Exponential Families on Resource-Constrained Systems

Dissertation

zur Erlangung des Grades eines

D o k t o r s d e r N a t u r w i s s e n s c h a f t e n

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Nico Philipp Piatkowski

Dortmund

2018

Tag der mündlichen Prüfung:	23.04.2018
Dekan:	Prof. Dr.-Ing. Gernot A. Fink
Gutachter:	Prof. Dr. Katharina Morik
	Prof. Dr. Stefano Ermon

Abstract

This work is about the estimation of exponential family models on resource-constrained systems. Our main goal is learning probabilistic models on devices with highly restricted storage, arithmetic, and computational capabilities—so called, ultra-low-power devices. Enhancing the learning capabilities of such devices opens up opportunities for intelligent ubiquitous systems in all areas of life, from medicine, over robotics, to home automation—to mention just a few. We investigate the inherent resource consumption of exponential families, review existing techniques, and devise new methods to reduce the resource consumption. The resource consumption, however, must not be reduced at all cost. Exponential families possess several desirable properties that must be preserved: Any probabilistic model encodes a conditional independence structure—our methods keep this structure intact. Exponential family models are theoretically well-founded. Instead of merely finding new algorithms based on intuition, our models are formalized within the framework of exponential families and derived from first principles. We do not introduce new assumptions which are incompatible with the formal derivation of the base model, and our methods do not rely on properties of particular high-level applications. To reduce the memory consumption, we combine and adapt reparametrization and regularization in an innovative way that facilitates the sparse parametrization of high-dimensional non-stationary time-series. The procedure allows us to load models in memory constrained systems, which would otherwise not fit. We provide new theoretical insights and prove that the uniform distance between the data generating process and our reparametrized solution is bounded. To reduce the arithmetic complexity of the learning problem, we derive the integer exponential family, based on the very definition of sufficient statistics and maximum entropy estimation. New integer-valued inference and learning algorithms are proposed, based on variational inference, proximal optimization, and regularization. The benefit of this technique is larger, the weaker the underlying system is, e.g., the probabilistic inference on a state-of-the-art ultra-low-power microcontroller can be accelerated by a factor of 250. While our integer inference is fast, the underlying message passing relies on the variational principle, which is inexact and has unbounded error on general graphs. Since exact inference and other existing methods with bounded error exhibit exponential computational complexity, we employ near minimax optimal polynomial approximations to yield new stochastic algorithms for approximating the partition function and the marginal probabilities. Changing the polynomial degree allows us to control the complexity and the error of our new stochastic method. We provide an error bound that is parametrized by the number of samples, the polynomial degree, and the norm of the model's parameter vector. Moreover, important intermediate quantities can be precomputed and shared with the weak computational device to reduce the resource requirement of our method even further. All new techniques are empirically evaluated on synthetic and real-world data, and the results confirm the properties which are predicted by our theoretical derivation. Our novel techniques allow a broader range of models to be learned on resource-constrained systems and imply several new research possibilities.

Contents

List of Acronyms

ASIC	Application-specific integrated circuits
BLprop	Bit-length propagation
BFGS	Broyden-Fletcher-Goldfarb-Shanno
BNC	Bayesian network classifier
BP	Belief propagation
CPU	Central processing unit
DBN	Dynamic Bayesian network
DCT	Discrete cosine transformation
DFS	Depth first search
DRAM	Dynamic random-access memory
EA	Evolutionary algorithm
FISTA	Fast iterative shrinkage thresholding algorithm
FLOPS	Floating point operations per second
FPGA	Field programmable gate array
FRAM	Ferroelectric random-access memory
GLM	Generalized linear model
GPU	Graphics processing units
IEEE	Institute of Electrical and Electronics Engineers
JT	Junction tree
KL-divergence	Kullback-Leibler-divergence
KL-function	Kurdyka-Łojasiewicz-function
LBP	Loopy belief propagation
LDS	Linear dynamical system
LP	Linear programming
MAP	Maximum a posteriori
MCMC	Markov chain Monte Carlo
MCU	Microcontroller unit
MF	Mean field
MIP	Mixed-integer programming
ML	Maximum likelihood
MRF	Markov random field
MSE	Mean squared error
NLP	Natural language processing
NNZ	Number of non-zeros
PAM	Perturb-and-MAP

RCDM	Randomized coordinate descent method
SCATS	Sydney coordinated adaptive traffic system
SGD	Stochastic gradient descent
SQM	Stochastic quadrature method
SRAM	Static random-access memory
STRF	Spatio-temporal random field
TRW	Tree-reweighting
ULP	Ultra-low-power
WISH	Weighted integrals and sums by hashing

List of Figures

List of Tables

List of Algorithms

List of Symbols

$\mathbb{R}$	The set of real numbers
$\mathbb{Q}$	The set of rational numbers
$\mathbb{N}$	The set of natural numbers
i, j, n, m, d, N, T	Positive integer variable or constants
$[n]$	The set $\{1, 2, \dots, n\}$
κ	Stepsize
λ	Regularization parameter
$\boldsymbol{M}, \boldsymbol{Q}, \boldsymbol{D}$	Matrices
$\boldsymbol{I}$	Identity matrix
$\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{W}$	Random variables
$\mathcal{X}, \mathcal{Y}, \mathcal{W}$	State spaces of a random variables
$\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}$	Realizations of a random variables
$\boldsymbol{x}^\top$	Transpose of $\boldsymbol{x}$
$\mathcal{D}$	A data set (multi-set) that contains N realizations of $\boldsymbol{X}$
$\boldsymbol{X} \perp\!\!\!\perp \boldsymbol{Y}$	Stochastic independence of random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$
$G = (V, E)$	Graph G with vertex set V and edge set E
K_n	Fully connected graph with n vertices
v, u, w	Vertices of G
$\mathcal{N}(v)$	Neighbors of v; all vertices u which are connected to vertex v
U, W	Subsets of $\mathbb{N}$ or V

$\mathcal{C}(G)$	Set of all cliques of graph G
ψ_C	Factor that belongs to clique C
$\overline{\mathcal{C}}(G)$	Set of all cliques with minimal factors
η	Reparametrization; function from $\mathbb{R}^{d'}$ to $\mathbb{R}^d$
$\mathbb{P}$	Generic probability measure
p	Generic probability density
$\boldsymbol{\theta}$	d-dimensional parameter vector
$p_{\boldsymbol{\theta}}$	Probability density with parameter $\boldsymbol{\theta}$
$\phi(\boldsymbol{X})$	Sufficient statistic of $\boldsymbol{X}$
$\boldsymbol{\mu}^*, \mathbb{E}[\phi(\boldsymbol{X})]$	Expected $\phi(\boldsymbol{X})$ w.r.t. the density of $\boldsymbol{X}$
$\hat{\boldsymbol{\mu}}, \mathbb{E}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})]$	Expected $\phi(\boldsymbol{X})$ w.r.t. the density with parameter $\boldsymbol{\theta}$
$\tilde{\boldsymbol{\mu}}, \tilde{\mathbb{E}}[\phi(\boldsymbol{X})], \mathbb{E}_{\mathcal{D}}[\phi(\boldsymbol{X})]$	Empirical expectation of $\phi(\boldsymbol{X})$ w.r.t. a data set $\mathcal{D}$
$\boldsymbol{\zeta}$	Vector of polynomial coefficients
χ^i_ϕ	Potential of i-dimensional tuple density w.r.t. ϕ
$[\![\cdot]\!]$	Equivalence class
$\binom{n}{k}, (n\ k)^\top$	Binomial coefficient "n choose k"
$\left\{\begin{matrix} n \\ k \end{matrix}\right\}, \{n\ k\}^\top$	Stirling number of second kind
$\langle \boldsymbol{\theta}, \phi(\boldsymbol{x}) \rangle$	Inner product between the vectors $\boldsymbol{\theta}$ and $\phi(\boldsymbol{x})$
$f \circ g$	Composition of functions f and g

1 Introduction

This work is about machine learning on devices that underlie specific resource constraints. In machine learning, data, collected from any reasonable data generating process, is used to *learn* the *parameters* of a function or *model* that in some well-defined way depends on the data. Learning is guided by a prespecified *loss function* that quantifies the quality of a specific choice of parameters—it measures how well the parameters fit to the data. Machine learning models can solve several abstract tasks: We might want to assign tags or labels to observed data, detect structure or interactions within data, or predict how the underlying data generating process might behave in the future. Most often, good model parameters are found via *numerical optimization*, i.e., the process of approaching the loss function's minimum by exploiting information about its derivatives. It is common that certain nice properties or quality assertions can be guaranteed for optimal parameter settings—such guarantees might be required if a method has to be applied in practice.

While the term "machine learning" subsumes a huge variety of models, parametrizations, loss functions and optimization methods, we restrict ourselves to probabilistic models that belong to the exponential family. On the one hand, probabilistic models allow us to represent *complex dependencies* between observable and unobservable entities explicitly. They let us *quantify uncertainty* in a consistent and *theoretically well-founded* manner, while being completely *agnostic* about the particular prediction or modelling task. On the other hand, the simple formal description of exponential families allows us to identify all sources of resource consumption. The exponential family arises from *first principles*, without incorporating assumptions or high-level rationalization efforts. Hence, results derived in this setting are valid for a manifold of tasks and models. Some specific instantiations of them have indeed been explored in the literature.

Even with the restriction to probabilistic models, the number of existing machine learning *applications* on or for resource-constrained systems is huge—it ranges from stereo matching [120, 39], image classification [58, 199, 197], image segmentation [196, 15], and speech recognition [219]; over localization [109, 30, 5], tracking [193, 101], health monitoring [218, 188, 249], and energy modelling [148]; to agriculture [88, 1], robotics [61, 62, 136, 15, 14], and sensor networks [38] as well as behavioral modelling [147, 52, 179, 54, 55, 72, 118, 53]. The large variety of possible applications of machine learning makes it more difficult but less essential to discuss each of them in-depth.

In general, running machine learning models in resource-constrained computational systems opens challenges in terms of both, execution time and energy consumption. Learning and applying the model directly on the ubiquitous device that actually measures the data has several advantages. First, it reduces the communication cost and thus the energy consumption. Second, not transferring data to a central server increases privacy and autonomy. Third, the measuring device itself can benefit from machine learning, e.g., learning from operation system log files allows for a more efficient "personalized" power management. These advantages are accompanied by restrictions of the computational capabilities which shrink the set of models that can be considered to be feasible.

However, the established computational paradigm of machine learning and other computationally demanding areas of computer science is to run algorithms on highly parallel compute servers with a maximum supply of cores, main memory, storage and interconnectivity. Here, maximum means as much as the current state-of-the-art architecture can handle. Moreover, some of the fastest systems to date are explicitly designed for machine learning and data analysis tasks [47, 46]. They are built to deliver the highest possible performance, but their resource consumption is far from optimal, especially when human-like performance is required on non-trivial tasks. While the typical adult human brain runs on less than 20 watts, the IBM Watson machine that defeated Jeopardy! champions depends on ninety compute servers, each of which consumes around one thousand watts [105]. AlphaGo, the system that has beaten professional Go players, runs on 1920 CPUs and another 280 GPUs [211]. While AlphaGo's exact energy consumption is unknown, the hardware suggests, that hundreds of thousands of watts are required. Indeed, the energy efficiency of high-end compute hardware, measured via FLOPS (floating-point operations per second) per watt, has increased during the last decade. At the same time, the complexity of state-of-the-art machine learning models do also increase—at least if model complexity is measured by the number of parameters or the depth of function compositions. However, human-like performance with human-like resource requirements can only be achieved if we adapt machine learning methods to the underlying hardware architecture.

It is true that in computer science, we usually want to abstract the software from the hardware whenever possible. If one wants to study the underlying algorithmic concept, this is a reasonable approach. However, if the properties of an algorithm—say, quality guarantees of a machine learning method—are already well understood, one may consider the computational capabilities of the underlying system explicitly and make full use of the whole system. Regarding the state-of-the-art machine learning systems mentioned above, explicit knowledge about the number of hardware threads, memory hierarchy, memory sizes, bus systems as well as network connectivity should be taken into account. Even new machine learning data structures and algorithms are designed with hardware specifics in mind [48, 43, 207, 76]. Tailoring algorithms to an underlying hardware architecture has become a trend also in programming graphics processing units (GPU), video game consoles, smartphones, cyber physical systems, or quantum computers. Moreover, with the ever increasing number of battery powered systems, consumer electronics, portable medical devices, robots, and several kinds of unmanned vehicles, multiple new areas for machine learning techniques emerge, where powerful hardware is occasionally not available—either because of its restricted size, weight or energy consumption. Hence, we should include knowledge about such resource constraints in our machine learning models, as we already do in case of hardware accelerators and memory hierarchies. Instead of fueling machine learning with electrical power, we should identify which computational capabilities are really necessary.

Requirements on high-level computational resources surely depend on the specific task. Some tasks have a high computational complexity, while others can be solved instantly. Some learning tasks inherently require large amounts of memory while others don't. These aspects are caused by the machine learning model and the corresponding algorithms, but the actual energy consumption depends on the low-level computational architecture, specified via clock-rates, memory types, functional units, and the number of clock-cycles that are required for arithmetic instructions [45]. Dynamic random-access memory (DRAM), for example, has to be refreshed in order to keep the values in memory—a procedure which consumes much more energy than, e.g., just reading the data [201, 202]. Hence, even the time for which values have to be stored can have an impact on the energy consumption. Such observations are of special importance when the underlying architecture is based on *microcontroller units* (MCU), *field programmable gate arrays* (FPGA), or *application-specific integrated circuits* (ASIC), which are usually equipped with the amount of memory that is required for a designated task.

At the time of writing this thesis, machine learning under resource constraints is discussed by some authors [113, 31, 238, 36], but none of them proposes to *adapt the model itself* to resource requirements and the underlying hardware architecture. Instead, merely algorithms or specific implementations are investigated. Some authors consider a so called *budget*, either of computation time that is available for prediction, or on the number of data samples that are available for training. Other authors develop and study specialized hardware implementations of algorithms for probabilistic models. E.g., FPGA and ASIC implementations of the sum-product algorithm [130] are presented in [200, 220, 51, 117], an FPGA implementation of Gibbs sampling [74] in [112, 220], and experimental results on resistive switching memory devices can be found in [65].

In contrast to these "pointwise" approaches, we want to move beyond including knowledge about the underlying hardware architecture into algorithms. We aim at a principled investigation of the machine learning model itself, with respect to the resources it consumes, and resource constraints that could arise. Exemplary constraints that could arise are:

- if the memory requirement of a model grows with the number of observed data points, it is a constraint to have a finite, constant amount of memory;
- if we need to evaluate a transcendental function, it is indeed a constraint to have no floating-point coprocessor;
- if computationally demanding algorithms are required, it is a constraint when the CPU has a low clock rate.

After constraints have been identified, we review existing methods for the reduction of resource consumption, discuss their shortcomings, and propose new techniques to leverage the constraints. However, the proposed techniques should not harm the model itself.

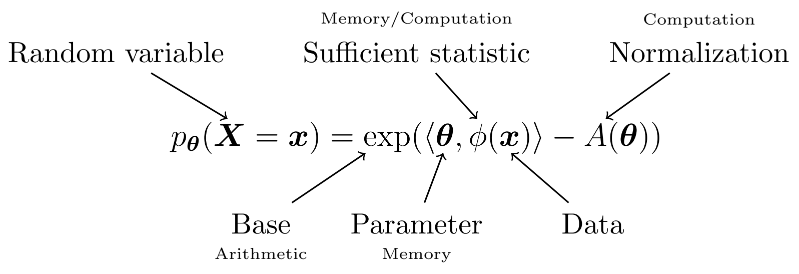
Memory/Computation
Computation
Random variable
Sufficient statistic
Normalization
$p_{\boldsymbol{\theta}}(\boldsymbol{X} = \boldsymbol{x}) = \exp(\langle \boldsymbol{\theta}, \phi(\boldsymbol{x}) \rangle - A(\boldsymbol{\theta}))$
Base
Parameter
Data
Arithmetic
Memory

Figure 1.1: Essential components of exponential family models for a (n-dimensional) random variable $\boldsymbol{X}$.

1.1 Approach and Techniques

Our central object of study is the exponential family [227]. To understand which resources are consumed and why they are consumed, one has to disassemble the model into its parts. The essential components of an exponential family model, together with their primary resource requirement, are sketched in Fig. 1.1. Within this thesis, we analyze these components, grouped by their primary resource consumption, where we devote a separate chapter to each resource. Therein, existing methods to reduce the amount of required resources are reviewed and discussed. The insights gained from these discussions allow us to extend the state-of-the-art by adapting the essential components of the exponential family to specific constraints. The resource consumption, however, must not be reduced at all cost. Exponential family members possess several desirable properties that should be preserved by our proposed extensions. More precisely:

1. Any probabilistic model encodes a conditional independence structure—our methods must keep this structure intact.

2. Exponential family models are theoretically well-founded. Instead of merely proposing new algorithms, our extensions and their effects must be formalized within the framework of exponential families.

3. Exponential family models are derived from first principles—our techniques must not introduce new assumptions which are incompatible with the formal derivation of the base model.

4. Our extensions should not rely on properties of particular high-level applications or tasks—this does not exclude low-level characteristics like discrete random variables or time-series data.

By restricting ourselves to techniques which satisfy the above requirements, we ensure their generality and compatibility with each other and with existing methods. Moreover, this eases the analysis of the theoretical implications of our techniques w.r.t. the model's resource consumption and quality.

Some exponential family related tasks have a rather high computational complexity—at least in their most general realization. In practice, these hard tasks are either completely avoided or approximated and a lot of research is devoted to the study of such *approximate inference* techniques. We indeed rely on some existing techniques, like variational inference or sampling, and extend them to fit specific resource constraints. Motivated by the nature of the essential components which are shown in Fig. 1.1, our extensions are based on reparametrization, regularization, as well as numerical and arithmetic approximation:

- **Reparametrization** is a standard concept of the exponential family. Instead of using a plain vector, say $\boldsymbol{\theta}$, to parametrize the model, one employs another (possibly lower dimensional) vector $\boldsymbol{\Delta}$ which can be "uncompressed" by some function η to generate the natural parameter $\boldsymbol{\theta} = \eta(\boldsymbol{\Delta})$. We use this technique to remove redundancies in $\boldsymbol{\theta}$ which are then not contained in $\boldsymbol{\Delta}$.

- In machine learning, **regularization** is known to prevent models from overfitting and to induce sparsity [203, 157, 86]. In the context of probabilistic models, regularization has been shown to detect the conditional independence structure between random variables [203, 144, 189, 240]. However, when the correct structure is known, further regularization will remove conditional indecencies from the model. Since one of our requirements is to keep the conditional independence structure intact, we propose a combination of regularization and reparametrization which results in sparse models without touching the structure. This reduces the memory consumption of the model. Regularization also plays a central role when dealing with arithmetic limitations.

- **Arithmetic approximation** is inherent in digital computers. Real-valued numbers are stored with a finite amount of bits. If the native precision of a number exceeds the bit-length, rounding errors are unavoidable without increasing the bit-length of the representation. In settings where many calculations are approximate or based on corrupted or noisy data, even ordinary 32 or 64 precision might be spurious. Recent results in machine learning suggest that the precision can safely be reduced without harming the quality of the resulting model [80, 138]. We drive this idea to an extreme, in which all computations are required to be integer-valued. This reduces the required complexity of the underlying computational circuit and prevents overfitting.

- Polynomial functions underlay a multitude of mathematical theories and applications. They are, among other things, used to find approximations to complicated functions [142], while possessing many appealing properties like error bounds and relatively easy estimation procedures. We employ a **polynomial approximation** of the core of any exponential family, namely to the potential function $\exp(\langle\boldsymbol{\theta}, \phi(\boldsymbol{x})\rangle)$, to derive a novel sampling-based inference technique. This reduces the computational complexity and guarantees error bounds, while maintaining the full conditional independence structure.

While these techniques aim at reducing the usage of some target resource, they also introduce other side-effects to the model and its resource consumption. E.g., restricting every computation to 16 bit integer arithmetic removes the need for a floating-point coprocessor, lowers the memory requirements (compared to 32 bit floating-point numbers), and introduces an approximation error. In the respective chapters, we discuss these side-effects, and derive quality guarantees.

1.2 Organization

Throughout this thesis, we analyze to what extend the resource consumption of exponential family models can be reduced via reparametrization, regularization, and numerical and arithmetic approximation. Our techniques are motivated by and derived from the model's essential components and their primary resource requirements (cf. Fig. 1.1). Three chapters, each devoted to one resource, establish the body of this thesis. Within each chapter, we investigate which parts of the model consume the respective resource, review existing methods, and identify possible improvements. We then propose extensions to the state-of-the-art to address the identified shortcomings, derive their theoretical and algorithmic implications, and conduct experimental analyses. These chapters are framed by a background chapter, containing the mathematical and statistical foundations, and a discussion chapter, in which we summarize our work and discuss our methods in the context of ultra-low-power (ULP) microcontroller units. The contents can be summarized as follows:

Background. Chapter 2 consists of the material that is required for a sound presentation of the rest of the thesis. The presentation of this material is by necessity brief, but it provides sufficient details to follow the methods and analysis presented in subsequent chapters. The chapter begins with an overview on basic terms and definitions of probability, information theory, conditional independence structures, and probabilistic inference, altogether building the essential foundation of probabilistic models. Exponential family models, their estimation, and their key properties are presented in Section 2.4. Parameter estimation and regularization are important topics throughout the this theses. They depend strongly on results and techniques from numerical optimization which we introduce in Section 2.3. In the remainder, we focus on techniques from numerical integration and polynomial approximation which form the base for a new inference algorithm.

Memory. The amount of random-access memory that is available to a system is limited. In addition refreshing the state of DRAM cells constantly consumes energy [201, 202]. In Chapter 3, we investigate all parts of the model that need to be stored and require actual memory. While the memory requirement for the data is fixed by the so-called sufficient statistic, the number of model parameters can be controlled via reparametrization. Classic reparametrization approaches from statistical physics [103] and natural language processing [198] are reviewed, and discussed in the context of generic time-series sensor data which is gathered by mobile or autonomous systems. The discussion leads to a new kind of piecewise-linear reparametrization. Combining the new reparametrization with l_1-regularization allows the model to detect redundancies in its natural parameters which results in sparse models that cannot be found using plain regularization. The technique, called spatio-temporal random field (STRF) [169], has already proven to be useful for modelling car traffic and traffic congestion [168, 7, 190, 137], usage of mobile network cells [147], and smartphone app usage [174]. Moreover, we explain how to extend STRF to non-stationary linear dynamical systems [173].

Arithmetic. The second resource type arises from the actual operations that are required to evaluate the model. The core of the exponential family is the exponential function, which is transcendental and inherently requires real-valued arithmetic. Circuits for floating-point arithmetic are larger and more complex than circuits for integer arithmetic and therefore, require more resources. Hence, resource-constrained systems may not be equipped with floating-point units; they have to emulate floating-point arithmetic via integer arithmetic, which is very costly in terms of clock cycles. Thus, machine learning with low-precision arithmetic and quantization recently became a vivid research area [80, 49, 37, 138]. In Chapter 4, we derive new integer-only algorithms and corresponding error bounds for learning and inference in exponential families members with integer parameters [170, 171]. Integer-only inference is based on approximating the bit-lengths of marginal probabilities, while the parameter estimation procedure is based on an integer regularization method—both techniques are entirely new.

Computation and Quality. We discuss the interplay of computational complexity and approximation quality in Chapter 5. Processing units are restricted in terms of their physical size and the heat they generate. Computing exact (marginal) probabilities is $\#\mathbf{P}$-complete in general [212, 229]. For certain sub-classes, however, polynomial-time algorithms are known [227, 191, 78], and significant research efforts have been conducted in order to derive approximations for more general cases. However, most existing methods either provide no error bounds or exhibit a computational complexity that prevents them from running on resource-constrained systems. We consider a near-minimax optimal polynomial approximation to the potential function of the exponential family. This allows us to derive new deterministic and randomized algorithms for the partition function and the marginal probabilities. The randomized algorithm relies on low-dimensional Monte Carlo sampling and exhibits low resource requirements. The approximation error of the method is bounded and depends on the polynomial degree, the norm of the model's parameter vector, and the number of Monte Carlo samples.

1.3 Contributions

Our results have implications for both, the abstract topic of this thesis (probabilistic models on resource-constrained systems) and for a more general understanding of the fine grained complexity of exponential family models. We summarize the most important contributions in the order in which they appear in the thesis. In the third chapter,

- we combine and adapt reparametrization and regularization in an innovative way that facilitates a sparse parametrization of high-dimensional non-stationary time-series, without altering the conditional independence structure. The procedure is completely novel and allows us to load models in memory constrained systems, which would otherwise not be possible.
- We provide new theoretical results, showing that our piecewise linear reparametrization is universal, in the sense that it is a bijection between the space of all natural parameter vectors and the space of all reparametrized vectors.
- Moreover, we prove that any sparsity in the reparametrized vector implies a neglectable difference in the parameters of consecutive time steps, and that the uniform distance between the parameter vector of the data generating process and the reparametrized solution is bounded. Both statements hold with high probability.
- We derive the Lipschitz constants of the reparametrized model to facilitate fast parameter learning without stepsize adaption.
- Even in the non-constrained setting, our technique reduces the effective degrees of freedom and hence, eases the learning problem. This contributes in a general manner to the area of probabilistic spatio-temporal modelling.

In Chapter 4, based on a classic result by Pitman [175],

- we show that the exponential family can be derived with any base. Following this result, we present the base-2 exponential family. By restricting the parameter space to the positive integers, we show that the exponential function (which inherently requires real-valued arithmetic) can be replaced by a simple bit-shift operation.
- We phrase the integer restriction as another reparametrization, and derive a bound on the difference of log-likelihood values of the true model and the integer model.
- Based on loopy belief propagation (LBP), we derive the bit-length propagation algorithm for probabilistic inference. The corresponding message computations can be carried out without any real-valued computation. By assuming uniform and independent rounding errors, we show that the expected Kullback-Leibler divergence between the result of bit-length propagation and the corresponding LBP result is bounded.
- Existing methods for mixed-integer optimization rely on real-valued computations, which is by assumption slow on resource-constrained systems. Hence, we introduce an entirely new method for optimization over the integers. To this end, we employ a blockwise proximal gradient descent method with a new type of regularization. Formally, the optimization is still carried out over the real-valued space. However,

we show that for a specific choice of regularization weight, all solutions (and all intermediate values) generated by our new integer gradient descent method are integers.

- Beyond integer-valued exponential families, our new optimization method is a general alternative to branch-and-bound and outer-approximation techniques, and may hence be regarded as contribution to the field of mixed-integer programming.

In the fifth chapter, based on results in numerical approximation theory,

- we derive a near-minimax optimal polynomial approximation to the potential function. While this idea is well known for the approximation of intractable integrals, it has never been applied to the partition function of the exponential family for discrete random variables.
- We derive an error bound on our new approximation to the partition function. The error depends on the degree of the underlying polynomial approximation, which in turn depends on the norm of the parameter vector. Since the norm can be controlled via regularization, we discover yet another connection between regularization and the resource consumption of the model.
- We identify a new property of sufficient statistics that allows us to derive an algorithm for the partition function whose complexity is polynomial in the parameter dimension and exponential in the polynomial degree. Moreover, we show that the sufficient statistic of any discrete state Markov random field has this property, and we discover under which circumstances continuous sufficient statistics also have this property.
- Based on a new probability mass function over k-ary index tuples, we derive a Monte Carlo sampling method whose result has a bounded relative distance to the true partition function with high probability. Due to its low resource requirements, the sampling method is well suited for resource-constrained systems.
- Moreover, we show how our algorithms can be used for marginal inference and approximate maximum likelihood estimation.
- Since only a few approaches are known for the approximation of partition functions, the new method can be regarded as contribution to the field of probabilistic inference in general.

Finally, in Chapter 6,

- we discuss our findings in the context of ULP microcontroller units, and present new perspectives for future research.
- Specifically, we study by how much the proposed methods increase the size of models that can be applied on real-world ULP systems and perform exemplary benchmarks. The results show, that models can be applied on systems which are inconceivable without our modifications.

- Since basically no results for general exponential family models on ULP systems exists, our findings can guide the choice of appropriate hardware architectures for future machine learning applications in resource-constrained systems.

In summary, the work presented here forms a cohesive investigation of learning and applying exponential family models under resource constraints. While each part has its own implications, they are inherently connected by regularization, i.e., the idea that constraints on the model space should reflect our knowledge about specific resource constraints. We hope that the results herein will prove useful for many more problems than those we have explicitly addressed. Some ideas for future directions are outlined in Chapter 6.

1.4 Acknowledgements

During the investigations which finally resulted in this thesis, intermediate states of research have been submitted or published in the form of conference and journal papers. We summarize at the end of each chapter in how far this thesis extends the already published results. I thank all of my coauthors who supported me in developing some basic steps towards exponential families on resource-constrained systems. The basics of regularization and reparametrization of spatio-temporal models are based on research done together with Sangkyun Lee and Katharina Morik [168, 169]. The experimental results on car traffic data are inspired by work with Alexander Artikis, François Schnitzler and Thomas Liebig [7, 190, 137]. The extension of the spatio-temporal model to multivariate Gaussian data is joint work with François Schnitzler [173]. Work on integer undirected models has also been conducted with Sangkyun Lee and Katharina Morik [170, 171]. The integer regularization technique is based on binary regularization which is joint work with Sibylle Hess [93]. Initial experiments on smartphone data were done together with Stefan Michaelis, Jochen Streicher, Olaf Spinczyk and Katharina Morik [147, 174]. My work has been funded by the Deutsche Forschungsgemeinschaft (DFG) via the collaborative research center SFB 876, project A1.

2 Background

Probability theory is the basic language in which our models and ideas are phrased. It is essential for almost all sciences and unifies the idea that most natural phenomena are either too complex or simply not fully observed to construct deterministic models. We provide the basic terms and definitions of probability and information theory in the appendix. Here, we start with a summary of our notation, followed by the core concepts, namely probabilistic graphical models and exponential families.

2.1 Notation

Uppercase, boldface letters, like $\boldsymbol{X}$, denote random variables (Definition 7.2). Each random variable has its own state space, which is denoted by a calligraphic version of the random variable's letter, e.g., $\mathcal{X}$ denotes the state space of $\boldsymbol{X}$. State spaces are sets that contain all possible values, also called realizations, a random variable can take. Generic realizations are denoted by lowercase boldface letters, like $\boldsymbol{x}$. The event that a random variable $\boldsymbol{X}$ takes a specific value $\boldsymbol{x} \in \mathcal{X}$ is denoted by $\boldsymbol{X} = \boldsymbol{x}$, and the probability density[1] of this event is $p(\boldsymbol{x}) = p(\boldsymbol{X} = \boldsymbol{x})$. Conditional probabilities are denoted by $p(\boldsymbol{X} = \boldsymbol{x} \mid O)$ for an arbitrary event O. Details about probability measures and densities can be found in Section 7.1. If multiple realizations of a random variable are collected, we enumerate them and address them via superscript, i.e., $\boldsymbol{x}^1, \boldsymbol{x}^2, \boldsymbol{x}^3$ are three arbitrary but fixed realizations of the random variable $\boldsymbol{X}$.

Here, any random variable has a non-zero, integer dimension $n > 0$. The i-th dimension of $\boldsymbol{X}$ is addressed by subscript: $\boldsymbol{X}_i$ $(1 \leq i \leq n)$. Moreover, multiple dimensions may be addressed at once via sets of indices $U \subset \mathbb{N}$, e.g., $\boldsymbol{X}_U$. The same notation is used to address single or multiple dimensions of $\mathcal{X}$ and $\boldsymbol{x}$, e.g., $\mathcal{X}_i, \mathcal{X}_U, \boldsymbol{x}_i, \boldsymbol{x}_U$—$\mathcal{X}_i$ is the state space of the i-th component of $\boldsymbol{X}$, and $\boldsymbol{x}_i$ is a possible realization of $\boldsymbol{X}_i$.

2.2 Graphical Models and Exponential Families

Any n-dimensional random variable $\boldsymbol{X}$ can obey a large number of conditional independences (Definition 7.5). Independence assertions between components of a random variable are denoted by

$$\boldsymbol{X}_i \perp\!\!\!\perp \boldsymbol{X}_j \quad \text{and} \quad \boldsymbol{X}_i \perp\!\!\!\perp \boldsymbol{X}_j \mid \boldsymbol{X}_z \tag{2.1}$$

[1]As explained in the appendix, the terms "probability measure", "probability density", and "probability mass function" are identical in case of discrete random variables. We will hence use the term "density" for both, discrete and continuous random variables.

for regular and conditional independences, respectively. Apparently, it involves several notational inconveniences to work directly with formal statements like (2.1) when $\boldsymbol{X}$ is high dimensional. It is hence natural to ask for a more compact and intuitive representation of conditional independences. The formalism of probabilistic graphical models provides a unifying framework for theoretical derivations, algorithms, and the estimation of large-scale multivariate statistical models [227].

Definition 2.1 (Graphical Model) *Let $\boldsymbol{X}$ be an n-dimensional random variable, and $G = (V, E)$ a graph with vertex set $V = [n]$ and edge set E. The n vertices of G are identified with the n dimensions of $\boldsymbol{X}$—vertex $v \in V$ corresponds to $\boldsymbol{X}_v$. The set of independences which are encoded by a graph is denoted by $I(G)$.*

In undirected models, also known as Markov random fields (MRF), the edge set encodes conditional independences of each vertex given its neighbors $\mathcal{N}(v) = \{u \in V \mid v, u \in E\}$, i.e.,

$$\forall v \in V : \boldsymbol{X}_v \perp\!\!\!\perp \boldsymbol{X}_{V \setminus (\{v\} \cup \mathcal{N}(v))} \mid \boldsymbol{X}_{\mathcal{N}(v)} \ . \tag{2.2}$$

The set $\mathcal{N}(v)$ is also called Markov blanket of vertex v (denoted by $\mathbb{M}_G(v)$), and independence assertions like (2.2) are called local Markov properties.

In directed graphs, each vertex $v \in V$ is conditionally independent of all other vertices given its parents (Parents$(v) = \{u \in V \mid (u, v) \in E\}$), its children (Children$(v) = \{u \in V \mid (v, u) \in E\}$), and the other parents of its children. I.e., the Markov blanket of a vertex v in a directed graph is

$$\mathbb{M}_G(v) = \text{Parents}(v) \cup \text{Children}(v) \bigcup_{u \in \text{Children}(v)} \text{Parents}(u) \setminus \{v\}.$$

Besides their intuitive representation, the particular way how graphical models encode conditional dependences imply algebraic properties of the underlying probability density, which will eventually lead to efficient algorithms as well as a canonical representations of data.

Theorem 2.1 (Directed Factorization [122]) *Let $\boldsymbol{X}$ be a random variable with density p and let G be a directed graph. Let further $I(\boldsymbol{X})$ be the set of local conditional independence assertions of $\boldsymbol{X}$, and let $I(G)$ be the set of conditional independence assertions implied by G. The density of $\boldsymbol{X}$ factorizes according to G, i.e.,*

$$p(\boldsymbol{x}) = \prod_{v \in V} p_v(\boldsymbol{x}_v \mid \boldsymbol{x}_{\text{Parents}(v)}),$$

if and only if $I(G) \subseteq I(\boldsymbol{X})$.

In directed models, each factor p_v is itself a probability density, and is hence normalized. In case of undirected models, a similar yet significantly different statement can be made.

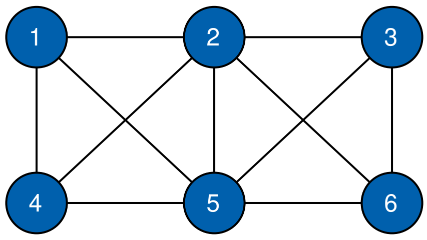

Figure 2.1: An undirected graph $G = (V, E)$, with $V = \{1, 2, 3, 4, 5, 6\}$. The cliques $A = \{1, 2, 4, 5\}$ and $B = \{2, 3, 5, 6\}$ share the edge $\{2, 5\}$.

Theorem 2.2 (Undirected Factorization [84]) *Let $\boldsymbol{X}$ be a random variable with strictly positive density p, G an undirected graph, and $\mathcal{C}(G)$ the set of its cliques. Let further $I(\boldsymbol{X})$ be the set of conditional independence assertions of $\boldsymbol{X}$, and let $I(G)$ be the set of conditional independence assertions implied by G. The density of $\boldsymbol{X}$ factorizes according to G, i.e.,*

$$p(\boldsymbol{x}) = \frac{1}{Z}\psi(\boldsymbol{x}) = \frac{1}{Z}\prod_{C\in\mathcal{C}(G)} \psi_C(\boldsymbol{x}_C),$$

if and only if $I(G) \subseteq I(\boldsymbol{X})$. Here, $Z = \int \psi(\boldsymbol{x})\,\mathrm{d}\,\nu(\boldsymbol{x})$ is a normalization constant, and ψ a positive function that factorizes into positive functions ψ_C.

Note that in both Theorems (2.1 and 2.2), the independences encoded by the graphical structure are not asked to contain all independences of $\boldsymbol{X}$ to yield a valid factorization. However, the more independences are captured by the graph, the finer is the corresponding factorization, and hence, the lower the complexity of probabilistic inference. In the extreme case, G is fully connected, which means that no conditional independences are captured, i.e., $I(G) = \emptyset$.

In contrast to directed models, the factors ψ_C of undirected models are not (necessarily) normalized. This particularity allows for more flexibility in parametrizing and estimating undirected models, but the joint density p requires explicit normalization via the so-called partition function Z (Section 2.2.2). The factors will play an important role for our contributions.

Definition 2.2 (Potential Function) *Let $\psi : \mathcal{X} \to \mathbb{R}_+$ be a non-negative function that factorizes along the cliques of an undirected graph G, i.e., $\psi(\boldsymbol{x}) = \prod_{C\in\mathcal{C}(G)} \psi_C(\boldsymbol{x}_C)$. ψ is called potential function of the corresponding undirected graphical model, and its factors ψ_C are called clique potentials or clique factors. The number of input values $|C|$ is the dimension of the clique factor ψ_C.*

Remark 2.1 *In general, any clique factor ψ_C is allowed to factorize further, e.g., $\psi_C(\boldsymbol{x}_C) = \psi_A(\boldsymbol{x}_A)\psi_B(\boldsymbol{x}_B)$, $A, B \subset C$. The factors ψ_A and ψ_B are "smaller" than ψ_C, in that they have a smaller dimension than ψ_C. A clique factor ψ_C that cannot be written as the product of smaller factors is called minimal, and we denote the set of all*

$$\phi_U(\boldsymbol{x}_U^1) = \begin{pmatrix}0\\\mathbf{1}\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\end{pmatrix} \quad \phi_U(\boldsymbol{x}_U^2) = \begin{pmatrix}0\\0\\0\\0\\0\\0\\0\\\mathbf{1}\\0\\0\\0\\0\end{pmatrix} \quad \phi_U(\boldsymbol{x}_U^3) = \begin{pmatrix}0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\\mathbf{1}\\0\end{pmatrix} \quad \mathcal{X}_U = \begin{array}{ll}\{(A,0,+1), & \\ (A,0,-1), & \leftarrow \boldsymbol{x}_U^1\\ (A,1,-1), & \\ (A,1,+1), & \\ (B,0,+1), & \\ (B,0,-1), & \\ (B,1,-1), & \\ (B,1,+1), & \leftarrow \boldsymbol{x}_U^2\\ (C,0,+1), & \\ (C,0,-1), & \\ (C,1,-1), & \leftarrow \boldsymbol{x}_U^3\\ (C,1,+1)\} & \end{array}$$

Figure 2.2: Sufficient statistic of a vertex set $U = \{u, v, w\}$ with vertex state spaces $\mathcal{X}_u = \{A, B, C\}$, $\mathcal{X}_v = \{0, 1\}$, $\mathcal{X}_w = \{-1, +1\}$. Let $\boldsymbol{x}_U^1 = (A, 0, -1)$, $\boldsymbol{x}_U^2 = (B, 1, +1)$, and $\boldsymbol{x}_U^3 = (C, 1, -1)$. The entries of ϕ_U correspond to indicator functions for each possible state in $\mathcal{X}_U = \mathcal{X}_u \otimes \mathcal{X}_v \otimes \mathcal{X}_w$, listed on the right.

cliques with minimal factors by $\overline{\mathcal{C}}(G) \subseteq \mathcal{C}(G)$. An undirected model is called pairwise, if any minimal factor involves at most two variables.

Since the dimensionality of factors plays an important role for the resource consumption of the model, we will always use a factorization in terms of cliques with minimal factors, e.g., $\psi(\boldsymbol{x}) = \prod_{C \in \overline{\mathcal{C}}(G)} \psi_C(\boldsymbol{x}_C)$. Moreover, all minimal factors that belong to the same clique are merged.

For brevity, we will omit the term "clique with minimal factor" in the sequel, and simply refer to those as the clique factors of simply factors of the model.

This situation ins depicted in Fig. 2.1. Assume that the factors of the cliques $A = \{1, 2, 4, 5\}$ and $B = \{2, 3, 5, 6\}$ factorize further into edge factors, i.e., the corresponding model is pairwise. Since the clique $D = \{2, 5\}$ is shared by A and B, two different factors $\psi_{A,2,5}(\boldsymbol{x}_D)$ and $\psi_{B,2,5}(\boldsymbol{x}_D)$, both corresponding to the clique D, would appear in the overall factorization of the model. Whenever this happens, we define a new factor $\psi_D(\boldsymbol{x}_D) = \psi_{A,2,5}(\boldsymbol{x}_D)\psi_{B,2,5}(\boldsymbol{x}_D)$. Hence, the final pairwise factorization $\psi(\boldsymbol{x}) = \prod_{e \in E} \psi_e(\boldsymbol{x}_e)$, will contain (at most) one potential for each edge $e \in E$—no matter how many cliques share the edge e.

There is an alternative graphical representation of undirected models in terms of $\overline{\mathcal{C}}(G)$, via so-called factor graphs [130]. Since both representations are equivalent, we denote undirected models via undirected graphs.

2.2.1 The Exponential Family of Densities

Definition 2.2 is merely the declaration of what we consider a potential function. Indeed, any arbitrary non-negative function ψ of n variables induces an undirected graphical model, regardless of its factorization properties. If ψ does not factorize at all, the corresponding model consists of a single clique and the graphical structure is K_n (the fully connected graph). In case ψ factorizes, each factor depends on a subset of all n variables. Each of these subsets of variables defines a clique, and the edges of the graphical model are then implied by the set of cliques.

In this thesis, we focus on undirected models, and the term graphical model always refers to an undirected model, unless otherwise explicitly mentioned. We do now proceed to the main formalism of this thesis:

Definition 2.3 (Exponential Family) *Let $\boldsymbol{X}$ be a random variable and p its density. Moreover, let $p = p_{\boldsymbol{\theta}}$ be parametrized by a vector $\boldsymbol{\theta} \in \mathbb{R}^d$. Then, $p_{\boldsymbol{\theta}}$ belongs to an exponential family of densities, if it can be written in the following form:*

$$p_{\boldsymbol{\theta}}(\boldsymbol{X} = \boldsymbol{x}) = \exp(\langle \boldsymbol{\theta}, \phi(\boldsymbol{x}) \rangle - A(\boldsymbol{\theta})) = \frac{1}{Z(\boldsymbol{\theta})} \psi(\boldsymbol{x}) \ . \tag{2.3}$$

Here, $A(\boldsymbol{\theta}) = \log Z(\boldsymbol{\theta})$ is the log-partition function, $\psi(\boldsymbol{x}) = \exp(\langle \boldsymbol{\theta}, \phi(\boldsymbol{x}) \rangle)$ the potential function, and $\phi(\boldsymbol{x})$ is the sufficient statistic of the model.

The sufficient statistic encodes the conditional independence structure of $\boldsymbol{X}$, i.e., ϕ is actually a function of G. Nevertheless, the notation $\phi_G(\boldsymbol{X})$ is omitted, whenever the underlying structure is not ambiguous. More formally:

Definition 2.4 (Sufficiency) *Let $\boldsymbol{X}$ be a random variable with state space $\mathcal{X}$, let further $\boldsymbol{x}$ be an arbitrary realization of $\boldsymbol{X}$, and let ϕ be some function from $\mathcal{X}$ to $\mathbb{R}^d$. Moreover, let the density of $\boldsymbol{X}$ be parametrized by $\boldsymbol{\theta}$. Then, ϕ is sufficient for $\boldsymbol{\theta}$, if*

$$p(\boldsymbol{\theta} \mid \boldsymbol{x}, \phi(\boldsymbol{x})) = p(\boldsymbol{\theta} \mid \phi(\boldsymbol{x})) \ .$$

I.e., given $\phi(\boldsymbol{x})$, the parameter $\boldsymbol{\theta}$ becomes independent of $\boldsymbol{x}$.

We know from Theorem 2.2 that undirected models factorize over the cliques of their conditional independence structure. It is, however, not yet clear how such a factorization can be achieved with exponential families. In case of discrete random fields $\boldsymbol{X}$, there is a canonical representation, based on simple indicator functions, which we now explain.

Definition 2.5 (Sufficient Statistics for Discrete Data) *Let $\boldsymbol{X}$ be a discrete, n-dimensional random variable, $G = (V, E)$ its undirected conditional independence structure, $U \subseteq V$, and $\mathbb{1}_{\{\boldsymbol{x}_U = \boldsymbol{x}'_U\}}$ an indicator function that evaluates to 1 if and only if $\boldsymbol{x}_U = \boldsymbol{x}'_U$ and 0 otherwise. Let $\mathcal{X}_U = \bigotimes_{v \in U} \mathcal{X}_v$ be the joint joint state space of the vertex set U; $\mathcal{X}_U$ contains $|\mathcal{X}_U| = \prod_{v \in U} |\mathcal{X}_v|$ elements. The overcomplete, binary, sufficient statistic of vertex set U, $\phi_U : \mathcal{X} \to \{0,1\}^{|\mathcal{X}_U|}$, is defined via*

$$\phi_U(\boldsymbol{x}) = \left(\prod_{v \in U} \mathbb{1}_{\{\boldsymbol{x}_v = \boldsymbol{x}'_v\}} : \forall \boldsymbol{x}' \in \mathcal{X}_U \right)^\top \ .$$

$\phi_U(\boldsymbol{x})$ is hence a $|\mathcal{X}_U|$-dimensional column vector that contains exactly one bit for each possible joint state of the variables in U. $\phi_U(\boldsymbol{x})$ contains exactly one 1-entry—all other entries are 0. The order in which the joint state indicators appear in $\phi_U(\boldsymbol{x})$ is arbitrary but fixed. For ease of notation, the domain of ϕ_U might be reduced to $\mathcal{X}_U$, i.e., $\phi_U(\boldsymbol{x}) = \phi_U(\boldsymbol{x}_U)$. Sufficient statistics may be overcomplete. Overcompleteness means that there exists some vector $\boldsymbol{a} \in \mathbb{R}^d$ and a constant $b \in \mathbb{R}$, such that $\langle \boldsymbol{a}, \phi(\boldsymbol{x})\rangle = b$ holds for all $\boldsymbol{x} \in \mathcal{X}$.

In the remainder of this thesis, overcomplete sufficient statistics are used without loss of generality. Any overcomplete statistic can be reduced to an equivalent minimal formulation by eliminating components of ϕ until no affine dependencies remain (cf. Appendix B.1 in [227]). Although minimality sounds like an appealing property when the focus lies on the consumption of resources, it is mainly an arithmetic property. Moreover, the overcomplete sufficient statistics that we consider in this thesis, are a merely theoretical tool and not fully instantiated in practice. They hence consume almost no resources. An exemplary sufficient statistic for different realizations of a generic vertex set $U = \{u, v, w\}$ is shown in Fig. 2.2. Note that only the 1-entries have to be known to represent the statistic—we will revisit this observation in Chapter 3, when we discuss the memory consumption of exponential family models.

The statistics of all clique factors are stacked together, to construct the sufficient statistic of the model.

Definition 2.6 (Sufficient Statistic for Discrete Graphical Models) *Let $\boldsymbol{X}$ be a discrete, n-dimensional random variable with undirected conditional independence structure $G = (V, E)$. The overcomplete sufficient statistic $\phi : \mathcal{X} \to \mathbb{R}^d$ of $\boldsymbol{X}$ is*

$$\phi(\boldsymbol{x}) = (\phi_C(\boldsymbol{x})^\top : \forall C \in \overline{\mathcal{C}}(G))^\top .$$

$\phi(\boldsymbol{x})$ is hence a d-dimensional column vector, with $d = \sum_{C \in \overline{\mathcal{C}}(G)} |\mathcal{X}_C|$, that contains indicator functions for the states of all cliques in $\overline{\mathcal{C}}(G)$. The order in which the sufficient statistics of cliques appear in $\phi(\boldsymbol{x})$ is arbitrary but fixed. Single components may be addressed via specific joint states in the subscript, i.e., $\phi(\boldsymbol{x})_{C=\boldsymbol{x}'} = \prod_{v \in C} \mathbb{1}_{\{\boldsymbol{x}_v = \boldsymbol{x}'_v\}}$.

The sufficient statistic defined above is universal—it can be constructed for any discrete random variable $\boldsymbol{X}$, i.e., the existence of this kind of sufficient statistic is not an assumption.

Definition 2.7 (Potential Function of Exponential Families) *Let $\boldsymbol{X}$ be a discrete, n-dimensional random variable with undirected conditional independence structure $G = (V, E)$. Moreover, let $\boldsymbol{\theta} \in \mathbb{R}^d$, with $d = \sum_{C \in \overline{\mathcal{C}}(G)} |\mathcal{X}_C|$, be the parameter vector of an exponential family density of $\boldsymbol{X}$ and ϕ its sufficient statistic. The vector $\boldsymbol{\theta}$ is composed out of sub-vectors $\boldsymbol{\theta}_C \in \mathbb{R}^{|\mathcal{X}_C|}$, where each dimension of $\boldsymbol{\theta}_C$ is the weight for a specific joint state of the variables contained in clique $C \in \overline{\mathcal{C}}(G)$. The order in which the weights appear in $\boldsymbol{\theta}_C$ is fixed by the order in which the joint state indicators appear in the corresponding sufficient statistic $\phi_C(\boldsymbol{x})$. Each clique factor can hence be written as*

$$\psi_C(\boldsymbol{x}) = \exp(\langle \boldsymbol{\theta}_C, \phi_C(\boldsymbol{x}_C)\rangle) .$$

Feeding the whole joint state $\boldsymbol{x}$ (instead of $\boldsymbol{x}_C$) into the potential function is merely done to simplify the notation. In addition, we will allow to partition the argument of ψ in any conceivable way. E.g., when $C = \{u, v, w\}$, the following equivalent expressions are all identical to $\psi_C(\boldsymbol{x})$:

$$\psi_C(\boldsymbol{x}_C) = \psi_C(\boldsymbol{x}_u, \boldsymbol{x}_v, \boldsymbol{x}_w) = \psi_C(\boldsymbol{x}_{\{u,v\}}, \boldsymbol{x}_w) = \psi_C(\boldsymbol{x}_{\{u,w\}}, \boldsymbol{x}_v) = \psi_C(\boldsymbol{x}_{\{w,v\}}, \boldsymbol{x}_u) \; .$$

One should get the idea that the order in which we feed the joint state of C to ψ has no relevance, as long as the argument suffices to reconstruct the values of the sufficient statistic. We will use this kind of notation throughout the whole thesis whenever applicable, i.e., with any function that accepts realizations of multi-dimensional random variables.

With the above definitions, every undirected model for any discrete random variable $\boldsymbol{X}$ with strictly positive density p, can be represented by an exponential family member:

$$\begin{aligned} p(\boldsymbol{x}) &= \frac{1}{Z} \prod_{C \in \overline{\mathcal{C}}(G)} \psi_C(\boldsymbol{x}_C) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{C \in \overline{\mathcal{C}}(G)} \exp(\langle \boldsymbol{\theta}_C, \phi_C(\boldsymbol{x}_C) \rangle) \\ &= \frac{1}{Z(\boldsymbol{\theta})} \exp\left(\sum_{C \in \overline{\mathcal{C}}(G)} \langle \boldsymbol{\theta}_C, \phi_C(\boldsymbol{x}_C) \rangle \right) = \exp(\langle \boldsymbol{\theta}, \phi(\boldsymbol{x}) \rangle - A(\boldsymbol{\theta})) \end{aligned}$$

The reason why the above equality holds for any discrete $\boldsymbol{X}$ is, that any arbitrary, strictly positive function f with domain $\mathcal{X}_C$ can be written in the form $\exp(\langle \boldsymbol{\theta}_C, \phi_C(\boldsymbol{x}_C) \rangle)$ via $\boldsymbol{\theta}_{C=\boldsymbol{x}'} = \log f(\boldsymbol{x}'), \forall \boldsymbol{x}' \in \mathcal{X}_C$ which guarantees that

$$\exp(\langle \boldsymbol{\theta}_C, \phi_C(\boldsymbol{x}') \rangle) = f(\boldsymbol{x}'), \forall \boldsymbol{x}' \in \mathcal{X}_C \; .$$

In contrast to the discrete case, there is no universal sufficient statistic for continuous random variables. It is, however, possible to specify some function space Φ of reasonable expressiveness—like the space of all feedforward artificial neural networks—and choose $\phi \in \Phi$ that minimizes a desired loss function.

The following theorem shows, that exponential family models have a natural connection to the concept of entropy.

Theorem 2.3 (Maximum Entropy Derivation) *Let $\boldsymbol{X}$ be an n-dimensional random variable. Let further $\phi : \mathcal{X} \to \mathbb{R}^d$ be a function on the state space of $\boldsymbol{X}$ with arbitrary but fixed expectation $\boldsymbol{c} \in \mathbb{R}^d$. If a density p has maximum entropy among all densities which satisfy $\mathbb{E}_p[\phi(\boldsymbol{X})] = \boldsymbol{c}$, then p belongs to an exponential family of densities.*

Proof. Let $\mathcal{P} = \{p : \mathcal{X} \to (0; +\infty]\}$ be the set of all functions which map the state space of $\boldsymbol{X}$ to the positive real line. For any element p of $\mathcal{P}$, let $\mathbb{P}$ denote the corresponding probability measure, uniquely defined—up to a ν-null set—by $\mathbb{P}(S) = \int_S p \, \mathrm{d}\,\nu$

for all $S \subseteq \mathcal{X}$. Let $\mathcal{H}_{\boldsymbol{X}}[p]$ be the entropy of $\boldsymbol{X}$ under density p. Now, consider the optimization problem:

$$\max_{p \in \mathcal{P}} \mathcal{H}_{\boldsymbol{X}}[p] \text{ s.t. } \mathbb{E}\left[\phi\left(\boldsymbol{x}\right)\right]_i = \boldsymbol{c}_i, \forall 1 \leq i \leq d \text{ and } \int_{\mathcal{X}} p \,\mathrm{d}\,\nu = 1$$

We construct the Lagrangian $\mathcal{L}\left(p, \boldsymbol{\theta}, C\right) = \mathcal{H}_{\boldsymbol{X}}[p] - \mathcal{E}[p] - \mathcal{Z}[p]$ with $\boldsymbol{\theta} \in \mathbb{R}^d$, $C \in \mathbb{R}$,

$$\mathcal{E}\left[p\right] = \sum_{i=1}^{d} \boldsymbol{\theta}_i \left(\boldsymbol{c}_i - \mathbb{E}\left[\phi\left(\boldsymbol{X}\right)\right]_i\right), \text{ and } \mathcal{Z}\left[p\right] = C\left(\int_{\mathcal{X}} p \,\mathrm{d}\,\nu - 1\right) .$$

The functional derivative of some functional F w.r.t. f, i.e., $(\mathrm{d}\,F/\,\mathrm{d}\,f)$, is defined via $\int_{\mathcal{X}}(\mathrm{d}\,F/\,\mathrm{d}\,f)q \,\mathrm{d}\,\nu = \partial F(f+\epsilon q)/\partial\epsilon \mid_{\epsilon=0}$ for some test function q. We will, however, not go into any details concerning the existence of the functional derivative, nor will we make any attempt to characterize the space of test functions which are allowed. A precise discussion of the topic can be found in [8]. The functional derivative of the entropy is

$$\begin{aligned}
\int_{\mathcal{X}} \frac{\mathrm{d}\,\mathcal{H}}{\mathrm{d}\,p} q \,\mathrm{d}\,\nu &= -\frac{\partial}{\partial\epsilon}\left(\int_{\mathcal{X}} (p+\epsilon q)\log\left(p+\epsilon q\right) \mathrm{d}\,\nu\right) \Big|_{\epsilon=0} \\
&= -\int_{\mathcal{X}} \frac{\partial p + \epsilon q}{\partial\epsilon} \Big|_{\epsilon=0} \log\left(p+\epsilon q\right) + \left(p+\epsilon q\right)\frac{\partial \log\left(p+\epsilon q\right)}{\partial\epsilon} \Big|_{\epsilon=0} \mathrm{d}\,\nu \\
&= \int_{\mathcal{X}} -\left(1+\log p\right) q \,\mathrm{d}\,\nu \ ,
\end{aligned}$$

and thus $(\mathrm{d}\,\mathcal{H}/\,\mathrm{d}\,p) = -(1+\log p)$. Similarly, for the expectation constraint $\mathcal{E}$, we have

$$\begin{aligned}
\int_{\mathcal{X}} \frac{\mathrm{d}\,\mathcal{E}}{\mathrm{d}\,p} q \,\mathrm{d}\,\nu &= -\sum_{i=1}^{d} \boldsymbol{\theta}_i \frac{\partial}{\partial\epsilon}\left(\int_{\mathcal{X}} \phi(\boldsymbol{x})_i (p(\boldsymbol{x}) + \epsilon q(\boldsymbol{x})) \,\mathrm{d}\,\nu(\boldsymbol{x})\right) \Big|_{\epsilon=0} \\
&= \int_{\mathcal{X}} -\langle\boldsymbol{\theta}, \phi(\boldsymbol{x})\rangle q(\boldsymbol{x}) \,\mathrm{d}\,\nu(\boldsymbol{x}) \ .
\end{aligned}$$

And the derivative is hence $(\mathrm{d}\,\mathcal{E}/\,\mathrm{d}\,p) = -\langle\boldsymbol{\theta}, \phi(\boldsymbol{x})\rangle$. Finally, for the normalization constraint, we find

$$\int_{\mathcal{X}} \frac{\mathrm{d}\,\mathcal{Z}}{\mathrm{d}\,p} q \,\mathrm{d}\,\nu = C\frac{\partial}{\partial\epsilon}\left(\int_{\mathcal{X}} (p+\epsilon q) \,\mathrm{d}\,\nu - 1\right) \Big|_{\epsilon=0} = \int_{\mathcal{X}} Cq \,\mathrm{d}\,\nu \ ,$$

and at any stationary point, we must have $(\mathrm{d}\,\mathcal{L}/\mathrm{d}\,p) = 0$, i.e.,

$$\begin{aligned}
-1 - \log p\left(\boldsymbol{x}\right) + \langle\boldsymbol{\theta}, \phi\left(\boldsymbol{x}\right)\rangle - C &= 0 \\
\Leftrightarrow p\left(\boldsymbol{x}\right) &= \exp\left(\langle\boldsymbol{\theta}, \phi\left(\boldsymbol{x}\right)\rangle - (1+C)\right) \ .
\end{aligned}$$

Plugging this back into the normalization constraint shows, that C is a function of $\boldsymbol{\theta}$:

$$\int_{\mathcal{X}} p \,\mathrm{d}\,\nu = 1 \Leftrightarrow \int_{\mathcal{X}} \exp\langle\boldsymbol{\theta}, \phi\left(\boldsymbol{x}\right)\rangle = \exp\left(1+C\right) \Leftrightarrow \log \sum_{\boldsymbol{x}\in\mathcal{X}} \exp\left(\langle\boldsymbol{\theta}, \phi\left(\boldsymbol{x}\right)\rangle\right) = 1 + C$$

It is common to make this functional dependence explicit by defining $A(\boldsymbol{\theta}) = 1 + C$, which yields the exponential family form (2.3)

$$p(\boldsymbol{x}) = \exp\left(\langle \boldsymbol{\theta}, \phi(\boldsymbol{x})\rangle - A(\boldsymbol{\theta})\right) \ .$$

■

Having maximum entropy is a convenient property but does not ultimately qualify exponential families as an appropriate model for resource-constrained systems. We will revisit exponential families and their sufficient statistics with a closer look on memory complexity in Section 3.1.

2.2.2 Probabilistic Inference

Inference is an ambiguous term—in an abstract sense, it refers to the derivation of knowledge from a probability measure. More specifically, it denotes the computation of one or more of the following quantities (or extensions thereof):

- Marginal probabilities $p(\boldsymbol{X}_i = \boldsymbol{x}_i)$ or conditional marginal probabilities $p(\boldsymbol{X}_i = \boldsymbol{x}_i \mid \boldsymbol{X}_j = \boldsymbol{x}_j)$ (cf. Definition 7.4),
- Partition function $Z(\boldsymbol{\theta}) = \int \psi(\boldsymbol{x}) \,\mathrm{d}\, \nu(\boldsymbol{x})$ (cf. Theorem 2.2),
- Maximum a posteriori (MAP) state, i.e., $\max_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x})$,
- Maximum likelihood parameter, i.e., $\max_{\boldsymbol{\theta}\in\mathbb{R}^d} \prod_{\boldsymbol{x}\in\mathcal{D}} p_{\boldsymbol{\theta}}(\boldsymbol{x})$.

We will now explain these tasks, corresponding algorithmic techniques, and resource requirements. In what follows, $\boldsymbol{X}$ is an n-dimensional random variable with joint state space $\mathcal{X} = \bigotimes_{i=1}^{n} \mathcal{X}_i$, joint density p, and base measure ν.

Partition Function and Marginal Probabilities

As seen in Theorem 2.2, the partition function is the quantity that ensures normalization of a probability density. Knowledge of the partition function is required in order to compute actual probabilities.

Lemma 2.1 (Normalization) *Let $\psi : \mathcal{X} \to \mathbb{R}_+$ be an arbitrary non-negative function. ψ can be converted into a probability density p_ψ over $\mathcal{X}$ via normalization*

$$p(\boldsymbol{x}) = \frac{1}{\int_{\mathcal{X}} \psi \,\mathrm{d}\,\nu} \psi(\boldsymbol{x}) = \frac{1}{Z}\psi(\boldsymbol{x}) \ .$$

The quantity

$$Z = \int_{\mathcal{X}} \psi \,\mathrm{d}\,\nu$$

is called partition function of p.

When the integration is carried out over a subset $U \subset \{1, 2, \dots, n\}$ of variables with the others being fixed to a state $\boldsymbol{x}_{\bar{U}}$, the normalized result is the marginal density of $\boldsymbol{x}_{\bar{U}}$:

$$p_{\bar{U}}(\boldsymbol{x}_{\bar{U}}) = \frac{1}{Z} \int_{\mathcal{X}_U} \psi(\boldsymbol{x}_U, \boldsymbol{x}_{\bar{U}}) \, \mathrm{d}\, \nu_U(\boldsymbol{x}_U) \ .$$

This shows why the computation of marginal probabilities and the partition function are closely related—any algorithm for Z can be transformed into an algorithm for the marginal density $p_{\bar{U}}$ by fixing the values of U during integration.

It depends on the actual ψ whether the computation of Z is easy or hard. In case of continuous random variables, ψ is often chosen such that the integration is possible in closed-form. Famous examples of such distributions are the Gaussian (a.k.a. Normal), Poisson, Laplace, Beta, Exponential, Weibull, Von Mises-Fisher, Levy, Cauchy, Gumbel, and Dirichlet distributions. One could even say that one important reason why those probability distributions are so well-known and frequently used, is that efficient algorithms for the computation of their normalized density exist.

For discrete random variables, such efficient algorithms cannot be found in general. Without additional assumptions, computing the partition function of a discrete random variable is a $\#\mathbf{P}$-complete[2] problem—it requires the same amount of time as counting the number of accepting computation paths of a non-deterministic polynomial time Turing machine [212, 33].

Special cases have been discovered for which efficient algorithms exist [191]. If the induced graph G contains no loops, i.e., is tree-structured, Z can be evaluated exactly in polynomial time via the belief propagation algorithm [165]—whose runtime is polynomial in the number of edges and polynomial in the state space sizes. The same algorithmic technique, e.g., message passing along the conditional independence structure, underlies a frequently applied heuristic inference algorithm, namely loopy belief propagation. Although LBP is often termed an approximate inference algorithm, it is no approximation algorithm in the classical sense, because it delivers no performance guarantees.

Belief Propagation The belief propagation (BP) algorithm, introduced by Pearl [165], is based on two basic ingredients. First, the factorization of undirected models as defined in Theorem 2.2. The BP algorithm is originally formulated for tree-structured graphs $G = (V, E)$, where the maximum cliques are simply the edges E. Consequently, the density factorizes over the edges[3]

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{\{v,u\} \in E} \psi_{vu}(\boldsymbol{x}_{vu})$$

and the partition function is thus

$$Z = \sum_{\boldsymbol{x} \in \mathcal{X}} \prod_{\{v,u\} \in E} \psi_{vu}(\boldsymbol{x}_{vu}) \ . \tag{2.4}$$

[2]The definitions of fundamental complexity classes can be found in [229].

[3]It is common in the literature to write the factorization of pairwise models in terms of vertices and edges. But since potentials of non-maximal cliques (here: vertices) may be absorbed into the potentials of their supercliques (here: edges), an explicit inclusion of vertex factors is not necessary.

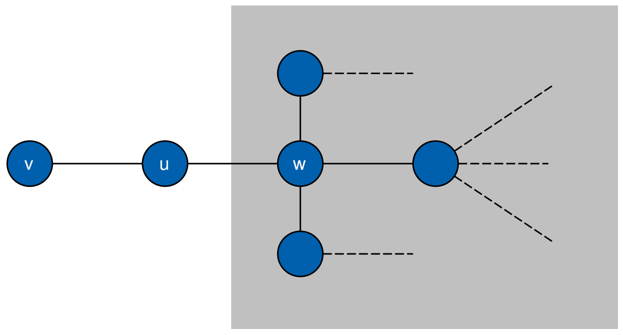

Figure 2.3: An undirected graph $G = (V, E)$ with vertex set $V = \{u, v, w, \dots\}$. The neighborhood of u is $\mathcal{N}(u) = \{v, w\}$. Dashed lines indicate edges to other vertices. The grey area indicates the subtree, rooted at vertex w.

The summation in (2.4) may be rewritten in form of explicit summations over each variable. Each factor depends on two variables, and we may employ the second basic ingredient, the distributive law, to pull the factors inside the summation. W.l.o.g., let v be a leaf vertex, u its sole neighbor, and $\mathcal{N}(u)$ the set of neighbors of u. This situation is depicted in Fig. 2.3. We set

$$\Psi_{w \not\to u}(\boldsymbol{x}_u) = \sum_{\boldsymbol{x}' \in \mathcal{X}_{V \setminus \{v,u\}}} \prod_{\{z,h\} \in E \setminus \{\{v,u\}\}} \psi_{zh}(\boldsymbol{x}'_{zh}) \tag{2.5}$$

to be the summation over the states of all vertices in the subtree rooted at w that originates when we cut the edge between w and u (the grey area in Fig. 2.3). Z may then be rewritten in terms of $\Psi_{w \not\to u}(\boldsymbol{x}_u)$:

$$Z = \sum_{\boldsymbol{x}_v \in \mathcal{X}_v} \prod_{u \in \mathcal{N}(v)} \sum_{\boldsymbol{x}_u \in \mathcal{X}_u} \psi_{vu}(\boldsymbol{x}_v, \boldsymbol{x}_u) \prod_{w \in \mathcal{N}(u) \setminus \{v\}} \Psi_{w \not\to u}(\boldsymbol{x}_u) \; . \tag{2.6}$$

Note that (2.5) has the same sum-product form as the full partition function (2.4), and we may proceed recursively like before. I.e., we cut the edges between w and any of its neighbors $z \neq u$ and apply (2.5). This can be repeated until all remaining subtrees are singletons. Since G is a tree, the recursion will terminate after a finite number of steps. The terms $\Psi_{w \not\to u}(\boldsymbol{x}_u)$ are usually interpreted as message from w to u, such that the final BP update becomes

$$\boldsymbol{m}_{u \to v}(\boldsymbol{x}_v) = \sum_{\boldsymbol{x}_u \in \mathcal{X}_u} \psi_{vu}(\boldsymbol{x}_v, \boldsymbol{x}_u) \prod_{w \in \mathcal{N}(u) \setminus \{v\}} \boldsymbol{m}_{w \to u}(\boldsymbol{x}_u) \; . \tag{2.7}$$

With this notation, we have $Z = \sum_{\boldsymbol{x}_v \in \mathcal{X}_v} \prod_{u \in \mathcal{N}(v)} \boldsymbol{m}_{u \to v}(\boldsymbol{x}_v)$ at any vertex v. The BP algorithm thus consists of computing the messages in reverse depth first search (DFS) order, where the DFS root may be any vertex. Computing Z with BP has therefore a worst-case time complexity of $\mathcal{O}(|E||\mathcal{X}_{\max}|^2|\mathcal{N}_{\max}|)$ and requires $\mathcal{O}(|E||\mathcal{X}_{\max}|)$ space, where $\mathcal{X}_{\max}$ is the largest vertex state space and $\mathcal{N}_{\max}$ is the largest neighborhood.

Algorithm 2.1: Loopy Belief Propagation

```
input Graph $G = (V, E)$, parameter $\boldsymbol{\theta} \in \mathbb{R}^d$, precision $\varepsilon > 0$, iterations $I \in \mathbb{N}$
output Vector $\hat{\boldsymbol{\mu}} \in [0;1]^d$ of all singleton and pairwise marginals
 1: $\boldsymbol{m}^{\text{new}} \leftarrow \mathbf{0}$ ; $\boldsymbol{m}^{\text{old}} \leftarrow \boldsymbol{\infty}$ ; $\epsilon \leftarrow \infty$ ; $i \leftarrow 1$
 2: repeat
 3:   $\boldsymbol{m}^{\text{old}} \leftarrow \boldsymbol{m}^{\text{new}}$
 4:   for $\{v, u\} \in E$ do
 5:     for $x \in \mathcal{X}_v$ do
 6:       update message $\boldsymbol{m}^{\text{new}}_{u\to v}(x)$ according to (2.9)
 7:     end for
 8:     for $x \in \mathcal{X}_u$ do
 9:       update message $\boldsymbol{m}^{\text{new}}_{v\to u}(x)$ according to (2.9)
10:     end for
11:   end for
12:   $\epsilon \leftarrow \|\boldsymbol{m}^{\text{new}} - \boldsymbol{m}^{\text{old}}\|_\infty$
13:   $i \leftarrow i + 1$
14: until $\epsilon < \varepsilon$ or $i > I$
```

The same procedure can be applied to compute the marginal density of some fixed event $\boldsymbol{X}_U = \boldsymbol{x}_U$, $U \subseteq V$, by simply omitting the summation over the states of vertices in U. If the set U contains a single vertex v, the corresponding marginal density becomes

$$p_v(\boldsymbol{x}_v) = \frac{1}{Z} \prod_{u \in \mathcal{N}(v)} \boldsymbol{m}_{u\to v}(\boldsymbol{x}_v) = \frac{\prod_{u\in\mathcal{N}(v)} \boldsymbol{m}_{u\to v}(\boldsymbol{x}_v)}{\sum_{x'\in\mathcal{X}_v} \prod_{u\in\mathcal{N}(v)} \boldsymbol{m}_{u\to v}(x')} \,, \tag{2.8}$$

which follows directly from plugging (2.7) into the definition of the marginal density (Definition 7.4) of an undirected model. It is also worth mentioning that the messages may be scaled by any arbitrary constant α without altering the resulting marginals, due to cancellation, e.g., $\prod_{u\in\mathcal{N}(v)} \alpha\boldsymbol{m}_{u\to v}(\boldsymbol{x}_v)/\sum_{x'\in\mathcal{X}_v}\prod_{u\in\mathcal{N}(v)} \alpha\boldsymbol{m}_{u\to v}(x') = p_v(\boldsymbol{x}_v)$. This fact allows the normalization of messages.

It is important that the whole derivation of the BP messages cannot be correct if G is not a tree, because every non-tree has a subgraph that does not contain leaf vertices. We therefore cannot simplify (2.4) by the distributive law. However, it is common to run BP in loopy graphs and compute the normalized messages in some arbitrary order. The corresponding algorithm is called *loopy belief propagation* [130], shown in Algorithm 2.1. In this setting, one has to recompute messages multiple times, because incoming messages might change—depending on previous message computations. Hence, we have to store two separate message vectors, $\boldsymbol{m}^{\text{new}}$ and $\boldsymbol{m}^{\text{old}}$, which are swapped after each LBP iteration.

$$\boldsymbol{m}^{\text{new}}_{u\to v}(\boldsymbol{x}_v) = \sum_{\boldsymbol{x}_u\in\mathcal{X}_u} \psi_{vu}(\boldsymbol{x}_v, \boldsymbol{x}_u) \prod_{w\in\mathcal{N}(u)\setminus\{v\}} \boldsymbol{m}^{\text{old}}_{w\to u}(\boldsymbol{x}_u) \,, \tag{2.9}$$

The procedure of message-re-computation may or may not converge. If it converges, its fixpoints are local minima of the so-called Bethe free energy [245, 92]. Despite its heuristic nature, LBP and its generalizations [130, 246] are well known for delivering good results in practice.

When marginals and partition function are computed from LBP fixpoints, they can in principle be arbitrarily far from the correct values. However, properties of the potential function ψ like bounded dynamic range or log-supermodularity can be used to derive convergence guarantees or lower bounds on the true partition function [102, 183, 184].

BP is attractive for probabilistic inference in resource-constrained systems due to its rather low resource requirements. Even in loopy graphs, one may prefer to run LBP for a few iterations, instead of more sophisticated methods (described below), because of its low overhead and good empirical performance. We present a specialized version of (L)BP, based on integer arithmetic, in Chapter 4. Therein, only the bit-length of a message is computed, instead of the actual message.

Junction Tree Algorithm Evaluating the partition function of loopy models exactly does not necessarily require a naive summation over the state space; there is another, more efficient, technique. Any loopy graph can be converted into a tree, the so-called *junction tree* (JT) [134, 227, 122]. As with BP in ordinary trees, inference on the junction tree has a time complexity that is polynomial in the maximal state space size of its vertices. The maximal vertex state space size of a junction tree is, however, exponential in the size of the largest clique of a triangulation[4] of G, a.k.a. exponential in the treewidth of G. Hence, if the treewidth of a loopy model is small, exact inference via the junction tree algorithm is rather efficient. Choosing a triangulation that results in a minimal treewidth is a **NP**-hard problem, but a valid triangulation can be found with time and memory complexity linear in the number of vertices [50, 19, 91]. Moreover, the maximal cliques of triangulated graphs can be computed in polynomial time w.r.t. to $|V|$ and $|\mathcal{N}_{\max}|$. We provide the pseudocode of the junction tree construction in Algorithm 2.2.

Since the junction tree is a tree, the derivation of BP applies and the same reasoning as in (2.4) $\rightarrow$ (2.5) $\rightarrow$ (2.6) $\rightarrow$ (2.7) may be used to derive the corresponding message update rules. Note, however, that each vertex of the junction tree $J(G)$ corresponds to a vertex subset of the original input graph G. The state spaces of neighboring vertices will overlap by construction, and the final messages look hence slightly different from the ordinary BP messages (2.7). For junction tree vertices $U, C, W \subset V$, the Shafer-Shenoy messages [192] are

$$\boldsymbol{m}_{U\to C}(\boldsymbol{x}_{C\setminus U}) = \sum_{\boldsymbol{x}_{U\setminus C}\in\mathcal{X}_{U\setminus C}} \psi_U(\boldsymbol{x}_U) \prod_{W\in\mathcal{N}(U)\setminus\{C\}} \boldsymbol{m}_{W\to U}(\boldsymbol{x}_{U\setminus W}) \,.$$

The clique potentials ψ_U are constructed from the potentials of G, according to $\psi_U = \prod_i \psi_{C_i}$ for all C_i which are contained in the junction tree vertex U. If any C_i is contained in more than one junction tree vertex, it is multiplied only to exactly one of the junction

[4]A triangulation of a graph $G = (V, E)$ is another graph $G' = (V, E')$ with $E \subseteq E'$, such that any induced cycle of G' has exactly three vertices.

Algorithm 2.2: Junction Tree Construction

```
input Graph $G = (V, E)$
output Junction tree $J(G)$
1: $G_\Delta \leftarrow$ triangulate$(G)$
2: $\mathcal{C}(G_\Delta) \leftarrow$ findcliques$(G_\Delta)$
3: $n \leftarrow |\mathcal{C}(G_\Delta)|$
4: Graph $H \leftarrow K_n$ // fully connected graph with $n$ vertices
5: for $C_i \in \mathcal{C}(G_\Delta)$ do
6:     for $C_j \in \mathcal{C}(G_\Delta)$ do
7:         $\boldsymbol{w}_{i,j} \leftarrow |C_i \cap C_j|$
8:     end for
9: end for
10: $J(G) \leftarrow$ MWST$(H,\boldsymbol{w})$ // maximum weight spanning tree of $H$ with weights $\boldsymbol{w}$
```

tree potentials. Other versions of junction tree messages exist [227], but the complexity of all variants depends on the quality of the triangulation.

The actual computational complexity of inference on the junction tree can be computed directly from the vertex state spaces and the neighborhood sizes. On a resource-constrained system, we can compute the junction tree, based on sub-optimal triangulation, and compute its resource requirements. If they exceed the capabilities of the system, we may resort to LBP or a simpler conditional independence structure. LBP results on a real-world microcontroller unit are provided in Chapter 6.

Variational Methods It should be clear from the previous paragraphs, that the conditional independence structure is responsible for both, an accurate representation of the probability measure, and the computational complexity. How can the structure be modified such that the complexity is reduced while not degrading the quality too much? One way to formalize this compromise is variational inference. We shall provide the basic underlying ideas of this field and relate it to resource-constrained systems. A more detailed introduction to the topic can be found in [227]. Here, the main starting point is Jensen's inequality [106].

Definition 2.8 (Convexity) *Let $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^d$. A differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if and only if*

$$f(\boldsymbol{a}) \geq f(\boldsymbol{b}) + \langle \nabla f(\boldsymbol{b}), \boldsymbol{a} - \boldsymbol{b} \rangle \ .$$

Moreover, f is convex if and only if its gradient is monotonically non-decreasing, i.e., $\langle \nabla f(\boldsymbol{a}) - \nabla f(\boldsymbol{b}), \boldsymbol{a} - \boldsymbol{b} \rangle \geq 0$. The negative of a convex function is a concave function and vice-versa.

Here, $\nabla f(\boldsymbol{a})$ is the gradient of f, i.e., the column vector of all first partial derivatives of f at $\boldsymbol{a}$.

$$\nabla f(\boldsymbol{a}) = \left(\frac{\partial f(\boldsymbol{a})}{\partial \boldsymbol{a}^\top} \right)^\top$$

Lemma 2.2 (Jensen's Inequality [106]**)** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function and $\boldsymbol{X}$ a d-dimensional random variable with density p. Then*

$$f(\mathbb{E}[\boldsymbol{X}]) \leq \mathbb{E}[f(\boldsymbol{X})] \,.$$

Now, consider a probability measure $\mathbb{F}$ on the same σ-algebra as $\mathbb{P}$, dominated by ν, and with density q. The logarithm of the partition function can then be lower-bounded by applying Jensen's inequality:

$$\log Z = \log \int_{\mathcal{X}} \psi \frac{q}{q} \,\mathrm{d}\,\nu \geq \int_{\mathcal{X}} q \log \frac{\psi}{q} \,\mathrm{d}\,\nu = \mathbb{E}_{\mathbb{F}}[\log \psi(\boldsymbol{X})] + \mathcal{H}[\mathbb{F}] \,,$$

where $\mathbb{E}_{\mathbb{F}}$ denotes the expectation w.r.t. to the probability measure $\mathbb{F}$. Note that the choice of $\mathbb{F}$ was arbitrary and that equality is attained for $\mathbb{F} = \mathbb{P}$. Denoting the space of all non-deterministic probability measures by $\mathcal{P}$, we arrive at the fundamental variational principle

$$\log Z = \max_{\mathbb{F} \in \mathcal{P}} \mathbb{E}_{\mathbb{F}}[\log \psi(\boldsymbol{X})] + \mathcal{H}[\mathbb{F}] \,. \tag{2.10}$$

The variational principle may equivalently be derived via conjugate duality [181], but in the context of this thesis, the former derivation suffices.

When the density of $\mathbb{P}$ belongs to an exponential family, plugging the potential function $\psi(\boldsymbol{x}) = \exp(\langle \boldsymbol{\theta}, \phi(\boldsymbol{x}) \rangle)$ into (2.10) simplifies the variational principle to

$$\log Z(\boldsymbol{\theta}) = \max_{\mathbb{F} \in \mathcal{P}} \langle \boldsymbol{\theta}, \boldsymbol{\mu}' \rangle + \mathcal{H}[\mathbb{F}] \,. \tag{2.11}$$

Therein, $\boldsymbol{\mu}' = \mathbb{E}_{\mathbb{F}}[\phi(\boldsymbol{X})]$ is the expected sufficient statistic w.r.t. $\mathbb{F}$. When $\boldsymbol{X}$ is discrete and ϕ is a binary, $\boldsymbol{\mu}'$ contains the marginal probabilities of the corresponding clique states, e.g., $\boldsymbol{\mu}'_{C=\boldsymbol{y}} = \mathbb{P}(\boldsymbol{X}_C = \boldsymbol{y})$. In this case, the set of all realizable $\boldsymbol{\mu}'$ vectors for the dependence structure G is the so-called marginal polytope

$$\mathcal{M}(G) = \{ \boldsymbol{\mu}' \in \mathbb{R}^{d(G)} \mid \exists \mathbb{F} \in \mathcal{P} : \mathbb{E}_{\mathbb{F}}[\phi_G(\boldsymbol{X})] = \boldsymbol{\mu}' \} \,, \tag{2.12}$$

where we excluded boundary marginals, and the connection between the model dimension d and the structure G was made explicit. We drop the dependence of $\mathcal{M}$ on G whenever the graph is clear from the context. When we define $\mathcal{H}(\boldsymbol{\mu}')$ as the entropy of the probability measure with marginals $\boldsymbol{\mu}'$, (2.11) can be reformulated solely in terms of $\boldsymbol{\mu}'$

$$\log Z(\boldsymbol{\theta}) = \sup_{\boldsymbol{\mu}' \in \mathcal{M}} \langle \boldsymbol{\theta}, \boldsymbol{\mu}' \rangle + \mathcal{H}(\boldsymbol{\mu}') \,. \tag{2.13}$$

This makes the connection between the marginals and the partition function evident. It can be shown that the negative entropy corresponds to the dual function $A^*(\boldsymbol{\mu}')$ of $A(\boldsymbol{\theta}) = \log Z(\boldsymbol{\theta})$ for all $\boldsymbol{\mu}'$ that lie in the interior of $\mathcal{M}$ [227].

Several variational approximations can be derived from (2.13) by relaxing the set $\mathcal{M}$ to classes $\tilde{\mathcal{M}}$ of tractable marginals. Inner approximations of $\mathcal{M}$ lead to lower bounds on $\log Z$ from which we may choose a member that maximizes the lower bound, and hence, being as close as possible to $\log Z$. Maybe the most prominent example is the naive mean field (MF) approximation; the class of joint densities that factorize fully over their singleton marginals, i.e., $p(\boldsymbol{x}) = \prod_{v\in V} p_v(\boldsymbol{x}_v)$. Moreover, structured mean field and other approaches are known which are less restrictive but computationally harder. Most of these approximations are based on non-convex variational problems [227] and suffer from multiple local optima. Given that the underlying exact variational principle (2.13) is convex, it is natural to consider variational approximations that retain this convexity. If the set $\mathcal{M}$ is replaced by a convex outer bound, and in addition, $\mathcal{H}$ is replaced by an upper bound, the approximate variational principle provides an upper bound on the log partition function [224, 226]. Recall that the sufficient statistic (2.6) is actually a function of the conditional independence structure G, and so is $\boldsymbol{\theta}$, since $\boldsymbol{\theta}$ contains a weight for each indicator in ϕ. For any subgraph H of G, let $\boldsymbol{\theta}|_H$ be the projection of $\boldsymbol{\theta}$ to H, i.e., all components of $\boldsymbol{\theta}$ which correspond to cliques C which are not present in H, are dropped—the same kind of projection may be applied to $\boldsymbol{\mu}'$. Due to duality between $-\mathcal{H}$ and $\log Z(\boldsymbol{\theta})$ for realizable $\boldsymbol{\mu}'$, we have

$$\begin{aligned}-\mathcal{H}(\boldsymbol{\mu}'|_H) &= \sup_{\boldsymbol{\theta}|_H\in\mathbb{R}^{d(H)}} \langle \boldsymbol{\theta}|_H, \boldsymbol{\mu}'|_H\rangle - \log Z(\boldsymbol{\theta}|_H)\\ &= \sup_{\boldsymbol{\theta}\in\mathbb{R}^d} \langle \boldsymbol{\theta}, \boldsymbol{\mu}'\rangle - \log Z(\boldsymbol{\theta}) \leq -\mathcal{H}(\boldsymbol{\mu}')\\ &\quad\ \text{s.t. } \boldsymbol{\theta}_C = \mathbf{0}, \forall C \notin H\ .\end{aligned}$$

We see that the entropy of a submodel with structure H is always larger than the entropy of the full model with structure G. In addition, for any set of subgraphs $\mathcal{G}(G)$ of G, let

$$\mathcal{M}(\mathcal{G}(G)) = \{\boldsymbol{\mu}' \in \mathbb{R}^{d(G)} \mid \exists \mathbb{F} \in \mathcal{P} : \forall H \in \mathcal{G}(G) : \mathbb{E}_{\mathbb{F}}[\phi_H(\boldsymbol{X})] = \boldsymbol{\mu}'|_H\}\ .$$

Each member of $\boldsymbol{\mu}' \in \mathcal{M}(\mathcal{G}(G))$ has the property that the expected sufficient statistics of all $H \in \mathcal{G}(G)$ are identical for the cliques they have in common. They may, however, contain different cliques which are then allowed to have arbitrary, realizable marginals. If ρ is an arbitrary probability mass of the discrete random variable $\boldsymbol{G}$ with state space $\mathcal{G}(G)$, we have $\mathcal{H}(\boldsymbol{\mu}') \leq \mathbb{E}_\rho[\mathcal{H}(\boldsymbol{\mu}'|_{\boldsymbol{G}})]$. Finally, the convex approximate variational principle is

$$\mathcal{B}_{\mathcal{G}(G)}(\boldsymbol{\theta}, \rho) = \sup_{\boldsymbol{\mu}'\in\mathcal{M}(\mathcal{G}(G))} \langle \boldsymbol{\theta}, \boldsymbol{\mu}'\rangle + \mathbb{E}_\rho[\mathcal{H}(\boldsymbol{\mu}'|_{\boldsymbol{G}})]\ .$$

$\mathcal{B}(\boldsymbol{\theta}, \rho)$ can be used in place of $A(\boldsymbol{\theta})$ whenever the latter cannot be evaluated due to resource constraints [221]. Especially convenient approximations $\mathcal{B}_{\mathcal{G}(G)}(\boldsymbol{\theta}, \rho)$ arise, when $\mathcal{G}(G) = \mathcal{T}(G)$ is a set of a spanning trees of G [224, 226] or even a single tree [244]. This particular convex approximate variational principle is known as tree-reweighting (TRW). Moreover, a special BP version, namely tree-reweighted belief propagation [225] can be derived, which solves the corresponding approximate variational problem via message passing.

Sampling Methods As seen in the previous paragraphs, the main source of complexity in marginalization and normalization of discrete random variables is the enumeration of an exponentially large set. Belief propagation, the junction tree algorithm, and general variational methods address this task by exploiting or simplifying the structure of the set that has to be enumerated. But in any case, all variable assignments are eventually fully enumerated. Instead of such exhaustive combinatorial calculations, one may draw N samples $\boldsymbol{x}^i$ from the state space $\mathcal{X}$ at random, compute the solution on the samples, and extrapolate the obtained result onto the full set. The process of gathering information from a set of samples to mimic a specific quantity is called estimation.

Definition 2.9 (Estimator) *Let $\boldsymbol{X}$ be a random variable with measure $\mathbb{P}$ and $\boldsymbol{Y} = \boldsymbol{Y}(\boldsymbol{X})$ a function of $\boldsymbol{X}$. Both random variables are not independent of each other, i.e., $\boldsymbol{X} \not\perp\!\!\!\perp \boldsymbol{Y}$. Let further $\mathcal{D}$ be a multi set that contains random realizations of $\boldsymbol{X}$. Any function $\hat{\boldsymbol{Y}} = \hat{\boldsymbol{Y}}(\mathcal{D})$ that is used to recover $\boldsymbol{Y}$ from $\mathcal{D}$ is called estimator for $\boldsymbol{Y}$.*

- *An estimator is called unbiased, if we can expect that it yields the correct value, i.e., $\mathbb{E}[\hat{\boldsymbol{Y}}] = \boldsymbol{Y}$.*
- *Here, an estimator is called (weakly) consistent, if the probability that the estimation error $|\hat{\boldsymbol{Y}} - \boldsymbol{Y}|$ is arbitrary small, converges to 1 in the limit of infinite samples, i.e.,*
$$\forall \epsilon > 0 : \lim_{|\mathcal{D}| \to \infty} \mathbb{P}(|\hat{\boldsymbol{Y}} - \boldsymbol{Y}| < \epsilon) = 1 \; .$$

Some examples of sampling based estimators are:

1. Estimating the expectation for arbitrary functions, by evaluating f on each sample $\boldsymbol{x} \in \mathcal{D}$ and computing the average function value $\hat{\mathbb{E}}[f(\boldsymbol{X})] = \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x} \in \mathcal{D}}^{N} f(\boldsymbol{x})$.
2. Estimating marginal probabilities $\hat{p}_U(\boldsymbol{x}_U) = \hat{\mathbb{E}}[\mathbb{1}_{\{\boldsymbol{X}_U = \boldsymbol{x}_U\}}]$ for $U \subseteq V$, by observing and counting the number of cases in which the specific variable assignment $\boldsymbol{x}_U$ appears.
3. Estimating the partition function, by summing the potentials $\psi(\boldsymbol{x})$ of uniform samples and finally rescaling the average potential by the size of the state space, i.e., $\hat{Z} = |\mathcal{X}| \hat{\mathbb{E}}[\psi(\boldsymbol{X})]$.
4. Estimating the most likely joint state by choosing the sample with the largest estimated density $\hat{\boldsymbol{x}}^* = \max_{\boldsymbol{x} \in \mathcal{D}} \hat{p}(\boldsymbol{x})$.

Underneath the here explained techniques lies the Monte Carlo principle. In each of the above cases, the estimator is reduced to a sample mean estimation, i.e., the estimation of an expected value, which is unbiased and weakly consistent [6]. It is indeed very important that we can generate proper samples that follow a probability measure of our choice.

The computational resource consumption of sampling stems from the number of samples N, and the actual algorithm used to generate the samples. The most basic techniques are rejection sampling and inversion sampling. Rejection sampling draws samples from a given, easy-to-sample proposal distribution $\mathbb{F}$ and converts them to samples from a not necessary normalized potential function ψ, by rejecting samples from $\mathbb{F}$ which do

Algorithm 2.3: Rejection Sampling

```
input  Proposal measure $\mathbb{F}$ with density $q$, potential $\psi$, number of samples $N$,
       upper bound $M$
output  Set of samples $\mathcal{D}$
1: $\mathcal{D} \leftarrow \emptyset$ ; $i \leftarrow 1$
2: while $i \leq N$ do
3:    Draw sample $\boldsymbol{x}^i \sim \mathbb{F}$
4:    Draw $u \sim \mathbb{U}_{(0;1]}$ // uniform random distribution on the interval $(0;1]$
5:    if $u < \psi(\boldsymbol{x})/(Mq(\boldsymbol{x}))$ then
6:       $\mathcal{D} \leftarrow \mathcal{D} \cup \{\boldsymbol{x}^i\}$ // accept sample $\boldsymbol{x}^i$
7:       $i \leftarrow i + 1$
8:    end if
9: end while
```

not fit to our target distribution (Algorithm 2.3). The algorithm is easy to implement but requires a constant bound M on the quotient $\psi(\boldsymbol{x})/q(\boldsymbol{x})$ which might not always be easy to obtain. Moreover, the runtime can be large when many samples are rejected. Whenever a probability measure is sufficiently simple, we may completely omit the rejection step via inversion sampling. The procedure consists of drawing a uniform random number $u \in (0;1)$, and inverting the distribution function $\mathbb{P}(\boldsymbol{X} \leq \boldsymbol{x})$ at u. I.e., when U is uniform in $(0;1)$, then $\mathbb{P}^{-1}(U)$ is a sample from $\mathbb{P}$. In general, the inverse of $\mathbb{P}$ might not be analytically available. Nevertheless, if the random variable $\boldsymbol{X}$ is discrete with $|\mathcal{X}|$ distinct states, the first $\boldsymbol{x}^l$ that satisfies $\sum_{i=1}^{l} \mathbb{P}(\boldsymbol{X} = \boldsymbol{x}^i) > U$ is a sample from $\mathbb{P}$. Here, "first" is meant w.r.t. some arbitrary but fixed ordering of states. Of course, when the state space $\mathcal{X}$ is large, say exponential in n, this technique has an unreasonable large runtime. One may encounter situations in which some structure of the underlying measure can be exploited to leverage this issue. This technique will be revisited in Chapter 5, when we present a new inference method that combines numerical integration with fast inversion sampling.

If numerical data, sampled from $\mathbb{P}$ (the proposal), is already available, it can be converted to samples from another measure $\mathbb{F}$ via importance sampling. To this end, samples $\boldsymbol{x}$ are weighted by their importance weight $f(x)/p(x)$. Taking the empirical expectation of the weighted samples asymptotically cancels out the true density of the samples. This technique can be applied to estimate the partition function. When $\mathcal{D}$ is a set of samples from $\mathbb{P}$ and ψ is the potential function, the importance sampling estimate of the partition function is $\hat{Z} = (1/|\mathcal{D}|) \sum_{\boldsymbol{x} \in \mathcal{D}} (\psi(\boldsymbol{x})/p(\boldsymbol{x}))$.Note that $p(\boldsymbol{x}) = 1/|\mathcal{X}|$ yields the simple estimator, mentioned in the examples above. The estimator is unbiased but its variance can be infinite whenever the divergence between p and f is large. A discussion of importance sampling and its variants can be found in [151, 139] and references therein.

As opposed to plain Monte Carlo methods, Markov Chain Monte Carlo (MCMC) meth-

Algorithm 2.4: Gibbs Sampler

```
input Set of variables $V$, conditional marginals $p_v(\boldsymbol{x}_v) \mid \boldsymbol{x}_{\mathcal{N}(v)}, \forall v, \boldsymbol{x}$,
    joint state $\boldsymbol{x}^0$, number of resampling steps $R$
output Sample $\boldsymbol{x}$
1: $\boldsymbol{x} \leftarrow \boldsymbol{x}^0$
2: for $i = 1$ to $R$ do
3:     for $v \in V$ do
4:         Sample $\boldsymbol{x}_v$ according to $p(\boldsymbol{x}_v \mid \boldsymbol{x}_{\mathcal{N}(v)})$ // computed via (2.14)
5:     end for
6: end for
```

ods generate a sequence of samples in which each sample depends on the previous one—the samples form a chain. After B steps, the elements of the chain are valid but dependent samples from our target distribution. To overcome the dependence, a fixed number of samples may be discarded before the next sample is stored. Maybe the most popular MCMC method is the Metropolis-Hastings algorithm [146, 87], which forms the basis of most practical MCMC algorithms. An overview can be found in [6]. Here, we will shortly present the Gibbs sampler [74].

For Gibbs sampling, recall that undirected models obey the (local) Markov property, i.e., $p_v(\boldsymbol{x}_v \mid \boldsymbol{x}_{V\setminus\{v\}}) = p_v(\boldsymbol{x}_v \mid \boldsymbol{x}_{\mathcal{N}(v)})$. Moreover, let $\mathcal{C}(v)$ be the set of cliques which contain vertex v, the conditional marginal is simply

$$p(\boldsymbol{x}_v \mid \boldsymbol{x}_{\mathcal{N}(v)}) = \frac{\prod_{C\in\mathcal{C}(v)} \psi_C(\boldsymbol{x}_v, \boldsymbol{x}_{\mathcal{N}(v)})}{\sum_{y\in\mathcal{X}_v} \prod_{C\in\mathcal{C}(v)} \psi_C(y, \boldsymbol{x}_{\mathcal{N}(v)})} \,, \tag{2.14}$$

which can be computed in time $\mathcal{O}(|\mathcal{X}_v||\mathcal{C}(v)|)$. The Gibbs sampler starts with an arbitrary joint state $\boldsymbol{x}^0$ and updates each variable by sampling from (2.14). If this process is repeated multiple times, the resulting joint state $\boldsymbol{x}$ will be a valid sample from $\mathbb{P}$, independent of $\boldsymbol{x}^0$. Gibbs sampling is generic and requires only the space to store the n-dimensional sample $\boldsymbol{x}$, the model parameters $\boldsymbol{\theta} \in \mathbb{R}^d$ and the neighborhoods $\mathcal{N}(v)$. The memory complexity is hence $\mathcal{O}(n+d+2m)$. Based on a set of samples $\mathcal{D}$ generated from Gibbs sampling (or other MCMC techniques), the (inverse) partition function may be estimated via the Ogata-Tanemura method [160, 177]:

$$\frac{1}{\hat{Z}(\boldsymbol{\theta})} = \frac{1}{|\mathcal{X}||\mathcal{D}|} \sum_{\boldsymbol{x}\in\mathcal{D}} \frac{1}{\psi(\boldsymbol{x})}$$

However, an estimate of the number of samples B that have to be discarded must be available. Determining this number is rather cumbersome. In practice one can apply statistical test, but none of them provides entirely reliable diagnostics whether the chain has stabilized. A body of theoretical work tries to lower bound the number of steps required for the distribution of the Markov chain to be close to the target [108]. Nevertheless, such results are highly problem specific, and do in general not lead to an actual number of burn-in steps, but rather asymptotic statements.

Maximum a posteriori states

Sampling procedures generate joint states according to their density. Taking multiple samples, joint states with higher density will appear more often. The most likely joint $\boldsymbol{x}^*$ of the model, i.e., the state of highest density, is called maximum a posteriori state.

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}\in\mathcal{X}} p_{\boldsymbol{\theta}}(\boldsymbol{X}=\boldsymbol{x}) = \arg\max_{\boldsymbol{x}\in\mathcal{X}} \langle \boldsymbol{\theta}, \phi(\boldsymbol{x})\rangle \ , \tag{2.15}$$

Moreover, we may choose any subset $O \subset V$ of variables whose states are known or observed, and a set of variables $H \subset V \setminus O$ whose states are unknown or hidden. We could then ask for the most likely state of the hidden variables, given the observation.

$$\boldsymbol{x}_H^* = \arg\max_{\boldsymbol{x}_H\in\mathcal{X}_H} p_{\boldsymbol{\theta}}(\boldsymbol{X}_H=\boldsymbol{x}_H \mid \boldsymbol{X}_O = \boldsymbol{x}_O) \tag{2.16}$$

We call (2.16) the conditional MAP problem. By means of (2.16), any Markov random field can be treated as a set of classifiers, where we can choose which variables represent classes and which variables represent features. This is especially helpful when the particular prediction task is not known at learning time.

Instead of sampling the states of variables, one can try to exploit the structure of ϕ and $\boldsymbol{\theta}$ to find or estimate the (conditional) MAP. The corresponding decision problem is **NP**-hard, so unless we employ exhaustive search techniques, we should in general not expect to find the true MAP state fast. However, a large body contains sophisticated techniques to address this problem. We will shortly summarize the most important results.

Using the same reasoning that we employed in the derivation of belief propagation (Section 2.2.2), one can show that BP, when we replace $+$ by max, solves the MAP problem exactly whenever the conditional independence structure is a tree. To distinguish both BP versions, we call the original messages (2.7) sum-product messages and the maximization version max-product messages. Technically, this can be seen as a transition from the $(+, \times)$-semiring to the $(\min, +)$-semiring [4, 130]. This fact is especially attractive in the context of resource-constrained systems, since the machine code for sum-product and max-product messages is almost identical and can hence be shared between the algorithms.

If the structure is loopy, we may indeed compute the junction tree (Algorithm 2.2) and compute the max-product Shafer-Shenoy messages to reveal the exact maximizing joint state. If we cannot afford to construct the junction tree, we may run loopy belief propagation instead. As with LBP, max-product LBP delivers very good empirical results [242].

It happens that in fact multiple $\boldsymbol{x}$ turn-out to be "most likely", at least approximately. Conditions and techniques have been studied under which the most likely joint state can be uniquely decoded via simple maximization over the vertex marginals of max-product messages [231, 223, 145, 123]. Many of them are specific to certain sub-classes of undirected models. An important theoretical result shows, that the fixpoints of max-product LBP on loopy conditional independence structures are so-called neighborhood maximums of the posterior probability. This means that the posterior probability of the max-product assignment is guaranteed to be greater than all other assignments in a particular large region around that assignment. The region includes all assignments that differ from the max-product assignment in any subset of nodes that form no more than a single loop in the graph. In some graphs, this neighborhood is exponentially large [231]. The latter result is a sound theoretical justification for the MAP approximation via LBP. However, tight bounds on the quality are not available. Moreover, LBP might not always converge.

Inspired by linear programming (LP) relaxations, block coordinate descent-like algorithms are known to always converge and find the exact MAP state whenever there is only one unique optimum [77, 128]. Moreover, LP approximations to the MAP problem enjoy quality guarantees. Specifically, they provide upper bounds on the MAP value and optimality certificates. Furthermore, they often work for graphs with large tree-width as shown in [195].

Instead of LP relaxations, the convexified approximate variational principle can also be applied to approximate the MAP [232]. Several analyses and convergent algorithms for the tree-reweighted approach have been proposed [223, 126, 123, 124]. Those techniques are based on the appealing property that (2.15) is a convex function of $\boldsymbol{\theta}$, since it is the maximum over linear functions. Hence, Jensen's inequality can be employed by imposing a probability distribution over simpler conditional independence structures (cf. Section 2.2.2). Due to its low resource overhead and guaranteed convergence, the sequential tree-reweighted message passing (TRW-S) technique [123] is an appropriate alternative to max-product LBP in the context of resource-constrained systems.

Finally, instead of finding or approximating one solution of (2.15), we can ask for the top-M most probable joint states. All of the above mentioned techniques can be modified such that a set of possible MAP assignments can be found [243, 71, 68, 16].

Since our main concern is learning, we will not investigate the MAP problem any further in the remainder of this thesis. We will instead focus on the other inference tasks, namely computation of the partition function, marginal probabilities, and parameter estimation.

2.3 Numerical Optimization and Regularization

The basic terms and tools of probability theory and exponential families are now established, and we proceed by introducing techniques from numerical optimization. These are essential to accomplish the actual estimation of $\boldsymbol{\theta}$ and G—the parameters and the conditional independence structure. Starting point for both tasks is a multi set $\mathcal{D}$—the data set—which contains samples from an n-dimensional random variable $\boldsymbol{X}$. We will discuss state-of-the art methods in the context of limited resources, and review results which have a strong impact on the resource consumption of parameter and structure estimation.

2.3.1 Parameter Estimation

It is the standard procedure in machine learning to specify a loss function ℓ, depending on a data set $\mathcal{D}$ and a parameter (vector) $\boldsymbol{\theta}$. From the set of all valid parameters, we choose $\hat{\boldsymbol{\theta}}$, the one that minimizes the loss. In the course of estimating a probability density, the most basic and intuitive method is the maximum likelihood (ML) principle. A key assumption is the independence of samples; if multiple samples are known to depend on each other, a larger model has to be constructed the takes care of these dependences. Due to independence of the data points, the probability density of our data set factorizes into the densities of each sample, i.e., $p_{\boldsymbol{\theta}}(\mathcal{D}) = \prod_{\boldsymbol{x}\in\mathcal{D}} p_{\boldsymbol{\theta}}(\boldsymbol{x})$. Choosing the parameter $\boldsymbol{\theta}$ that maximizes the joint density of $\mathcal{D}$ is the core of the ML estimation. The actual goal is to find the most likely parameter, given the data, i.e.,

$$\hat{\boldsymbol{\theta}} \in \arg\max_{\boldsymbol{\theta}\in\mathbb{R}^d} p(\boldsymbol{\theta} \mid \mathcal{D}) = \arg\max_{\boldsymbol{\theta}\in\mathbb{R}^d} \frac{p(\mathcal{D} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \; . \tag{2.17}$$

Following the original ML setting, we are agnostic about $p(\boldsymbol{\theta})$ and treat it as a constant—an assumption that will be dropped later on. The division by $p(\mathcal{D})$ is irrelevant for the maximization, since it does not depend on $\boldsymbol{\theta}$. Due to monotonicity and numerical convenience, we may maximize $\log p$ instead of p. Finally, to better match the intuition of loss minimization, we minimize the negative, average log-density $-(1/|\mathcal{D}|)\log p(\mathcal{D})$. Substituting the density of exponential family members (2.3), the ML loss function becomes

$$\ell(\boldsymbol{\theta};\mathcal{D}) = -\frac{1}{|\mathcal{D}|}\sum_{\boldsymbol{x}\in\mathcal{D}}(\langle\boldsymbol{\theta},\phi(\boldsymbol{x})\rangle - A(\boldsymbol{\theta})) = -\langle\boldsymbol{\theta},\tilde{\boldsymbol{\mu}}\rangle + A(\boldsymbol{\theta}) \; , \tag{2.18}$$

where the last equality holds by linearity of the dot product and the abbreviation $\tilde{\boldsymbol{\mu}} = \frac{1}{|\mathcal{D}|}\sum_{\boldsymbol{x}\in\mathcal{D}}\phi(\boldsymbol{x})$ for the sample mean estimator of the sufficient statistic. The gradient is $\nabla\ell(\boldsymbol{\theta};\mathcal{D}) = \nabla A(\boldsymbol{\theta}) - \tilde{\boldsymbol{\mu}}$. Note that an exact computation of the (log) likelihood is in general not possible due to the **#P**-hardness of the (log) partition function, and a large body of work about its approximation exists ([104, 226], Section 2.2.2, Chp. 5).

Now, we would like to find the parameter $\hat{\boldsymbol{\theta}}$ that minimizes the loss, i.e., $\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^d}\ell(\boldsymbol{\theta};\mathcal{D})$. To apply methods of numerical optimization, we need to derive expressions for the partial[5] derivatives of A.

Lemma 2.3 (Moments and the Partition Function [227]) *The first and second derivatives of the log-partition function are the first moment and the second central moment, respectively, of the random variable $\phi(\boldsymbol{X})$ w.r.t. the density $p_{\boldsymbol{\theta}}$.*

$$\frac{\partial A(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}_i} = \mathbb{E}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})]_i \qquad \frac{\partial^2 A(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}_i\partial\boldsymbol{\theta}_j} = \mathbb{C}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})]_{ij} \tag{2.19}$$

Where $\mathbb{C}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})]$ is the covariance matrix of $\phi(\boldsymbol{X})$ w.r.t. $p_{\boldsymbol{\theta}}$.

[5]It might happen that the number of exponential family parameters is 1, like in a Poisson distribution. To reduce notational clutter, we will not distinguish between a derivative and a partial derivative. Differentiation w.r.t. a variable w is always denoted as partial derivative $\partial/\partial w$.

Proof.

$$\begin{aligned}\frac{\partial A(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_i} &= \frac{\partial}{\partial \boldsymbol{\theta}_i} \log \int_{\mathcal{X}} \psi \,\mathrm{d}\,\nu = \frac{\frac{\partial}{\partial \boldsymbol{\theta}_i} \int_{\mathcal{X}} \exp(\langle \boldsymbol{\theta}, \phi(\boldsymbol{x})\rangle)\,\mathrm{d}\,\nu}{Z(\boldsymbol{\theta})} = \int_{\mathcal{X}} \phi_i p_{\boldsymbol{\theta}} \,\mathrm{d}\,\nu \\ \frac{\partial^2 A(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_i \partial \boldsymbol{\theta}_j} &= \frac{\partial}{\partial \boldsymbol{\theta}_j} \mathbb{E}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})_i] = \int_{\mathcal{X}} \phi_i \frac{\psi \phi_j + \psi \mathbb{E}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})_j]}{Z(\boldsymbol{\theta})} \,\mathrm{d}\,\nu \\ &= \int_{\mathcal{X}} p_{\boldsymbol{\theta}} \phi_i \phi_j - p_{\boldsymbol{\theta}} \phi_i \mathbb{E}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})_j] \,\mathrm{d}\,\nu = \mathbb{E}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})_i \phi(\boldsymbol{X})_j] - \mathbb{E}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})_i] \mathbb{E}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})_j]\end{aligned}$$

■

The partial derivatives of $A(\boldsymbol{\theta})$ corresponds to the expected values of $\phi(\boldsymbol{X})_i$ and hence $\nabla A(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})] = \hat{\boldsymbol{\mu}}$. The same relation can be established via convex conjugacy, i.e., $\hat{\boldsymbol{\mu}} = \arg\sup_{\boldsymbol{\mu}' \in \mathcal{M}} \langle \boldsymbol{\theta}, \boldsymbol{\mu}' \rangle + \mathcal{H}(\boldsymbol{\mu}')$ (2.13).

It is straightforward to show via convex function calculus that A is a convex function of $\boldsymbol{\theta}$, and as such, any local minimum is also a global minimum. When the graphical model is tree-structured, optimal vertex and edge parameters can be computed in closed-form [227]:

$$\hat{\boldsymbol{\theta}}_{v=x} = \log \tilde{\boldsymbol{\mu}}_{v=x} \qquad \hat{\boldsymbol{\theta}}_{vu=xy} = \log \frac{\tilde{\boldsymbol{\mu}}_{vu=xy}}{\tilde{\boldsymbol{\mu}}_{v=x} \tilde{\boldsymbol{\mu}}_{u=y}} \tag{2.20}$$

If the graphical structure is not a tree, the above closed-form no longer applies. It is possible to derive approximate parameters in closed-form, based on the convex approximate variational principle from Section 2.2.2. Those may be interpreted as the expectation of a parameter vector over all spanning-trees of a graph G, i.e., $\bar{\boldsymbol{\theta}} = \sum_{T \in \mathcal{T}(G)} q(T) \boldsymbol{\theta}(T)$, where q is some density, all parameters of $\boldsymbol{\theta}(T)$ which do not correspond to edges of the spanning tree T are 0, and the others are computed via (2.20). The closed-form of $\bar{\boldsymbol{\theta}}$ is then

$$\bar{\boldsymbol{\theta}}_{v=x} = \log \tilde{\boldsymbol{\mu}}_{v=x} \qquad \bar{\boldsymbol{\theta}}_{vu=xy} = q_{vu} \log \frac{\tilde{\boldsymbol{\mu}}_{vu=xy}}{\tilde{\boldsymbol{\mu}}_{v=x} \tilde{\boldsymbol{\mu}}_{u=y}} \,,$$

where $q_{vu} = \sum_{T \in \mathcal{T}(G)} q(T) \mathbb{1}_{\{\{v,u\} \in T\}}$ is the marginal probability that edge $\{v, u\}$ appears in any tree in $\mathcal{T}(G)$. Although the initial work on tree-reweighted approximations assumed that $\mathcal{T}(G)$ contains all of G's spanning trees, it suffices that $\mathcal{T}(G)$ is just some set of trees such that each edge of G is contained in at least one tree in $\mathcal{T}(G)$ [123].

Tree structured models and approximations based on them are well understood and the corresponding inference and parameter estimation problems allow for efficient algorithms. Now, let us assume for the remainder of this chapter that no closed-form solution can be found. In such cases, we employ (accelerated) first-order methods and proximal methods, or second-order methods like BFGS [158]. Second-order methods ensure very fast convergence, but require an prohibitively large amount of resources for high-dimensional models. As an example, the Hessian, e.g., the matrix of all second derivatives $\nabla^2 A(\boldsymbol{\theta})_{ij} = \frac{\partial^2 A(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_i \partial \boldsymbol{\theta}_j}$, of a model with 10^4 parameters requires over 380 megabytes memory at 32 bit floating-point precision—far too much if we assume that only a few kilobytes are available. Moreover, on highly resource-constrained systems, even the simplest full-dimensional vector operations are very expensive. Approximate second-order methods like L-BFGS thus require too many resources, and we resort to (accelerated) proximal first-order methods [153, 17, 164] or techniques which perform only partial updates of the parameter vector [155]. Before we take a closer look at these methods, we discuss what kind of quality we should expect from the estimate.

Properties of the Estimator

Given the undirected conditional independence structure of a random variable $\boldsymbol{X}$, we know by (2.8) that the true probability density factorizes according to the cliques of G, and that it can be rewritten in the exponential family form. This suggests that there is a "true" $\boldsymbol{\theta}^*$ which was used to generate our actual data set $\mathcal{D}$. Classic statistical results assert that maximum likelihood estimation is consistent whenever the likelihood has a unique global optimum. In this case, the ML estimate $\hat{\boldsymbol{\theta}}$ converges in probability to the parameter $\boldsymbol{\theta}^*$ that generated the data. More details on estimation and consistency can be found in [67]. Here, the objective function (2.18) is only strictly convex, if a minimal sufficient statistic is used. While it is true that any non-minimal statistic can be converted to an equivalent minimal statistic (cf. Appendix B.1 in [227]), the actual conversion is rather technical and the resulting statistics loose their interpretability in terms of marginal probabilities. Since non-minimal (overcomplete) sufficient statistics have convenient properties that we will exploit in the sequel, we will sacrifice strict convexity for interpretability. But this does not result in a harder estimation procedure. The objective function is still convex, which means that there is a set of optimal solutions—each corresponding to a different representation of the same model. This gives us the freedom to choose from the set of all optimal parametrizations, the one which implies the largest reduction in memory, arithmetic, or computational complexity. When we find any minimizer of (2.18) based on overcomplete statistics, other optimal solutions may be discovered via shift-invariance.

Lemma 2.4 (Shift Invariance) *Let $\boldsymbol{X}$ be a random variable with exponential family density, parameter $\boldsymbol{\theta} \in \mathbb{R}^d$, and undirected structure G. For any clique $C \in \overline{\mathcal{C}}(G)$ and any vector $\boldsymbol{r} \in \mathbb{R}^d$, let $\boldsymbol{r}_C$ be the sub-vector of components that correspond to the states of C according to Definition 2.6. Let $R_\phi = \{\boldsymbol{r} \in \mathbb{R}^d \mid \forall C \in \overline{\mathcal{C}}(G) : \exists : \boldsymbol{r}_C \in \mathbb{R}^{|\mathcal{X}_C|} : \boldsymbol{r}_C = (r_C, r_C, \dots, r_C)^\top\}$. Exponential family members with overcomplete sufficient statistics are shift-invariant against vectors from R_ϕ, that is, $\forall \boldsymbol{r} \in R : \forall \boldsymbol{x} \in \mathcal{X} : p_{\boldsymbol{\theta}+\boldsymbol{r}}(\boldsymbol{x}) = p_{\boldsymbol{\theta}}(\boldsymbol{x})$.*

Proof. The overcomplete sufficient statistic contains, for all cliques $C \in \overline{\mathcal{C}}(G)$, one binary indicator for each clique state $\boldsymbol{x}_C \in \mathcal{X}_C$ (Definition 2.5). Since any $\boldsymbol{x}$ implies exactly one state for each clique, the vector $\phi(\boldsymbol{x})$ contains exactly one 1-entry per clique, which implies $\langle \boldsymbol{r}, \phi(\boldsymbol{x})\rangle = \sum_{C\in\overline{\mathcal{C}}(G)} r_C$ for all $\boldsymbol{r} \in R_\phi$. Thus, for all $\boldsymbol{r} \in R_\phi$ and all $\boldsymbol{x} \in \mathcal{X}$,

$$p_{\boldsymbol{\theta}+\boldsymbol{r}}(\boldsymbol{x}) = \frac{\exp(\langle \boldsymbol{\theta}+\boldsymbol{r}, \phi(\boldsymbol{x})\rangle)}{\sum_{\boldsymbol{y}\in\mathcal{X}} \exp(\langle \boldsymbol{\theta}+\boldsymbol{r}, \phi(\boldsymbol{y})\rangle)} = \frac{\exp\left(\langle \boldsymbol{\theta}, \phi(\boldsymbol{x})\rangle + \sum_{C\in\overline{\mathcal{C}}(G)} r_C\right)}{\sum_{\boldsymbol{y}\in\mathcal{X}} \exp\left(\langle \boldsymbol{\theta}, \phi(\boldsymbol{y})\rangle + \sum_{C\in\overline{\mathcal{C}}(G)} r_C\right)} = p_{\boldsymbol{\theta}}(\boldsymbol{x}) \; .$$

■

Shift-invariance is also denoted by $\equiv$, e.g., $\forall \boldsymbol{r} \in R : p_{\boldsymbol{\theta}+\boldsymbol{r}} \equiv p_{\boldsymbol{\theta}}$.

Hence, as long as we add the same constant to all parameters which belong to the same clique, the density is unchanged. However, R_ϕ does not contain all vectors which are neutral elements w.r.t. parameter shifting. The full set R^*_ϕ of all vectors $\boldsymbol{r}$ which satisfy $p_{\boldsymbol{\theta}+\boldsymbol{r}} \equiv p_{\boldsymbol{\theta}}$ is the solution set of a system of linear equations. More precisely,

$$p_{\boldsymbol{\theta}+\boldsymbol{r}} \equiv p_{\boldsymbol{\theta}} \Leftrightarrow \boldsymbol{r} \in R^*_\phi \Leftrightarrow \exists c \in \mathbb{R} : \underbrace{\begin{pmatrix} - \phi(\boldsymbol{x}^1)^\top - \\ - \phi(\boldsymbol{x}^2)^\top - \\ \vdots \\ - \phi(\boldsymbol{x}^{|\mathcal{X}|})^\top - \end{pmatrix}}_{\boldsymbol{M}} \boldsymbol{r} = \begin{pmatrix} c \\ c \\ \vdots \\ c \end{pmatrix} . \tag{2.21}$$

The matrix $\boldsymbol{M}$ has $|\mathcal{X}|$ rows and d columns. Each row of $\boldsymbol{M}$ corresponds to the (transposed) sufficient statistic vector of some $\boldsymbol{x} \in \mathcal{X}$. The set R^*_ϕ contains all vectors $\boldsymbol{r}$ with $p_{\boldsymbol{\theta}+\boldsymbol{r}} \equiv p_{\boldsymbol{\theta}}$, for any arbitrary but fixed sufficient statistic—the random variable is allowed to have any state space $\mathcal{X}$. In contrast, the set R_ϕ from Lemma 2.4 works only when ϕ is the binary overcomplete sufficient statistic. The qualitative difference is that Lemma 2.4 gives us a way to construct actual shift-invariant vectors, while R^*_ϕ from (2.21) is of rather theoretical interest—it allows us to characterize the set of all optimal solutions, whenever at least one minimum $\hat{\boldsymbol{\theta}}$ of (2.18) is known.

Nevertheless, keep in mind that even when we know an exact minimizer, the model that generated our data might differ from our estimate. A simple and intuitive notion of how "correct" our estimate is, can be derived from the problem formulation itself. Recall that the gradient of the ML objective is $\nabla \ell(\boldsymbol{\theta}; \mathcal{D}) = \hat{\boldsymbol{\mu}} - \tilde{\boldsymbol{\mu}}$. At any stationary point, we must have $\hat{\boldsymbol{\mu}} = \tilde{\boldsymbol{\mu}}$, which means that our model is able to reproduce the empirical marginals, which in-turn can be shown to be close to the true marginals, as long as we have enough data [241].

Lemma 2.5 (Estimation of Expected Statistics [241]) *Let $\boldsymbol{X}$ be a random variable with state space $\mathcal{X}$, $\mathcal{D} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^N\}$ a data set, and $\phi : \mathcal{X} \to \mathbb{R}^d$ some function. The true expected value of $\phi(\boldsymbol{X})$ is denoted by $\boldsymbol{\mu}^* = \mathbb{E}[\phi(\boldsymbol{X})]$ and the corresponding empirical estimator via $\tilde{\boldsymbol{\mu}} = \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x} \in \mathcal{D}} \phi(\boldsymbol{x})$. Then,*

$$\|\tilde{\boldsymbol{\mu}} - \boldsymbol{\mu}^*\|_\infty \leq \sqrt{\frac{(c+1)\log d}{2|\mathcal{D}|}}$$

with probability of at least $1 - 2\exp(-c \log d)$ and any $c > 0$.

Proof. $\tilde{\boldsymbol{\mu}}$ is unbiased due to $\mathbb{E}[\tilde{\boldsymbol{\mu}}] = \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x} \in \mathcal{D}} \mathbb{E}[\phi(\boldsymbol{X})] = \boldsymbol{\mu}^*$. According to Hoeffding's inequality [94],

$$\mathbb{P}(|\tilde{\boldsymbol{\mu}}_i - \mathbb{E}[\tilde{\boldsymbol{\mu}}]_i| > t) \leq 2\exp(-2|\mathcal{D}|t^2)$$

for all $t > 0$. Since this holds for any dimension i, we can apply the union bound to get

$$\mathbb{P}(\exists i \in [d] : |\tilde{\boldsymbol{\mu}}_i - \boldsymbol{\mu}^*_i| > t) \leq 2\exp(-2|\mathcal{D}|t^2 + \log d) .$$

The lemma follows for $t = \sqrt{(c+1)\log d/(2|\mathcal{D}|)}$. ■

2.3.2 First-Order Methods

The basic idea of first-order methods is to optimize a linear approximation of the objective function. Starting from an arbitrary parameter vector $\boldsymbol{\theta}^0$, an iterative procedure generates the series $\boldsymbol{\theta}^0, \boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \ldots$; in the simplest case, the series is generated via

$$\boldsymbol{\theta}^{i+1} = \boldsymbol{\theta}^i - \kappa_i \nabla \ell(\boldsymbol{\theta}^i; \mathcal{D}) \; . \tag{2.22}$$

If the objective function is truly linear, we may choose $\kappa_i = 1$ and arrive at the optimal solution directly after the first step. If the function is non-linear, $\boldsymbol{\theta}^i$ will converge to a stationary point of ℓ for a reasonable choice of step-sizes $\kappa_0, \kappa_1, \kappa_2, \ldots$. Since our objective function ℓ is convex, each stationary point is a global minimizer. Choosing a good series of step-sizes κ_i can be a delicate process and might involve non-trivial search strategies. Luckily, conditions are known under which a single constant κ suffices.

Definition 2.10 (Lipschitz Continuity) *A function $f : \mathbb{R}^d \to \mathbb{R}^l$ is Lipschitz continuous, if there exists a constant $L \in \mathbb{R}$ such that for all $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^d$,*

$$\|f(\boldsymbol{a}) - f(\boldsymbol{b})\| \leq L \|\boldsymbol{a} - \boldsymbol{b}\| \; ,$$

for any arbitrary but fixed norm $\|\cdot\|$. If not otherwise explicitly stated, Lipschitz continuity is declared w.r.t. the l_2-norm (or equivalently, the Frobenius norm in case of matrix arguments). L is called Lipschitz constant. By the mean value inequality (cf. Proposition 5.3.11 in [8]), if f is continuously differentiable and $L = \sup_{\boldsymbol{c} \in \mathbb{R}^d} \|\nabla f(\boldsymbol{c})\|$, then f is Lipschitz continuous with constant L.

Results about the Lipschitz continuity of exponential family members can be found in the literature [96]. We could, however, not find any results on the Lipschitz continuity of the gradient of $\ell(\boldsymbol{\theta}; \mathcal{D})$ and hence, present our own derivation.

Lemma 2.6 (Lipschitz Continuous Log-Likelihood Gradient) *Let $\boldsymbol{X}$ be a discrete random variable with binary, overcomplete sufficient statistic. $\nabla \ell(\boldsymbol{\theta}; \mathcal{D})$ is Lipschitz continuous with constant $L = 2|\overline{\mathcal{C}}(G)|$.*

Proof. Following Definition 2.10, we have $L = \sup_{\boldsymbol{\theta} \in \mathbb{R}^d} \|\nabla^2 \ell(\boldsymbol{\theta})\|_F$. By plugging-in the Hessian of ℓ from Lemma 2.3, we get

$$\begin{aligned}
L = \sup_{\boldsymbol{\theta} \in \mathbb{R}^d} \|\mathbb{C}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})]\|_F &= \sup_{\boldsymbol{\theta} \in \mathbb{R}^d} \|\mathbb{E}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})\phi(\boldsymbol{X})^\top] - \mathbb{E}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})]\mathbb{E}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})]^\top\|_F \\
&\leq \sup_{\boldsymbol{\theta} \in \mathbb{R}^d} \|\mathbb{E}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})\phi(\boldsymbol{X})^\top]\|_F + \|\mathbb{E}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})]\|_2^2 \\
&< \sup_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathbb{E}_{\boldsymbol{\theta}}[\|\phi(\boldsymbol{X})\phi(\boldsymbol{X})^\top\|_F] + |\overline{\mathcal{C}}(G)| = 2|\overline{\mathcal{C}}(G)| \; .
\end{aligned}$$

The second line follows from the triangle inequality, from submultiplicativity of the Frobenius norm, and from the Forbenius norm's invariance against matrix transposition. To derive the third line, observe that $\|\mathbb{E}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})]\|_F^2 = \sum_{i=1}^d \hat{\boldsymbol{\mu}}_i^2$ is the summation of squared marginals. $\hat{\boldsymbol{\mu}}_i \in (0;1)$ implies $\sum_{i=1}^d \hat{\boldsymbol{\mu}}_i^2 < \sum_{i=1}^d \hat{\boldsymbol{\mu}}_i$. Summing up the marginals of any clique C yields $\sum_{\boldsymbol{y}\in\mathcal{X}_C} \hat{\boldsymbol{\mu}}_{C=\boldsymbol{y}} = 1$, and the summation over the marginals of all cliques is thus the number of factors $|\overline{\mathcal{C}}(G)|$. Every norm is convex, and hence Jensen's inequality (Lemma 2.2) implies that $\|\mathbb{E}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})\phi(\boldsymbol{X})^\top]\|_F \leq \mathbb{E}_{\boldsymbol{\theta}}[\|\phi(\boldsymbol{X})\phi(\boldsymbol{X})^\top\|_F]$. Finally, $\|\phi(\boldsymbol{x})\phi(\boldsymbol{x})^\top\|_F = |\overline{\mathcal{C}}(G)|$ by Definition 2.6. ■

The above lemma is derived for discrete state models with overcomplete sufficient statistics. In case of generic statistics for arbitrary data, it is straightforward to derive the upper bound $d(\phi_{\max}^2 + \hat{\boldsymbol{\mu}}_{\max}^2)$ on L, where $\phi_{\max}$ and $\hat{\boldsymbol{\mu}}_{\max}$ are upper bounds on $\phi(\boldsymbol{x})_i$ and $\hat{\boldsymbol{\mu}}_i$, respectively.

Based on the gradient's Lipschitz constant, we instantiate a result from numerical optimization to see how many iterations of (2.22) we need to achieve a specific precision.

Theorem 2.4 (Convergence of Gradient Descent [156]) *Let $\ell(\boldsymbol{\theta};\mathcal{D})$ be the objective function* (2.18)*, and $\hat{\boldsymbol{\theta}} \in \mathbb{R}^d$ a globally optimal parameter vector. The iterates $\boldsymbol{\theta}^i$ generated by gradient descent* (2.22) *with fixed stepsize $\kappa = 1/L = 1/(2|\overline{\mathcal{C}}(G)|)$ satisfy*

$$\ell(\boldsymbol{\theta}^i;\mathcal{D}) - \ell(\hat{\boldsymbol{\theta}};\mathcal{D}) \leq \frac{|\overline{\mathcal{C}}(G)|}{i}\|\boldsymbol{\theta}^0 - \hat{\boldsymbol{\theta}}\|_2^2\,.$$

Hence, achieving $\ell(\boldsymbol{\theta}^i;\mathcal{D}) - \ell(\hat{\boldsymbol{\theta}};\mathcal{D}) \leq \epsilon$ for any $\epsilon > 0$ requires $\mathcal{O}(|\overline{\mathcal{C}}(G)|/\epsilon)$ steps.

Proof. Lemma 2.6 asserts that $\nabla\ell$ has Lipschitz constant $L = 2|\overline{\mathcal{C}}(G)|$. For ease of exposition, let $g(\boldsymbol{a}) = (L/2)\|\boldsymbol{a}\|_2^2 - \ell(\boldsymbol{a};\mathcal{D})$. Multiplying Definition 2.10 by $\|\boldsymbol{a}-\boldsymbol{b}\|_2$, and applying the Cauchy-Schwarz inequality yields $\langle\nabla\ell(\boldsymbol{a};\mathcal{D}) - \nabla\ell(\boldsymbol{b};\mathcal{D}), \boldsymbol{a}-\boldsymbol{b}\rangle \leq L\|\boldsymbol{a}-\boldsymbol{b}\|_2^2$. Plugging in $\nabla\ell(\boldsymbol{a};\mathcal{D}) = L\boldsymbol{a} - \nabla g(\boldsymbol{a})$ shows, that $\nabla g(\boldsymbol{a})$ is monotonically non-decreasing and thus convex (Definition 2.8). Resubstituting $g(\boldsymbol{a})$ into the definition of convexity leads to a quadratic upper bound on ℓ, namely

$$\ell(\boldsymbol{a};\mathcal{D}) \leq \ell(\boldsymbol{b};\mathcal{D}) + \langle\nabla\ell(\boldsymbol{b};\mathcal{D}), \boldsymbol{a}-\boldsymbol{b}\rangle + (L/2)\|\boldsymbol{a}-\boldsymbol{b}\|_2^2\,. \tag{2.23}$$

Substituting $\boldsymbol{a} = \boldsymbol{\theta}^{i+1}$ (cf. (2.22)) and $\boldsymbol{b} = \boldsymbol{\theta}^i$ into this upper bound, leads to

$$\begin{aligned}
\ell(\boldsymbol{\theta}^{i+1};\mathcal{D}) &\leq \ell(\boldsymbol{\theta}^i;\mathcal{D}) - (\kappa - (L\kappa^2)/2)\|\nabla\ell(\boldsymbol{\theta}^i;\mathcal{D})\|_2^2 & (2.24)\\
&\leq \ell(\hat{\boldsymbol{\theta}};\mathcal{D}) + \langle\nabla\ell(\boldsymbol{\theta}^i;\mathcal{D}), \boldsymbol{\theta}^i - \hat{\boldsymbol{\theta}}\rangle - (\kappa/2)\|\nabla\ell(\boldsymbol{\theta}^i;\mathcal{D})\|_2^2 & (2.25)\\
&= \ell(\hat{\boldsymbol{\theta}};\mathcal{D}) + (1/(2\kappa))(\|\boldsymbol{\theta}^i - \hat{\boldsymbol{\theta}}\|_2^2 - \|\boldsymbol{\theta}^{i+1} - \hat{\boldsymbol{\theta}}\|_2^2) & (2.26)
\end{aligned}$$

Where we used the definition of $\boldsymbol{\theta}^{i+1}$ from (2.22). We see from (2.24) that the series of function values $\ell(\boldsymbol{\theta}^0;\mathcal{D}), \ell(\boldsymbol{\theta}^1;\mathcal{D}), \ell(\boldsymbol{\theta}^2;\mathcal{D}), \ldots$ is non-increasing. Any function value $\ell(\boldsymbol{\theta}^j;\mathcal{D})$ with $j < i$ cannot be closer to $\ell(\hat{\boldsymbol{\theta}};\mathcal{D})$ than the function value of the i-th iterate $\ell(\boldsymbol{\theta}^i;\mathcal{D})$, and so is the their average. Finally, we get

$$\ell(\boldsymbol{\theta}^i;\mathcal{D}) - \ell(\hat{\boldsymbol{\theta}};\mathcal{D}) \leq \frac{1}{i}\sum_{j=1}^{i-1}\ell(\boldsymbol{\theta}^j;\mathcal{D}) - \ell(\hat{\boldsymbol{\theta}};\mathcal{D}) \leq \frac{\|\boldsymbol{\theta}^0 - \hat{\boldsymbol{\theta}}\|_2^2}{2i\kappa} \tag{2.27}$$

by applying (2.26) to each term in the sum. ■

Gradient descent stores the current parameter vector $\boldsymbol{\theta}^i$ and the gradient. According to the theorem, the number of iterations that we need until a specific distance ϵ to the optimal function value is achieved, is linear in the size of the conditional independence structure, linear in $1/\epsilon$, and indeed quadratic in $\|\boldsymbol{\theta}^0 - \hat{\boldsymbol{\theta}}\|_2$. Each iteration requires access to $\nabla\ell(\boldsymbol{\theta}^i; \mathcal{D}) = \mathbb{E}_{\boldsymbol{\theta}^i}[\phi(\boldsymbol{X})] - \tilde{\boldsymbol{\mu}}$ for computing the parameter update (2.22), which in turn requires probabilistic inference to compute $\mathbb{E}_{\boldsymbol{\theta}^i}[\phi(\boldsymbol{X})]$ (Section 2.2.2). There are so-called pseudo-likelihood approaches, which destroy all but direct dependencies between variables to ease parameter estimation [20]. Such procedures may yield very good results in specific application. Nevertheless, breaking the random field into parts renders the (long-range) dependencies of graphical models (Section 2.2) unreliable, and is, in general, a crude approximation of the joint density. For comparison: using loopy belief propagation will result in inconsistent marginals for cliques with more than two vertices, nevertheless, long range dependencies are preserved in the sense that every clique potential influences every marginal probability. Probabilistic inference has high resource requirements in general, and it is hence important to understand which parts influence the number of training iterations.

The distance to the optimum is considered as a constant, since we cannot assume to have a priori knowledge about the location of the optimum—initializing $\boldsymbol{\theta}^0$ such that it is close to $\hat{\boldsymbol{\theta}}$ is not possible. W.l.o.g., we initialize $\boldsymbol{\theta}^0$ to the zero vector, and hence, the convergence speed depends on the squared l_2-norm of the optimal solution $\|\hat{\boldsymbol{\theta}}\|_2^2$. The first point that we can control is the desired precision ϵ. We have to keep in mind that the optimization problem is based on data samples, and is in fact a statistical estimation problem. Hence, the precision up to which we can estimate the true parameter is dictated by the quality of our data, which may render arbitrary small ϵ values spurious or useless [152, 2]. As an example, any optimum of (2.18) guarantees, that the marginals $\hat{\boldsymbol{\mu}}$, generated by our model, and the empirical marginals $\tilde{\boldsymbol{\mu}}$ coincide. But we know from Lemma 2.5, that there is some discrepancy between the empirical marginals which we observe, and the true underlying marginals $\boldsymbol{\mu}^*$. Hence, it is questionable if we really want to match $\tilde{\boldsymbol{\mu}}$ up to very small ϵ—a topic that will also be addressed in the next Section.

Due to the linearity in L and $1/\epsilon$, the runtime of ordinary first-order methods can be large. Accelerated first-order methods exist, which achieve sub-linear convergence, i.e., the number of iterations required to reach ϵ precision is $\mathcal{O}(\sqrt{L/\epsilon})$—this result is known to be optimal for gradient descent-type methods for convex programs under the Lipschitz gradient condition [156]. The acceleration is achieved, by introducing a sequence of auxiliary variables $\boldsymbol{\alpha}^0, \boldsymbol{\alpha}^1, \ldots$ and weights $\beta_0, \beta_1, \beta_2, \ldots$, to include an extrapolation of the last two iterates into the parameter update. So for $\beta_0 = 1$ and arbitrary $\boldsymbol{\alpha}^0 \in \mathbb{R}^d$, iterating

$$\boldsymbol{\theta}^i = \boldsymbol{\alpha}^i - \kappa_i \nabla\ell(\boldsymbol{\alpha}^i; \mathcal{D}), \quad \beta_{i+1} = \frac{1 + \sqrt{4\beta_i^2 + 1}}{2}, \quad \boldsymbol{\alpha}^{i+1} = \boldsymbol{\theta}^i + \frac{\beta_i - 1}{\beta_{i+1}}(\boldsymbol{\theta}^i - \boldsymbol{\theta}^{i-1}) \tag{2.28}$$

generates the accelerated sequence of solutions $\boldsymbol{\theta}^0, \boldsymbol{\theta}^1, \ldots$, which converges faster to $\hat{\boldsymbol{\theta}}$. Note that β_i satisfies $\beta_{i+1}^2 = \beta_{i+1} + \beta_i^2$.

Theorem 2.5 (Convergence of Accelerated Methods [153]) *Let $\ell(\boldsymbol{\theta};\mathcal{D})$ be the objective function* (2.18)*, and $\hat{\boldsymbol{\theta}} \in \mathbb{R}^d$ a globally optimal parameter vector. The iterates $\boldsymbol{\theta}^i$ generated by accelerated gradient descent* (2.28) *with fixed stepsize $\kappa = 1/L = 1/(2|\overline{\mathcal{C}}(G)|)$ satisfy*

$$\ell(\boldsymbol{\theta}^i;\mathcal{D}) - \ell(\hat{\boldsymbol{\theta}};\mathcal{D}) \leq \frac{4|\overline{\mathcal{C}}(G)|}{(i+2)^2}\|\boldsymbol{\alpha}^0 - \hat{\boldsymbol{\theta}}\|_2^2 \,.$$

Hence, achieving $\ell(\boldsymbol{\theta}^i;\mathcal{D}) - \ell(\hat{\boldsymbol{\theta}};\mathcal{D}) \leq \epsilon$ for any $\epsilon > 0$ requires $\mathcal{O}((|\overline{\mathcal{C}}(G)|/\epsilon)^{\frac{1}{2}})$ steps.

Proof. We follow the proof of Theorem 2.4 with $\boldsymbol{a} = \boldsymbol{\theta}^{i+1}$ and $\boldsymbol{b} = \boldsymbol{\alpha}^{i+1}$ until inequality (2.25). Rearranging the terms therein, we obtain

$$\langle \nabla\ell(\boldsymbol{\alpha}^{i+1};\mathcal{D}), \boldsymbol{\alpha}^{i+1} - \hat{\boldsymbol{\theta}}\rangle \geq \ell(\boldsymbol{\theta}^{i+1};\mathcal{D}) - \ell(\hat{\boldsymbol{\theta}};\mathcal{D}) + (\kappa/2)\|\nabla\ell(\boldsymbol{\alpha}^{i+1};\mathcal{D})\|_2^2 \,. \tag{2.29}$$

As opposed to gradient descent, the sequence of objective function values $\ell(\boldsymbol{\theta}^i;\mathcal{D})$, generated by the accelerated method, is not monotonic. We hence cannot rely on the same arguments as in (2.27) to show the bound—a more direct relation between the iterates is required. To derive this, let $\boldsymbol{r}^i = (\beta_i - 1)(\boldsymbol{\theta}^{i-1} - \boldsymbol{\theta}^i)$. By (2.24) and convexity of ℓ, we have

$$(\kappa/2)\|\nabla\ell(\boldsymbol{\alpha}^{i+1};\mathcal{D})\|_2^2 \leq \ell(\boldsymbol{\theta}^i;\mathcal{D}) - \ell(\boldsymbol{\theta}^{i+1};\mathcal{D}) - \frac{1}{\beta_{i+1}}\langle \nabla\ell(\boldsymbol{\alpha}^{i+1};\mathcal{D}), \boldsymbol{r}^i\rangle \,. \tag{2.30}$$

Some rearrangements show, that

$$\begin{aligned}
\|\boldsymbol{r}^{i+1} - \boldsymbol{\theta}^{i+1} + \hat{\boldsymbol{\theta}}\|_2^2 &= \|\boldsymbol{r}^i - \boldsymbol{\theta}^i + \beta_{i+1}\kappa\nabla\ell(\boldsymbol{\alpha}^{i+1};\mathcal{D}) + \hat{\boldsymbol{\theta}}\|_2^2 \\
&= \|\boldsymbol{r}^i - \boldsymbol{\theta}^i + \hat{\boldsymbol{\theta}}\|_2^2 + \beta_{i+1}^2\kappa_2\|\nabla\ell(\boldsymbol{\alpha}^{i+1};\mathcal{D})\|_2^2 \\
&\quad + 2(\beta_{i+1} - 1)\kappa\langle\nabla\ell(\boldsymbol{\alpha}^{i+1};\mathcal{D}), \boldsymbol{r}^i\rangle \\
&\quad + 2\beta_{i+1}\kappa\langle\nabla\ell(\boldsymbol{\alpha}^{i+1};\mathcal{D}), \hat{\boldsymbol{\theta}} - \boldsymbol{\alpha}^{i+1}\rangle \,.
\end{aligned}$$

Further rearrangements and applying (2.29) to the last line yields

$$\begin{aligned}
&\|\boldsymbol{r}^{i+1} - \boldsymbol{\theta}^{i+1} + \hat{\boldsymbol{\theta}}\|_2^2 - \|\boldsymbol{r}^i - \boldsymbol{\theta}^i + \hat{\boldsymbol{\theta}}\|_2^2 \\
= \; & -2\beta_{i+1}\kappa(\ell(\boldsymbol{\theta}^{i+1};\mathcal{D}) - \ell(\hat{\boldsymbol{\theta}};\mathcal{D})) + 2(\beta_{i+1} - 1)\kappa\langle\nabla\ell(\boldsymbol{\alpha}^{i+1};\mathcal{D}), \boldsymbol{r}^i\rangle \\
& + (\beta_{i+1}^2 - \beta_{i+1})\kappa_2\|\nabla\ell(\boldsymbol{\alpha}^{i+1};\mathcal{D})\|_2^2 \,.
\end{aligned}$$

Substituting (2.30) and $\beta_{i+1}^2 = \beta_{i+1} + \beta_i^2$, results in

$$\begin{aligned}
&\|\boldsymbol{r}^{i+1} - \boldsymbol{\theta}^{i+1} + \hat{\boldsymbol{\theta}}\|_2^2 - \|\boldsymbol{r}^i - \boldsymbol{\theta}^i + \hat{\boldsymbol{\theta}}\|_2^2 \\
\leq \; & 2(\beta_{i+1}^2 - \beta_{i+1})\kappa(\ell(\boldsymbol{\theta}^i;\mathcal{D}) - \ell(\boldsymbol{\theta}^{i+1};\mathcal{D})) - 2\beta_{i+1}\kappa(\ell(\boldsymbol{\theta}^{i+1};\mathcal{D}) - \ell(\hat{\boldsymbol{\theta}};\mathcal{D})) \\
= \; & 2\kappa\beta_i^2(\ell(\boldsymbol{\theta}^i;\mathcal{D}) - \ell(\hat{\boldsymbol{\theta}};\mathcal{D})) - 2\kappa\beta_{i+1}^2(\ell(\boldsymbol{\theta}^{i+1};\mathcal{D}) - \ell(\hat{\boldsymbol{\theta}};\mathcal{D}))
\end{aligned}$$

This inequality can now be chained to bound the distance between the i-th and the optimal function value.

$$
\begin{aligned}
2\kappa\beta_i^2(\ell(\boldsymbol{\theta}^i;\mathcal{D}) - \ell(\hat{\boldsymbol{\theta}};\mathcal{D})) &\le 2\kappa\beta_i^2(\ell(\boldsymbol{\theta}^i;\mathcal{D}) - \ell(\hat{\boldsymbol{\theta}};\mathcal{D})) + \|\boldsymbol{r}^i - \boldsymbol{\theta}^i + \hat{\boldsymbol{\theta}}\|_2^2 \\
&\le 2\kappa\beta_{i-1}^2(\ell(\boldsymbol{\theta}^{i-1};\mathcal{D}) - \ell(\hat{\boldsymbol{\theta}};\mathcal{D})) + \|\boldsymbol{r}^{i-1} - \boldsymbol{\theta}^{i-1} + \hat{\boldsymbol{\theta}}\|_2^2 \\
&\dots \\
&\le 2\kappa\beta_0^2(\ell(\boldsymbol{\theta}^0;\mathcal{D}) - \ell(\hat{\boldsymbol{\theta}};\mathcal{D})) + \|\boldsymbol{r}^0 - \boldsymbol{\theta}^0 + \hat{\boldsymbol{\theta}}\|_2^2 \\
&\le \|\boldsymbol{\alpha}^0 - \hat{\boldsymbol{\theta}}\|_2^2
\end{aligned}
$$

The last inequality uses (2.29) and the fact that $\boldsymbol{r}^0 = 0$ and $\beta_0 = 1$. The theorem follows from $\beta_i \ge (1/2)(i+2)$. ■

Recent results indicate, that the asymptotic rate of convergence is even slightly faster then shown above [12]—$o(i^{-2})$ instead of $\mathcal{O}(i^{-2})$. Moreover, monotonic accelerated first-order methods have been developed [17].

Although Theorem 2.4 and 2.5 are known for long, we decided to reproduce them here due to their fundamental impact on the resource consumption of parameter estimation. While the generic versions of these Theorems are phrased with a generic gradient Lipschitz constant L, we presented the convergence results in terms of our own upper bound $2|\overline{\mathcal{C}}(G)|$.

Improving the runtime is only possible by additional assumptions on the loss function, e.g., strong convexity—a property that fails to hold for many loss functions and also fails in our case. It is possible to prove faster convergence based on restricted strong convexity [152], which holds for a broader set of loss functions [2]. The improvements are, however, problem instance specific and so are not further discussed here.

Besides ϵ, the second point that influences the number of iterations is the number of cliques; more specifically, the number of factors. It arises through the appearance of the gradient's Lipschitz constant in the quadratic upper bound (2.23).

In the following Chapters, it will be useful to conduct a block-wise optimization of $\boldsymbol{\theta}$, where the blocks of $\boldsymbol{\theta}$ are constituted from the clique factors, e.g., $\boldsymbol{\theta} = (\boldsymbol{\theta}_C)_{C\in\overline{\mathcal{C}}(G)}$. To this end, applying the Definition of Lipschitz continuity (Definition 2.10) yields the corresponding block Lipschitz constants.

Lemma 2.7 (Clique-wise Gradient Lipschitz Constants) *Let $\ell(\boldsymbol{\theta};\mathcal{D})$ be the negative log-likelihood of an exponential family with binary overcomplete sufficient statistic, and let C be some clique in $\overline{\mathcal{C}}(G)$. Then,*

$$
\|\nabla\ell(\boldsymbol{\theta};\mathcal{D})_C - \nabla\ell(\boldsymbol{\theta}';\mathcal{D})_C\|_2 < \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2 \;,
$$

which implies $L_C < 1$.

Proof. Let us consider the entries of the covariance matrix of the random vector $\phi_C(\boldsymbol{X})$.

$$
\begin{aligned}
\mathbb{C}_{\boldsymbol{\theta}}[\phi_C(\boldsymbol{X})]_{i,j} &= \mathbb{E}_{\boldsymbol{\theta}}[\phi_C(\boldsymbol{X})\phi_C(\boldsymbol{X})^\top]_{i,j} - \mathbb{E}_{\boldsymbol{\theta}}[\phi_C(\boldsymbol{X})]_i\mathbb{E}_{\boldsymbol{\theta}}[\phi_C(\boldsymbol{X})]_j^\top \\
&= \left(\sum_{\boldsymbol{x}\in\mathcal{X}} p_{\boldsymbol{\theta}}(\boldsymbol{x})\phi_C(\boldsymbol{X})_i\phi_C(\boldsymbol{X})_j\right) - \left(\sum_{\boldsymbol{x}\in\mathcal{X}} p_{\boldsymbol{\theta}}(\boldsymbol{x})\phi_C(\boldsymbol{x})_i\right)\left(\sum_{\boldsymbol{x}\in\mathcal{X}} p_{\boldsymbol{\theta}}(\boldsymbol{x})\phi_C(\boldsymbol{x})_j\right)
\end{aligned}
$$

with $i, j \in [|\mathcal{X}_C|]$. If $i = j$,

$$\mathbb{C}_{\boldsymbol{\theta}}[\phi_C(\boldsymbol{X})]_{i,i} = \underbrace{\left(\sum_{\boldsymbol{x}\in\mathcal{X}} p_{\boldsymbol{\theta}}(\boldsymbol{x})\phi_C(\boldsymbol{X})_i\right)}_{p_i} - \underbrace{\left(\sum_{\boldsymbol{x}\in\mathcal{X}} p_{\boldsymbol{\theta}}(\boldsymbol{x})\phi_C(\boldsymbol{x})_i\right)^2}_{p_i^2} < p_i \; . \tag{2.31}$$

Note that if $i \neq j$, the term

$$\sum_{\boldsymbol{x}\in\mathcal{X}} p_{\boldsymbol{\theta}}(\boldsymbol{x})\phi_C(\boldsymbol{X})_i\phi_C(\boldsymbol{X})_j$$

must be zero, since either $\phi_C(\boldsymbol{X})_i = 1$ or $\phi_C(\boldsymbol{X})_j = 1$, but not both. Hence, if $i \neq j$,

$$\mathbb{C}_{\boldsymbol{\theta}}[\phi_C(\boldsymbol{X})]_{i,j} = -\left(\sum_{\boldsymbol{x}\in\mathcal{X}} p_{\boldsymbol{\theta}}(\boldsymbol{x})\phi_C(\boldsymbol{x})_i\right)\left(\sum_{\boldsymbol{x}\in\mathcal{X}} p_{\boldsymbol{\theta}}(\boldsymbol{x})\phi_C(\boldsymbol{x})_j\right) = -p_i p_j \; . \tag{2.32}$$

Now,

$$1 = 1^2 = \left(\sum_{i=1}^{|\mathcal{X}_C|} p_i\right)^2 = \sum_{i=1}^{|\mathcal{X}_C|}\sum_{j=1}^{|\mathcal{X}_C|} p_i p_j = \sum_{i=1}^{|\mathcal{X}_C|} p_i^2 + \sum_{i=1}^{|\mathcal{X}_C|}\sum_{j\neq i} p_i p_j > \sum_{i=1}^{|\mathcal{X}_C|} p_i^2 + \sum_{i=1}^{|\mathcal{X}_C|}\sum_{j\neq i} p_i^2 p_j^2 \; . \tag{2.33}$$

Using (2.31), (2.32) and (2.33), we have

$$\|\mathbb{C}_{\boldsymbol{\theta}}[\phi_C(\boldsymbol{X})]\|_F < \sqrt{\sum_{i=1}^{|\mathcal{X}_C|} p_i^2 + \sum_{i=1}^{|\mathcal{X}_C|}\sum_{j\neq i}(-p_i p_j)^2} \;<\; 1 \; ,$$

and thus $L_C = \sup_{\boldsymbol{\theta}_C \in \mathbb{R}^{|\mathcal{X}_C|}} \|\mathbb{C}_{\boldsymbol{\theta}}[\phi_C(\boldsymbol{X})]\|_F < 1$. ■

2.3.3 Regularization

Maximum likelihood estimation has appealing properties, but it might be reasonable to establish additional criteria for the model that we want to estimate. Particular subsets of parameters can have special characteristics, which make us prefer them over others, even when this means to sacrifice optimality in terms of (2.18). The concept of penalizing models which lack certain desired properties is called regularization. It will turn out, that regularization is inherent in the course of analyzing probabilistic models in the context of resource-constrained systems.

In the literature, maybe the most famous desired property of machine learning models is sparsity—the estimated parameter vector contains "many" 0 entries. Sparsity can have various implications: (1) First of all, the obvious algebraic advantage that parameters which are 0 may drastically simplify the underlying computation (Section 2.3.4), since some parts of an expression need not be evaluated; (2) Setting parameters to 0 while still achieving good predictive performance shows, that the data or features which are weighted by the parameters are unnecessary, hence, sparsity acts like a feature selection [203, 222]; (3) Given the fact that most data sets are not perfect due to measurement errors or filthy editing, we might not want a model that fits perfectly to the data and its imperfections. Enforcing sparsity can therefore be interpreted as suppressing noisy and shortcoming of the data to prevent overfitting; (4) Finally, theoretical insights and practical results show, that sparsity can reduce the sample complexity of the estimation problem. More specifically, the covering number of a parametrized family of functions whose parameters have bounded l_1-norm, is in $\mathcal{O}(\log d)$. Based on results from [176], it can be shown that the number of samples required to achieve a certain accuracy ϵ is lower bounded by $\Omega(\log d)$. A proof of this statement for logistic regression can be found in [157].

Definition 2.11 (Regularization) *Let $f : \mathbb{R}^d \to \mathbb{R}$ be the objective function of a minimization problem. Let further $R(\boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}_+$ be a non-negative, convex function, and $\lambda > 0$. The problem*

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} f(\boldsymbol{\theta}) + \lambda R(\boldsymbol{\theta}) \tag{2.34}$$

is called regularization of the original optimization problem. R is the regularizer and λ is the regularization weight.

The above definition is also called the Lagrangian form of regularization [86]. Intuitively, (2.34) is the Lagrangian of a constrained optimization problem [181]. To see this, let $\lambda = (\alpha(B)/B)$, $B > 0$, and $\alpha : \mathbb{R}_+ \to \mathbb{R}_+$, then,

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^d} f(\boldsymbol{\theta}) + \lambda R(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^d} f(\boldsymbol{\theta}) + \lambda R(\boldsymbol{\theta}) - \alpha(B) \\ &= \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^d} f(\boldsymbol{\theta}) + \alpha(B)((1/B)R(\boldsymbol{\theta}) - 1) = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^d} f(\boldsymbol{\theta}) \text{ s.t. } R(\boldsymbol{\theta}) \leq B\end{aligned}$$

where $\alpha(B)$ is akin to a Lagrange multiplier associated with the constraint $R(\boldsymbol{\theta}) \leq B$. When f and R are convex, and there exists at least one $\boldsymbol{\theta}$ with $R(\boldsymbol{\theta}) < B$, then the existence of a (dual-optimal) $\alpha(B) \geq 0$ is guaranteed by Slater's condition. In case of parameter learning for exponential families, it should thus be possible to find regularization weights which correspond to feasible constraints on the parameter set when R is convex. We investigate the effects of a particular non-convex regularization in Chapter 4.

The most frequently encountered convex regularizers are powers of l_p-norms, e.g.,

$$\|\boldsymbol{\theta}\|_p = \left(\sum_{i=1}^{d} |\boldsymbol{\theta}_i|^p\right)^{\frac{1}{p}} ,$$

especially the l_1-regularization $R(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_1$ or squared-l_2-regularization $R(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2^2$. Squared-$l_2$-regularization penalizes parameters with large absolute value. Moreover, it has the extremely appealing property that any squared-l_2-regularized convex objective function becomes strictly convex. To see this, note that $\nabla^2\lambda\|\boldsymbol{\theta}\|_2^2 = \lambda\boldsymbol{I}$ where $\boldsymbol{I}$ is the $d \times d$ identity matrix. The sum of any positive-semidefinite matrix and $\lambda\boldsymbol{I}$ must be positive definite since $\lambda > 0$, which implies that the regularized objective is strictly convex.

In addition, the l_2-norm is a smooth function of $\boldsymbol{\theta}$ and the corresponding regularized problem may readily be solved via (accelerated) first-order methods. Contrary to the latter, l_1-regularization is non-smooth. First-order methods for the smooth optimization of non-smooth functions employ so-called proximal operators $\text{prox}_{\lambda R}$ [154, 164]. The plain proximal gradient method updates the model parameters according to

$$\boldsymbol{\theta}^{i+1} = \text{prox}^{\kappa}_{\lambda R}(\boldsymbol{\theta}^i - \kappa_i \nabla \ell(\boldsymbol{\theta}^i; \mathcal{D})) \ .$$

The actual instance of the prox-operator depends on the particular regularization R which we like to employ.

Definition 2.12 (Proximal Operator [164]) *Let $R : \mathbb{R}^d \to \mathbb{R}$ be a closed, proper, convex function and $\lambda > 0$. The proximal operator $\text{prox}_R : \mathbb{R}^d \to \mathbb{R}^d$ of R is defined via*

$$\text{prox}^{\kappa}_{\lambda R}(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\gamma} \in \mathbb{R}^d} \lambda R(\boldsymbol{\gamma}) + \frac{1}{2\kappa} \|\boldsymbol{\theta} - \boldsymbol{\gamma}\|_2^2 \ , \tag{2.35}$$

where κ corresponds to the step size of an underlying first-order method. $\text{prox}^{\kappa}_{\lambda R}$ is strongly convex and not everywhere infinite, so it has a unique minimizer for every $\boldsymbol{\theta}$, even if $\text{dom}\, R \subset \mathbb{R}^d$.

Although κ is part of prox, we will write $\text{prox}_{\lambda R}$ instead of $\text{prox}^{\kappa}_{\lambda R}$ to simplify notation. For suitable choices of f, $\text{prox}_f(\boldsymbol{x})$ can be shown to realize first or second order optimization steps for some objective function that is implied by f. Intuitively, $\text{prox}_{\lambda R}(\boldsymbol{\theta})$ is a point that compromises between minimizing R and being near to $\boldsymbol{\theta}$, while λ is a tradeoff parameter between both goals. Depending on the particular regularization, a closed form for prox can be found. In case of l_1-regularization, this closed-form is

$$\text{prox}_{\lambda\|\cdot\|_1}(\boldsymbol{\theta})_i = \begin{cases} \boldsymbol{\theta}_i - \lambda & , \boldsymbol{\theta}_i \geq +\lambda \\ 0 & , |\boldsymbol{\theta}_i| < \lambda \\ \boldsymbol{\theta}_i + \lambda & , \boldsymbol{\theta}_i \leq -\lambda \end{cases} , \tag{2.36}$$

and for the l_2 norm,

$$\text{prox}_{\lambda\|\cdot\|_2}(\boldsymbol{\theta})_i = \begin{cases} 1 - \lambda / \|\boldsymbol{\theta}\|_2 & , \|\boldsymbol{\theta}\|_2 \geq \lambda \\ 0 & , \|\boldsymbol{\theta}\|_2 < \lambda \end{cases} . \tag{2.37}$$

The derivation of the above closed-form and a detailed discussion of proximal operators in general can be found in [164]. Note that the proximal operator of the l_1 norm sets a parameter to 0 whenever its absolute value lies within the interval $[-\lambda, \lambda]$. The behavior of $\text{prox}_{\lambda\|\cdot\|_2}(\boldsymbol{\theta})_i$ is similar but more global, in the sense that the new value of the i-th parameter depends on the overall vector $\boldsymbol{\theta}$ and not just on $\boldsymbol{\theta}_i$.

Iterating $\boldsymbol{\theta}^{i+1} = \operatorname{prox}_{\lambda R}(\boldsymbol{\theta}^i)$ will converge to a minimum of R. This can be combined with the $\boldsymbol{\theta}$-update step in gradient descent (2.22) and the accelerated first-order method (2.28) to yield optimization algorithms for regularized objectives. In case of the accelerated gradient descent on the l_1-regularized likelihood (2.18), the update becomes $\boldsymbol{\theta}^{i+1} = \operatorname{prox}_{\lambda\|\cdot\|_1}(\boldsymbol{\alpha}^i - \kappa^i \nabla \ell(\boldsymbol{\alpha}_i; \mathcal{D}))$, while the rest of the method stays as it is. It can be shown that this procedure, known as the fast iterative shrinkage thresholding algorithm (FISTA), inherits its convergence properties (cf. Theorem 2.5) from the underlying first-order method [17, 12]. Moreover, in Chapter 4, we make use of the fact that proximal first-order methods can converge to critical points even in cases when the corresponding regularization R is non-convex [11, 25].

In summary, regularization can be incorporated with low overhead, as long as R and prox_R are easy to evaluate. In the cases explained above, the memory requirements increase by the need to store λ, and the time required to evaluate the norms and the corresponding proximal operators is $\mathcal{O}(d)$.

While the above introduction to regularization is based on a heuristic augmentation of the objective function, we may also interpret the regularizer as the logarithm of some prior density on the model parameters.

Probabilistic Interpretation

In maximum likelihood estimation, we choose the model that maximizes the probability of our data (2.17). Instead of choosing a single model, one may choose a probability density over models in order to account for uncertainty which is inherent to our estimate. In Bayesian statistics, the likelihood $p(\mathcal{D} \mid \boldsymbol{\theta})$ is combined with a prior $p(\boldsymbol{\theta} \mid \boldsymbol{\lambda})$ over model parameters, controlled by hyperparameters $\boldsymbol{\lambda} \in \mathbb{R}^k$. The density over the model space—the posterior distribution—may then be derived via Bayes rule

$$p(\boldsymbol{\theta} \mid \mathcal{D}, \boldsymbol{\lambda}) = \frac{p(\mathcal{D} \mid \boldsymbol{\theta}, \boldsymbol{\lambda}) p(\boldsymbol{\theta} \mid \boldsymbol{\lambda})}{p(\mathcal{D} \mid \boldsymbol{\lambda})} = \frac{p(\mathcal{D} \mid \boldsymbol{\theta}, \boldsymbol{\lambda}) p(\boldsymbol{\theta} \mid \boldsymbol{\lambda})}{\int_{\mathbb{R}^d} p(\mathcal{D} \mid \boldsymbol{\theta}, \boldsymbol{\lambda}) p(\boldsymbol{\theta} \mid \boldsymbol{\lambda}) \, \mathrm{d}\,\nu(\boldsymbol{\theta})}\,, \tag{2.38}$$

where the denominator $p(\mathcal{D} \mid \boldsymbol{\lambda})$ is the marginal likelihood, i.e., the density of the data for given hyperparameters. Instead of predicting the most likely state $\boldsymbol{x}^*$ based on a single model $\boldsymbol{\theta}$ (2.15), the model in the so-called posterior predictive distribution

$$p(\boldsymbol{x} \mid \mathcal{D}, \boldsymbol{\lambda}) = \int_{\mathbb{R}^d} p(\boldsymbol{x} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathcal{D}, \boldsymbol{\lambda}) \, \mathrm{d}\,\nu(\boldsymbol{\theta}) \tag{2.39}$$

is marginalized out. It should be clear that the integration in (2.38) and (2.39) is rather expensive in general, but in case of exponential families, a conjugate prior exist which may simplify the derivation of the posterior distribution. In case of (finite) discrete state spaces, the conjugate prior is the Dirichlet distribution—well known applications in the field of probabilistic topic models are based on this fact [23]. In general, one could resort to sampling methods in order to approximate the integrals (cf. Section 2.2.2).

Another way to incorporate the prior into the estimation process is known as maximum a posteriori estimation (not be confused with MAP states, cf. Section 2.2.2).

$$\hat{\boldsymbol{\theta}} \in \arg\max_{\boldsymbol{\theta}\in\mathbb{R}^d} p(\mathcal{D} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{\lambda}) \ .$$

Note, that if we plug-in an exponential family member for the data likelihood and construct the negative average log-likelihood, the resulting expression

$$\ell_{\mathrm{MAP}}(\boldsymbol{\theta}; \mathcal{D}) = -\langle \boldsymbol{\theta}, \tilde{\boldsymbol{\mu}} \rangle + A(\boldsymbol{\theta}) - \log p(\boldsymbol{\theta} \mid \boldsymbol{\lambda}) \tag{2.40}$$

looks familiar. This is indeed the ordinary maximum likelihood loss plus a function of $\boldsymbol{\theta}$, parametrized by $\boldsymbol{\lambda}$. The space of possible priors is of course huge, but certain choices will reveal objectives, which we already explained before. Assume, for example, that $p(\boldsymbol{\theta} \mid \boldsymbol{\lambda})$ is a multivariate Gaussian with zero-mean and scaled $(d \times d)$-identity covariance matrix $(2\lambda)^{-1}\boldsymbol{I}$. In this case $\boldsymbol{\lambda} = (2\lambda)^{-1}$ is 1-dimensional and $\log p(\boldsymbol{\theta} \mid \boldsymbol{\lambda}) = -(d/2)\log(\pi\lambda^{-1}) - \lambda\|\boldsymbol{\theta}\|_2^2$. Since the first term is constant (w.r.t. $\boldsymbol{\theta}$), the sets of maximizing arguments of (2.40) and of the l_2-regularized ML objective are identical. Thus, by choosing an independent Gaussian prior we rediscover the squared-l_2-regularized ML objective. Choosing a component-wise Laplacian prior with location 0 and scale $\boldsymbol{\lambda} = \lambda^{-1}$ yields $\log p(\boldsymbol{\theta} \mid \boldsymbol{\lambda}) = -d\log(2\lambda^{-1}) - \lambda\|\boldsymbol{\theta}\|_1$, which is equivalent to maximum likelihood estimation with l_1-regularization.

2.3.4 Structure Estimation

Samples from a distribution can not only be used to identify particular members of an exponential family, but also to estimate the conditional independence structure G of an n-dimensional random variable $\boldsymbol{X}$. There are at most $m_{\max} = n!/(2(n-2)!)$ edges and thus $2^{m_{\max}}$ different possible conditional independence structures. Clearly, an exhaustive search through the space of all structures is not feasible. Classic structure estimation procedures are based on pairwise independence tests, i.e., if the variables $\boldsymbol{X}_v$ and $\boldsymbol{X}_u$ satisfy a specific criterion, the edge $\{v, u\}$ is included in the undirected graph. Recent variants of this idea directly construct triangulated graphs, based on likelihood-ratio tests [166, 167] or local search heuristics [182, 205]. Such methods usually come without any guarantees, since a clique of $k > 2$ vertices may exist in the conditional independence structure, although some of its members are independent. Formally, k vertices are mutually independent if and only if the joint density of all subsets factorizes into a product of vertex marginals. Cliques that contain pairs of independent edges cannot readily be detected by methods which are based on pairwise independence tests. In the special case of Gaussian models, some guarantees can be made for structures, which are derived from partial correlation coefficients [234].

Optimal tree structures can be found via the Chow-Liu algorithm, which is based on choosing the tree structure which minimizes the Kullback-Leibler divergence to the optimal structure. Let $\mathbb{P}$ be the true, unknown probability measure with density p, and $\mathbb{F}_T$ a measure whose density q_T factorizes w.r.t. to a tree structure T over the vertex and edge marginals of the true distribution, i.e., p_v and p_{vu}. According to Definition 7.7,

$$\mathrm{KL}[\ \mathbb{P} \parallel \mathbb{F}_T\ ] = \int_{\mathcal{X}} p \log \frac{p}{q_T} \,\mathrm{d}\,\nu = -\mathcal{H}[\mathbb{P}] + \mathbb{E}[-\log q_T(\boldsymbol{X})]\ ,$$

and because q_T factorizes over the vertex- and edge-marginals of p,

$$\mathrm{KL}[\ \mathbb{P} \parallel \mathbb{F}_T\ ] = -\mathcal{H}[\mathbb{P}] + \sum_{v\in V} \mathcal{H}[\mathbb{P}_v] - \sum_{vu\in E(T)} \mathcal{I}[\boldsymbol{X}_v, \boldsymbol{X}_u]\ .$$

The last equality can be derived via $q_T(\boldsymbol{x}) = \prod_{v\in V} p_v(\boldsymbol{x})^{-\deg(v)} \prod_{vu\in E} p_{vu}(\boldsymbol{x}_v, \boldsymbol{x}_u)$ where $\deg(v)$ is the degree of vertex v, and $E(T)$ is the edge set of the tree T. $\mathcal{I}[\boldsymbol{X}_v, \boldsymbol{X}_u]$ is the mutual information between vertex v and vertex u (cf. Definition 7.8). It is clear, that $\mathcal{H}[\mathbb{P}]$ is unknown—it depends on the true, unknown, conditional independence structure. Nevertheless, it does not depend on our choice of T and neither do the vertex entropies. Hence, minimizing the Kullback-Leibler divergence over the set of all spanning trees is equivalent to choosing the tree whose edge set maximizes the sum of edge-wise mutual information $\arg\min_{T\in\mathcal{T}(G)} \mathrm{KL}[\ \mathbb{P} \parallel \mathbb{F}_T\ ] = \arg\max_{T\in\mathcal{T}(G)} \sum_{vu\in E(T)} \mathcal{I}[\boldsymbol{X}_v, \boldsymbol{X}_u]$. The latter is a maximum weight spanning tree problem over the complete graph with n vertices. Computing the edge weights $\mathcal{I}[\boldsymbol{X}_v, \boldsymbol{X}_u]$ indeed requires access to the pairwise marginals p_{vu} of the true density. Those have to be estimated from the data set $\mathcal{D}$ as explained in Section 2.2.2. Since the sample means are consistent estimators of the marginal probabilities, it can be shown that the outcome of the aforementioned procedure is a consistent estimator of the true underlying conditional independence structure, if the true structure is a tree.

It is worth noting that the factorization of q_T into vertex and edge marginals is only valid if T is a tree. In case of arbitrary structures, a junction tree factorization over cliques and their intersections is possible [227], but the number of possible cliques that we could include into the model is exponential, while the number of possible edges is only quadratic—this explains why choosing the best tree has a polynomial runtime.

The above methods are combinatorial in the sense that they enumerate possible edges of the estimated structure. Numerical methods for the structure estimation task are based on a simple, yet effective observation.

Lemma 2.8 (Edge Deletion) *Let $p_{\boldsymbol{\theta}}$ be an exponential family member with structure $G = (V, E)$ and parameter $\boldsymbol{\theta}$. Assume w.l.o.g. that $\{v, u\} \in E$. Let further $p_{\boldsymbol{\theta}^-}$ be an exponential family member with structure $G^- = (V, E \setminus \{v, u\})$ and parameter $\boldsymbol{\theta}^-$. Let both parameter vectors agree on the common cliques, i.e., $\boldsymbol{\theta} = (\boldsymbol{\theta}^-, \boldsymbol{\theta}^+)$, where $\boldsymbol{\theta}^+$ is the parameter vector of all cliques C which are not in G^-. If the parameters for each $C \in \overline{\mathcal{C}}(G) \setminus \overline{\mathcal{C}}(G^-)$ are set to some constant $c \in \mathbb{R}$, then $p_{\boldsymbol{\theta}} = p_{\boldsymbol{\theta}^-}$.*

Proof. Let $\phi(\boldsymbol{x})^-$ be the sufficient statistic of cliques in $\overline{\mathcal{C}}(G^-)$, and $\phi(\boldsymbol{x})^+$ the sufficient statistic of cliques in $\overline{\mathcal{C}}(G) \setminus \overline{\mathcal{C}}(G^-)$. For all $\boldsymbol{x} \in \mathcal{X}$,

$$
\begin{aligned}
p_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \frac{\exp(\langle \boldsymbol{\theta}, \phi(\boldsymbol{x})\rangle)}{\sum_{\boldsymbol{y}\in\mathcal{X}} \exp(\langle \boldsymbol{\theta}, \phi(\boldsymbol{y})\rangle)} = \frac{\exp(\langle \boldsymbol{\theta}^-, \phi(\boldsymbol{x})^-\rangle + \langle \boldsymbol{\theta}^+, \phi(\boldsymbol{x})^+\rangle)}{\sum_{\boldsymbol{y}\in\mathcal{X}} \exp(\langle \boldsymbol{\theta}^-, \phi(\boldsymbol{y})^-\rangle + \langle \boldsymbol{\theta}^+, \phi(\boldsymbol{y})^+\rangle)} \\
&= \frac{\exp(\langle \boldsymbol{\theta}^-, \phi(\boldsymbol{x})^-\rangle + c|\overline{\mathcal{C}}(G) \setminus \overline{\mathcal{C}}(G^-)|)}{\sum_{\boldsymbol{y}\in\mathcal{X}} \exp(\langle \boldsymbol{\theta}^-, \phi(\boldsymbol{y})^-\rangle + c|\overline{\mathcal{C}}(G) \setminus \overline{\mathcal{C}}(G^-)|)} = p_{\boldsymbol{\theta}^-}(\boldsymbol{x}) \ .
\end{aligned}
$$

■

For any parameter vector $\boldsymbol{\theta}$, the lemma gives us a condition to check, if the model that corresponds to $\boldsymbol{\theta}$ contains parameters for unnecessary cliques—all parameters of an unnecessary clique would be equal. Shift-invariance (Lemma 2.4) tells us, that we may subtract any constant from all parameters which belong to the same clique, without altering the probability density $p_{\boldsymbol{\theta}}$. We may hence assume w.l.o.g., that all parameters of an unnecessary clique are actually 0. The process of parameter estimation from data is, however, subject to noise which arises from an insufficient amount of data (Lemma 2.5). We therefore cannot expect that the estimated parameters of an unnecessary clique would be exactly equal.

Ordinary l_1-regularization enforces sparsity, but in order to detect unnecessary cliques in the parametrization, enforcing single parameters to become 0 might leave us with cliques where all but a few parameters are zero. The so-called group l_1/l_2-regularization can enforce sparsity over whole blocks of parameters and may hence be applied to suppress the imperfections in our data and reveal which cliques might not be required to model the density.

$$
R_{l_1/l_2}(\boldsymbol{\theta}) = \sum_{C\in\overline{\mathcal{C}}(G)} \|\boldsymbol{\theta}_C\|_2
$$

The corresponding proximal operator is $\operatorname{prox}_{\lambda\|\cdot\|_2}(\cdot)$ (2.37), applied to the parameter vector of each clique factor separately.

$$
\operatorname{prox}_{\lambda R_{l_1/l_2}}(\boldsymbol{\theta}) = (\operatorname{prox}_{\lambda\|\cdot\|_2}(\boldsymbol{\theta}_C) : C \in \overline{\mathcal{C}}(G)) \tag{2.41}
$$

After learning $\boldsymbol{\theta}$ under group l_1/l_2-regularization, we may read off the conditional independence structure by joining the edges of all clique factors which have non-zero parameter vector. Since multiple cliques will have non-empty intersection, the resulting edge set could be rather dense. A study of more sophisticated group regularizers which account for the overlap between groups can be found in [13].

Various structure estimation techniques rely on the above ideas [203, 144, 189, 240]. Assuming that the true parameter vector indeed contains several zero entries, it can be shown, that those zeros will be discovered by regularization based methods, even if the model dimension d is much larger than the number of samples [222], and even when approximate inference techniques are applied [241].

2.4 Polynomial Approximation

The last set of basics which we discuss in this chapter is devoted to polynomials. At a first glance, it seems that polynomials have not much in common with the exponential family. The connection will, however, become evident in Chapter 5, when we employ a polynomial approximation of the potential function to derive a new approximate inference algorithm. A broad treatment of polynomials, and approximation algorithms based on them can be found in [142, 185].

Definition 2.13 (Polynomial) *Let $f_{\boldsymbol{\zeta}} : \mathbb{R} \to \mathbb{R}$ be a function, parametrized by the $k+1$-dimensional vector $\boldsymbol{\zeta} \in \mathbb{R}^{k+1}$. If $f_{\boldsymbol{\zeta}}$ can be written in the form*

$$f_{\boldsymbol{\zeta}}(z) = \sum_{i=0}^{k} \boldsymbol{\zeta}_i z^i = \boldsymbol{\zeta}_0 + \boldsymbol{\zeta}_1 z + \boldsymbol{\zeta}_2 z^2 + \cdots + \boldsymbol{\zeta}_k z^k \ ,$$

then $f_{\boldsymbol{\zeta}}$ is a degree-k polynomial in z, with coefficients $\boldsymbol{\zeta}$. Polynomials are closed under scalar multiplication and addition.

The closedness under addition follows from associativity, i.e., the sum of two polynomial $f_{\boldsymbol{\zeta}}(z) + g_{\gamma}(z)$ is again a polynomial, by grouping the same powers of z together and summing their corresponding coefficients. A large class of functions has a natural representation as polynomial-like power series.

Definition 2.14 (Analytic) *Let $\boldsymbol{\zeta}_i$ be some sequence of real coefficients. A function $f : \mathbb{R} \to \mathbb{R}$ is called real analytic on $[l; u] \subset \mathbb{R}$, if, for any $z_0 \in [l; u]$, the series*

$$F(z) = \sum_{i=0}^{\infty} \boldsymbol{\zeta}_i (z - z_0)^i$$

converges pointwise and locally uniform to $f(z)$ if z is sufficiently close to z_0. That is, the distance between z and z_0 should not exceed the radius of convergence.

Alternatively, analytic functions are characterized by a Lipschitz-like property; the absolute value of their k-th derivative at any z is upper bounded by $C^{k+1}k!$, where $C > 0$ [127].

When we truncate the series expansion of an analytic function after the $(k+1)$-th term, we are left with a degree k polynomial which is a (local) approximation of f. Since polynomials might be easier to evaluate than the original function, insights about the general quality that we can achieve with polynomial approximations are of particular significance.

Theorem 2.6 (Weierstrass Approximation Theorem [142]) *Let $f : [l; u] \to \mathbb{R}$ be a continuous function on the interval $[l; u]$. For every $\varepsilon > 0$, there exists a polynomial $f_{\boldsymbol{\zeta}}$ on $[l; u]$ whose distance to f uniformly bounded by ε. The absolute distance between f and its approximation, i.e., $\varepsilon(z) = |f(z) - f_{\boldsymbol{\zeta}}(z)|$, will be called approximation error of $f_{\boldsymbol{\zeta}}$ at z. Then,*

$$\forall \varepsilon > 0 : \exists f_{\boldsymbol{\zeta}} : \sup_{z \in [l;u]} |f(z) - f_{\boldsymbol{\zeta}}(z)| < \varepsilon \ .$$

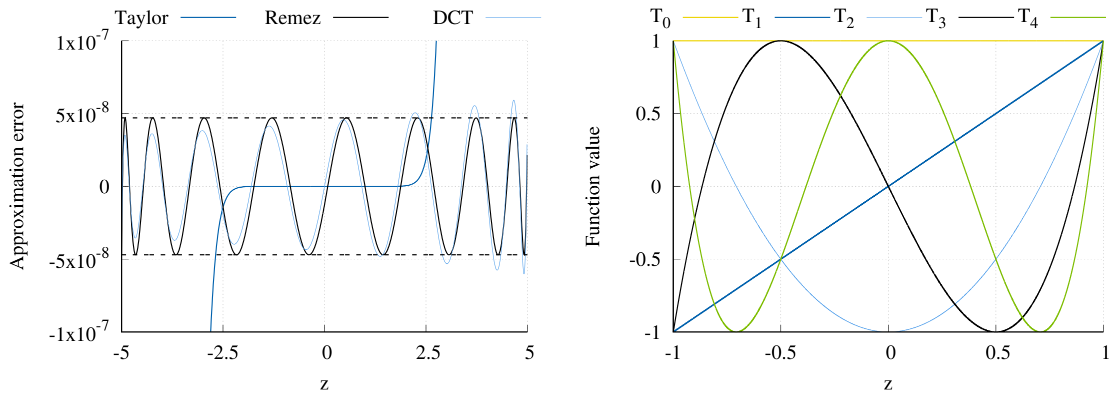

Figure 2.4: Left: Visualization of the error $f(z) - \hat{f}(z)$ of pointwise (Taylor) and uniform (DCT and Remez) approximations on the interval $[-5, 5]$. The dashed horizontal lines indicate the worst case error of the uniform approximation. The approximation with DCT coefficients does not exactly obey the equioscillation property, whereas the Remez approximation cannot be distinguished from the minimax approximation. Right: The first 5 Chebyshev polynomials.

This theorem tells us that every continuous function on $[l; u]$ can be approximated arbitrary well by a polynomial—a consequence of the more general statement that the set of all polynomials is dense in the set of real-valued continuous functions on $[l; u]$. Multivariate versions exist, but in the context of this thesis, univariate polynomials suffice.

Indeed, if we want to materialize the above theorem, we need a way to find the coefficients of a polynomial. A well known class of polynomial approximations includes the degree k truncated Taylor series. The coefficients of a Taylor expansion can be directly derived from the definition of analytic function. Investigating the i-th derivative of F at z_0, i.e., $(\partial^i F(z)/\partial z^i)\,\big|_{z=z_0}$, reveals that

$$T[f](z) = \sum_{i=0}^{\infty} \frac{1}{i!} \frac{\partial^i F}{\partial z^i} (z - z_0)^i \ .$$

Although the convergence is called pointwise and locally uniform, the compact neighborhood of z_0 in which we can guarantee a reasonable accuracy is rather small. This might be already clear from the fact, that the whole series is constructed from knowledge about f at a single point z_0. Pointwise and uniform convergence are directly related to the error of a truncated series expansion—the difference is visualized in Figure 2.4 (left). We can see, that the error of uniform approximations is evenly distributed over the whole domain, while a pointwise approximation concentrates it accuracy around $z_0 = 0$ but has unbounded error beyond a, in general unknown, neighborhood around z_0. The oscillating error shown in the figure will turn out to be an identifying property of the best (in l_∞ sense) polynomial approximation.

2.4.1 Chebyshev Polynomials

In what follows, we construct (near) minimax optimal polynomial approximations. They distinguish themselves by an oscillating approximation error. Since Chebyshev polynomials oscillate on the interval $[-1;1]$, they are a good starting point for the construction of (near) optimal approximations.

Indeed, it can be shown that an optimal (w.r.t. the l_p-norm) degree-k polynomial approximation $h_{\boldsymbol{\zeta}}$ of any function f on $[l;u]$ always exists [142].

Definition 2.15 (Minimax Approximation) *For any continuous function f from $[l;u]$ to $\mathbb{R}$, let $h_{\boldsymbol{\zeta}^*}$ be a polynomial approximation to f with coefficients*

$$\boldsymbol{\zeta}^* = \arg\min_{\boldsymbol{\zeta}\in\mathbb{R}^{k+1}} \max_{z\in[l;u]} |f(z) - h_{\boldsymbol{\zeta}}(z)| = \arg\min_{\boldsymbol{\zeta}\in\mathbb{R}^{k+1}} \|f - h_{\boldsymbol{\zeta}}\|_\infty \ .$$

$h_{\boldsymbol{\zeta}^*}$ is called minimax approximation of f.

Moreover, the minimax polynomial approximation can be identified via the extrema of it error function $\varepsilon(z)$.

Theorem 2.7 (Polynomial Approximation and Equioscillation [110]) *For any continuous function $f : [l;u] \to \mathbb{R}$, there exists a unique minimax optimal degree-k polynomial approximation $h_{\boldsymbol{\zeta}^*}$, which is uniquely characterized by the equioscillation property. That is, there are $k+2$ points in $[l;u]$ at which $f(z) - h_{\boldsymbol{\zeta}^*}(z)$ attains its maximum absolute value with alternating signs.*

The equioscillation property of the error function is both, necessary and sufficient, for $h_{\boldsymbol{\zeta}^*}$ to be unique and minimax optimal. We will now identify a class of polynomial which can in-turn be used to construct optimal approximations.

Definition 2.16 (Chebyshev Polynomial) *The degree-k Chebyshev polynomial is specified by the fundamental recurrence relation*

$$T_0(z) = 1, \quad T_1(z) = z, \quad T_k(z) = 2zT_{k-1}(z) - T_{k-2}(z) \ .$$

A non recursive form is $T_k(x) = \cos(k \arccos(x))$. Moreover, T_k oscillates on $[-1;1]$ between its $k+1$ extrema, which are the points

$$y_i = \cos\frac{i\pi}{k} \ .$$

Due to their oscillation property (Figure 2.4, right), Chebyshev polynomials are an important building block in the construction and analysis of minimax optimal approximations. The explicit, non-recursive form of Chebyshev polynomials is rather cumbersome. However, the recursion $T_k(z)$ eventually involves a summation of scaled powers of z, and can hence be rewritten in the form of an ordinary polynomial (Definition 2.13). Chebyshev polynomials have an extraordinary large variety of convenient properties, like rapidly decreasing and individually converging coefficients [73], which make them ubiquitous in virtually any field of numerical analysis. An excellent discussion of Chebyshev polynomials in general, can be found in [142].

Definition 2.17 (Chebyshev Expansion) *Let f be a function on $[-1;1]$. The Chebyshev expansion of f is the series*

$$H(z) = \frac{\gamma_0}{2}\sum_{i=1}^{\infty}\gamma_i T_i(z)$$

with

$$\gamma_i = \frac{2}{\pi}\int_{-1}^{1}(1-z^2)^{-\frac{1}{2}}f(z)T_i(z)\,\mathrm{d}\,z\ . \tag{2.42}$$

Truncating the above series after the first $k+1$ terms is called degree-k Chebyshev approximation with exact coefficients. Note that since polynomials are closed under scalar multiplication and addition (cf. Definition 2.13), degree-k Chebyshev approximations are itself polynomials of degree-k. There are several functions for which the exact coefficients γ_i in (2.42) may be determined explicitly, although this is not possible in general due to the integration. It is, however, always possible to perform a numerical approximation of the integral in (2.42). If we define $z = \cos\omega$, it follows that

$$\gamma_i = \frac{1}{\pi}\int_{0}^{2\pi} f(\cos\omega)cos(i\omega)\,\mathrm{d}\,\omega\ ,$$

since the integrand is even and has period 2π. The latter integral may be approximated by the trapezoidal rule on a grid of $2k+1$ equally spaced points

$$\omega_j = \frac{(j-(1/2))\pi}{k},\quad j\in\{1,2,\ldots,2k+1\} \tag{2.43}$$

which are the zeros of T_k. Some calculation [142] then yields the coefficients of a degree-k Chebyshev approximation[6].

Definition 2.18 (Chebyshev Approximation) *Let f be a function on $[-1;1]$. The degree-k Chebyshev approximation of f is the polynomial*

$$h_\zeta(z) = \frac{\zeta_0}{2}\sum_{i=1}^{k}\zeta_i T_i(z)$$

with

$$\zeta_i = \frac{2}{k+1}\sum_{j=1}^{k+1} f(z_j)T_i(z_j),\quad i\in\{0,1,2,\ldots,k\} \tag{2.44}$$

and $z_j = \cos\omega_j$ as specified in (2.43).

[6]The exact same coefficients may be derived by equating the function values $f(z_j)$ with $\sum_{i=0}^{k}\zeta_i T_i(z_j)$, multiplying by $(2/(k+1))T_l(z_j)$, and summing over j. Due to discrete orthogonality properties of Chebyshev polynomials, we have $\sum_{j=1}^{k} T_i(z_j)T_l(z_j) = 0 \Leftrightarrow i \neq l$, which leads directly to the coefficients (2.44). Nevertheless, we preferred to emphasize the connection between the approximate coefficients and the exact coefficients of the Chebyshev expansion.

In fact, $h_{\boldsymbol{\zeta}}(z)$ reproduces f exactly at the zeros z_j and is hence also termed the Chebyshev interpolation polynomial. The procedure that computes the approximate coefficients (2.44) is also known as discrete cosine transformation (DCT).

We can now relate the above approximation to the minimax optimal one.

Definition 2.19 (Near-Minimax Approximation) *For any continuous function $f : [l; u] \to \mathbb{R}$, $h_{\boldsymbol{\zeta}^*}$ its minimax approximation and $h_{\boldsymbol{\zeta}}$ another polynomial approximation to f. $h_{\boldsymbol{\zeta}}$ is said to be near-minimax with relative distance $\rho > 0$, if*

$$\|f - h_{\boldsymbol{\zeta}}\|_\infty \leq (1 + \rho)\|f - h_{\boldsymbol{\zeta}^*}\|_\infty \, .$$

The approximation with coefficients (2.44) can be shown to be near-minimax with ρ in $\mathcal{O}(\log k)$ [178]. In practice, the truncated Chebyshev series with exact coefficients (2.42) and the Chebyshev interpolation polynomial with approximate coefficients are virtually identical and to all intents and purposes interchangeable, as long as f is sufficiently smooth [142].

Moreover, upper bounds on the error can be given, if we incorporate knowledge about the function that we like to approximate.

Theorem 2.8 (Polynomial Approximation Error [206, 237]) *Let $f : [-1; 1] \to \mathbb{R}$ be a $K + 1$ times differentiable, absolutely continuous function, with bounded K-th derivative $\|\frac{\partial^K f}{\partial z^K}\|_T \leq V_K < \infty$, where the bound is w.r.t. the norm*

$$\|g\|_T = \int_{-1}^{1} |\frac{\partial g(z)}{\partial z}| / \sqrt{1 - z^2} \, \mathrm{d}\, z \, .$$

The error ε of a the degree-k Chebyshev approximation (Definition 2.18) with $k \geq K+1$ is bounded via

$$\|f - h_{\boldsymbol{\zeta}}\|_\infty \leq \varepsilon = \frac{4V_K}{K\pi k!} \, .$$

Application Note

The coefficients $\boldsymbol{\zeta}$, which are coefficients of Chebyshev polynomials, can be converted to coefficients $\tilde{\boldsymbol{\zeta}}$ in terms of powers of z, by collecting terms and summing the corresponding coefficients. Every Chebyshev interpolation can thus be equivalently expressed as

$$h_{\tilde{\boldsymbol{\zeta}}}(z) = \sum_{i=0}^{k} \tilde{\zeta}_i z^i \, .$$

Finally, recall that we defined the Chebyshev approximation for functions over $[-1; 1]$. Nevertheless, the bijection $\Pi_{[l;u]\to[a;b]} : [l; u] \to [a; b]$ can be composed with any function, to project its domain from any interval $[l; u]$ to any other interval $[a; b]$.

$$\Pi_{[l;u]\to[a;b]}(z) = \frac{z - l}{u - l}(b - a) + a \tag{2.45}$$

Hence, when we want to approximate the function f with domain $[l; u]$, we will instead approximate $f \circ \Pi_{[-1;1]\to[l;u]}$ which is a function with domain $[-1; 1]$. Since the resulting approximation $h_{\boldsymbol{\zeta}}$ has domain $[-1; 1]$, the final approximation of f is the composition $h_{\boldsymbol{\zeta}} \circ \Pi_{[l;u]\to[-1;1]}$.

2.4.2 Remez Algorithm

An alternative, more algorithmic, method to estimate the coefficients of the minimax approximation, based on numerical optimization, is the Remez exchange algorithm [69]. Some nodes z_i are selected, at which the polynomial approximation should exactly match the target function. These nodes and the corresponding polynomial coefficients $\boldsymbol{\zeta}_i$ are then iteratively adjusted. Moreover, the procedure outputs an estimate of the worst-case error of the resulting approximation. The pseudocode is given is shown Algorithm 2.5. Based on the definition of minimax optimality (Definition 2.15), the algorithm constructs a polynomial that achieves error $\pm\varepsilon$ at the nodes z_i (line 6). However, the absolute error might be larger in the neighborhood of each z_i, and the nodes are hence moved towards the next local optimum (line 8). It is important to understand, that we choose the nodes that maximize the error of the polynomial approximation. This step can be carried out via numerical optimization.

The procedure is reminiscent of active learning [44], but instead of adding new labeled examples, the number of nodes is fixed. This is required, to ensure, that the system of linear equations (line 6) that we solve to get the updated polynomial coefficients, has a unique solution. The number of nodes is implied by the desired degree k of the resulting polynomial. Termination of the Remez algorithm occurs, when the minimum error, achieved at any node, is close to the maximum error. This indicates that the approximation is close to uniform, and hence near-minimax optimal.

In terms of resource consumption, (2.44) is far easier to evaluate than Algorithm 2.5, since DCT/FFT has a time complexity of $\mathcal{O}(k \log k)$, and requires no storage beside the coefficients themselves. To get an idea about the qualitative difference between both methods, the corresponding approximations are shown in Figure 2.4 (left). Clearly, both error functions oscillate, but the extrema of the DCT approximation are not exactly equal, while the Remez approximation is minimax w.r.t. to 64 bit floating-point precision.

Algorithm 2.5: Remez Exchange Algorithm

```
input Twice differentiable function $f$, degree $k$, interval $[l; u]$, threshold $\tau$.
output Coefficients $\boldsymbol{\zeta}$ and error estimate $\varepsilon_{\max}$.
 1: for $i = 1$ to $k+2$ do
 2:     $\omega_i \leftarrow (i - (1/2))\pi/k$
 3:     $z_i \leftarrow \cos \omega_i$
 4: end for
 5: repeat
 6:     solve $\begin{pmatrix} 1 & z_1 & z_1^2 & \dots & z_1^k & (-1)^0 \\ 1 & z_2 & z_2^2 & \dots & z_2^k & (-1)^1 \\ \dots & & & & & \dots \\ 1 & z_{k+2} & z_{k+2}^2 & \dots & z_{k+2}^k & (-1)^{k+1} \end{pmatrix} \begin{pmatrix} \zeta_0 \\ \zeta_1 \\ \dots \\ \zeta_k \\ \varepsilon \end{pmatrix} = \begin{pmatrix} f(z_1) \\ f(z_2) \\ \dots \\ f(z_{k+2}) \end{pmatrix}$
 7:     for $i = 1$ to $k+2$ do
 8:         move $z_i$ to the next local optimum of $h_{\boldsymbol{\zeta}}(z) - f(z)$,
            depending on $\operatorname{sgn}(\varepsilon(-1)^i)$
 9:     end for
10:     $\varepsilon_{\min} \leftarrow \min_i |h_{\boldsymbol{\zeta}}(z_i) - f(z_i)|$
11:     $\varepsilon_{\max} \leftarrow \max |h_{\boldsymbol{\zeta}}(z_i) - f(z_i)|$
12: until $\varepsilon_{\max} - \varepsilon_{\min} < \tau$
```

3 Memory Constraints

Variables, parameters, data and any miscellaneous piece of information that is required to perform probabilistic inference, parameter or structure estimation have to be stored in memory. Reading and writing data to random-access memory requires electric power to charge or discharge specific parts of a memory chip. In the most common type—dynamic random-access memory—deposited charge leaks across the parts of the chip, and disappears. In order to store data for anything other than a very short time, every cell must be periodically read and then re-written, a process known as refresh [115]. Refresh has to happen multiple times per second, and is thus the major resource consuming factor of DRAM. Alternative technologies, e.g., static random-access memory (SRAM) or ferroelectric random-access memory (FRAM) remove the need to refresh cells in exchange for other drawbacks. SRAM cells require more transistors per bit, compared to DRAM cells and, at higher frequencies, SRAM can consume as much power as DRAM. In addition to its reduced power consumption, FRAM is non-volatile—when the power is lost, the data in its cells is kept intact. However, FRAM chips are subject to storage capacity limitations, since the ferroelectric effects tend to disappear when the size of an FRAM cell is above a certain threshold. Moreover, reading data from FRAM is destructive, i.e., bits have to be rewritten directly after reading, which consumes additional power. The total power consumption of FRAM is, however, still many times lower compared to DRAM.

We conclude that any system which is either subject to physical size constraints, or strong constraints regarding its power supply, cannot have a large amount of main memory. In Chapter 6, we present results on ultra-low-power devices which offer 8 kB SRAM for intermediate variables, and up to 256 kB FRAM for program code and data. Having in mind that the memory consumption has an impact on the physical size and power consumption of the resource-constrained system which should host the model, we identify sources of memory consumption in exponential family models. Existing approaches to reduce the memory requirements are discussed, and our own class of approaches for the exploitation of redundancies in the parameters are presented.

We will now investigate the memory requirements of exponential family models. When we measure the memory consumption, we assume that elements of $\mathbb{R}$ and $\mathbb{N}$ are atomic and have constant size. This is indeed true when we assume the usage of primitive data types (e.g., 8 bit integers, 32 bit floats, etc.). In case of multi-precision types which allow the representation of numbers with arbitrary precision, one has to keep track of how much memory each variable consumes. Here, we do not consider implementation specific memory requirements for pointers, iteration counters, or overhead of dynamic data structures. Recalling the definitions from Chapter 2, evaluating the potential function $\psi(\boldsymbol{x}) = \exp(\langle \boldsymbol{\theta}, \phi(\boldsymbol{x}) \rangle)$ of an exponential family member requires in general

- n elements to represent the instance $\boldsymbol{x}$,
- d elements for the intermediate representation of $\phi(\boldsymbol{x})$ in general, or only the $|\overline{\mathcal{C}}(G)|$ indices of the 1-entries whenever $\boldsymbol{X}$ is discrete,
- $|\overline{\mathcal{C}}(G)|$ elements to represent the domain-sizes of the clique factors,
- d elements to store the parameter vector $\boldsymbol{\theta}$,
- and one element for the function value itself.

In case of discrete variables, we also store the state space sizes $|\mathcal{X}_v|$ of each variable—n values in total. By denoting the memory consumption of a function f by $\mathbf{M}_{\mathrm{f}}$, evaluating ψ with binary overcomplete ϕ at $\boldsymbol{x}$ requires to store at least

$$\mathbf{M}_{\psi} = 1 + 2n + d + 2|\overline{\mathcal{C}}(G)|$$

elements. For example, a 10×10 grid model with binary variables and 4 weights per edge, assuming that each element corresponds either to a 32 bit integer or 32 bit floating-point number, requires

$$\mathbf{M}_{\psi} \times 4 \text{ Byte} = (2 \times 100 + 2 \times 180 + 720 + 1) \times 4 \text{ Byte} = 5.124 \text{ kB}.$$

Indeed, evaluating $\psi(\boldsymbol{x})$ is far from tasks like probabilistic inference or parameter estimation. Nonetheless, $\mathbf{M}_{\psi}$ can serve as a lower bound for the memory complexity of more sophisticated tasks, especially when we want to asses the compatibility of a model with a specific system.

Regarding the probabilistic inference, we provide lower bounds on the memory consumption of various methods in Table 3.1. In almost all cases, the memory required to compute $\hat{\boldsymbol{\mu}}$ is the same as for the partition function, plus the additional storage for the actual marginals. Exceptions are naive mean field, which outputs only vertex marginals (cf. Section 2.2.2) and the junction tree method, which is assumed to output the marginals of the junction tree vertices. Those may actually be used to compute marginals of any vertex set via an additional marginalization step. In case of MAP estimation, the memory consumption of Gibbs sampling is an outlier, as all samples have to be stored to determine which one appeared most often. The additional $|E|$-term in the TRW complexities stems from the edge appearance probabilities, required by TRW-BP.

Let us now consider regularized parameter estimation and structure estimation via (proximal) first-order methods. Those require to store at least the gradient, the stepsize, and the optional regularization weight λ, i.e., $\mathbf{M}_{\mathrm{GD}} = d + 2$ (GD $\equiv$ gradient descent). Accelerated first-order methods, as defined in (2.28), require two additional vectors of size d and the extrapolation weight β, hence, $\mathbf{M}_{\mathrm{AGD}} = \mathbf{M}_{\mathrm{GD}} + 2d + 1$ (AGD $\equiv$ accelerated gradient descent). If these memory requirements exceed the memory that is available in our system, we may resort to randomized coordinate descent methods (RCDM) [155]. Those require just a single (random) component of the gradient at each iteration, and achieve an ϵ distance to the optimum after $\mathcal{O}(d/\epsilon)$ iterations, which is slower than

Table 3.1: Memory requirements of inference methods and tasks.

Method	Z **(Partition)**	$\hat{\boldsymbol{\mu}}$ **(Marginals)**	$\boldsymbol{x}^*$ **(MAP)**
BP	$(1+\lvert\mathcal{N}_{\max}\rvert)\lvert\mathcal{X}_{\max}\rvert$	$d+\mathbf{M}_{\text{BP[Z]}}$	$n+\mathbf{M}_{\text{BP[Z]}}$
LBP	$2\sum_{\{v,u\}\in E}\lvert\mathcal{X}_v\rvert+\lvert\mathcal{X}_u\rvert$	$d+\mathbf{M}_{\text{LBP[Z]}}$	$n+\mathbf{M}_{\text{LBP[Z]}}$
JT	$(1+\lvert\mathcal{N}^J_{\max}\rvert)\lvert\mathcal{X}^J_{\max}\rvert$	$\mathbf{M}_{\text{JT[Z]}}+\sum_{v\in V(J)}\lvert\mathcal{X}_v\rvert$	$n+\mathbf{M}_{\text{JT[Z]}}$
Naive-MF	$\sum_{v\in V(J)}\lvert\mathcal{X}_v\rvert$	$\mathbf{M}_{\text{Naive-MF[Z]}}$	$n+\mathbf{M}_{\text{Naive-MF[Z]}}$
TRW-BP	$\lvert E\rvert+\mathbf{M}_{\text{BP[Z]}}$	$d+\lvert E\rvert+\mathbf{M}_{\text{BP[Z]}}$	$n+\lvert E\rvert+\mathbf{M}_{\text{BP[Z]}}$
GIBBS	$1+\mathbf{M}_{\text{RNG}}$	$d+\mathbf{M}_{\text{RNG}}$	$\lvert\mathcal{S}\rvert n+\mathbf{M}_{\text{RNG}}$

gradient descent but has the same dependence on ϵ. Hence, such methods offer an alternative, whenever a system cannot store multiple d-dimensional vectors.

Assuming that no closed form solution can be found, the minimal memory requirements to train an exponential family model is therefore

$$\mathbf{M}_{\text{TRAIN}} = \mathbf{M}_{\psi} + \mathbf{M}_{\text{GIBBS}[\hat{\boldsymbol{\mu}}_\text{i}]} + \mathbf{M}_{\text{RCDM}} = \mathcal{O}(\mathbf{M}_{\psi}) \ .$$

$\mathbf{M}_{\text{GIBBS}[\hat{\boldsymbol{\mu}}_\text{i}]} = 1 + \mathbf{M}_{\text{RNG}}$ is the memory required to estimate a single entry of $\hat{\boldsymbol{\mu}}_i = \mathbb{E}[\phi(\boldsymbol{X})_i]$ via Gibbs sampling, where $\mathbf{M}_{\text{RNG}}$ is the memory consumption of a random number generator. $\mathbf{M}_{\text{TRAIN}}$ is thus roughly $\mathbf{M}_{\psi}$ whenever $\mathbf{M}_{\text{RNG}}$ is small, which strengthens our intuition that $\mathbf{M}_{\psi}$ is a meaningful lower bound on the memory consumption of exponential family models.

3.1 Sufficiency

In the general machine learning setting, some data has to be stored before we start to train our model. In case of large data sets, stochastic gradient descent (SGD) methods update the parameter vector after a single data point (or a so-called mini batch) has been read. The data can be discarded after the update and the next batch is constructed to compute the next update. SGD methods are known to work well in practice, and, moreover, some guarantees about their convergence rate can be made [27]. Nevertheless, SGD and other mini batch methods are motivated by the fact that the computation time complexity for evaluating the gradient depends on the number of examples—an intuition that does not hold in case of the (regularized) maximum likelihood loss of generative exponential families. From Section 2.3.1, we know that the gradient of exponential family models is the difference between the empirical expectation and the expectation computed from the model. The complexity for computing the gradient is governed by the complexity of computing $\mathbb{E}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})]$ via probabilistic inference, since the cost for computing $\tilde{\boldsymbol{\mu}} = \frac{1}{|\mathcal{D}|}\sum_{i=1}^{d}\phi(\boldsymbol{X})$ is neglectable.

We will now see, that the ability to learn the model from aggregated data is a unique property of the exponential family. More specifically, any model that does not belong to an exponential family must have unbounded memory requirements as the number of data points tends to infinity.

Theorem 3.1 (Existence of Sufficient Statistics [175, 129]) *Let $\boldsymbol{X}$ random variable, n-dimensional, and with continuously differentiable density $p_{\boldsymbol{\theta}}$. Let further $\mathcal{D}$ be a data set with N samples $\boldsymbol{x}^i$ from $\boldsymbol{X}$. There is a function $\phi : \mathcal{X} \to \mathbb{R}^d$ with $\Phi(\mathcal{D}) = \sum_{i=1}^N \phi(\boldsymbol{x}^i)$ whose finite dimension d is independent of N. Φ is sufficient for $\boldsymbol{\theta}$, i.e., $p(\boldsymbol{\theta} \mid \mathcal{D}, \Phi(\mathcal{D})) = p(\boldsymbol{\theta} \mid \Phi(\mathcal{D}))$, if and only if $p_{\boldsymbol{\theta}}$ belongs to an exponential family of densities.*

Proof. ($\Leftarrow$) Recall that the data set enters the likelihood of an exponential family member (2.18) solely in form of $\tilde{\boldsymbol{\mu}} = \frac{1}{N}\sum_{i=1}^N \phi(\boldsymbol{x}^i) = \frac{1}{N}\Phi(\mathcal{D})$. In other words, knowledge of $\tilde{\boldsymbol{\mu}}$ is enough to estimate $\boldsymbol{\theta}$. Recall that the scalar $1/N$ was chosen just for notational convenience. Hence, $N\tilde{\boldsymbol{\mu}}$ and thus $\Phi(\mathcal{D})$ must be sufficient[7] for $\boldsymbol{\theta}$.

($\Rightarrow$) Now, suppose $p_{\boldsymbol{\theta}}$ is an arbitrary parametrized probability density. Assume that $\Phi(\mathcal{D}) = \sum_{i=1}^N \phi(\boldsymbol{x}^i)$ is sufficient for $\boldsymbol{\theta}$, with a finite dimension d that does not depend on N. For the log-likelihood $\ell(\boldsymbol{\theta}; \mathcal{D}) = \sum_{i=1}^N \log p_{\boldsymbol{\theta}}(\boldsymbol{x}^i)$, we have

$$\nabla \ell(\boldsymbol{\theta}; \mathcal{D}) = \left(\frac{\partial \ell(\boldsymbol{\theta}; \mathcal{D})}{\partial \boldsymbol{\theta}^\top}\right)^\top = \sum_{i=1}^N \frac{\nabla p_{\boldsymbol{\theta}}(\boldsymbol{x}^i)}{p_{\boldsymbol{\theta}}(\boldsymbol{x}^i)} \,. \tag{3.1}$$

$\Phi(\mathcal{D})$ is sufficient for $\boldsymbol{\theta}$, and thus $\nabla \ell(\boldsymbol{\theta}; \mathcal{D})$ can be rewritten as a function $H(\Phi(\mathcal{D}), \boldsymbol{\theta})$ that does depend on $\mathcal{D}$ only through $\Phi(\mathcal{D})$.

Now, consider a second data set $\mathcal{A}$. From $\Phi(\mathcal{D} \cup \mathcal{A}) = \Phi(\mathcal{D}) + \Phi(\mathcal{A})$, we have

$$H(\Phi(\mathcal{D}) + \Phi(\mathcal{A}), \boldsymbol{\theta}) = \nabla \ell(\boldsymbol{\theta}; \mathcal{D} \cup \mathcal{A}) = \nabla \ell(\boldsymbol{\theta}; \mathcal{D}) + \nabla \ell(\boldsymbol{\theta}; \mathcal{A}) = H(\Phi(\mathcal{D}), \boldsymbol{\theta}) + H(\Phi(\mathcal{A}), \boldsymbol{\theta})$$

which implies that H is additive in its first argument. Differentiating H w.r.t. $\boldsymbol{x}^i$ and applying the chain rule, yields

$$\frac{\partial^2 \ell(\boldsymbol{\theta}; \mathcal{D})}{\partial \boldsymbol{\theta} \partial \boldsymbol{x}^{i\top}} = \frac{\partial H(\Phi(\mathcal{D}), \boldsymbol{\theta})}{\partial \boldsymbol{x}^{i\top}} = \left(\frac{\partial \Phi(\mathcal{D})}{\partial \boldsymbol{x}^{i\top}}\right)^\top \left(\frac{\partial H(\Phi(\mathcal{D}), \boldsymbol{\theta})}{\partial \Phi(\mathcal{D})^\top}\right)^\top \,.$$

It follows from (3.1) that the left-hand side is a function of $\boldsymbol{x}^i$ and $\boldsymbol{\theta}$. By definition, $(\partial \Phi(\mathcal{D}) / \partial \boldsymbol{x}^{i\top}) = (\partial \phi(\boldsymbol{x}^i) / \partial \boldsymbol{x}^{i\top})$ is a function of $\boldsymbol{x}^i$. Since H is additive in $\Phi(\mathcal{D})$, $\partial H(\Phi(\mathcal{D}), \boldsymbol{\theta}) / \partial \Phi(\mathcal{D})^\top = h(\boldsymbol{\theta})$ must be a matrix-valued function h, of $\boldsymbol{\theta}$ only[8]. Integrating $\partial H(\Phi(\mathcal{D}), \boldsymbol{\theta}) / \partial \Phi(\mathcal{D})^\top$ w.r.t. $\Phi(\mathcal{D})$, we obtain

$$\nabla \ell(\boldsymbol{\theta}; \mathcal{D}) = H(\Phi(\mathcal{D}), \boldsymbol{\theta}) = \int \frac{\partial H(\Phi(\mathcal{D}), \boldsymbol{\theta})}{\partial \Phi(\mathcal{D})^\top} \,\mathrm{d}\,\Phi(\mathcal{D}) = h(\boldsymbol{\theta})\Phi(\mathcal{D}) + C(\boldsymbol{\theta}) \,,$$

[7] While Φ is the actual sufficient statistic that aggregates the data set $\mathcal{D}$, the function ϕ extracts information from a single instance $\boldsymbol{x}$. Since $\phi(\boldsymbol{x}) = \Phi(\{\boldsymbol{x}\})$, both functions are usually called sufficient statistic.

[8] To see this, recall that a partial derivative is defined via $\partial f(\boldsymbol{a}) / \partial \boldsymbol{a}_i = \lim_{h \to 0} f(\boldsymbol{a} + \boldsymbol{e}_i h) - f(\boldsymbol{a})/h$, where $\boldsymbol{e}_i$ is the i-th unit vector. Now, if f is additive, we have $\partial f(\boldsymbol{a}) / \partial \boldsymbol{a}_i = \lim_{h \to 0} f(\boldsymbol{e}_i h)/h$, which is independent of $\boldsymbol{a}$. Hence, the gradient of an additive function must be a constant w.r.t. to the variable to which we are differentiating.

where the constant of integration might be a function of $\boldsymbol{\theta}$. From (3.1) and Φ,

$$\sum_{i=1}^{N} \nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{x}^i) = \sum_{i=1}^{N} h(\boldsymbol{\theta})\phi(\boldsymbol{x}^i) + C(\boldsymbol{\theta})$$
$$\Leftrightarrow \nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{x}^i) = h(\boldsymbol{\theta})\phi(\boldsymbol{x}^i) + C(\boldsymbol{\theta}) \ .$$

We have an equivalence between the last two lines, because the order of our data points was arbitrary. Both summations in the first line must be equal for all possible permutations of the data points, which implies the second line. Finally, integrating w.r.t. $\boldsymbol{\theta}$, we arrive at

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}^i) = \langle h(\boldsymbol{\theta})^\top \boldsymbol{\theta}, \phi(\boldsymbol{x}^i)\rangle + \langle \boldsymbol{\theta}, C(\boldsymbol{\theta})\rangle + C_0(\boldsymbol{x}^i)$$
$$\Leftrightarrow p_{\boldsymbol{\theta}}(\boldsymbol{x}^i) = b(\boldsymbol{x}^i)\exp(\langle \eta(\boldsymbol{\theta}), \phi(\boldsymbol{x}^i)\rangle - A(\boldsymbol{\theta}))$$

with $\eta(\boldsymbol{\theta}) = h(\boldsymbol{\theta})^\top \boldsymbol{\theta}$, $b(\boldsymbol{x}^i) = \exp(C_0(\boldsymbol{x}))$, and $A(\boldsymbol{\theta}) = -\langle \boldsymbol{\theta}, C(\boldsymbol{\theta})\rangle$. Thus, $p_{\boldsymbol{\theta}}$ belongs to an exponential family of densities. ■

In a nutshell, the existence of a sufficient statistic implies that the underlying probability density belongs to an exponential family, and vice versa. The proof is based on a proof by Pitman [175], but we provide a shorter way of proving that the gradient is additive, which simplifies the proof. Sometimes, exponential family members are presented with the extra factor $b(\boldsymbol{x})$ that arises from the constant of integration $C_0(\boldsymbol{x})$. We suppress it in this thesis, since it can be absorbed into $\langle \boldsymbol{\theta}, \phi(\boldsymbol{x})\rangle$.

The function η plays a central role in the upcoming Sections—it allows us to unpack the actual parameter from a presumably compressed representation. In the case $\eta(\boldsymbol{\theta}) = \boldsymbol{\theta}$, $\boldsymbol{\theta}$ is called the natural parameter of the exponential family member.

Theorem 3.1 asserts that we can learn exponential family models from arbitrary large amounts of data, while requiring a constant amount of memory. Moreover, only exponential family members have this property. This is especially relevant in the context of memory constrained systems, when we assume that new data arrives on a regular basis.

3.2 Reparametrization

The memory complexity depends crucially on the model dimension d. Nevertheless, the dimension of the actual parameters $\boldsymbol{\theta}$ and the dimension of the sufficient statistic $\phi(\boldsymbol{x})$ are not necessarily equal. Equipped with an appropriate reparametrization, low dimensional parameters may be projected to the d-dimensional space.

Definition 3.1 (Reparametrization) *Let $\boldsymbol{\theta} \in \mathbb{R}^d$ be the natural parameters of an exponential family member. The vector $\boldsymbol{\Delta} \in \mathbb{R}^k$ is called reparametrization of $\boldsymbol{\theta}$ w.r.t. $\boldsymbol{\eta} : \mathbb{R}^k \to \mathbb{R}^d$, if $\boldsymbol{\eta}(\boldsymbol{\Delta}) = \boldsymbol{\theta}$. If $\boldsymbol{\eta}$ is a bijection, we say that the reparametrization is universal.*

Before we discuss reparametrizations further, we take a quick look at the sufficient statistic instead. We mentioned in Section 2.4 that the sufficient statistics that we consider in this thesis have almost no resource requirements, but one may ask to change

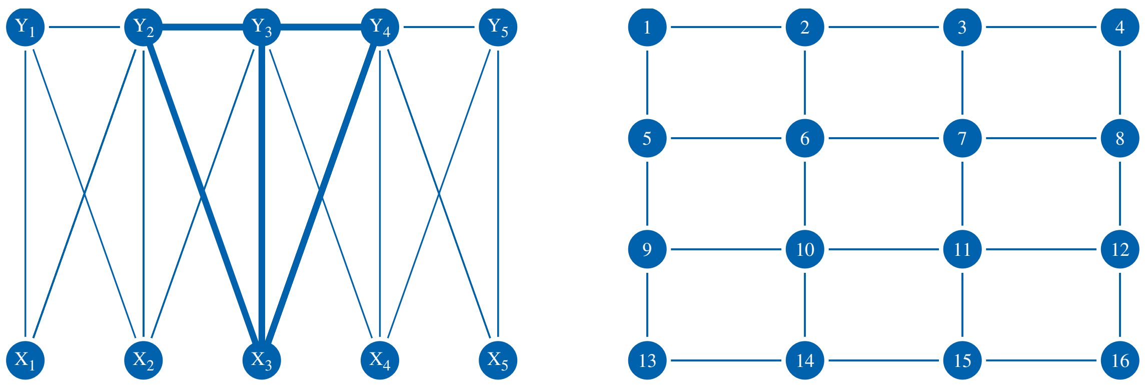

Figure 3.1: Sequence structure (left) and grid structure (right).

ϕ like a reparametrization changes $\boldsymbol{\theta}$ in order to save resources. However, modifying sufficient statistics is a more delicate process than performing a reparametrization. The reason lies in the semantics of sufficiency (Definition 2.4). Assuming that ϕ is based on the correct undirected structure, there is not much room for changes. When we drop components of $\phi(\boldsymbol{X})$, we alter the conditional independence structure and sacrifice sufficiency. Some authors try to simplify ϕ by exploiting the fact that some variables can behave alike. So-called *lifting* techniques can detect symmetries in the model. E.g., some variables v, u could have the same expected sufficient statistic, $\mathbb{E}[\phi(\boldsymbol{X})]_v = \mathbb{E}[\phi(\boldsymbol{X})]_u$. The corresponding variables may then be grouped together to lower the complexity of probabilistic inference [3]. However, such complexity reductions are artifacts of particular applications or data sets. Two variables can only be truly symmetric, when they are involved into exactly the same Markov blanket, and when they follow the same marginal density. Even if all these requirements are met, symmetries will break when the symmetric variables are conditioned on different values during probabilistic inference. The question to what extend fractional symmetries can be used to lower the resource consumption, is subject to ongoing research.

In short, the dimension of a sufficient statistic cannot be reduced without altering the conditional independence structure, which is ineligible as explained in Section 1.1.

Now, we focus on the parametrization of the model. The parameter dimension decides whether a model can run on a memory constrained system. Moreover, even on regular workstations or servers, it is desirable to have the parameters in a fast storage, e.g., cache memory, since each parameter is usually accessed multiple times during inference.

3.2.1 Parameter Tying

Let us review reparametrization techniques from natural language processing (NLP) and statistical physics. When graphical models, like hidden Markov models or conditional random fields, are applied for segmenting and labeling sequence data, each element of the sequence is represented by its own random variable and variables are connected to their direct ancestor (if any). Each variable of the sequence is moreover connected to one or more sequence features—a small sequence structure is shown in Fig. 3.1 (left).

Such structures occur frequently in probabilistic natural language processing [132, 198]. In these applications, the sequence variables ($\boldsymbol{Y}_i$ in Fig. 3.1) represent latent semantic or grammatical states, and the sequence features $\boldsymbol{X}_i$ represent words (or corresponding features) of a sentence in some natural language. Two observations are very important if one wants to apply such models: (1) The state space of each $\boldsymbol{X}_i$ is extraordinary large—it can involve several thousand states. (2) Different sentences will have different lengths. Both facts prevent a naive probabilistic model to work well. First, we would need an extra model for each possible sentence length $l \in \mathbb{N}$. Second, each of these models would have $d_l = \mathcal{O}(l|\mathcal{X}_o||\mathcal{Y}_h|^2)$ parameters, where $|\mathcal{X}_o|$ is the number of different words or word feature values, and $|\mathcal{Y}_h|$ is the number of distinct states of each sequence element. Even if we could estimate all these large sequence models separately, the models would not perform well on data which contains words in an order that never appeared in the training data. A solution to this problem is the so called parameter tying, also known as parameter sharing. The key observation is that all cliques in the model essentially represent the same predecessor/successor relation among sequence elements—visualized by the two cliques with bold edges is Fig. 3.1. So instead of the naive procedure described above, we can estimate one set of $|\mathcal{X}_o||\mathcal{Y}_h|^2$ parameters which is shared by all predecessor cliques $\{\boldsymbol{Y}_{i-1}, \boldsymbol{Y}_i, \boldsymbol{X}_i\}$, and one set of parameters that is shared by all successor cliques $\{\boldsymbol{Y}_{i+1}, \boldsymbol{Y}_i, \boldsymbol{X}_i\}$. Note that this defines a reparametrization with the new parameter vector $\boldsymbol{\Delta}$ of dimension $2|\mathcal{X}_o||\mathcal{Y}_h|^2$. The parameters of the naive model are then generated from the reparametrization $\eta(\boldsymbol{\Delta})$, which simply uses the same parameters for all predecessor cliques and the same parameters for all successor cliques.

Another instance of reparametrization can be found in the Ising model [103]. Originally a mathematical model of ferromagnetism in statistical mechanics, Ising models are a popular model in the graphical model literature, due to its simple characteristics. Ising models consists of n^2 binary variables, interconnected by a grid structure (Fig. 3.1, right). The variables represent the moments of atomic spins, and the edge weights $\boldsymbol{\theta}_{vu}$ have a physical interpretation in terms of ferromagnetism. Here, we present the edge weights in form of 2×2 matrices.

$$\boldsymbol{\theta}_{vu} = \eta(\boldsymbol{\beta}_{vu}) = \begin{pmatrix} 0, -\boldsymbol{\beta}_{vu} \\ -\boldsymbol{\beta}_{vu}, 0 \end{pmatrix} \qquad \boldsymbol{\theta}_{vu} = \eta(\beta) = \begin{pmatrix} 0, -\beta \\ -\beta, 0 \end{pmatrix} \tag{3.2}$$

In a general Ising model, each edge $\{v, u\}$ is reparametrized by a separate parameter $\boldsymbol{\beta}_{vu} \in \mathbb{R}$, which allows to control the density of full joint configurations in which neighboring vertices have the same state. If $\boldsymbol{\beta}_{vu} > 0$, joint configurations in which v and u have the same state are more likely, while if $\boldsymbol{\beta}_{vu} < 0$, the opposite holds (Note that Lemma 2.8 tells us, that $\boldsymbol{\beta}_{vu} = 0$ means that the edge is not existing at all). In a simplified Ising model, the reparametrization is unified over all edges (cf. (3.2), right). That is, a single scalar $\beta \in \mathbb{R}$ determines all edge weights via the reparametrization $\eta(\beta) \in \mathbb{R}^d$. In this case, β is also called the inverse temperature of the model [125].

Clearly, the reparametrizations in the above examples can reduce the memory requirements for the model parameters by orders of magnitude (from quadratic to constant). Moreover, less parameters imply a simplification of the estimation problem. The idea of variables at different locations sharing the same weights and detecting the same pattern in different parts of an image is also inherent in deep convolutional networks [135]—a popular machine learning technique at the time of writing this thesis.

The reparametrizations discussed so far, are motivated by specific applications. There are multiple applications for the sequence model reparametrization, but it relies on the fact that the states of the hidden variables $\boldsymbol{Y}_i$ do not depend on the actual position of a sequence elements. The reparametrization of Ising models is only meaningful, when we assume that the interaction between the variables is symmetric. As soon as $\boldsymbol{\theta}_{vu=10} \neq \boldsymbol{\theta}_{vu=01}$, $\eta(\boldsymbol{\beta})$ cannot represent the corresponding exponential family member. Convolutional nets are designed for images and sequences, too. Tasks which do not obey certain positional invariances may not be learnable by convolution layers. In summary, the major reason for why these methods will fail in certain situations is, that some natural parameters are not contained in the image of η. In other words, the “correct” model $\boldsymbol{\theta}^*$ has no preimage under η—the above reparametrizations are not universal. These observations also motivate the requirement from Section 1.1 that our extensions should not rely on properties of particular high-level applications or tasks.

Based on these insights, we present and study a universal reparametrization, which is designed for multivariate sequence data, as it is often collected by small, ubiquitous, resource-constrained systems. Typical scenarios include either measurements from a single system, taken at various spatial locations at different points in time, as well as measurements collected by multiple systems over time at fixed locations.

3.3 Multivariate Sensor Data

Typical applications of resource-constrained systems involve the collection of data from multiple sources. Instead of collecting data only once, a system may monitor a dynamic process, which comprises a series of measurements at multiple points in time.

Sensor-based monitoring and prediction has become a hot topic in a large variety of applications. According to the slogan Monitor, Mine, Manage [35], series of data from heterogeneous sources are to be put to good use in diverse applications. A view of data mining towards distributed sensor measurements is presented in [143]. There are several approaches to distributed stream mining based on work like, e.g., [235] or [186]. The goal in these approaches is a general model (or function) which is built on the basis of local models while restricting communication costs. Most often, the global model allows to answer threshold queries, but also clustering of nodes is sometimes handled. Such models are global and not designed for the prediction of measurements at a particular location. In contrast, we want to a probabilistic model for full joint realizations of all sensors at all points in time. Based on such models, we may identify unlikely situations or outliers in a stream of sensor readings. Detecting events in streams of data [140] has accordingly been modeled, e.g. in the context of monitoring hygiene in a hospital [228]. However, in our case, the monitoring does not imply certain events. We do not aim at finding patterns that define an event, although they may show up as a side effect. The analysis of mobile sensor measurements has been framed as spatio-temporal trajectory mining by, e.g., [75]. There, frequent patterns are mined from movements of pedestrians or cars. The places are not given a priori, but interesting places could be derived from frequent crossings. Thus, multivariate sensor data appears in a multitude of tasks which can be phrased in terms of exponential family models. Moreover, spatial relations are naturally expressed by graphical models. For instance, discriminative graphical models have been used for object recognition over time [56], but also generative models have been applied to video or image data [248, 99]. We known from the previous section that the number of training instances does not influence the model complexity. However, the number of parameters can exceed millions easily, which emphasizes the importance of an efficient model representation.

In the literature, approaches that aim at the reduction of model parameters are based on the identification of sparse conditional independence structures which in turn imply sparse parameter vectors. Some important directions are discussed in the following. General regularization-based methods for sparse estimation [204, 70] and approaches which are tailored for dynamic systems arose in the last decades. In time-varying dynamic Bayesian networks [194], the authors describe how to find the conditional independence structure of continuous, spatio-temporal data by performing a kernel reweighting scheme for aggregating observations across time and applying ℓ_1-regularization for sparse structure estimation. In subsequent work, it is shown how to transfer their ideas to spatio-temporal data with discrete domains [121]. The objective function that is used in the latter approach contains a regularization term for the difference of the parameter vectors of consecutive time-slices. It is therefore reminiscent of the spatio temporal reparametrization that we propose in Section 3.4. However, the estimation is performed locally for each vertex and the resulting local models are heuristically combined to arrive at a global model. It can however be shown that local procedures can indeed be enough to consistently estimate the neighborhood of each vertex [180]. Other authors show how to incorporate the non-informative Jeffreys hyperprior into the estimation procedure [236]. But the resulting sparsity cannot be controlled via a regularization weight. Their simulation results indicate, that the proposed method tends to underestimate the number of non-zero parameters.

Approaches mentioned so far assume, that a specific segmentation of the data in suitable time-slices is already available. Fearnhead [66] developed efficient dynamic programming algorithms for the computation of the posterior over the number and location of changepoints in time series. Based on this line of research, it is shown in [239] how Fearnhead's algorithms can be generalized to multidimensional time series. Specifically, the authors model the conditional independence structure using sparse, ℓ_1-regularized, Gaussian graphical models. The techniques presented therein can be used to identify the maximum a posteriori segmentation of time-series, which is required to apply any of the algorithms mentioned above.

We will now explain the construction of a general class of undirected models for spatio-temporal data. Therein, it is important to understand that the terms "spatial" and "temporal" are rather metaphoric in this context. In order to avoid any ambiguity, we call those structures *generalized sequences*. Afterwards, we show how sparsity can be induced into the corresponding models without altering the structure.

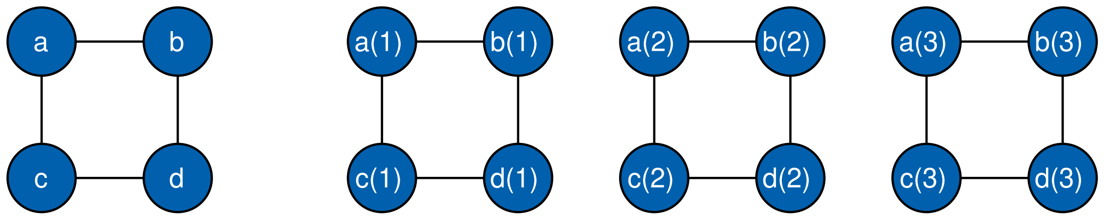

Figure 3.2: Left: Exemplary base graph G_0 with $V_0 = \{a, b, c, d\}$. Right: Exemplary generalized sequence of G_0 with $T = 3$ and $E_{\mathrm{ST}} = \emptyset$.

3.3.1 Generalized Sequence Structures

Sequence models are popular in the natural language processing community [132, 198]. There, consecutive words or corresponding word features are connected to a sequence of labels that reflects an underlying domain of interest like entities or part of speech tags. Now, consider a set of sensors $\{v_1, v_2, \dots, v_n\}$, connected to a resource-constrained system—we will identify each sensor with a random variable in a graphical model. Moreover, those sensors generate measurements over time. For simplicity, assume that the measurements are synchronized in some way; that is, all sensors are queried at arbitrary but fixed points in time $t = 1, 2, \dots, T$. Note that we ask the sequence of measurements to be finite. One may interpret this as data collected from a finite physical process, or other well defined time frame, e.g., data collected over one day or one week, etc. Since we treat the sensor measurements as realizations of a multivariate random variable, they obey, at any point in time t, some conditional independence structure $G(t)$. If the environment (e.g. sensor positions and the underlying laws of nature) do not change over time, it is reasonable to assume that the conditional independence assertions between sensors do also not change over time.

In analogy to NLP models, it is appealing to think of these so-called time-slices, like words in a sentence, to form a temporal chain $G(1) - G(2) - \dots - G(T)$ of joint measurements.

Definition 3.2 (Generalized Sequence) *Let $G_0 = (V_0, E_0)$ be the conditional independence structure and $T \in \mathbb{N}$. G_0 is called base graph. Let the graph $G = (V, E)$ be constructed from G_0, T, and edge sets $E_h \subset V_0 \times V_0$, according to $V = \{v(t) \mid v \in V_0 \wedge t \in [T]\}$ and $E = E_S \cup E_{ST}$ with*

$$\begin{aligned} E_S &= \{\{v(t), u(t)\} \mid \{v, u\} \in E_0 \wedge t \in [T]\}, \\ E_{ST} &= \{\{v(t), u(t')\} \mid v, u \in V_0 \wedge v \neq u \wedge |t - t'| = h \wedge (v, u) \in E_h \wedge t, t' \in [T]\}. \end{aligned}$$

G is called generalized sequence or spatio-temporal graph of G_0. In this context, indices $t \in [T]$ are called time-points. The set of all vertices $v(t) \in V$ that correspond to the same time-point t is called t-th layer or time-slice. Edge sets E_h contains all edges between cliques whose temporal distance is h.

The above definition allows us to construct sequences of graphs with a well-defined connectivity between sequence elements. The interpretation of V and E_{S} is rather straight-

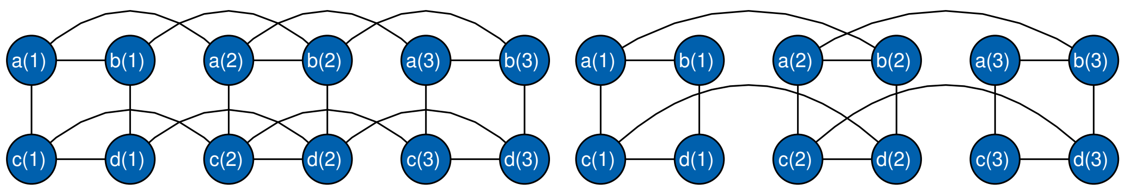

Figure 3.3: Left: Exemplary generalized sequence of G_0 (Fig. 3.2, left) with $T = 3$ and $E_1 = \{(a, a), (b, b), (c, c), (d, d)\}$. Right: Exemplary generalized sequence of G_0 (Fig. 3.2, left) with $T = 3$ and $E_1 = \{(a, b), (c, d)\}$.

forward: V contains T copies $v(t)$ of each vertex in V_0, and E_S contains T copies of each edge in E_0. To understand the role of E_ST and E_h, consider the situation depicted in Fig. 3.2. Therein, we see a small base graph with four vertices (left), and a corresponding generalized sequence with three time-points (right). Since vertices of different time-points are not connected, we have $E_\mathrm{ST} = \emptyset$. The edges contained in E_ST are controlled via the sets E_h. The pairs $(v, u) \in V_0 \times V_0$ from the set E_h act as templates for edges between vertices at different time-points t, t' with temporal distance h, i.e., $|t - t'| = h$. For example, if we want to connect the copies of each vertex with their direct temporal successors (temporal distance $h = 1$), we set $E_1 = \{(a, a), (b, b), (c, c), (d, d)\}$. The resulting generalized sequence is shown in Fig. 3.3 (left). Note, however, that the templates are not invariant w.r.t. the order of vertices. The set $E_1 = \{(a, b), (c, d)\}$ implies that there will be edges between $a(t)$ and $b(t + 1)$, but not between $b(t)$ and $a(t + 1)$ (cf. Fig. 3.3, right).

With this notation, complex spatio-temporal dependencies can be defined. Simple linear structures introduce temporal dependencies between consecutive time-points only. They combine a simple autoregressive structure with spatio-temporal neighborhoods.

Definition 3.3 (Linear Spatio-Temporal Graph) *Let $G_0 = (V_0, E_0)$ be a base graph and $T \in \mathbb{N}$. A generalized sequence with*

$$E_1 = \{(v, v) \mid v \in V_0\} \cup \{(v, u), (u, v) \mid \{v, u\} \in E_0\}$$

and $E_h = \emptyset$ for $h > 1$, is called linear generalized sequence or linear spatio-temporal graph.

Like vertices $v(t)$, the dimensions of the sufficient statistic are also accessed via their time index. More specifically,

Definition 3.4 (Sufficient Statistics of Generalized Sequences) *Let $G = (V, E)$ be a generalized sequence of length T. Let $G(t) = (\{v(t') \in V \mid t' \geq t\}, E(t))$ be the graph that contains all edges between layer t and all layers $t' \geq t$. The sufficient statistic of the t-th layer is then*

$$\phi_t(\boldsymbol{x}) = (\phi_C(\boldsymbol{x}) : \forall C \in \overline{\mathcal{C}}(G(t))) \; .$$

In accordance with Definition 2.6, the clique statistics may appear in $\phi_t(\boldsymbol{x})$ in any arbitrary but fixed order—the weights in the corresponding parameter vectors $\boldsymbol{\theta}(t)$ appear in the same order. Finally, the sufficient statistic of the model is $\phi(\boldsymbol{x}) = (\phi_t(\boldsymbol{x}) : 1 \leq t \leq T)$.

The sets of cliques per layer $\overline{\mathcal{C}}(G(1)), \overline{\mathcal{C}}(G(2)), \ldots, \overline{\mathcal{C}}(G(T))$ are non-overlapping by construction, which allows us to write

$$\langle \phi(\boldsymbol{X}), \boldsymbol{\theta} \rangle = \sum_{t=1}^{T} \langle \phi_t(\boldsymbol{X}), \boldsymbol{\theta}(t) \rangle \ .$$

The base graph G_0, the number of layers T, the edge templates $E_h, 1 \leq h < T$, and the sizes of the vertex state spaces $\mathcal{X}_v, \forall v \in V_0$ determine the number of model parameters d. In order to compute this quantity, we consider the state spaces of the clique factors.

$$d = \sum_{C \in \overline{\mathcal{C}}(G)} |\mathcal{X}_C| = \sum_{t=1}^{T} \sum_{C \in \overline{\mathcal{C}}(G(t))} |\mathcal{X}_C| \tag{3.3}$$

In case of a linear spatio-temporal graph with pairwise potentials, in which all vertices share a common state space $\mathcal{X}_v$, the above expression for the dimension reduces to

$$d = \big((T-1)(|V_0| + 3|E_0|) + |E_0|\big)|\mathcal{X}_v|^2 \ . \tag{3.4}$$

Generalized sequence models allow us to express various kinds of multivariate dynamical systems. Clearly, the conditional independence structures defined above, can be seen as an undirected version of dynamic Bayesian networks (DBN) [150]. In contrast to DBN, we explicitly allow the parameters to change over time. Moreover, the parameters of undirected models are not normalized. Reparametrization and regularization cannot be applied directly to DBNs, since these techniques can hardly maintain the normalization of conditional probability tables (which embody the parameters of a DBN). Although Bayesian networks may be rewritten as exponential family members, the approaches presented in the sequel are not well suited for directed models with normalized parameters.

3.3.2 Redundancy

Generalized sequence models enjoy a great flexibility, because at any time step t, the parameters may change arbitrarily. When we consider any clique C and fix one of its states $\boldsymbol{y} \in \mathcal{X}_c$, the corresponding parameter $\boldsymbol{\theta}_{C=\boldsymbol{y}}(t)$ may vary as a function of t (cf. Fig. 3.4). On the one hand, this allows the model to include sudden changes in the marginal density $p_{\boldsymbol{\theta}}(\boldsymbol{X}_t)$—here, $\boldsymbol{X}_t$ are the random variables associated with the vertices in layer $G(t)$. When such models underlie a natural parametrization, the number of parameters is large, since we need to store one element per clique-state-timepoint tuple as seen in (3.3). Moreover, it turns out that the natural parameters may suffer from a specific type of redundancy. To get an intuition for this, fix any clique C together

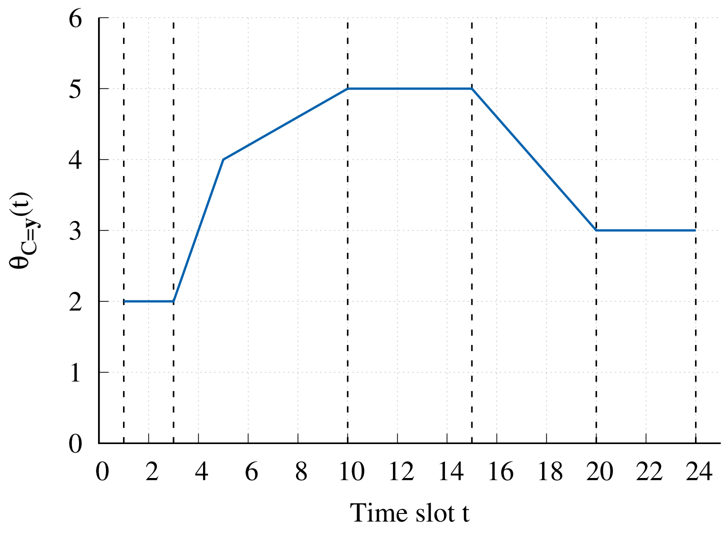

Figure 3.4: Exemplary parameter sequence $\boldsymbol{\theta}_{C=\boldsymbol{y}}(t)$ as a function of t for $T = 24$. Dashed vertical lines indicate regions in which the parameter is constant.

with a corresponding state $\boldsymbol{y} \in \mathcal{X}_C$, and some layer t. If the data is over sampled, i.e., the measurements where taken faster than the actual sensor value can change, or, if the dynamics of the underlying process are simply constant over a certain time period, the parameters for several points in time will be identical, as shown in Fig. 3.4. Therein, the intervals $\{[1,3],[10,15],[20,24]\}$, in which the parameter $\boldsymbol{\theta}_{C=\boldsymbol{y}}(t)$ is constant, are indicated by pairs of dashed lines. Fig. 3.4 is indeed an idealized scenario—actual parameter estimates will expose some variations due to finite samples, or some non-constant but deterministic behavior between certain points in time. In practice, such effects do occur at several sensors, states and time-points [169].

The redundancy structure cannot be exploited or detected via plain l_1-regularization, since all parameters are non-zero. However, if we would have known such deterministic regions before parameter estimation, we could have saved a lot of memory by proposing a reparametrization $\eta(\boldsymbol{\Delta})$ which stores a single value per constant region, and projects this value to all redundant parameters.

In what follows, we present a reparametrization for generalized sequence models which is based on this idea [169]. Moreover, a priori knowledge about the deterministic regions is not required at all. Our combined approach of reparametrization and regularization is able to detect them automatically during parameter estimation, and will produce a sparse parameter vector.

3.4 Compressible Reparametrization

The consistency of conditional independence structure estimates has been studied in the literature [250, 241, 85], but the detection of sparsity that goes beyond that setting is rather rare. Recall that if we assume that the correct structure is known, ordinary

$$\begin{pmatrix}
\begin{matrix}D(3)\text{ for }C_1=x_1\\ 3\times 3\text{ matrix}\end{matrix} & & & & & & \\
& \cdots & & & & & \\
& & \begin{matrix}D(3)\text{ for }C_1=x_4\\ 3\times 3\text{ matrix}\end{matrix} & & & & \\
& & & \cdots & & & \\
& & & & \begin{matrix}D(3)\text{ for }C_4=x_1\\ 3\times 3\text{ matrix}\end{matrix} & & \\
& & & & & \cdots & \\
& & & & & & \begin{matrix}D(3)\text{ for }C_4=x_4\\ 3\times 3\text{ matrix}\end{matrix}
\end{pmatrix}$$

Figure 3.5: Global decay matrix $\boldsymbol{D}^\circ$ for the structure shown in Fig. 3.2. We denote the edges of the corresponding base graph by C_1, C_2, C_3, C_4. Each clique (edge) exists at three time-points, which implies that each decay matrix is 3×3. We assume that all edges share a common state space $\mathcal{X}_e = \{x_1, x_2, x_3, x_4\}$. $\boldsymbol{D}^\circ$ hence consists of $4 \times 4 = 16$ block matrices on its diagonal and all other entries are 0.

regularization-based sparsity induction will change the conditional independence structure, since zeros in the parameters correspond to independences in the model. Here, we present a sparse reparametrization which keeps the structure of generalized sequences intact. Moreover, the underlying idea can be transferred to other structures as well. Motivated by the possible redundancy in the parameters of sequence models, we study reparametrizations which are piecewise linear functions of the time index.

Definition 3.5 (Piecewise Linear Reparametrization [169]) *Let G be a generalized sequence of length T, and let $\boldsymbol{D}(h) \in [0;1]^{h \times h}$ be a lower unitriangular[9] matrix. $\boldsymbol{D}(h)$ is called decay matrix. The piecewise linear reparametrization of parameters for a clique C is specified by the vector $\boldsymbol{\Delta}_{C=\boldsymbol{x}'} \in \mathbb{R}^t$ via*

$$\boldsymbol{\theta}_{C=\boldsymbol{x}'} = \eta_{\boldsymbol{D}(h)}(\boldsymbol{\Delta}_{C=\boldsymbol{x}'}) = \boldsymbol{D}(h)\boldsymbol{\Delta}_{C=\boldsymbol{x}'} \tag{3.5}$$

with $h = T - (\max\{t' \mid v(t') \in C\} - \min\{t' \mid v(t') \in C\})$. To extend the notation from a single clique to the full model, we construct a $d \times d$ block diagonal matrix $\boldsymbol{D}^\circ$ (Fig. 3.5), where each diagonal block is a copy of $\boldsymbol{D}(h)$, which corresponds to a clique factor C with state $\boldsymbol{x}'$. This allows us to write $\boldsymbol{\theta} = \boldsymbol{D}^\circ\boldsymbol{\Delta}$. $\eta_{\boldsymbol{D}}$ is called spatio-temporal reparametrization. The corresponding reparametrized exponential family members are called spatio-temporal random fields.

[9]An unitriangular matrix is triangular, and all entries on its main diagonal are 1.

The natural parameters $\boldsymbol{\theta}_{C=\boldsymbol{x}'}$ at time t for any clique C are hence given as weighted sum of some entries of $\boldsymbol{\Delta}_{C=\boldsymbol{x}'}$.

$$\boldsymbol{\theta}_{C=\boldsymbol{x}'} = \begin{bmatrix} \boldsymbol{\theta}_{C=\boldsymbol{x}'}(1) \\ \boldsymbol{\theta}_{C=\boldsymbol{x}'}(2) \\ \vdots \\ \boldsymbol{\theta}_{C=\boldsymbol{x}'}(T) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Delta}_{C=\boldsymbol{x}'}(1) \\ \boldsymbol{D}_{2,1}(h)\boldsymbol{\Delta}_{C=\boldsymbol{x}'}(1) + \boldsymbol{\Delta}_{C=\boldsymbol{x}'}(2) \\ \vdots \\ \sum_{i=1}^{T} \boldsymbol{D}_{T,i}(h)\boldsymbol{\Delta}_{C=\boldsymbol{x}'}(i) \end{bmatrix}$$

As opposed to the reparametrizations that we encountered in Section 3.2, the above can represent any natural parameter.

Lemma 3.1 (Universality of the Reparametrization) *The reparametrization provided in Definition 3.5 is universal.*

Proof. Indeed, any $\boldsymbol{\Delta} \in \mathbb{R}^d$ can be mapped to some $\boldsymbol{\theta} \in \mathbb{R}^d$ by multiplication with $\boldsymbol{D}^{\circ}$ according to Definition 3.5. To see that the converse also holds, note that for each $t \in [T]$, $\det \boldsymbol{D}(h) = \prod_{i=1}^{t} \boldsymbol{D}(h)_{i,i} = 1$, due to unitriangularity. Each $\boldsymbol{D}(h)$ is thus invertible and so is the block diagonal matrix $\boldsymbol{D}^{\circ}$. So for any given natural parameter $\boldsymbol{\theta}_{C=\boldsymbol{y}}$, we can find the corresponding reparametrization via $\boldsymbol{\Delta}_{C=\boldsymbol{y}} = \boldsymbol{D}^{-1}\boldsymbol{\theta}_{C=\boldsymbol{y}}$. That is, $\eta_{\boldsymbol{D}}$ is bijective and hence universal. ■

Since $\eta_{\boldsymbol{D}}$ is universal, any natural parameter can be represented via some $\boldsymbol{\Delta}$. Moreover, $\eta_{\boldsymbol{D}}$ is a linear function of $\boldsymbol{\Delta}$. The convexity of a function is preserved by composing it with a linear function. Hence, the reparametrized negative average log-likelihood $\ell(\eta_{\boldsymbol{D}}(\boldsymbol{\Delta}); \mathcal{D}) = A(\eta_{\boldsymbol{D}}(\boldsymbol{\Delta})) - \langle \eta_{\boldsymbol{D}}(\boldsymbol{\Delta}), \tilde{\boldsymbol{\mu}} \rangle$ is a convex function of $\boldsymbol{\Delta}$.

Up to now, we have not saved any memory since $\boldsymbol{\Delta}$ and $\boldsymbol{\theta}$ have the same dimension. By imposing l_1- and l_2-regularization on the reparametrized objective, we arrive at the problem

$$\min_{\boldsymbol{\Delta} \in \mathbb{R}^d} \underbrace{A(\eta_{\boldsymbol{D}}(\boldsymbol{\Delta})) - \langle \eta_{\boldsymbol{D}}(\boldsymbol{\Delta}), \tilde{\boldsymbol{\mu}} \rangle + \frac{\lambda_2}{2}\|\boldsymbol{\Delta}\|_2^2 + \lambda_1\|\boldsymbol{\Delta}\|_1}_{\ell^{\mathrm{ST}}(\boldsymbol{\Delta};\mathcal{D})} \ . \tag{3.6}$$

The following theorem shows that the intuition which we used to design our reparametrization (cf. Fig. 3.4) has indeed the desired effect—it allows us to convert redundancy into sparsity by detecting neglectable changes in consecutive natural parameters. Moreover, a polynomial number of samples suffices to achieve a small estimation error with high probability. In contrast to existing regularization-based approaches, solving (3.6) does not alter the conditional independence structure to find a sparse model.

Theorem 3.2 (STRF Consistency) *Consider a random variable $\boldsymbol{X}$ with exponential family density, parameter $\boldsymbol{\theta}^* \in \mathbb{R}^d$ whose reparametrization has minimal norm among all equivalent parameters, and generalized sequence structure of length T. We are given a data set $\mathcal{D}$ with $N = |\mathcal{D}|$ samples from $\boldsymbol{X}$. Suppose $\|\nabla^2 A(\boldsymbol{\theta}^*)^{-1}\|_\infty \leq \kappa$ and $\|\boldsymbol{\Delta}\|_\infty \leq \gamma$, and set $\lambda_1 = 4T\sqrt{\log(d)/N}$ and $\lambda_2 = \gamma^{-1}\lambda_1$. If $N \geq 324\kappa^4 d^{12}\log(d)/(T-d^2)^2$, then, for an arbitrary decay matrix $\boldsymbol{D}$:*

- *The distance between the true parameter $\boldsymbol{\theta}^*$ and the estimate $\eta_{\boldsymbol{D}}(\hat{\boldsymbol{\Delta}})$ is bounded, i.e.,*
$$\|\eta_{\boldsymbol{D}}(\hat{\boldsymbol{\Delta}}) - \boldsymbol{\theta}^*\|_\infty \leq 3\kappa d^2 \lambda_1 \ ,$$
- *and any sparsity in the estimate implies some redundancy in the true parameter, i.e.,*
$$\begin{aligned} &\hat{\boldsymbol{\Delta}}_{C=\boldsymbol{x}'}(t) = 0 \Rightarrow \\ |\boldsymbol{\theta}^*_{C=\boldsymbol{x}'}(t-1) - \boldsymbol{\theta}^*_{C=\boldsymbol{x}'}(t)| \ &\leq \ \frac{3d^2\kappa\lambda_1}{T} + (t-1)\left(\max_{i=1}^{t-1} |\hat{\boldsymbol{\Delta}}_{C=\boldsymbol{x}'}(i)| + \frac{3d^2\kappa\lambda_1}{T}\right) \ , \end{aligned}$$

for any clique C and time-point t. Both statements hold with probability at least $1-(2/d)$.

Proof. Since our sufficient statistic is overcomplete, there are infinitely many parameters $\boldsymbol{\theta}$ (specified via (2.21)) which are equivalent to the true parameter $\boldsymbol{\theta}^*$, i.e., $p_{\boldsymbol{\theta}} \equiv p_{\boldsymbol{\theta}^*}$. Among all these equivalent parameters, one has minimal $\frac{\lambda_2}{2}\|\eta^{-1}_{\boldsymbol{D}}(\boldsymbol{\theta})\|_2^2 + \lambda_1\|\eta^{-1}_{\boldsymbol{D}}(\boldsymbol{\theta})\|_1$ value, which we assume w.l.o.g. to be $\boldsymbol{\theta}^*$.

Due to l_2-regularization, the objective function (3.6) is strictly convex. The unique minimizer $\hat{\boldsymbol{\Delta}}$ satisfies the subgradient condition

$$\mathbf{0} \in \nabla A(\eta(\hat{\boldsymbol{\Delta}})) - \boldsymbol{D}^{\circ\top}\tilde{\boldsymbol{\mu}} + \lambda_2\hat{\boldsymbol{\Delta}} + \lambda_1\partial\|\hat{\boldsymbol{\Delta}}\|_1 \ , \tag{3.7}$$

where $\partial\|\hat{\boldsymbol{\Delta}}\|_1$ denotes the subdifferential[10] of $\|\cdot\|_1$ at $\hat{\boldsymbol{\Delta}}$. The boundaries of the subdifferential can be found via the left-sided and right-sided limits of the corresponding difference quotient. For any subgradient $\boldsymbol{G} \in \partial\|\hat{\boldsymbol{\Delta}}\|_1$, we have $\boldsymbol{G}_i = \pm 1$ if $\hat{\boldsymbol{\Delta}} \gtrless 0$, and otherwise $\boldsymbol{G}_i \in [-1;1]$, which implies $\lambda_1\langle \boldsymbol{G}, \hat{\boldsymbol{\Delta}}\rangle = \lambda_1\|\hat{\boldsymbol{\Delta}}\|_1$.

$\nabla A(\eta(\hat{\boldsymbol{\Delta}})) = \boldsymbol{D}^{\circ\top}\nabla_{\eta_{\boldsymbol{D}}(\hat{\boldsymbol{\Delta}})}A(\eta_{\boldsymbol{D}}(\hat{\boldsymbol{\Delta}}))$ denotes the gradient at $\hat{\boldsymbol{\Delta}}$, derived via the matrix chain-rule [141]. To handle the non-linearity of our objective, we consider the first-order Taylor expansion of $\nabla(A \circ \eta)$ around the true $\boldsymbol{\Delta}^* = \eta^{-1}_{\boldsymbol{D}}(\boldsymbol{\theta}^*)$, i.e.,

$$\nabla A(\eta(\boldsymbol{\Delta})) = \nabla A(\eta(\boldsymbol{\Delta}^*)) + \nabla^2 A(\eta(\boldsymbol{\Delta}^*))(\boldsymbol{\Delta} - \boldsymbol{\Delta}^*) + R_{\boldsymbol{\Delta}^*}(\boldsymbol{\Delta}) \ ,$$

where $R_{\boldsymbol{\Delta}^*}(\boldsymbol{\Delta})$ is the Lagrange remainder vector of the Taylor expansion. Its i-th element is specified by

$$\begin{aligned} R_{\boldsymbol{\Delta}^*}(\boldsymbol{\Delta})_i \ &= \ \sum_{j=1}^{d}\sum_{k=j}^{d} \frac{1}{1+\mathbb{1}_{\{j=k\}}} \frac{\partial^3}{\partial\boldsymbol{\theta}_i\partial\boldsymbol{\theta}_j\partial\boldsymbol{\theta}_k} A(\eta(\boldsymbol{\Delta}(c)))(\boldsymbol{\Delta}_j - \boldsymbol{\Delta}^*_j)(\boldsymbol{\Delta}_k - \boldsymbol{\Delta}^*_k) \\ &\leq \ \sum_{j=1}^{d}\sum_{k=j}^{d} \frac{3}{1+\mathbb{1}_{\{j=k\}}}(\boldsymbol{\Delta}_j - \boldsymbol{\Delta}^*_j)(\boldsymbol{\Delta}_k - \boldsymbol{\Delta}^*_k) = \frac{3}{2}\|\tilde{\boldsymbol{\Delta}} - \boldsymbol{\Delta}^*\|_1^2 \ , \end{aligned}$$

where $\boldsymbol{\Delta}(c) = (1-c')\boldsymbol{\Delta}^* + c'\boldsymbol{\Delta}$, and $c' \in (0,1)$ is an unknown but irrelevant constant. To simplify notation, we set $\boldsymbol{D}^{\circ\top}\boldsymbol{\mu}^* = \nabla A(\eta(\boldsymbol{\Delta}^*))$ and $\boldsymbol{Q} = \nabla^2 A(\eta(\boldsymbol{\Delta}^*))$ in the following.

[10] The subdifferential of convex functions is guaranteed to exist. More details on subdifferentials and the closure processes which define them, can be found in [149].

Now, let us define the ball $\mathcal{B}(r) = \{\boldsymbol{a} \mid \|\boldsymbol{a}\|_\infty \leq r\}$ that contains all vectors whose uniform norm is at most r. We further define the function

$$F(\boldsymbol{a}) = \boldsymbol{a} - \boldsymbol{Q}^{-1} \operatorname*{arg\,min}_{\boldsymbol{g} \in \partial \ell^{\mathrm{ST}}(\boldsymbol{\Delta}^* + \boldsymbol{a}; \mathcal{D})} \|\boldsymbol{g}\|_\infty .$$

Note that when a point $\hat{\boldsymbol{\Delta}}$ satisfies the subgradient condition (3.7), $\boldsymbol{a} = \hat{\boldsymbol{\Delta}} - \boldsymbol{\Delta}^*$ is a fixpoint of F. We show that any point $\boldsymbol{a} = \boldsymbol{\Delta} - \boldsymbol{\Delta}^* \in \mathcal{B}(r)$ is mapped into $\mathcal{B}(r)$. Let $r = 3\kappa d^2 \lambda_1 / T$, then

$$\begin{aligned}
\|F(\boldsymbol{\Delta} - \boldsymbol{\Delta}^*)\|_\infty &= \|(\boldsymbol{\Delta} - \boldsymbol{\Delta}^*) - \boldsymbol{Q}^{-1} \operatorname*{arg\,min}_{\boldsymbol{g} \in \partial \ell^{\mathrm{ST}}(\boldsymbol{\Delta}; \mathcal{D})} \|\boldsymbol{g}\|_\infty \|_\infty \\
&= \| - \boldsymbol{Q}^{-1}(R_{\boldsymbol{\Delta}^*}(\boldsymbol{\Delta}) - \boldsymbol{W} + \lambda_2 \boldsymbol{\Delta} + \lambda_1 \boldsymbol{G})\|_\infty \\
&\leq \|\boldsymbol{Q}^{-1}\|_\infty (\|R_{\boldsymbol{\Delta}^*}(\boldsymbol{\Delta})\|_\infty + \|\boldsymbol{W}\|_\infty + \lambda_2 \|\boldsymbol{\Delta}\|_\infty + \lambda_1 \|\boldsymbol{G}\|_\infty) \\
&\leq (3d^2/2)\kappa r^2 + 3\kappa\lambda_1 \ \leq \ r .
\end{aligned}$$

The last two inequalities follow from the upper bounds on $\|\boldsymbol{Q}^{-1}\|_\infty$ and $\|\boldsymbol{\Delta}\|_\infty$ as asserted by the Theorem, the upper bound on $\|\boldsymbol{W}\|_\infty$ via Lemma 2.5, as well as $\|\boldsymbol{G}\|_\infty \leq 1$ via $\boldsymbol{G} \in \partial \|\boldsymbol{\Delta}\|_1$.

Brouwer's fixpoint theorem asserts that at least one fixpoint of F is contained in $\mathcal{B}(3\kappa d^2 \lambda_1/T)$. Since the objective is strictly convex, there is only a single unique point $\hat{\boldsymbol{\Delta}}$ satisfying the zero subgradient condition $\boldsymbol{0} \in \partial \ell^{\mathrm{ST}}(\hat{\boldsymbol{\Delta}}; \mathcal{D})$. Thus, $\|\hat{\boldsymbol{\Delta}} - \boldsymbol{\Delta}^*\|_\infty \leq 3d^2\kappa\lambda_1/T$.

Based on this, we derive a bound in terms of $\boldsymbol{\theta}^*$. The matrix $\mathcal{D}^\circ$ is invertible, since it is block diagonal with unitriangular blocks. Hence,

$$\|\eta_{\boldsymbol{D}}(\hat{\boldsymbol{\Delta}}) - \boldsymbol{\theta}^*\|_\infty = \|\boldsymbol{D}^\circ(\hat{\boldsymbol{\Delta}} - \boldsymbol{\Delta}^*)\|_\infty \leq \|\boldsymbol{D}^\circ\|_\infty \|\hat{\boldsymbol{\Delta}} - \boldsymbol{\Delta}^*\|_\infty \leq 3d^2\kappa\lambda_1 ,$$

with $\|\boldsymbol{D}^\circ\|_\infty \leq T$, which proves the first part of the theorem. Moreover, $\tilde{\boldsymbol{\Delta}}_i$ and $\boldsymbol{\Delta}^*_i$ will have the same sign, whenever $\boldsymbol{\Delta}^*_i > 3d^2\kappa\lambda_1/T$.

By definition of $\eta_{\boldsymbol{D}}$,

$$\begin{aligned}
|\boldsymbol{\theta}^*_{C=\boldsymbol{x}'}(t) - \boldsymbol{\theta}^*_{C=\boldsymbol{x}'}(t-1)| &= \left| \sum_{i=1}^{t} \boldsymbol{D}_{t,i} \boldsymbol{\Delta}^*_{C=\boldsymbol{x}'}(i) - \sum_{j=1}^{t-1} \boldsymbol{D}_{t-1,j} \boldsymbol{\Delta}^*_{C=\boldsymbol{x}'}(j) \right| \\
&\leq |\boldsymbol{\Delta}^*_{C=\boldsymbol{x}'}(t)| + \left| \sum_{i=1}^{t-1} (\boldsymbol{D}_{t,i} - \boldsymbol{D}_{t-1,i}) \boldsymbol{\Delta}^*_{C=\boldsymbol{x}'}(i) \right| ,
\end{aligned}$$

for any clique C, where we suppressed the dimension of the decay matrix to simplify notation, i.e., $\boldsymbol{D} = \boldsymbol{D}(h)$. Due to $\|\hat{\boldsymbol{\Delta}} - \boldsymbol{\Delta}^*\|_\infty \leq 3d^2\kappa\lambda_1/T$, $\hat{\boldsymbol{\Delta}}_i = 0$ implies $|\boldsymbol{\Delta}^*_i| < 3d^2\kappa\lambda_1/T$. According to Hölder's inequality and the definition of decay matrices, we have

$$\begin{aligned}
\left| \sum_{i=1}^{t-1} (\boldsymbol{D}_{t,i} - \boldsymbol{D}_{t-1,i}) \boldsymbol{\Delta}^*_{C=\boldsymbol{x}'}(i) \right| \ &\leq \ (t-1) \max_{i=1}^{t-1} |\boldsymbol{\Delta}^*_{C=\boldsymbol{x}'}(i)| \\
&\leq \ (t-1)(\max_{i=1}^{t-1} |\hat{\boldsymbol{\Delta}}_{C=\boldsymbol{x}'}(i)| + 3d^2\kappa\lambda_1/T) ,
\end{aligned}$$

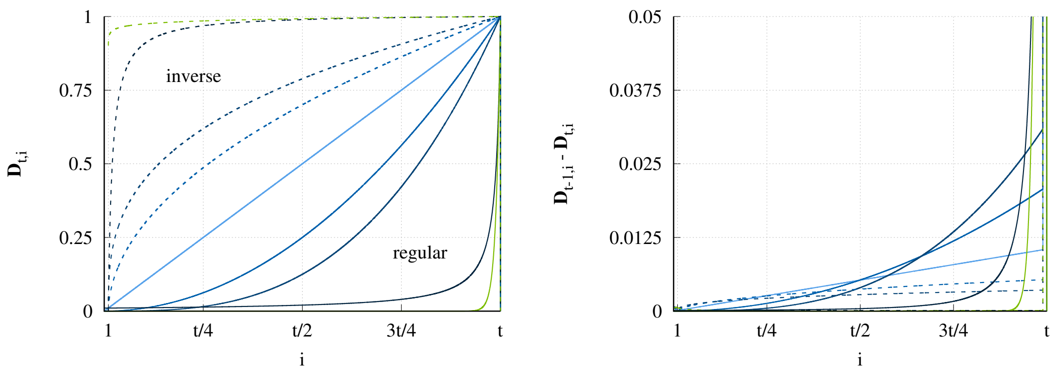

Figure 3.6: Left: Various decay types: linear, quadratic, cubic, rational, and exponential. The inverse decay types are depicted with dashed lines. Right: Differences $\boldsymbol{D}_{t-1,i} - \boldsymbol{D}_{t,i}$ for various decay types. The plot is cutted at 0.05 to improve readability.

which proves the second part of the Theorem. ■

Within the proof, we assumed that the true parameter $\boldsymbol{\theta}^*$ has smallest l_2 and l_1-norm among all equivalently optimal solutions. Note, however, that this was just to simplify the statement of the Theorem. By dropping this assumption, the statement of the theorem changes, such that the unique estimator $\eta_{\boldsymbol{D}}(\hat{\boldsymbol{\Delta}})$ has bounded distance to the unique parameter $\boldsymbol{\theta}^{**}$ that minimizes $\frac{\lambda_2}{2}\|\eta_{\boldsymbol{D}}^{-1}(\boldsymbol{\theta})\|_2^2 + \lambda_1\|\eta_{\boldsymbol{D}}^{-1}(\boldsymbol{\theta})\|_1$ among all "true" parameters.

Besides the nice properties asserted by the Theorem, our reparametrization introduces some computational overhead, due to the summation in (3.5). In particular, whenever an algorithm has to read a parameter $\boldsymbol{\theta}_{C=\boldsymbol{x}'}(t)$, it has do be decompressed instantly, which adds asymptotic runtime complexity $\mathcal{O}(t)$ to the access. However, if the available memory is small, the compressed representation $\boldsymbol{\Delta}$ could be the only way to hold the model in memory. On workstations or servers, the reparametrization can be beneficial when the compressed model fits into the CPU cache memory and the full parameter vector does not.

3.4.1 Decay Types

The statement of Theorem 3.2 is independent of the choice of the decay matrix $\boldsymbol{D}(h)$. Up to now, the only requirement on $\boldsymbol{D}(h) \in [0;1]^{h\times h}$ is its unitriangularity[11]. We now discuss some specific choices and their implications. In particular, we consider decay matrices in which the entries below the main diagonal are a non-decreasing function of the column index, that is $\boldsymbol{D}_{t,i} \leq \boldsymbol{D}_{t,i+1}$ whenever $i \leq t$ and 0 otherwise. This is implies, that the influence of each $\boldsymbol{\Delta}_{C=\boldsymbol{y}}(i)$ on any $\boldsymbol{\theta}_{C=\boldsymbol{y}}(t)$ is stronger, the smaller $t-i$ is. To put it differently, the influence decays with increasing distance.

[11]We will again suppress the dimension t when we denote the decay matrix, i.e., we write $\boldsymbol{D}$ instead of $\boldsymbol{D}(h)$.

$$\begin{array}{lllll} 1 &&&& \\ \frac{1}{2} & 1 &&& \\ \frac{1}{3} & \frac{2}{3} & 1 && \\ \frac{1}{4} & \frac{2}{4} & \frac{3}{4} & 1 & \\ \frac{1}{5} & \frac{2}{5} & \frac{3}{5} & \frac{4}{5} & 1 \end{array} \qquad \begin{array}{lllll} 1 &&&& \\ \frac{1}{4} & 1 &&& \\ \frac{1}{9} & \frac{4}{9} & 1 && \\ \frac{1}{16} & \frac{4}{16} & \frac{9}{16} & 1 & \\ \frac{1}{25} & \frac{4}{25} & \frac{9}{25} & \frac{16}{25} & 1 \end{array} \qquad \begin{array}{lllll} 1 &&&& \\ \frac{1}{8} & 1 &&& \\ \frac{1}{27} & \frac{8}{27} & 1 && \\ \frac{1}{64} & \frac{8}{64} & \frac{27}{64} & 1 & \\ \frac{1}{125} & \frac{8}{125} & \frac{27}{125} & \frac{64}{125} & 1 \end{array} \qquad \begin{array}{lllll} 1 &&&& \\ \frac{1}{2} & 1 &&& \\ \frac{1}{3} & \frac{1}{2} & 1 && \\ \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & 1 & \\ \frac{1}{5} & \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & 1 \end{array} \qquad \begin{array}{lllll} 1 &&&& \\ \frac{1}{e^1} & 1 &&& \\ \frac{1}{e^2} & \frac{1}{e^1} & 1 && \\ \frac{1}{e^3} & \frac{1}{e^2} & \frac{1}{e^1} & 1 & \\ \frac{1}{e^4} & \frac{1}{e^3} & \frac{1}{e^2} & \frac{1}{e^1} & 1 \end{array}$$

Figure 3.7: Exemplary decay matrices for $T = 5$, regular decay. From left to right: linear, quadratic, cubic, rational, and exponential. Omitted entries are zero.

Here, we consider linear $f_{\text{lin}}(x, z) = x/z$, quadratic $f_{\text{qua}}(x, z) = (x/z)^2$, cubic $f_{\text{cub}}(x, z) = (x/z)^3$, rational $f_{\text{rat}}(x, z) = 1/(z - x + 1)$, and exponential $f_{\text{exp}}(x, z) = \exp(x - z)$ decay.

Definition 3.6 (Decay Matrix Types) *Let $g_f = \Pi_{[1;T]\to[f(1,T);1]}$ be the projection from $[1; T]$ to $[f(1, T); 1]$ (2.45), and let $P[f] = g_f \circ f \circ g_f$ be the pre and post composition of a function f with g_f. For all $i \leq t$, a decay matrix $\boldsymbol{D}$ is called*

- *linear, if $\boldsymbol{D}_{t,i} = f_{lin}(i, t)$,*
- *quadratic, if $\boldsymbol{D}_{t,i} = f_{qua}(i, t)$, and inverse quadratic if $\boldsymbol{D}_{t,i} = P[f_{qua}^{-1}](i, t)$,*
- *cubic, if $\boldsymbol{D}_{t,i} = f_{cub}(i, t)$, and inverse cubic if $\boldsymbol{D}_{t,i} = P[f_{cub}^{-1}](i, t)$,*
- *rational, if $\boldsymbol{D}_{t,i} = f_{rat}(i, t)$, and inverse rational if $\boldsymbol{D}_{t,i} = P[f_{rat}^{-1}](i, t)$,*
- *and exponential, if $\boldsymbol{D}_{t,i} = f_{exp}(i, t)$, and inverse exponential if $\boldsymbol{D}_{t,i} = P[f_{exp}^{-1}](i, t)$.*

The non-inverse decay types are also called regular decay types.

Plots of the decay functions are illustrated in Fig. 3.6, and exemplary decay matrices $\boldsymbol{D}(5)$ are shown in Fig. 3.7. We can see in the plot, that for inverse decays, the weights of preceding parameters $\boldsymbol{\Delta}(i)$ have more impact on $\boldsymbol{\theta}(t)$, compared to their regular counterparts. In case of regular rational and regular exponential decay, only those $\boldsymbol{\Delta}(i)$ with a small difference between i and t have a strong influence of on $\boldsymbol{\theta}(t)$. Moreover, all decay types are non-zero for $1 \leq i \leq t$, and hence, a fraction of every $\boldsymbol{\Delta}(i)$ is eventually inherited by $\boldsymbol{\theta}(t)$.

Beside these intuitive observations, the decay types have an impact on the parameter estimation and on the quality guarantees. Regarding the latter, any appearance of the factor T in Theorem 3.2 arises through the upper bounds on $\|\boldsymbol{D}^{\circ\top}\|_\infty$ and $\|\boldsymbol{D}_t - \boldsymbol{D}_{t-1}\|_\infty$, respectively. Recall that the uniform matrix-norm of a $d \times d$-matrix $\boldsymbol{M}$ is the maximal row-wise l_1-norm $\max_{i=1}^{d} \|\boldsymbol{M}_{i,\cdot}\|_1$, and that $\boldsymbol{D}^\circ$ is block diagonal whose blocks are constituted from the decay matrices of each clique. It follows that $\|\boldsymbol{D}^{\circ\top}\|_\infty =$

Table 3.2: Upper bounds on the l_∞-norm of decay matrices and on the l_1-norm of their row-wise differences. Entries shown in this table are immediate consequences of the arithmetic series, the harmonic series, monotonic decay, and Definition 3.6.

Decay Type	$\|D(T)^\top\|_\infty$ **(I)**	$\|D(T)_t - D(T)_{t-1}\|_1$ **(II)**
Linear	$2 + T\exp(-1)$	$3/2$
Quadratic	$(T/4)+1$	$7/4$
Cubic	$T/3$	$15/8$
Rational	$\log(T)+1$	2
Exponential	$(\exp(-T)-1)/(\exp(-1)-1)$	2
Inv. Quadratic	$(T/2)+1$	$7/4$
Inv. Cubic	$3T/4$	$15/8$
Inv. Rational	T	$3/2$
Inv. Exponential	T	$1-\exp(-1)+1$

$\sum_{i=1}^{T} |\boldsymbol{D}_{i,T}|$. While this can readily be seen in Fig. 3.6 to be very close to T in case of inverse exponential or inverse rational decay, the norm is smaller in case of their regular counterparts. More precisely, in case of the regular rational decay,

$$\|\boldsymbol{D}^\circ\|_\infty = \sum_{i=1}^{T} |f_{\text{rat}}(i,T)| = \sum_{i=1}^{T} \frac{1}{i} < 1 + \log T \ ,$$

which follows from an upper bound on the harmonic series. Moreover,

$$\|\boldsymbol{D}_{T-1,\cdot} - \boldsymbol{D}_{T,\cdot}\|_1 = \sum_{i=1}^{T} |f_{\text{rat}}(i,T-1) - f_{\text{rat}}(i,T)| = 1 + \sum_{i=1}^{T-1} \frac{1}{T-i} - \frac{1}{T-i+1} = 2 - \frac{1}{T} \ .$$

Upper bounds on the these norms for other decay types are given in Table 3.2. Based on these bounds, we can devise a specialized version of Theorem 3.2.

Corollary 3.1 (STRF Consistency with Rational Decay) *Suppose the preconditions of Theorem 3.2 hold, but with $\lambda_1 = 4(\log(T)+1)\sqrt{\log(d)/N}$. In case of rational decay, and $N \geq 324\kappa^4 d^{12}\log(d)/(\log(T)+1-d^2)^2$:*

$$\begin{aligned} &\hat{\boldsymbol{\Delta}}_{C=\boldsymbol{x}'}(t) = 0 \Rightarrow \\ |\boldsymbol{\theta}^*_{C=\boldsymbol{x}'}(t-1) - \boldsymbol{\theta}^*_{C=\boldsymbol{x}'}(t)| \ &\leq \ \frac{3d^2\kappa\lambda_1}{\log(T)+1} + 2\left(\max_{i=1}^{t-1} |\hat{\boldsymbol{\Delta}}_{C=\boldsymbol{x}'}(i)| + \frac{3d^2\kappa\lambda_1}{\log(T)+1}\right) \ , \end{aligned}$$

for any clique C and time-point t with probability at least $1-(2/d)$.

Hence, the choice of decay type influences the sample complexity, and the necessary condition for sparsity. This motivates our empirical investigation of different decay types in Section 3.6.

After discussing reparametrizations of generalized sequence models with fixed decay types, one may argue that there is the possibility to estimate an appropriate decay matrix $\boldsymbol{D}$ from data. In this case, the optimization problem becomes non-convex, and we would not be able to state any consistency guarantees for the estimated parameters. If we fix either $\boldsymbol{D}$ or $\boldsymbol{\Delta}$ during optimization, the problem becomes convex, since the set of all unitriangular matrices over $[0;1]$ is convex. In this form, the problem is reminiscent of matrix factorization, and local optima of the regularized problem can be found via proximal alternating linearized minimization [25, 93]. However, we will not investigate this direction any further within this thesis, since it delivers no new insights into the inherent resource consumption of exponential families.

3.4.2 Reparametrization and Optimization

Before we proceed to the next topic, let us have short look at the impact of the reparametrization on the conditions for optimization. More specifically, the gradient, Hessian and Lipschitz constant of the reparametrized log-likelihood are products of the decay matrix and the derivatives of the ordinary log-likelihood. Let $\boldsymbol{\theta} = \eta(\boldsymbol{\Delta})$. The first two derivatives of the reparametrized likelihood are then

$$\nabla_{\boldsymbol{\Delta}}\ell(\eta(\boldsymbol{\Delta});\mathcal{D}) = \boldsymbol{D}^{\circ\top}\nabla_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta};\mathcal{D}) \quad \text{and} \quad \nabla^2\ell_{\boldsymbol{\Delta}}(\eta(\boldsymbol{\Delta});\mathcal{D}) = \boldsymbol{D}^{\circ\top}\nabla^2_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta};\mathcal{D})\boldsymbol{D}^{\circ}\ . \tag{3.8}$$

These equations follow directly from matrix differential calculus [141]. We see, that each element of $\nabla(\ell\circ\eta)$ is a weighted sum of elements of $\nabla\ell$. This becomes more clear when we focus on a single partial derivative w.r.t. $\boldsymbol{\Delta}_{C=\boldsymbol{x}'}(t)$, which corresponds to a clique assignment at time point t—w.l.o.g., we assume that the clique exists at all T time-points. Since the transposed decay matrix (which is upper triangular) appears in the gradient, all summands which correspond to time indices $t' < t$ are 0. The first partial derivative is thus

$$\frac{\partial\ell(\eta(\boldsymbol{\Delta});\mathcal{D})}{\partial\boldsymbol{\Delta}_{C=\boldsymbol{y}}(t)} = \sum_{i=t}^{T}\boldsymbol{D}_{i,t}\frac{\partial\ell(\boldsymbol{\theta};\mathcal{D})}{\partial\boldsymbol{\theta}_{C=\boldsymbol{y}}(i)}\ .$$

$\partial\ell(\eta(\boldsymbol{\Delta});\mathcal{D})/\partial\boldsymbol{\Delta}_{C=\boldsymbol{y}}(t)$ is hence influenced by all partial derivatives w.r.t. $\boldsymbol{\theta}_{C=\boldsymbol{x}'}(t')$ with $t' \geq t$—this shows how gradient information is inherited from all successive points in time which rely on $\boldsymbol{\Delta}_{C=\boldsymbol{y}}(t)$ to reconstruct their natural parameter. The particular choice of decay type implies the strength of the influence, though.

For the Hessian, we restrict ourselves to binary sufficient statistics. We known from Section 2.3.1, that the Hessian of the log-likelihood (2.19) is the covariance matrix of the random variable $\phi(\boldsymbol{X})$. The second partial derivatives of the reparametrized likelihood are

$$\begin{aligned}\left[\nabla^2_{\boldsymbol{\Delta}}\ell(\eta(\boldsymbol{\Delta});\mathcal{D})\right]_{ij} &= \sum_{k=1}^{d}\boldsymbol{D}^{\circ\top}_{i,k}\sum_{l=1}^{d}\mathbb{C}[\phi(\boldsymbol{X})]_{k,l}\boldsymbol{D}^{\circ}_{l,j} = \sum_{k=1}^{d}\sum_{l=1}^{d}\mathbb{C}[\phi(\boldsymbol{X})]_{k,l}\boldsymbol{D}^{\circ}_{k,i}\boldsymbol{D}^{\circ}_{l,j}\\ &\le \sum_{k=1}^{d}\sum_{l=1}^{d}\boldsymbol{D}^{\circ}_{k,i}\boldsymbol{D}^{\circ}_{l,j} = \sum_{k=1}^{T}\boldsymbol{D}_{k,i}\sum_{l=1}^{T}\boldsymbol{D}_{l,j} \le \|\boldsymbol{D}^{\top}\|^2_{\infty}\ ,\end{aligned}$$

where the first inequality holds, because $\phi(\boldsymbol{X})$ is assumed to be binary, and the last equality holds, because all but T entries in each row and column of $\boldsymbol{D}^{\circ}$ are zero (by construction of $\boldsymbol{D}^{\circ}$). We upper bound both summations by the l_∞-norm of $\boldsymbol{D}^{\top}$, which is tight, whenever the indices i and j belong to maximum l_1-norm rows of $\boldsymbol{D}^{\top}$. The corresponding values for various decay types can be found in column (I) of Table 3.2. Most decay types imply, that the second partial derivatives of the reparametrized objective are in $\mathcal{O}(T^2)$. Exceptions are the rational decay, with $\mathcal{O}(\log(T)^2)$, and exponential decay with $1-\mathcal{O}(\exp(-2T))$. This implies that the differences of curvature over time-points can be large for any but rational and exponential decay. Considering the contours of the corresponding objective functions, the latter obey better scaling: in other words, the corresponding objective functions will look more elliptical which results in faster convergence. We refer to [158] for more details about the relation between the scaling of objective functions and convergence rates.

Moreover, we can observe a change in the gradients Lipschitz constant.

Lemma 3.2 (Lipschitz Continuity of Linear Sequence Models) *Let $\boldsymbol{\Delta}$ be the parameter of an exponential family with linear sequence structure (Definition 3.3), $T>2$, and decay matrix $\boldsymbol{D}$. The gradient $\nabla\ell(\eta(\boldsymbol{\Delta});\mathcal{D})$ is Lipschitz continuous with constant $L=2dT|\overline{\mathcal{C}}(G)|$.*

Proof. Using (3.8), it readily follows that

$$\sup_{\boldsymbol{\theta}\in\mathbb{R}^d}\|\nabla^2\ell_{\boldsymbol{\Delta}}(\eta(\boldsymbol{\Delta});\mathcal{D})\|_F = \sup_{\boldsymbol{\theta}\in\mathbb{R}^d}\|\boldsymbol{D}^{\circ\top}\nabla^2_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta};\mathcal{D})\boldsymbol{D}^{\circ}\|_F \le 2|\overline{\mathcal{C}}(G)|\|\boldsymbol{D}^{\circ}\|^2_F\ .$$

To find an upper bound on $\|\boldsymbol{D}^{\circ}\|^2_F$, recall that the $d\times d$-matrix $\boldsymbol{D}^{\circ}$ contains one copy of $\boldsymbol{D}$ for each of the $\sum_{h=1}^{T-1}\sum_{C\in\overline{\mathcal{C}}_{+h}}|\mathcal{X}_C|$ clique-state combinations—all other entries of $\boldsymbol{D}^{\circ}$ are zero. Since the decay matrix $\boldsymbol{D}$ is triangular with values ≤ 1, we have

$$\begin{aligned}\|\boldsymbol{D}^{\circ}\|^2_F &= \overbrace{\sum_{h=1}^{T-1}\sum_{C\in\overline{\mathcal{C}}_{+h}}|\mathcal{X}_C|}^{\text{number of blocks in } \boldsymbol{D}^{\circ}}\ \|\boldsymbol{D}\|^2_F \le \frac{T(T+1)}{2}\sum_{h=1}^{T-1}\sum_{C\in\overline{\mathcal{C}}_{+h}}|\mathcal{X}_C|\\ &= \frac{T(T+1)}{2}\frac{d-|E_0||\mathcal{X}_v|^2}{(T-1)} \le dT\end{aligned}$$

The last inequality follows from (3.4), from $\sum_{h=1}^{T-1}\sum_{C\in\overline{\mathcal{C}}_{+h}}|\mathcal{X}_C| = (|V_0|+3|E_0|)|\mathcal{X}_v|^2$ (due to the assumed linearity of the STRF), and from $(T+1)/(T-1)\le 2$. ■

Note that the derived Lipschitz constant is rather large. This is mainly because the squared Frobenius norm of each decay matrix was upper bounded by $(T(T+1)/2)$. In an actual implementation, one may simply compute the exact Frobenius norm of $\boldsymbol{D}$ to derive a smaller Lipschitz constant. Recall that a small Lipschitz constant is desirable, because it implies a larger stepsize and hence, faster parameter learning, while guaranteeing convergence to an optimal solution.

3.5 Continuous State Spaces

No matter what type of data is modeled, as long as the conditional independence structure is a generalized sequence, the spatio-temporal reparametrization can be applied to the underlying exponential family member. In regression tasks—no matter if the target variable is univariate or multivariate—a basic approach is to model the conditional density of the target variable $\boldsymbol{Y}$, given an observed variable $\boldsymbol{X}$ as an exponential family member. Any choice of exponential family member can be interpreted as a specific additive random error on the target variable. This technique is also known as generalized linear model (GLM). Special kinds of GLM for sequential continuous data have been developed, so-called linear dynamical systems. In what follows, we discuss how the reparametrization of generalized sequences can be transferred to these kind of models [173].

Definition 3.7 (Linear Dynamical System) *Let $\boldsymbol{x} = (\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^T)$ be a length T sequence of n-dimensional real-valued vectors. Suppose that $\boldsymbol{x}$ is a realization of the random variable $\boldsymbol{X} = (\boldsymbol{X}^1, \boldsymbol{X}^2, \ldots, \boldsymbol{X}^T)$, and assume that its autonomous dynamics are fully specified by a finite, discrete-time, affine matrix equation*

$$\boldsymbol{x}^t = \boldsymbol{A}^{t-1}\boldsymbol{x}^{t-1} + \boldsymbol{\varepsilon}^t \qquad \textit{for } 1 < t \leq T$$

with transition matrices[12] $\boldsymbol{A}^t \in \mathbb{R}^{n\times n}$ and noise $\boldsymbol{\varepsilon}^t \in \mathbb{R}^n$. $\boldsymbol{x}^1$ is the initial state of the system, and each $\boldsymbol{\varepsilon}^t$ is drawn from the same multivariate Gaussian distribution $\boldsymbol{\varepsilon}^t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$. Any random variable $\boldsymbol{X}$ that is generated via the above process, is called linear dynamical system *(LDS). The dynamics are time invariant, if $\boldsymbol{A}^t = \boldsymbol{A}^{t-1}, \forall 1 \leq t \leq T$.*

The system generates a random variable due to the stochasticity in $\boldsymbol{\varepsilon}$. Moreover, the density of each $\boldsymbol{X}^t \mid \boldsymbol{x}^{t-1}$ is inherited from $\boldsymbol{\varepsilon}^t$. E.g., if $\boldsymbol{\varepsilon}^t$ is multivariate Gaussian with mean $\boldsymbol{0}$, then $\boldsymbol{X}^t$ is multivariate Gaussian with mean $\boldsymbol{A}^{t-1}\boldsymbol{x}^{t-1}$. With the above definition, the full joint density of $\boldsymbol{X}$ factorizes over time and may be written as

$$p(\boldsymbol{X} = \boldsymbol{x}) = p(\boldsymbol{X}^1 = \boldsymbol{x}^1) \prod_{t=1}^{T-1} p(\boldsymbol{X}^{t+1} = \boldsymbol{x}^{t+1} \mid \boldsymbol{X}^t = \boldsymbol{x}^t) \,. \tag{3.9}$$

[12]We will use $\boldsymbol{A} := (\boldsymbol{A}^1, \boldsymbol{A}^2, \ldots, \boldsymbol{A}^{T-1})$ to denote the concatenation of all transition matrices of the system.

Within this Section, we assume w.l.o.g. that $\boldsymbol{X}^1$ and each $\boldsymbol{\varepsilon}^t$ has multivariate Gaussian density with mean $\mathbf{0}$ and covariance matrix $\boldsymbol{\Sigma}$.

$$p_{\boldsymbol{\Sigma}}(\boldsymbol{\varepsilon}) = \frac{1}{\sqrt{(2\pi)^n \det \boldsymbol{\Sigma}}} \exp(-(1/2)\boldsymbol{\varepsilon}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\varepsilon}) \tag{3.10}$$

We will call (3.10) the standard form. In this case, $p_{\boldsymbol{\Sigma}}(\boldsymbol{X}^1 = \boldsymbol{x}^1)$ and each $p_{\boldsymbol{\Sigma},\boldsymbol{A}^{t-1}}(\boldsymbol{X}^t = \boldsymbol{x}^t \mid \boldsymbol{X}^{t-1} = \boldsymbol{x}^{t-1})$ are Gaussian too, and their product—the joint density of $\boldsymbol{X}$—is also Gaussian. The Gaussian is indeed in the exponential family, but note that (3.10) is not in exponential family form. Let the operator vec : $\mathbb{R}^{m\times n} \to \mathbb{R}^{mn}$ transform a matrix into a vector by stacking the columns of the matrix one underneath the other—vec($\boldsymbol{M}$) represents the matrix $\boldsymbol{M}$ in column-major order. Parameter and sufficient statistic $\phi : \mathbb{R}^n \to \mathbb{R}^d$ of the multivariate Gaussian are then

$$\boldsymbol{\theta} = \begin{pmatrix} -\frac{1}{2}\operatorname{vec}(\boldsymbol{\Sigma}^{-1}) \\ \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \end{pmatrix} \quad \text{and} \quad \phi(\boldsymbol{x}) = \begin{pmatrix} \operatorname{vec}(\boldsymbol{x}\boldsymbol{x}^\top) \\ \boldsymbol{x} \end{pmatrix} ,$$

respectively. Moreover, a closed form of the log partition function can be derived by the n-dimensional Gaussian integral:

$$\begin{aligned} \log Z(\boldsymbol{\theta}) &= \log \int \exp\left(\langle \boldsymbol{\theta}, \phi(\boldsymbol{x})\rangle\right) \mathrm{d}\boldsymbol{x} \\ &= \log \int \exp\left(-\frac{1}{2}\boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x} \ + \ \boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right) \mathrm{d}\boldsymbol{x} \\ &= \log\left(\sqrt{(2\pi)^n \det \boldsymbol{\Sigma}^{-1}} \exp\left(\frac{1}{2}\boldsymbol{\mu}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right)\right) . \end{aligned}$$

Plugging this into (2.3) and rearranging, the identity between the exponential family form and the standard form of the multivariate Gaussian density becomes evident.

$$\begin{aligned} p_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \exp\left(\langle \boldsymbol{\theta}, \phi(\boldsymbol{x})\rangle - \log Z(\boldsymbol{\theta})\right) \\ &= \frac{1}{\sqrt{(2\pi)^n \det \boldsymbol{\Sigma}}} \exp\left(-\frac{1}{2}\boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x} \ + \ \boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \ - \ \frac{1}{2}\boldsymbol{\mu}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right) \\ &= \frac{1}{\sqrt{(2\pi)^n \det \boldsymbol{\Sigma}}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) = p_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\boldsymbol{x}) \ . \end{aligned}$$

Based on this equivalence, the joint density (3.9) of an LDS can also be rewritten in terms of exponential families (2.3)

$$\begin{aligned} p_{\boldsymbol{A},\boldsymbol{\Sigma}}(\boldsymbol{X}^1 = \boldsymbol{x}^1, \boldsymbol{X}^2 = \boldsymbol{x}^2, \ldots, \boldsymbol{X}^t = \boldsymbol{x}^t) &= p_{\boldsymbol{\Sigma}}(\boldsymbol{x}^1) \prod_{t=1}^{T-1} p_{\boldsymbol{A}^t,\boldsymbol{\Sigma}}(\boldsymbol{x}^{t+1} \mid \boldsymbol{x}^t) \\ &= p_{\boldsymbol{\theta}^1}(\boldsymbol{x}^1) \prod_{t=1}^{T-1} p_{\boldsymbol{\theta}^{t+1}}(\boldsymbol{x}^{t+1} \mid \boldsymbol{x}^t) \\ &= \exp\left(\sum_{t=1}^{T} \langle \boldsymbol{\theta}^t, \phi_t(\boldsymbol{x}^t, \boldsymbol{x}^{t-1})\rangle - \log Z(\boldsymbol{\theta}^t)\right) , \end{aligned}$$

with

$$\boldsymbol{\theta}(t) = \begin{pmatrix} -\frac{1}{2}\operatorname{vec}(\boldsymbol{\Sigma}^{-1}) \\ \operatorname{vec}(\boldsymbol{\Sigma}^{-1}\boldsymbol{A}^{t-1}) \end{pmatrix} \qquad \text{and} \qquad \phi^t(\boldsymbol{x}^t, \boldsymbol{x}^{t-1}) = \begin{pmatrix} \operatorname{vec}(\boldsymbol{x}^t\boldsymbol{x}^{t\top}) \\ \operatorname{vec}(\boldsymbol{x}^t\boldsymbol{x}^{t-1\top}) \end{pmatrix} . \tag{3.11}$$

This representation has several drawbacks when compared to the standard form. An obvious disadvantage is, that multiple copies of $\boldsymbol{\Sigma}^{-1}$ are encoded into the parameters. When we go ahead and estimate the parameters $\boldsymbol{\theta}(t)$, additional constraints are required to ensure that the parts of $\boldsymbol{\theta}(t)$ that correspond to a copy of $\boldsymbol{\Sigma}^{-1}$ agree with each other. Moreover, $\boldsymbol{\Sigma}^{-1}$ needs to be invertible and hence, parts of $\boldsymbol{\theta}(t)$ are restricted to the cone of positive definite matrices. In short, the direct estimation of the natural LDS parameters is not beneficial, and we will focus on the standard form instead.

3.5.1 Parameter Estimation

Due to the standard form, the estimation procedure looks slightly different to the generic estimation of exponential family parameters. Objective function and gradient are affected by this alternative representation. The parameters $\boldsymbol{A}$ and $\boldsymbol{\Sigma}^{-1}$ are indeed estimated by minimizing the negative average log-likelihood $\ell^{\mathrm{LDS}}(\boldsymbol{A}, \boldsymbol{\Sigma}^{-1}; \mathcal{D}) = -\frac{1}{|\mathcal{D}|T} \log p_{\boldsymbol{A},\boldsymbol{\Sigma}^{-1}}(\mathcal{D})$, where we additionally divide by T for notational convenience. Substituting the standard form into ℓ^{LDS}, yields the objective function

$$\begin{aligned}
\ell^{\mathrm{LDS}}(\boldsymbol{A}, \boldsymbol{\Sigma}^{-1}; \mathcal{D}) &= -\frac{1}{|\mathcal{D}|T} \log \prod_{\boldsymbol{x}\in\mathcal{D}} p_{\boldsymbol{A},\boldsymbol{\Sigma}}(\boldsymbol{x}) \\
&= -\frac{1}{|\mathcal{D}|T} \sum_{\boldsymbol{x}\in\mathcal{D}} \left(\log p_{\boldsymbol{\Sigma}}(\boldsymbol{x}_1) + \sum_{t=1}^{T-1} \log p_{\boldsymbol{A},\boldsymbol{\Sigma}}(\boldsymbol{x}^{t+1} \mid \boldsymbol{x}^t) \right) \\
&= C - \frac{1}{2} \log\det \boldsymbol{\Sigma}^{-1} + \frac{1}{2|\mathcal{D}|T} \sum_{\boldsymbol{x}\in\mathcal{D}} \sum_{t=1}^{T} \boldsymbol{r}^{t\top}\boldsymbol{\Sigma}^{-1}\boldsymbol{r}^t
\end{aligned}$$

with residual vector $\boldsymbol{r}^t = \boldsymbol{x}^t - \boldsymbol{A}^{t-1}\boldsymbol{x}^{t-1}$ and constant $C = \frac{1}{2} n \log 2\pi$. To simplify notation, $p_{\boldsymbol{\Sigma}}(\boldsymbol{x}^1)$ is absorbed into the summation by setting $\boldsymbol{x}^0 := \mathbf{0}$ and $\boldsymbol{A}^0 := \mathbf{0}$. As with the generic objective function, the above is a convex function of the transition matrices and the inverse noise covariance matrix, due to the convexity of $-\log\det \boldsymbol{\Sigma}^{-1}$. The partial derivative[13] of ℓ^{LDS} w.r.t. each transition matrix $\boldsymbol{A}^t$, for $1 \le t < T$, is then

$$\begin{aligned}
\frac{\partial \ell^{\mathrm{LDS}}(\boldsymbol{A}, \boldsymbol{\Sigma}^{-1}; \mathcal{D})}{\partial \operatorname{vec}\left(\boldsymbol{A}^t\right)^\top} &= \frac{1}{2|\mathcal{D}|T} \sum_{\boldsymbol{x}\in\mathcal{D}} \frac{\partial \left(\boldsymbol{r}^{t+1\top}\boldsymbol{\Sigma}^{-1}\boldsymbol{r}^{t+1}\right)}{\partial \operatorname{vec}\left(\boldsymbol{A}^t\right)^\top} \\
&= -\frac{1}{|\mathcal{D}|T} \operatorname{vec}\left(\boldsymbol{\Sigma}^{-1} \sum_{\boldsymbol{x}\in\mathcal{D}} (\boldsymbol{x}^{t+1} - \boldsymbol{A}^t\boldsymbol{x}^t)\boldsymbol{x}^{t\top} \right)^\top
\end{aligned}$$

[13] Please be aware that the notation for a derivative w.r.t. a matrix is notoriously inconsistent and prone to errors. Here, we adopt the notation of [141] in that only derivatives w.r.t. vectors are allowed. Matrices must hence be vectorized before we take the derivative. A detailed discussion of this topic can be found in [141], pages 194–215.

and its partial derivative w.r.t. $\mathbf{\Sigma}^{-1}$ is

$$
\begin{aligned}
\frac{\partial \ell^{\mathrm{LDS}}(\boldsymbol{A}, \mathbf{\Sigma}^{-1}; \mathcal{D})}{\partial \operatorname{vec}\left(\mathbf{\Sigma}^{-1}\right)^\top} &= -\frac{1}{2}\frac{\partial \log\det \mathbf{\Sigma}^{-1}}{\partial \operatorname{vec}\left(\mathbf{\Sigma}^{-1}\right)^\top} + \frac{1}{2|\mathcal{D}|T}\sum_{\boldsymbol{x}\in\mathcal{D}}\sum_{t=1}^{T}\frac{\partial\left(\boldsymbol{r}^{t\top}\mathbf{\Sigma}^{-1}\boldsymbol{r}^t\right)}{\partial \operatorname{vec}\left(\mathbf{\Sigma}^{-1}\right)^\top} \\
&= \frac{1}{2}\operatorname{vec}\left(-\mathbf{\Sigma} + \frac{1}{|\mathcal{D}|T}\sum_{\boldsymbol{x}\in\mathcal{D}}\sum_{t=1}^{T}\boldsymbol{r}^t\boldsymbol{r}^{t\top}\right)^\top
\end{aligned}
\tag{3.12}
$$

An intuitive implication of the first-order condition $\partial\ell^{\mathrm{LDS}}(\boldsymbol{A}, \mathbf{\Sigma}^{-1}; \mathcal{D})/\partial \operatorname{vec}\left(\mathbf{\Sigma}^{-1}\right)^\top = 0$ is, that the minimizer $\mathbf{\Sigma}^*$ must be equal to the empirical second moment of the residual vector. We are now ready to discuss the spatio-temporal reparametrization of LDS.

3.5.2 Linear Dynamical Systems and Undirected Models

The reparametrization is defined for undirected generalized sequences. The LDS, however, defines a directed model, which follows directly from its factorization (cf. Theorem 2.1). Moreover, the corresponding undirected structure is not a generalized sequence.

Lemma 3.3 (LDS Non-Generality) *The undirected structure of an LDS is not a generalized sequence.*

Proof. The inverse covariance matrix $\mathbb{C}^{-1}$ of any multivariate Gaussian contains the partial correlation coefficients between all pairs of variables. Any off-diagonal entry $\mathbb{C}^{-1}_{i,j}$ of the inverse covariance matrix is zero, if and only if $\boldsymbol{X}_i \perp\!\!\!\perp \boldsymbol{X}_{-j} \mid \boldsymbol{X}_{V\setminus\{i,j\}}$. Hence, the conditional independence structure is given by the non-zero pattern of $\mathbb{C}^{-1}$ (cf. [133, 234]). Whenever two (multivariate) random variables $\boldsymbol{W}, \boldsymbol{Y}$ are jointly Gaussian with mean $\boldsymbol{\mu} = (\boldsymbol{\mu}_{\boldsymbol{W}}, \boldsymbol{\mu}_{\boldsymbol{Y}})$ and covariance matrix

$$
\mathbb{C} = \begin{pmatrix} \mathbb{C}_{\boldsymbol{W},\boldsymbol{W}} & \mathbb{C}_{\boldsymbol{W},\boldsymbol{Y}} \\ \mathbb{C}_{\boldsymbol{Y},\boldsymbol{W}} & \mathbb{C}_{\boldsymbol{Y},\boldsymbol{Y}} \end{pmatrix} ,
$$

the conditional mean and covariance of $\boldsymbol{Y} \mid \boldsymbol{W}$ are

$$
\begin{aligned}
\boldsymbol{\mu}_{\boldsymbol{Y}|\boldsymbol{W}=\boldsymbol{w}} &= \boldsymbol{\mu}_{\boldsymbol{Y}} + \mathbb{C}_{\boldsymbol{Y},\boldsymbol{W}}\mathbb{C}^{-1}_{\boldsymbol{W},\boldsymbol{W}}\left(\boldsymbol{w} - \boldsymbol{\mu}_{\boldsymbol{W}}\right) \\
\mathbb{C}_{\boldsymbol{Y}|\boldsymbol{W}=\boldsymbol{w}} &= \mathbb{C}_{\boldsymbol{Y},\boldsymbol{Y}} - \mathbb{C}_{\boldsymbol{Y},\boldsymbol{W}}\mathbb{C}^{-1}_{\boldsymbol{W},\boldsymbol{W}}\mathbb{C}_{\boldsymbol{W},\boldsymbol{Y}} .
\end{aligned}
$$

Now, by setting $\boldsymbol{Y} = \boldsymbol{X}^2$, $\boldsymbol{W} = \boldsymbol{X}^1$, and $\boldsymbol{\mu} = \boldsymbol{0}$, we see that $\boldsymbol{A}^2\boldsymbol{x}^1 = \boldsymbol{\mu}_{\boldsymbol{X}^2|\boldsymbol{X}^1=\boldsymbol{x}^1} = \mathbb{C}_{\boldsymbol{X}^2,\boldsymbol{X}^1}\mathbb{C}^{-1}_{\boldsymbol{X}^1,\boldsymbol{X}^1}\boldsymbol{x}^1$, which implies $\boldsymbol{A}^2 = \mathbb{C}_{\boldsymbol{X}^2,\boldsymbol{X}^1}\mathbb{C}^{-1}_{\boldsymbol{X}^1,\boldsymbol{X}^1}$. $\mathbb{C}_{\boldsymbol{X}^2|\boldsymbol{X}^1=\boldsymbol{x}^1}$ equals $\mathbf{\Sigma}$ by construction, and thus $\mathbf{\Sigma} = \mathbb{C}_{\boldsymbol{X}^2,\boldsymbol{X}^2} - \mathbb{C}_{\boldsymbol{X}^2,\boldsymbol{X}^1}\mathbb{C}^{-1}_{\boldsymbol{X}^1,\boldsymbol{X}^1}\mathbb{C}_{\boldsymbol{X}^1,\boldsymbol{X}^2}$. Moreover, $\mathbb{C}_{\boldsymbol{X}^1,\boldsymbol{X}^1}$ equals $\mathbf{\Sigma}$ (by assumption on $\boldsymbol{X}^1$), and hence

$$
\mathbb{C} = \begin{pmatrix} \mathbf{\Sigma} + \boldsymbol{A}^2\mathbf{\Sigma}^\top\boldsymbol{A}^{2\top} & \boldsymbol{A}^2\mathbf{\Sigma} \\ \mathbf{\Sigma}^\top\boldsymbol{A}^{2\top} & \mathbf{\Sigma} \end{pmatrix} . \tag{3.13}
$$

So we derived the covariance matrix of $(\boldsymbol{X}^1, \boldsymbol{X}^2)$. In generalized sequence models, the spatial structure, i.e., the conditional independences between variables in the same time-slice, is constant over time. Here, the spatial structure is defined by the diagonal blocks of $\mathbb{C}^{-1}$. By inversion of block matrices in 3.13, we see that the diagonal blocks of the inverse covariance matrix are equal, if and only if

$$\begin{aligned}\boldsymbol{\Sigma} - \boldsymbol{\Sigma}^\top \boldsymbol{A}^{2\top}(\boldsymbol{\Sigma} + \boldsymbol{A}^2\boldsymbol{\Sigma}^\top \boldsymbol{A}^{2\top})^{-1}\boldsymbol{A}^2\boldsymbol{\Sigma} &= \boldsymbol{\Sigma} + \boldsymbol{A}^2\boldsymbol{\Sigma}^\top\boldsymbol{A}^{2\top} - \boldsymbol{A}^2\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}^\top\boldsymbol{A}^{2\top} \\ \Leftrightarrow \boldsymbol{\Sigma}^{-1} + \boldsymbol{A}^{2\top}\boldsymbol{\Sigma}^{-1}\boldsymbol{A}^2 &= \boldsymbol{\Sigma}^{-1} \, .\end{aligned}$$

The equivalence follows from the Woodbury matrix identity. Nevertheless, the last equality can only hold, if $\boldsymbol{A}^{2\top}\boldsymbol{\Sigma}^{-1}\boldsymbol{A}^2 = \boldsymbol{0}$, which is not true in general. The diagonal blocks of $\mathbb{C}$ and thus the undirected spatial structure of the LDS are hence allowed to change over time, which contradicts the definition of generalized sequence models. ■

Nevertheless, we will see that the reparametrization can still be applied.

3.5.3 Reparametrization

Despite its directedness, 3.13 shows, that the undirected structure between time-slices is influenced by the transition matrices $\boldsymbol{A}^t$. Like with ordinary exponential families, regularization can be used to detect the non-zero pattern in $\boldsymbol{A}^t$. If we aim at a reduction of model parameters, any further regularization, e.g., with a higher regularization weight, will affect the conditional independence structure of the model. According to the requirements that we stated in the Chapter 1, the conditional independence structure should be kept intact.

The natural (exponential family) LDS parameters (3.11) are not well suited for reparametrization, due to the redundant encoding and positive-definiteness of the noise covariance matrix. Instead, we perform a reparametrization of the transition matrices, which can be used directly for the prediction of future states of the system, or to study the evolution of particular interactions between variables.

In analogy to Definition 3.5, the reparametrization is defined by $n \times n$ vectors $\boldsymbol{\Delta}_{i,j}$, each of dimension $T-1$. Let $\boldsymbol{A}_{i,j} \in \mathbb{R}^{T-1}$ be the vector that contains the (i,j) entry of each of the $T-1$ transition matrices, and let $\boldsymbol{D}$ be a decay matrix for $T-1$ time steps.

$$\boldsymbol{A}_{i,j} = \eta_{\boldsymbol{D}}(\boldsymbol{\Delta}_{i,j}) = \boldsymbol{D}\boldsymbol{\Delta}_{i,j} \tag{3.14}$$

We use $\boldsymbol{A} = \eta_{\boldsymbol{D}}(\boldsymbol{\Delta})$ to denote the full reparametrization of all transition matrices. To this end, we again construct the matrix $\boldsymbol{D}^\circ$ such that $\eta_{\boldsymbol{D}}(\boldsymbol{\Delta}) = \boldsymbol{D}^\circ\boldsymbol{\Delta}$. This results in the objective function

$$\ell^{\text{S-LDS}}(\eta_{\boldsymbol{D}}(\boldsymbol{\Delta}), \boldsymbol{\Sigma}^{-1}; \mathcal{D}) = \ell^{\text{LDS}}(\eta_{\boldsymbol{D}}(\boldsymbol{\Delta}), \boldsymbol{\Sigma}^{-1}; \mathcal{D}) + \frac{\lambda_2}{2}\|\boldsymbol{\Delta}\|_2^2 + \lambda_1\|\boldsymbol{\Delta}\|_1 \, .$$

The inverse covariance matrix $\boldsymbol{\Sigma}^{-1}$ is not reparametrized. On the one hand, maintaining its symmetry and positive definiteness within the reparametrization requires non-trivial changes to the reparametrization. Results on the estimation of sparse inverse covariance matrices where such topics are discussed may be found in [70, 34, 251].

Here, the partial derivative of $\ell^{\text{S-LDS}}$ w.r.t. $\boldsymbol{\Sigma}$ will be the same as in (3.12). The partial derivative w.r.t. $\boldsymbol{\Delta}$ follows directly from the chain rule (see, e.g., [141]).

$$
\begin{aligned}
&\frac{\partial \ell^{\text{S-LDS}}(\eta_{\boldsymbol{D}}(\boldsymbol{\Delta}), \boldsymbol{\Sigma}^{-1}; \mathcal{D})}{\partial \boldsymbol{\Delta}_{i,j}^{\top}} \\
&= \left(\frac{\partial \eta_{\boldsymbol{D}}(\boldsymbol{\Delta})}{\partial \boldsymbol{\Delta}_{i,j}^{\top}}\right)^{\top} \left(\frac{\partial \ell^{\text{LDS}}(\eta_{\boldsymbol{D}}(\boldsymbol{\Delta}), \boldsymbol{\Sigma}^{-1}; \mathcal{D})}{\partial \eta_{\boldsymbol{D}}(\boldsymbol{\Delta})_{i,j}^{\top}}\right)^{\top} + \lambda_2 \boldsymbol{\Delta}_{i,j} + \lambda_1 \partial |\boldsymbol{\Delta}_{i,j}| \\
&= -\frac{1}{|\mathcal{D}|T} \boldsymbol{D}^{\top} \operatorname{vec}\left(\boldsymbol{\Sigma}^{-1} \sum_{\boldsymbol{x} \in \mathcal{D}} (\boldsymbol{x}^{t+1} - \boldsymbol{A}^t \boldsymbol{x}^t) \boldsymbol{x}^{t\top}\right)^{\top} + \lambda_2 \boldsymbol{\Delta}_{i,j} + \lambda_1 \partial |\boldsymbol{\Delta}_{i,j}|
\end{aligned}
$$

The non-smooth part that arises from the l_1-regularization is handled by the proximal operator (2.36) as before. To estimate the Lipschitz constant, we derive an upper bound on the norm of the Hessian of $\ell^{\text{S-LDS}}(\eta_{\boldsymbol{D}}(\boldsymbol{\Delta}), \boldsymbol{\Sigma}^{-1}; \mathcal{D})$. First, consider $\ell^{\text{S-LDS}}(\boldsymbol{A}, \boldsymbol{\Sigma}^{-1}; \mathcal{D})$.

$$
\nabla^2_{\boldsymbol{A}^t} \ell^{\text{S-LDS}}(\boldsymbol{A}, \boldsymbol{\Sigma}^{-1}; \mathcal{D}) = \frac{\partial \ell^{\text{S-LDS}}(\boldsymbol{A}, \boldsymbol{\Sigma}^{-1}; \mathcal{D})}{\partial \operatorname{vec}(\boldsymbol{A}^t) \partial \operatorname{vec}(\boldsymbol{A}^t)^{\top}} = \boldsymbol{\Sigma}^{-1} \frac{1}{|\mathcal{D}|T} \sum_{\boldsymbol{x} \in \mathcal{D}} \boldsymbol{x}^t \boldsymbol{x}^{t\top} + \lambda_2 \boldsymbol{I} \; . \tag{3.15}
$$

The Hessian w.r.t. $\boldsymbol{A}$ is thus the product of the inverse noise covariance matrix and the empirical second moment of the data in time-slice t. An upper bound on this expression cannot be meaningful without any assumptions on $\|\boldsymbol{x}\|$ and $\|\boldsymbol{\Sigma}^{-1}\|$. However, in a practical setting, computing the Frobenius norm of (3.15) is rather straightforward, whenever $\boldsymbol{\Sigma}^{-1}$ is available. Given that this can be done, the Lipschitz constant (w.r.t. $\boldsymbol{\Delta}$) is then

$$
\begin{aligned}
L &\leq \sup_{\boldsymbol{\Delta}} \|\nabla^2_{\boldsymbol{\Delta}} \ell^{\text{S-LDS}}(\eta_{\boldsymbol{D}}(\boldsymbol{\Delta}), \boldsymbol{\Sigma}^{-1}; \mathcal{D})\|_F = \sup_{\boldsymbol{\Delta}} \|\boldsymbol{D}^{\circ\top} \nabla^2_{\eta_{\boldsymbol{D}}(\boldsymbol{\Delta})} \ell^{\text{S-LDS}}(\eta_{\boldsymbol{D}}(\boldsymbol{\Delta}), \boldsymbol{\Sigma}^{-1}; \mathcal{D}) \boldsymbol{D}^{\circ}\|_F \\
&\leq \sup_{\boldsymbol{\Delta}} \|\boldsymbol{D}^{\circ}\|_F^2 (\|\nabla^2_{\eta_{\boldsymbol{D}}(\boldsymbol{\Delta})} \ell^{\text{S-LDS}}(\eta_{\boldsymbol{D}}(\boldsymbol{\Delta}), \boldsymbol{\Sigma}^{-1}; \mathcal{D})\|_F + \lambda_2) \; .
\end{aligned}
$$

Experimental results for the reparametrized LDS can be found in [173]. Since the LDS is not an undirected model, we restrict ourselves to models for discrete data for the remainder of this Chapter.

3.6 Experimental Demonstration

The theoretical results that we derived in this section show that regularized spatio-temporal reparametrizations of generalized sequence models will detect temporal redundancies in the parameters. These results are independent of the specific choice of decay matrix and conditional independence structure. We conduct a series of experiments to demonstrate this behavior on synthetic and real-world data. More precisely, we are interested in answering the following questions empirically:

Q1 Can we observe an increase in sparsity (and hence, a reduction of memory consumption)?

Q2 Do compressed models still provide a reasonable quality?

Q3 How does the spatio-temporal reparametrization influence the computational complexity?

The first two questions are motivated by Theorem 3.2, that is, regularization will push the reparametrization towards zero whenever the variation of natural parameters is small over time. Moreover, the full, decompressed parametrization should be close to the true parameters. However, due to overcompleteness, there is an affine set of equivalent parameters, specified by (2.21), which represent the same density. It is hence not guaranteed that our estimate will be close to the parameter that was actually used to generate the data. Due to the relation between a parameter vector $\boldsymbol{\theta}$ and the induced vector of marginals $\boldsymbol{\mu}$, we will instead consider the normalized squared Euclidean distance—equivalent to the mean squared error (MSE)—between our estimated marginals $\hat{\boldsymbol{\mu}}$ and the empirical marginals $\tilde{\boldsymbol{\mu}}$ to investigate the quality of different models and to find an answer to question **Q2**. In terms of sparsity, optimization of the regularized objective finds a compromise between maximum likelihood and maximum sparsity. The true parameter vector might not obey small changes over time and will hence not allow for a compressed representation. However, due to overcompleteness, there might be an equivalent but compressible solution, even if $\boldsymbol{\theta}^*$ is not. We should hence expect to answer **Q1** in the affirmative. To measure the actual sparsity of any parameter vector $\boldsymbol{\theta} \in \mathbb{R}^d$, we consult its proportion of non-zero components (NNZ-Ratio), denoted by $\|\boldsymbol{\theta}\|_0/d$—this notation is common in the literature, although $\|\cdot\|_0$ is not homogeneous and hence not a norm.

Finally, by definition of the spatio-temporal reparametrization, each access to $\boldsymbol{\theta}_i(t)$ involves a decoding overhead of $\mathcal{O}(t)$, which indicates that the actual runtime of the compressed model will degrade compared to the vanilla MRF. Caching effects could nonetheless amortize this performance penalty. On the other hand the effective number of parameters (or degrees of freedom) is smaller in case of the regularized model [86]. Hence, the actual number of parameter updates can decrease and thus speed up the estimation procedure. Based on these considerations, we will report the total runtime (in milliseconds).

To summarize, our evaluation incorporates

- the relative number of non-zero components of $\boldsymbol{\theta}$ to measure the memory consumption of a model,
- the mean squared error between estimated and empirical marginals: $\mathrm{MSE}(\boldsymbol{\theta}) = \frac{1}{d}\sum_{i=1}^{d}(\hat{\boldsymbol{\mu}} - \tilde{\boldsymbol{\mu}})^2$ to quantify the general quality of a model,
- and the runtime per training iteration in milliseconds, to measure the computational complexity.

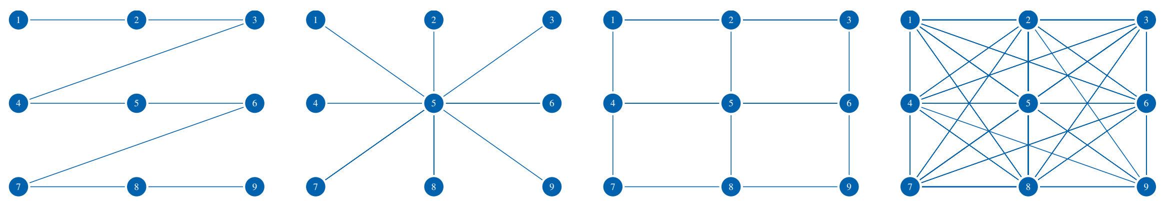
Figure 3.8: Base graphs for synthetic data generation. From left to right: chain, star, grid, full. The full structure contains all possible edges, although some of them are not visible in the figure.

3.6.1 Setup

The basic model that underlies each experiment is a discrete state exponential family with linear sequence structure and overcomplete sufficient statistic. Its parameters are estimated via accelerated first-order optimization. We investigate the following variants of the basic model:

M1 No regularization, no reparametrization—a vanilla MRF on spatio-temporal data

M2 Regularization, no reparametrization

M3 Regularization with spatio-temporal reparametrization—the STRF approach

In case of models **M2** and **M3**, we consider various regularization weights $\lambda \in \{10^{-2} \times 2^0, 10^{-2} \times 2^1, \dots, 10^{-2} \times 2^7\}$. Furthermore, model **M3** is instantiated with all decay types that we discussed in Section 3.4.1. Data from different sources is considered to demonstrate the robustness of the reparametrization. Each experiment is performed on an Intel Xeon E5-2697 v2 system. We provide a docker image with our own C++ implementation of models **M1**–**M3** for download at `https://sfb876.tu-dortmund.de/px`.

Synthetic Data

We conduct a series of experiments on synthetic data which allows us to embed temporal redundancies into the true parameter. For simplicity, all synthesized models contain only pairwise factors ψ_{vu}. The number of variables per time-slice is fixed to $n_0 = 9$ and the number of states per variable is fixed to 2. We consider four different temporal depths T, i.e., number of temporal layers, from small ($T = 16$) to large ($T = 128$). Small models consists of $n = 144$ variables and large models of $n = 1152$ variables.

We consider four different base graphs G_0, representing the spatial conditional independence structure, shown in Fig. 3.8. The graphs are chosen to cover different levels of complexity. In the simplest structure (chain), each vertex has degree at most 2. The star structure exhibits one vertex with relatively high degree. Grid structures consist of multiple loops but obey a vertex degree which is independent of n, while spatial neighborhood sizes in fully connected structures depend on the number of spatial variables.

The ground truth linear sequence structures $G = (V, E)$ are synthesized according to Definition 3.2, with base graph $G_0 \in \{\text{chain}, \text{star}, \text{grid}, \text{full}\}$ and $T \in \{16, 32, 64, 128\}$. We consider overcomplete representations in which each edge $e(t)$ is parametrized by a 4-dimensional vector $\boldsymbol{\theta}_{e(t)}$ whose entries are sampled independently from a Gaussian with mean 0 and variance 1. Here, $e(t)$ denotes an edge from time-slice t (cf. Definition 3.2).

Finally, we inject some temporal redundancies into the random model parameters. Recall that the sequence model contains up to T copies of each edge $e(t) \in E$. Redundancies are injected by visiting each temporal copy of each edge in increasing temporal order $t = 2, 3, \ldots$. Then, we draw a Bernoulli random variable $\boldsymbol{B}$ with parameter $q \in [0; 1]$. Whenever $\boldsymbol{B} = 1$, the edge at time t, e.g., $e(t)$, inherits the parameters of its temporal predecessor $e(t-1)$, i.e., $\boldsymbol{\theta}_{e(t)} \leftarrow \boldsymbol{\theta}_{e(t-1)} \Leftrightarrow \boldsymbol{B} = 1$.

Note that the parameter q allows us to control the amount of temporal redundancies in a compact manner. If $q = 0$, no edge will inherit the parameters from its predecessor, and all parameters are independent realizations of a standard normal random variable. By increasing q, dependencies between the parameters of consecutive edges arise. When $q = 1$, the parameters are actually time invariant—they are identical at each time step. In our experiments, we consider $q \in \{0, 0.25, 0.5, 0.75, 1\}$ to study the effects of different redundancy levels.

Eventually, 1000 samples are generated from each model via Gibbs sampling (Algorithm 2.4). During data generation, the first 100 samples are discarded. Between consecutive samples, all variables are resampled 16 times in a round-robin fashion to enforce independence of consecutive samples.

Although the true underlying graphical structure is known, the finite synthetic data sets might not capture all true conditional independences. Hence, we apply a closed-form l_1-regularized structure estimation [240] with regularization weight $\lambda = 10^{-1}$.

Real-World Data

In the second set of experiments, we estimate the models on data, collected from real sensor networks, namely:

D1 INSIGHT—Dublin City SCATS Data[14]

D2 VaVeL—Warsaw City Mobile Network Cell Data[15]

D3 Intel Lab—Temperature and Humidity Data[16]

While the memory consumption of the model is of rather minor importance in large, wired sensor networks, sparse models are still of interest when overfitting has to be avoided. Considering a process that generates one independent sample per day, only 365 training instances can be generated per year. If the corresponding sensor network is large, the number of training examples is likely to be too small for consistent maximum

[14]Section 4.1.1 in `http://www.insight-ict.eu/sites/default/files/deliverables/D5-1.pdf`
[15]Section 3.15 in `http://www.vavel-project.eu/sites/default/files/VaVeL_D1_3.pdf`
[16]`http://db.csail.mit.edu/labdata/labdata.html`

Table 3.3: Summary of real-world data sets.

Name	**# variables** (n)	**# readings**	**# data points** (N)
D1 INSIGHT	2367	316632468	134
D2 VaVeL	4988	3934145	43
D3 Intel Lab	56	2313682	36

likelihood estimation. This is indeed true for the data sets considered here, whose characteristics are shown in Table 3.3. Nevertheless, incorporating prior knowledge in form of regularization (Section 2.3.3) can help to prevent overfitting.

Let us shortly summarize some details about the sensor networks and the data collection. More details about the data can be found in the referenced online sources. The first set (**D1**) was collected between Jan 1 and Mai 14 in 2013 from SCATS (Sydney Coordinated Adaptive Traffic System) traffic sensors, which are embedded into several streets in the city of Dublin, Ireland. Each sensor outputs the average vehicle speed (**D1**.A) and relative sensor occupation (traffic density) (**D1**.B) per minute. Secondly, the set **D2** was collected between Mai 15 and June 26 of 2016 from selected mobile network cells, located in the urban area of the city of Warsaw, Poland. The measured data represents subscriber activity on the basis of network events which are triggered together with voice and xMS communication. For each cell, the number of events per hour was recorded. The last set (**D3**) was collected between Feb 28 and April 2 of 2004 from sensors deployed in the Intel Berkeley Research lab. Temperature (**D3**.A), humidity (**D3**.B), light and voltage values were collected once every 31 seconds. Moreover, we ignore sensor readings with implausible values, e.g., unreasonable high indoor temperatures.

For each of these data sets, we construct a linear sequence model with $T = 12$ layers, representing 24 hours, i.e., each layer corresponds to a two hour interval of a day. The data of each 2 hour interval is averaged and discretized via quantiles. To this end, we compute 0.0-, 0.2-, 0.4-, 0.6-, 0.8-, and 1.0-quantiles of 2-hour-averages for each sensor. These quantiles define up to 5 non-overlapping intervals, and each 2-hour average is then replaced by the corresponding interval number. Finally, missing values are replaced by the median measurement of the corresponding sensor.

Due to the small number of available data points (Table 3.3), the regularization based structure estimation that we used for the synthetic data did not result in meaningful conditional independence structures—the models where almost fully connected. Instead, we choose the optimal tree structure as base graph, computed via the Chow-Liu algorithm (cf. Section 2.3.4).

M1	- - - - -	Rational	▲	Inv. Rational	△
Linear	▼	Exponential	■	Inv. Exponential	□
Quadratic	◆	Inv. Quadratic	◇	M2	●
Cubic	⬟	Inv. Cubic	⬠		

Figure 3.9: Key for Figures 3.10 to 3.17 to indicate the results for models **M1**, **M2**, and **M3** with various decay types. Each decay type corresponds to an **M3** model.

Table 3.4: Empirical upper bounds on the variances of each experiment on synthetic data.

Name	$\tilde{\mathbb{V}}^*$[MSE]	$\tilde{\mathbb{V}}^*$[NNZ Ratio]	$\tilde{\mathbb{V}}^*$[millis/iter]
Chain	0.0008128239666544	0.16065211567913	29.3399973505194
Star	0.000968238996476002	0.158395315017228	176.53025861319
Grid	0.000789754555558402	0.127438307225791	11.6477079330723
Full	0.0007800014778184	0.114628416485514	18.2190416116637

3.6.2 Results

Results on the synthetic data sets are presented in Figures 3.10–3.13, and results on the real-world data sets are shown in Figures 3.16–3.17. The dashed line indicates the result for model **M1**, black solid circles represent results for model **M2**, and other symbols correspond to results for model **M3**—the common key of all plots is shown in Fig. 3.9. In total, the results represent 32805 learning runs. Error bars are ommited in favor of readability. Instead, we provide the empirical worst-case variances in Table 3.4. While variances of mean squared error and runtime-per-training-iteration are small, the variance of the number-of-non-zero-ratio (parameter density) sticks out. This is not a surprise, since the achievable sparsity depends on the actual parameters, and we compute the variances over random model parameters. The MSE and the runtime do indeed not depend on the actual model parameters and have hence much lower variances.

We explain and discuss the results w.r.t. to the questions stated at the beginning of this Section.

Q1 Can we observe an increase in sparsity?

Converting redundancy into sparsity is the main motivation for our regularized reparametrization approach. First, we investigate the response of **M1**–**M3** to an increase in the regularization weight λ. Corresponding results can be found in the NNZ-Ratio plots of Figures 3.10–3.17. And indeed, increasing λ leads to a lower proportion of non-zero parameters, and thus an increased sparsity. In all cases, **M2** and **M3** models exhibit a lower NNZ-ratio than the unregularized baseline MRF **M1**. Moreover, the sparsity of regularized models increases at the same rate.

Inverse decay types take more information from previous time steps into account. One could hence expect that they may lead to an increased sparsity, compared to their regular counterparts. But taking a closer look at the different decay types reveals, that there is no strict order between inverse and non-inverse decay types. All sparsity levels are close to each other, with the regular exponential decay leading to the highest sparsity, followed by the regular rational decay. In most cases, the **M2** models deliver a lower sparsity than our proposed reparametrized models. This behavior is stable over all graphical structures that we considered in our experiments.

Similar observations can be made when we consider sparsity as a function of the model's redundancy (Fig. 3.11). The rate at which the sparsity of **M2** and **M3** models increases is now different. **M2** models are almost rigid against an increased redundancy. At very high redundancy, the NNZ-Ratio drops at least for star and full structures, and stays at the same level on chain and grid structures. Instead, the average sparsity of **M3** models increases with increasing sparsity and exhibits a large increase in regimes of high redundancy on all graphical structures.

Lastly, we investigate how sparsity behaves as a function of the model's depth T (Fig. 3.12). For a low number of time steps ($T < 64$), the sparsity of models with regular linear, quadratic, and cubic decay is lower than that of the **M2** model and models with inverse decay. However, this changes when the depth is increased. In all experiments, the difference in sparsity between **M2** and **M3** models (with regular decay) diverged. This is a strong evidence for the intuition that deep spatio-temporal models of type **M2** become more complex with increasing depth, while **M3** models tend to become less complex (*easier*) when reparametrizations with regular decay types are used.

Altogether, we can conclude that an increase in sparsity can be observed for both, **M2** and **M3** models. Regular rational and exponential decays deliver the highest sparsity. But being sparse alone is not enough, sparse models should still provide a reasonable quality.

Q2 Do compressed models still provide a reasonable quality?

We do now investigate the quality of sparse models. To this end, we compare the mean squared error $\|\tilde{\boldsymbol{\mu}} - \hat{\boldsymbol{\mu}}\|_2^2/d$ between the empirical marginals $\tilde{\boldsymbol{\mu}}$ and the model's estimate $\hat{\boldsymbol{\mu}}$ for **M1**, **M2** and **M3** models, estimated via 1000 iterations of loopy belief propagation.

Results for **M2** and **M3** models can be found in the MSE plots of Figures 3.10–3.17). Treating the MSE as a function of λ (Fig. 3.10,3.15–3.17), it becomes evident that sparse **M2** models exhibit a considerably larger MSE than almost all **M3** models. The only exception constitute models with regular exponential decay. This confirms our intuition which we used to design the reparametrization. Strong regularization leads to very sparse **M2** models, but destroys the underlying conditional independence structure, while sparse **M3** models keep this structure intact. The plot also reveals that the decay types which provided the highest sparsity, namely rational and exponential decay, have also the largest MSE. However, the MSE of regular rational models is still below the MSE of models with plain l_1-regularization. It is also interesting that the uncompressed **M1** models provide a superior low MSE of $\approx 10^{-10}$ on synthetic data (the dashed line is so close to 0 that it cannot be seen in the plots), but on all real-world data sets, their error has the same order of magnitude than that of **M2** and **M3** models. This could indicate that the parameters of the model that generated the real-world data follow a substantially different distribution than the one that we used to generate the synthetic data.

Regarding the redundancy that we injected into our synthetic data, we can see from Fig. 3.11 that the MSE is almost constant w.r.t. the redundancy. If anything, then a slight decreasing trend can be observed for all models types. The same applies to the MSE as a function of the models depth (Fig. 3.12). Again, models with exponential decay and rational decay show a larger error than the other decay types.

All in all, **M2** models trade sparsity against quality, while our proposed **M3** models, especially those with rational decay, are capable of being sparse and achieving a reasonable small error at the same time.

Q3 How does the spatio-temporal reparametrization influence the computational complexity?

Since most things are not for free, we may reckon that some kind of tradeoff exists. We hence investigate the empirical runtime of the models. More precisely, we consider the average runtime (in milliseconds) per training iteration as shown in Figures 3.13–3.17. Results on MSE and NNZ-ratio are very close for different decay types. This changes when we consider the runtime. The experimental results on the synthetic data (Figures 3.13 and 3.14) reveal an ordering of decay and model types. In fact, **M2** models have the shortest runtime per iteration, followed by the **M1** model, followed by **M3** models with regular decay, followed by **M3** models with inverse decay. In some cases, the runtime difference between sparse **M3** models with rational or exponential decay and **M1** is below one millisecond.

This result is intuitive if one recalls what kind of computation has to be performed for each model. The asymptotic complexity of **M1** and **M2** models is equivalent, and since arithmetic which involves many zeros can be carried out faster, sparse **M2** models are faster than their dense **M1** counterparts. On the other hand, **M3** models have to "decode" the natural exponential family parameters at runtime and hence suffer from a computational overhead. Regular decay types assign the coefficient ≈ 0 to parameters which are far from the parameter at time t. Hence, decoding their natural parameters enjoys the same effects which make **M2** models faster than **M1** models. Inverse decay types assign non-zero weights to almost all preceding parameters and are hence inherently slower than reparametrizations with regular decay. This behavior can be observed in all experiments on synthetic data, no matter if we treat the runtime as a function of the regularization weight, as a function of the redundancy, or as a function of the models depth.

Similar observations can be made in the experimental results on real-world data.

But in some cases, the order is not preserved, e.g., models with regular decay can sometimes be slower than models with inverse decay. This can be explained with the rather shallow depth of our real-world models ($T = 12$), since a higher depth implies a larger performance penalty for inverse decay types.

More surprising is that in some cases, **M3** models are actually faster than **M1** and even **M2** models on the real-world data. This signifies that the parameters learned by **M3** models may ease the computation of loopy belief propagation in some undiscovered way—at least on our real-world data. However, we cannot preclude the possibility that this behavior is triggered by imperfections in the real-world data which are not present in the synthetic data.

To sum up, the theoretical runtime penalty of **M3** models can be observed in our experiments. In many cases, this penalty amounts to a few milliseconds, and in some other cases, the penalty is not existing at all. Our reparametrized models are thus a practical alternative to classic undirected models.

3.7 Discussion

Resource constraint systems can have highly limited storage capabilities. In this chapter, we investigated the inherent memory requirements of exponential family members. This led to a principled overview on the memory complexity of potential functions and inference methods. The concept of sufficient statistics allows the memory complexity of exponential family models to be independent of the number of training data points. Instead, we identified the model parameter vector as the most relevant source of memory consumption. Techniques for the reduction of the parameter dimension are known from applications in statistical physics and natural language processing—both based on the reparametrization technique and are restricted to specific applications. Multivariate sensor data arises frequently in resource-constrained systems, but a direct application of the known techniques is not possible. We proposed generalized sequence structures, a sub-class of exponential family models with a special type of conditional independence structure for multivariate sensor data. By combining piecewise linear reparametrizations and regularization, redundant parts of a model could be identified and converted to sparsity, which reduces the memory requirements of the model. We showed how these models can be estimated consistently with various reparametrization types, characterized via different decay matrices. Theoretical properties of some canonical decay types were investigated. We explained how the proposed technique can be applied to continuous state space models and discussed the relation to dynamic Bayesian networks. Extensive experiments on synthetic and real-world multivariate sequence data showed, that we are capable to learn sparse models while preserving a high quality. This could not be achieved with existing regularization techniques, since those destroy the underlying conditional independence structure while our models keep this structure intact. Our reparametrization has to be decoded at run time and hence exhibit a penalty on their time complexity that is proportional to the models depths. Nevertheless, when the memory of the underlying resource-constrained system is too small for the full model, our compressed models might be the only practical way. In such situations, a time-space-tradeoff can be acceptable.

The reparametrization was first published in [169]. Therein, essential theoretical properties like universality of reparametrizations and consistent estimation were omitted and the discussion and experiments were restricted to the rational decay type. Here, we put a strong emphasis on the learning of models, which is also reflected in the experiments that were conducted in this Chapter. Orthogonal results on the classification performance of generalized sequence models can be found in [147, 169, 7, 174, 173, 137].

A more technical approach to reduce the effective memory consumption of a model is the reduction of the bit-width of the underlying data type, e.g. 16 bit instead of 64 bit. We will discuss this opportunity further in the upcoming chapter in the context of arithmetic constraints.

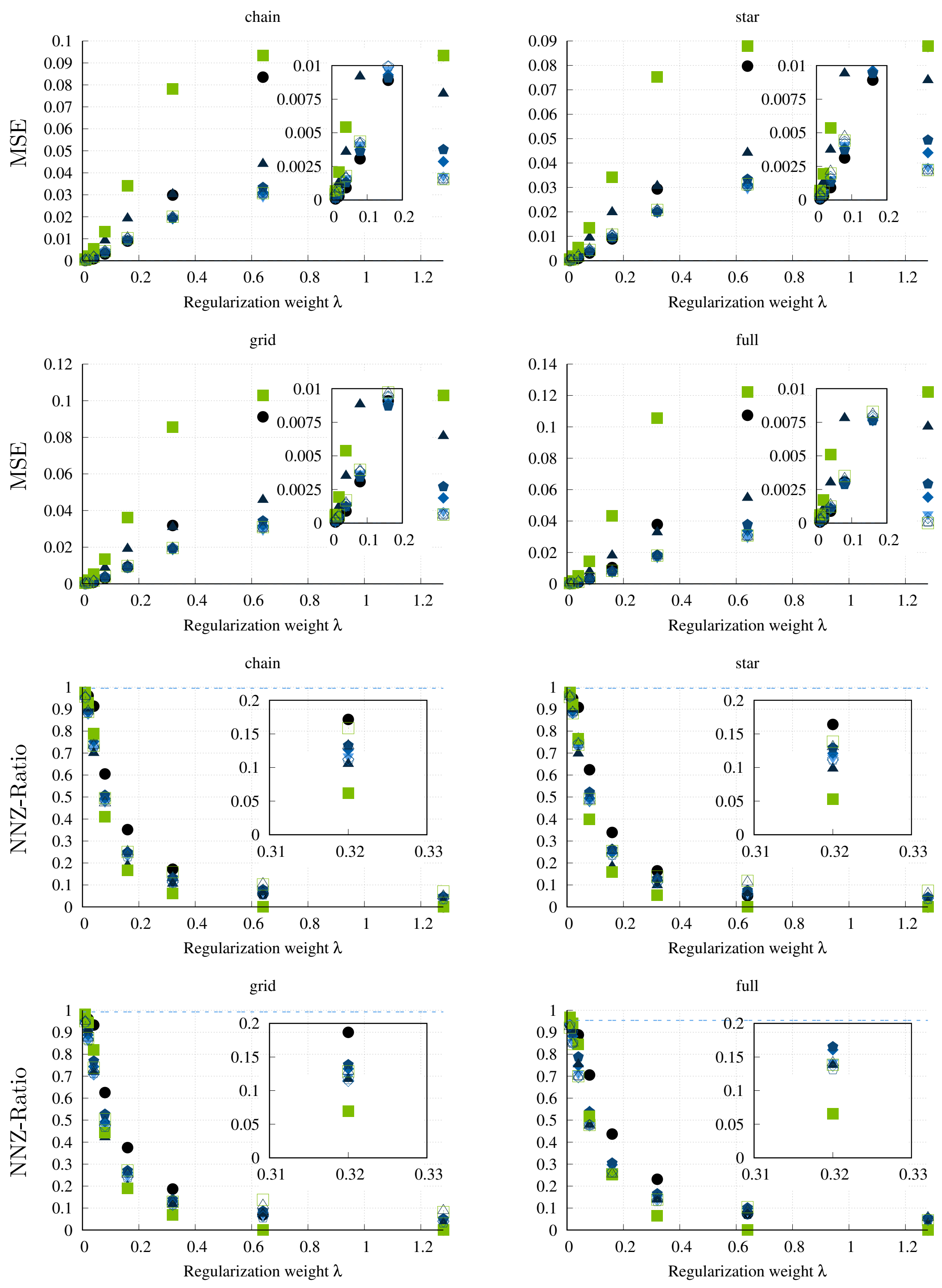

Figure 3.10: Average mean squared error in estimated marginals (y-axis, first two rows), and average number of non-zero parameters (y-axis, last two rows), as a function of the regularization weight λ (x-axis). The results are averaged over multiple runs for different time-slices $T \in \{16, 32, 64, 128\}$, and five levels of redundancy $\{0, 1/4, 1/2, 3/4, 1\}$. Different colors indicate different decay types (see Fig. 3.9).

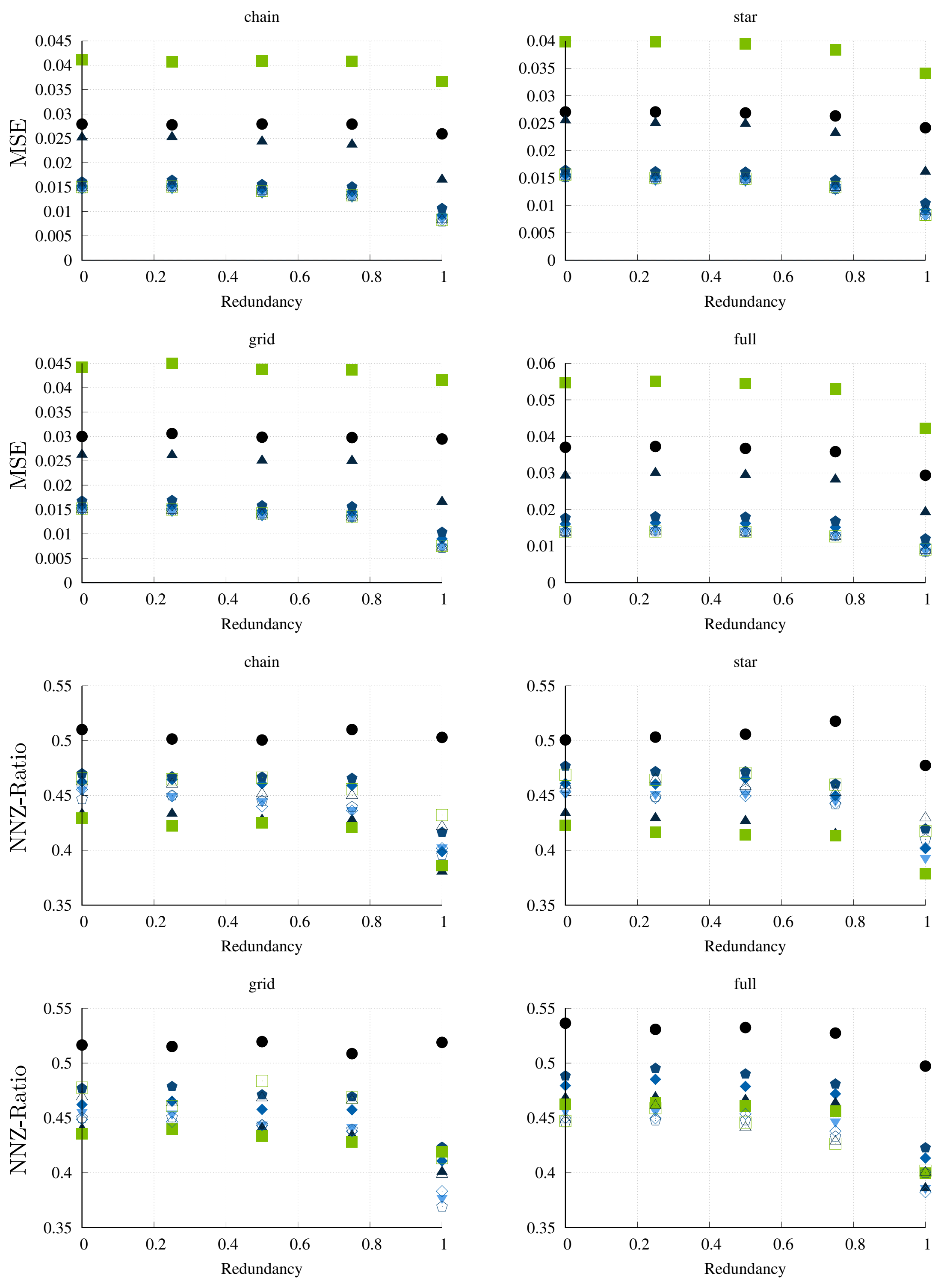

Figure 3.11: Average mean squared error in estimated marginals (y-axis, first two rows), and average number of non-zero parameters (y-axis, last two rows), as a function of the redundancy (x-axis). The results are averaged over multiple runs for different time-slices $T \in \{16, 32, 64, 128\}$, and various regularization weights $\{2^0 \times 10^{-2}, 2^1 \times 10^{-2}, \ldots, 2^7 \times 10^{-2}\}$. Different colors indicate different decay types (see Fig. 3.9).

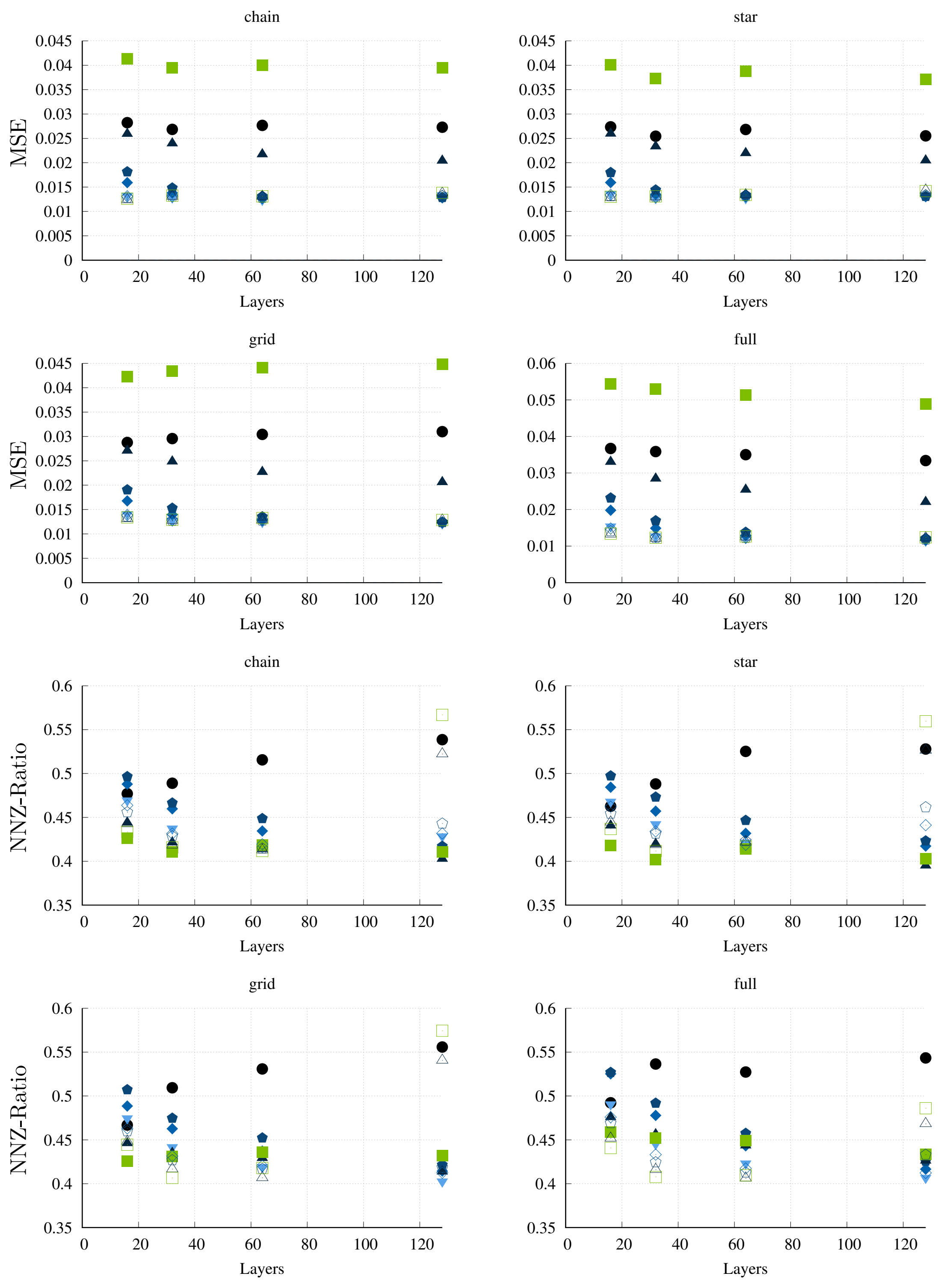

Figure 3.12: Average mean squared error in estimated marginals (y-axis, first two rows), and average number of non-zero parameters (y-axis, last two rows), as a function of the number of time-slices T (x-axis). The results are averaged over multiple runs for five levels of redundancy $\{0, 1/4, 1/2, 3/4, 1\}$, and various regularization weights $\{2^0 \times 10^{-2}, 2^1 \times 10^{-2}, \ldots, 2^7 \times 10^{-2}\}$. Different colors indicate different decay types (see Fig. 3.9).

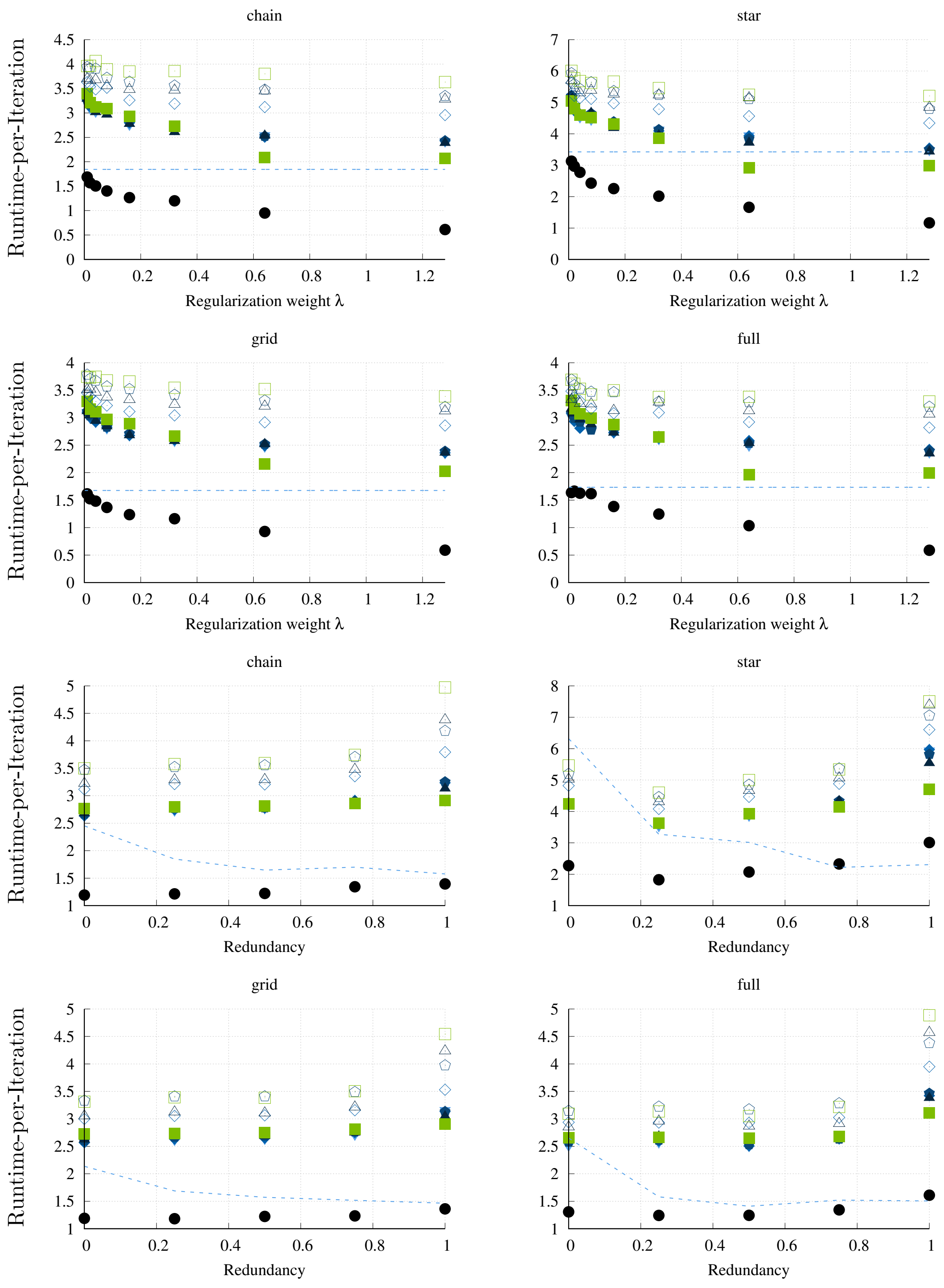

Figure 3.13: Average runtime per training iteration (milliseconds, y-axis) as a function of the regularization weight λ (x-axis, first two rows), and as a function of the redundancy (x-axis, last two rows). The results are averaged over multiple runs. Different colors indicate different decay types (see Fig. 3.9).

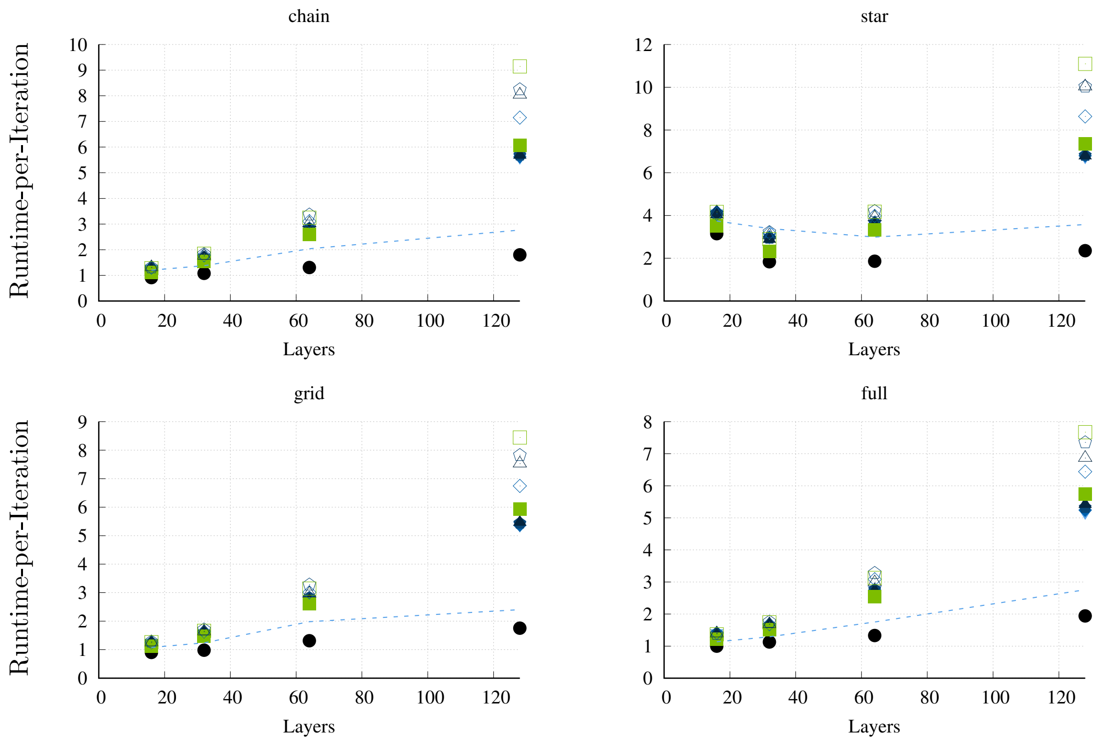

Figure 3.14: Average runtime per training iteration (milliseconds, y-axis) as a function of the model depth T (x-axis, first two rows). The results are averaged over multiple runs. Different colors indicate different decay types (see Fig. 3.9).

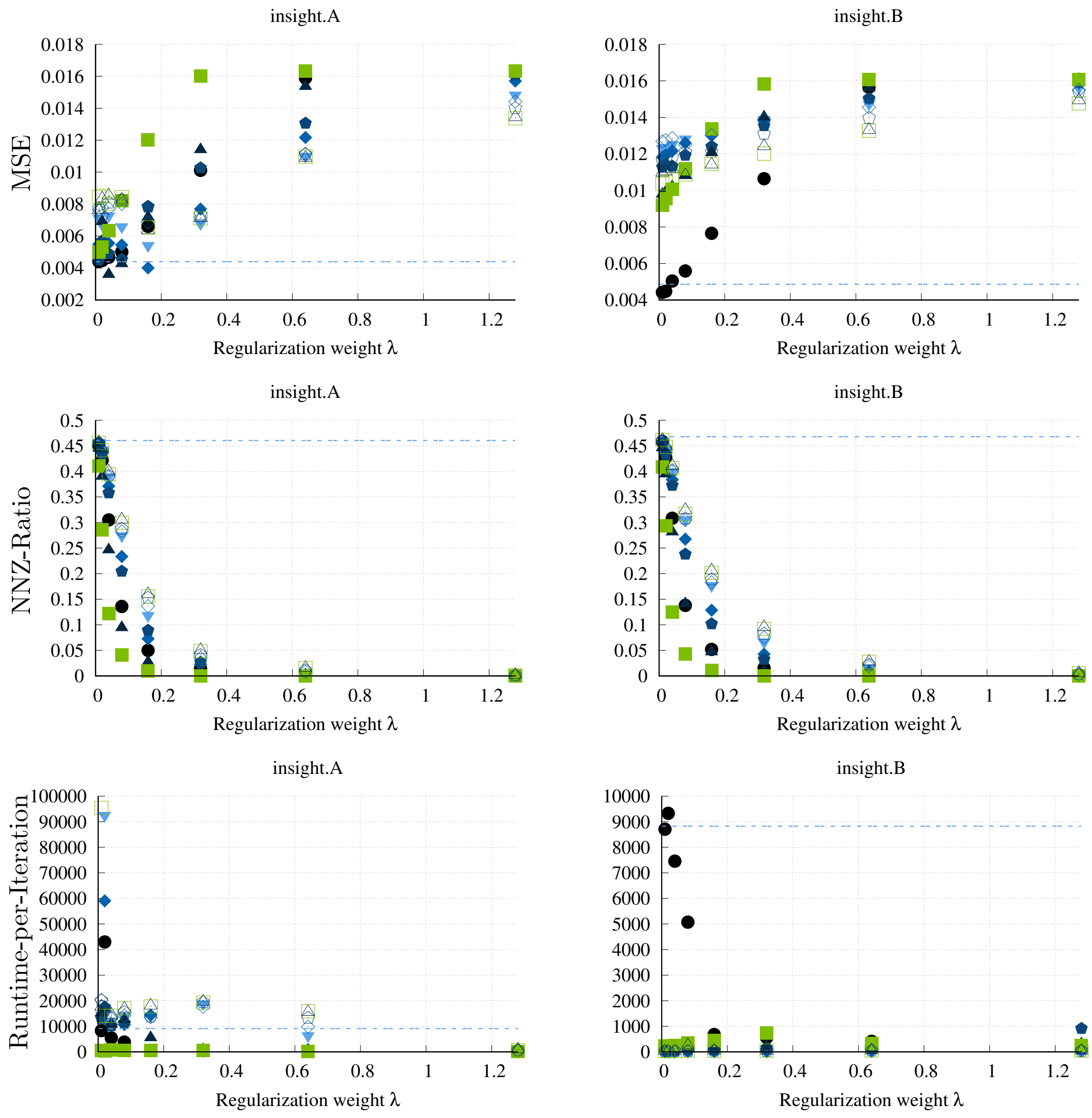

Figure 3.15: Experimental results on the INSIGHT data. Average MSE between estimated and empirical marginals (first row), average number of non-zero parameters (second row), and average runtime per training iteration in milliseconds (last row), as a function of the regularization weight (x-axis). Different colors indicate different decay types (see Fig. 3.9).

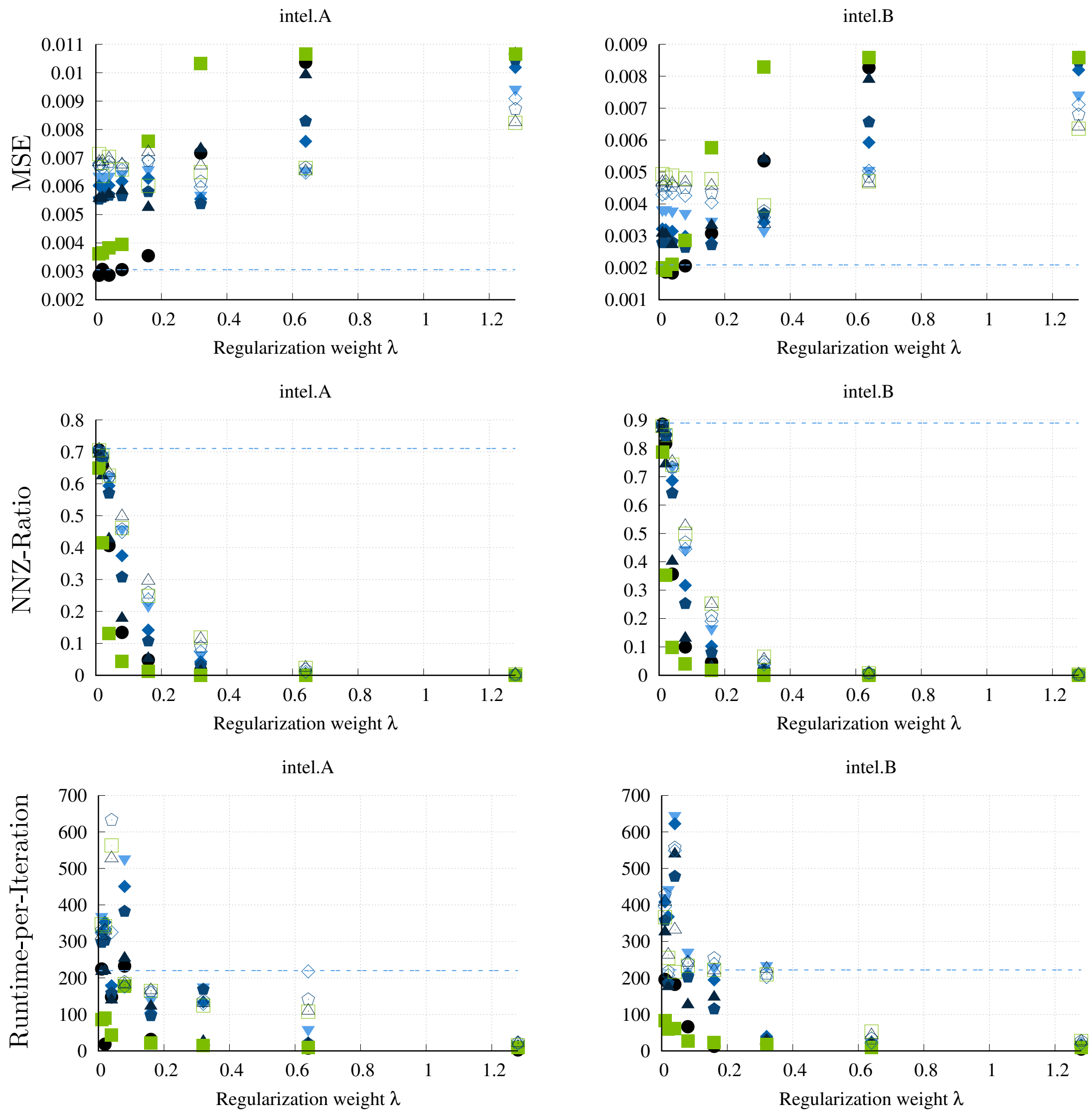

Figure 3.16: Experimental results on the Intel Lab data. Average MSE between estimated and empirical marginals (first row), average number of non-zero parameters (second row), and average runtime per training iteration in milliseconds (last row), as a function of the regularization weight (x-axis). Different colors indicate different decay types (see Fig. 3.9).

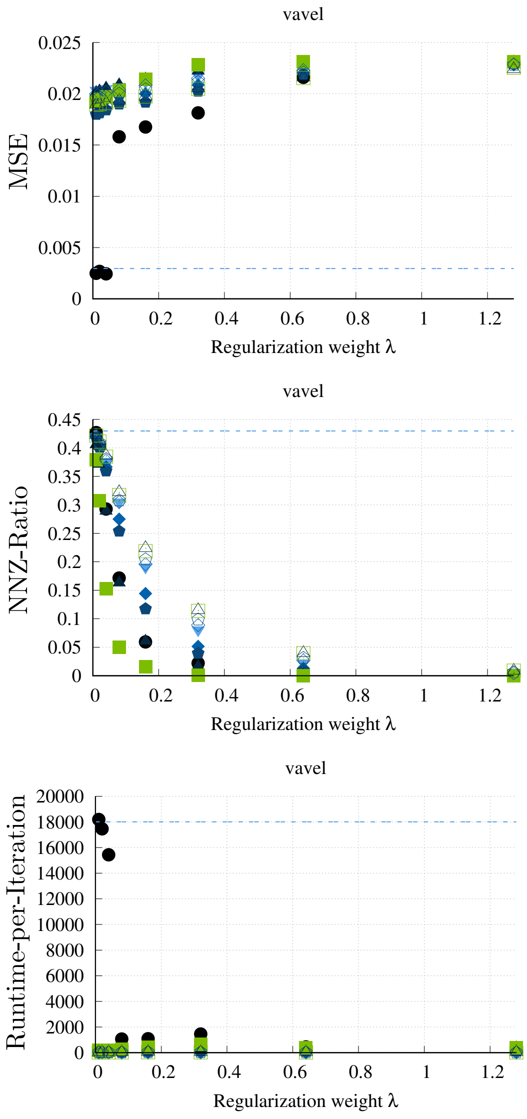

Figure 3.17: Experimental results on the VaVeL data. Average MSE between estimated and empirical marginals (first row), average number of non-zero parameters (second row), and average runtime per training iteration in milliseconds (last row), as a function of the regularization weight (x-axis). Different colors indicate different decay types (see Fig. 3.9).

4 Arithmetic Constraints

Computing the probability of joint states, marginal probabilities, and parameter estimation, require the arithmetic evaluation of the model, in particular, the evaluation of clique potentials $\psi_C(\boldsymbol{x}) = \exp(\langle \boldsymbol{\theta}_C, \phi_C(\boldsymbol{x}) \rangle)$. Any system on which we want to evaluate ψ_C should at least be able to perform multiplication, addition, and exponentiation. Moreover, since $\boldsymbol{\theta} \in \mathbb{R}^d$ and $\phi(\boldsymbol{x}) \in \mathbb{R}^d$ are real-valued vectors, a simulation of real-valued arithmetic is required via, say, the IEEE standard for floating-point arithmetic (IEEE 754). Although very basic, these operations can become prohibitively expensive when we consider computationally weak systems. In particular, ultra-low-power systems may not have hardware circuits for floating-point arithmetic. An obvious reason for such a constraint is the sheer physical size of a floating-point coprocessor—we might have to use a very small system whose size constraints have no room for a floating-point logic. Another reason is indeed the additional energy consumption, provoked by the additional hardware. It can be seen in Table 4.1 that the execution of floating point instructions consumes more energy than corresponding integer instructions.

Systems without floating-point units have to emulate the corresponding arithmetic operations in software, based on low level logic operations and integer arithmetic. Even if we resort to fixed-point arithmetic, multiple integer instructions have to be composed to emulated real-valued arithmetic. Compared to hardware implementations, the emulation requires substantially more clock cycles and hence, more time and more energy. Consequently, if we are forced to work with such constraints, we should resort to simple algorithms which do not require any real-valued arithmetic at all. At a first glace, this seems to inherently contradict our goal to study exponential family models on resource-constrained systems, since the evaluation of ψ_C requires d real-valued multiplications, d real-valued additions and one evaluation of the exponential function. Nevertheless, based on the very definition, we will derive and study a proper sub-class of the exponential family, in which the potential function can be evaluated solely with unsigned integer arithmetic[17].

Based on the integer subclass of exponential families, we propose and analyze probabilistic inference and optimization procedures, which allow us to learn parameters without any need for real-valued computation. To be more precise, we devise a variant of BP, called bit-length propagation, to compute approximations to the marginal probabilities and the maximum a posteriori assignment. Moreover, we derive an integer upper bound on the negative log-likelihood and investigate its minimization via our novel integer gradient descent method. In our experiments, we consider different levels of integrality to

[17]Note that each system with random-access memory must have at least one integer unit to perform address calculations.

Table 4.1: Approximate energy consumption of arithmetic instructions in picojoule for 45nm chips [97].

Data type	`int`		`float`	
Bit-width	8 bit	32 bit	16 bit	32 bit
Addition	0.03pJ	0.1pJ	0.4pJ	0.9pJ
Multiplication	0.2pJ	3.1pJ	1.1pJ	3.7pJ

characterize which models can reliably be learned with our new methods.

This opens up the opportunity of running probabilistic models on very small, resource-constrained systems. Moreover, the reduction of available decimal digits removes spurious arithmetic precision, which may readily be interpreted as regularization—it prevents the model to overfit to numerical particularities of the data. As a side-effect, the memory consumption is reduced by the use of data types with low bit-width.

4.1 Low-Precision Machine Learning

Understanding the impact of low-precision arithmetic, e.g. fixed-point arithmetic, is an active research area in machine learning [80, 37, 138]. However, most aspects of this topic have not been studied rigorously. The results are mostly heuristic, empirical and not directly applicable to undirected probabilistic models.

Probabilistic classifiers based on directed models, so called Bayesian network classifiers (BNC), have been analyzed w.r.t. the implications of parameter quantization [209, 208]. More precisely, worst-case and probabilistic bounds on the classification error for different bit-widths have been derived. Moreover, the authors compared the classification performance and the robustness of BNCs with generatively and discriminatively optimized parameters, i.e. parameters optimized for high data likelihood and parameters optimized for classification, with respect to parameter quantization. While this study delivers valuable theoretical insights about the effects of parameter quantization, the analyzed models require real-valued computation for parameter estimation and inference.

Similar results have been observed for neural networks [116, 80, 138]. The authors investigate different types of rounding. Their results suggest, that the classification performance of neural networks with 16 bit fixed-point arithmetic can deliver state-of-the art performance w.r.t. the prediction error on a test set. Moreover, it is shown that using low-precision arithmetic allows for energy efficient hardware implementations of deep neural networks, compared to ordinary CPU/GPU implementations. In [187], the authors provide a bound on the misclassification rate in presence of limited precision, for networks whose parameters are estimated under full precision, without any arithmetic constraint.

Some authors go even further and restrict the parameters or activation functions of neural networks to the domain $\{-1, +1\}$ [49, 100]. Despite this apparently strong limitation, near state-of-the-art results on typical benchmark data sets are achieved. Nevertheless, these methods rely internally on real-valued computations for both, parameter estimation and prediction.

In what follows, we close this gap, at least for undirected models, by proposing a proper sub-class of the exponential family, for which pure unsigned integer arithmetic suffices.

4.2 Integer Exponential Families

To construct a model that does not require any real-valued computation, we assume that our random variable $\boldsymbol{X}$ has a discrete state space $\mathcal{X}$. However, any density value generated by an exponential family member is, by definition, a fractional number[18]. To overcome this fact, our first intermediate goal is the derivation of exponential family members which are restricted to output rational density values.

As an initial step towards this goal, consider the derivation of the exponential families via the maximum entropy principle (Theorem 2.3) or from sufficient statistics (Theorem 3.1). In either way, the exponential function arises through the inversion of a $\log p$ term. The $\log p$ term itself originates from either the entropy or the log-likelihood.

An important observation is, that no matter which road we take to derive the exponential family, the choice of the logarithm's base is arbitrary. Without any specific reason, the natural logarithm (to the base e) is chosen in the proofs of Theorems 2.3 and 3.1. If we select $\log_b$ instead, it is straightforward to walk through the proofs and derive the base-b exponential family.

Corollary 4.1 (Base-b Families) *The exponential family is independent of the base. All exponential family densities can be written in base-b form, i.e.,*

$$p_{b,\boldsymbol{\theta}}(\boldsymbol{X} = \boldsymbol{x}) = b^{\langle \boldsymbol{\theta}, \phi(\boldsymbol{x}) \rangle - A_b(\boldsymbol{\theta})}$$

with

$$A_b(\boldsymbol{\theta}) = \log_b Z_b(\boldsymbol{\theta}) = \log_b \int_{\mathcal{X}} b^{\langle \boldsymbol{\theta}, \phi(\boldsymbol{x}) \rangle} \,\mathrm{d}\,\nu(\boldsymbol{x}) \;.$$

Moreover, we may transition arbitrarily between bases via the reparametrization

$$\eta_{a,b}(\boldsymbol{\theta}) = \frac{\log b}{\log a} \boldsymbol{\theta} \;.$$

I.e., $\forall \boldsymbol{x} \in \mathcal{X} : p_{a,\boldsymbol{\theta}}(\boldsymbol{x}) = p_{b,\eta_{a,b}(\boldsymbol{\theta})}(\boldsymbol{x})$. The set of all base-$b$ densities is denoted by $\mathcal{F}_b$. The set of all base-b densities with $b \in \mathbb{N}$ is called the base-$\mathbb{N}$ exponential family.

Of course, the above reparametrization is a direct consequence of $x^z = \exp(z \log x)$.

The next step towards integer exponential families is a restriction of our parameter space to the set of integers. Maybe surprising, the following lemma shows that we already reached our first goal.

[18] Recall that exp is never 0, and hence, $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ cannot be zero or one if p is in the exponential family.

Theorem 4.1 (Integer Parameters and Exponential Families [170, 171]) *Let $\boldsymbol{X}$ be a discrete random variable with base-$\mathbb{N}$ exponential family density $p_{b,\boldsymbol{\theta}}(\boldsymbol{x})$. When $\boldsymbol{\theta} \in \mathbb{N}^d$, then, for each $\boldsymbol{x} \in \mathcal{X}$, $p_{b,\boldsymbol{\theta}}(\boldsymbol{X} = \boldsymbol{x}) \in \mathbb{Q}_+ \cap (0;1)$. Moreover, for any set of variables $U \subseteq V$, the corresponding marginals $p_{b,\boldsymbol{\theta}}(\boldsymbol{X}_U = \boldsymbol{x}_U)$ are also in $\mathbb{Q}_+ \cap (0;1)$.*

Proof. The base-b density can be rewritten as a quotient $p_{b,\boldsymbol{\theta}}(\boldsymbol{X} = \boldsymbol{x}) = \psi_b(\boldsymbol{x})/Z_b(\boldsymbol{\theta})$. Binary overcomplete sufficient statistics (Definition 2.6) can represent any conditional independence structure of multivariate discrete random variables. Hence $\forall \boldsymbol{x} \in \mathcal{X} : \phi(\boldsymbol{x}) \in \{0,1\}^d$. Thus, $\langle \boldsymbol{\theta}, \phi(\boldsymbol{x}) \rangle$ is integer, and since $b \in \mathbb{N}$, we can conclude that the numerator of $p_{b,\boldsymbol{\theta}}(\boldsymbol{X} = \boldsymbol{x})$ is also integer. Moreover, $\mathbb{N}$ is closed under addition and hence the denominator $Z_b(\boldsymbol{\theta}) = \sum_{\boldsymbol{x} \in \mathcal{X}} b^{\langle \boldsymbol{\theta}, \phi(\boldsymbol{x}) \rangle}$ must be an integer too. This shows that $p_{b,\boldsymbol{\theta}}(\boldsymbol{X} = \boldsymbol{x})$ is in $\mathbb{Q}$, and since $p_{b,\boldsymbol{\theta}}$ is normalized and always non-zero, the density of any $\boldsymbol{x}$ must be in $\mathbb{Q} \cap (0;1)$. Any sum of rational numbers is rational and thus the marginals of $p_{b,\boldsymbol{\theta}}$ are rational too. ■

Choosing an integer base and imposing the constraint that our model parameters are integers implies that each density is a rational number. According to Corollary 4.1, we may choose an integer base without sacrificing the expressiveness of our model class. Allowing only integer parameters, however, seems to be a rather strong restriction of the model space—the sub-class of exponential family members with integer parameters is small in the sense that it has Lebesgue measure 0. Let us analyze the impact of this restriction.

Definition 4.1 (Integer Reparametrization) *Let $\boldsymbol{\theta} \in \mathbb{R}^d$ be the parameter of some exponential family member. The unsigned integer reparametrization $\eta_{\lfloor \cdot \rfloor}$ consists of flooring $\boldsymbol{\theta}$, and shifting it into the non-negative orthant.*

$$\eta_{\lfloor \cdot \rfloor}(\boldsymbol{\theta})_i = \lfloor \boldsymbol{\theta}_i \rfloor + \lceil | \min_{j=1}^{d} \boldsymbol{\theta}_j | \rceil$$

Indeed, $\eta_{\lfloor \cdot \rfloor}(\cdot)$ is not a bijection, and hence not universal according to Definition 3.1. One may ask whether $\eta_{\lfloor \cdot \rfloor}(\boldsymbol{\theta})$ delivers integer parameters which are not arbitrary far from $\boldsymbol{\theta}$ in terms of log-likelihood (2.18). The following intuitive result shows, that the amount of sacrificed likelihood depends on the number of cliques scaled by the flooring error.

Theorem 4.2 (Log-Likelihood Error) *Let $\lfloor \boldsymbol{\theta} \rfloor$ be the element-wise flooring of the entries in $\boldsymbol{\theta} \in \mathbb{R}^d$, and let further $\boldsymbol{\epsilon} = \boldsymbol{\theta} - \lfloor \boldsymbol{\theta} \rfloor$. Suppose $\boldsymbol{\theta}$ is the parameter of a discrete exponential family member with overcomplete binary sufficient statistic. Then,*

$$\ell(\eta_{\lfloor \cdot \rfloor}(\boldsymbol{\theta}); \mathcal{D}) - \ell(\boldsymbol{\theta}; \mathcal{D}) \leq 2\|\boldsymbol{\epsilon}\|_2 |\overline{\mathcal{C}}(G)| \;,$$

where equality is attained when $\boldsymbol{\theta}$ is in $\mathbb{N}^d$.

Proof. By Definition 4.1, each component of $\eta_{\lfloor\cdot\rfloor}(\boldsymbol{\theta})$ is shifted by $\lceil|\min_{j=1}^{d}\boldsymbol{\theta}_j|\rceil$. Recall that exponential family models with overcomplete sufficient statistic are shift-invariant (Lemma 2.4). Thus, $\ell(\eta_{\lfloor\cdot\rfloor}(\boldsymbol{\theta});\mathcal{D}) = \ell(\lfloor\boldsymbol{\theta}\rfloor;\mathcal{D})$. Combining this with the quadratic upper bound on convex functions with Lipschitz continuous gradient (2.23), we get

$$\ell(\eta_{\lfloor\cdot\rfloor}(\boldsymbol{\theta});\mathcal{D}) - \ell(\boldsymbol{\theta};\mathcal{D}) = \ell(\lfloor\boldsymbol{\theta}\rfloor;\mathcal{D}) - \ell(\boldsymbol{\theta};\mathcal{D}) \leq \langle\boldsymbol{\mu},\boldsymbol{\epsilon}\rangle + \frac{L}{2}\|\boldsymbol{\epsilon}\|_2^2 \,,$$

where $\boldsymbol{\mu} = \nabla\ell(\boldsymbol{\theta};\mathcal{D})$. We know from Lemma 2.6, that the Lipschitz constant of the gradient is upper bounded by $2|\overline{\mathcal{C}}(G)|$. The statement of the theorem follows by applying the Cauchy-Schwarz inequality to $\langle\boldsymbol{\mu},\boldsymbol{\epsilon}\rangle$, and observing that $\|\boldsymbol{\mu}\|_2 = |\overline{\mathcal{C}}(G)|$ (by overcompleteness) and $\|\boldsymbol{\epsilon}\|_2^2 \leq \|\boldsymbol{\epsilon}\|_2$, since $\boldsymbol{\epsilon} \in [0;1)^d$. ■

The result formalizes the idea that integer models should deliver reasonable results whenever the true parameter is not too far from being integer. However, this statement has some caveats. Clearly, the worst-case distance for any number to the nearest integer is $1/2$. At a first glace, it seems that the worst-case for the integer reparametrization would hence be any parameter vector with $\boldsymbol{\theta}_i = \boldsymbol{w}_i + 1/2$ and $\boldsymbol{w} \in \mathbb{N}^d$. But in this case, we can simply invoke the shift-invariance lemma (Lemma 2.4) to get an equivalent pure integer parameter vector $\boldsymbol{\theta} + (1/2, 1/2, \dots, 1/2)^\top$. It is thus not obvious if the error of an integer model will be large in practice—we will revisit this question in our experimental demonstration.

Before we discuss the probabilistic inference and parameter estimation in integer models, we make an observation that strengthens our hope in that integer models require simpler arithmetic operations than ordinary exponential family members.

Remark 4.1 (Left Bit-Shifts) *The potential function of base-2 exponential families, i.e.,*

$$\psi_2(\boldsymbol{x}) = 2^{\langle\boldsymbol{\theta},\phi(\boldsymbol{x})\rangle} \,,$$

can be evaluated via a left bit-shift, denoted by $a \ll b$ for $a, b \in \mathbb{N}$. E.g.,

$$\psi_2(\boldsymbol{x}) = 1 \ll \langle\boldsymbol{\theta},\phi(\boldsymbol{x})\rangle \,.$$

No arithmetic unit for exponentiation is required.

4.3 Integer Probabilistic Inference

We discuss how to compute the partition function and the marginal probabilities of integer models. In particular, we review the inference methods from Section 2.2.2 and check if they can be applied to perform inference without the need for real-valued arithmetic.

4.3.1 General Variational Inference

One can think about at least two orthogonal interpretations of variational inference in the context of integer models. First, we may define the rational marginal polytope

$$\mathcal{M}_b^{\mathbb{Q}} = \{\boldsymbol{\mu} \in \mathbb{Q}^d \mid \exists\mathbb{F} \in \mathcal{F}_b : \mathbb{E}_{\mathbb{F}}[\phi(\boldsymbol{X})] = \boldsymbol{\mu}\} \,,$$

which contains all marginals, realizable by members of $\mathcal{F}_b$. Indeed, $\mathcal{M}_b^{\mathbb{Q}} \subset \mathcal{M}$. Let us fix the parameter $\boldsymbol{\theta} \in \mathbb{R}^d$ of an arbitrary exponential family member. The rational variational approximation of $p_{\boldsymbol{\theta}}$ is then

$$\log Z(\boldsymbol{\theta}) \leq \sup_{\boldsymbol{\mu} \in \mathcal{M}_b^{\mathbb{Q}}} \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle + \mathcal{H}(\mu) \ ,$$

where equality can only hold if and only if $\nabla A(\boldsymbol{\theta}) \in \mathcal{M}_b^{\mathbb{Q}}$. Without any further approximation of $\langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle$ or $\mathcal{H}$, this procedure inherently requires real-valued arithmetic.

On the other hand, suppose that $\bar{\boldsymbol{\theta}} \in \mathbb{N}^d$ is the integer parameter vector of some base-$\mathbb{N}$ model. Assume that $\mathcal{T}_b^{\mathbb{Q}} \supset \mathcal{M}_b^{\mathbb{Q}}$ is some tractable superset of the rational marginal polytope, like the naive mean field approximation, or the pairwise consistent Bethe approximation that corresponds to loopy belief propagation (cf. [227]). More precisely, consider the marginalization constraints

$$\forall v \in V : \sum_{x \in \mathcal{X}_v} \boldsymbol{\mu}_{v=x} = 1 \quad \text{and} \quad \forall \{v,u\} \in E : \forall x \in \mathcal{X}_v : \boldsymbol{\mu}_{v=x} = \sum_{y \in \mathcal{X}_u} \boldsymbol{\mu}_{vu=xy} \ . \tag{4.1}$$

The rational set of locally consistent marginals is then

$$\mathcal{T}_b^{\mathbb{Q}} = \{ \boldsymbol{\mu} \in \mathbb{Q}^d \mid \boldsymbol{\mu} \text{ satisfies } (4.1) \} \ .$$

Combining this with the tree-based entropy decomposition[19]

$$\mathcal{H}_{\text{Tree}}(\boldsymbol{\mu}) = \sum_{v \in V} \mathcal{H}_{\boldsymbol{\mu}}[\boldsymbol{X}_v] - \sum_{vu \in E(T)} \mathcal{I}_{\boldsymbol{\mu}}[\boldsymbol{X}_v, \boldsymbol{X}_u] \ ,$$

known from Section 2.3.4, we arrive at the rational Bethe variational problem

$$\sup_{\boldsymbol{\mu} \in \mathcal{T}_b^{\mathbb{Q}}} \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle + \mathcal{H}_{\text{Tree}}(\boldsymbol{\mu}) \ .$$

Due to the entropy term, real-valued arithmetic is still required to solve the above problem in its current form. It can be shown, however, that any fixpoint of loopy belief propagation corresponds to a critical point of the above problem [247, 227]. And if the underlying conditional independence structure is a tree, LBP delivers the exact marginals. Indeed, this result has been derived for general Markov random fields. Nevertheless, according to Theorem 4.1, marginals of base-$\mathbb{N}$ models with integer parameters are rational, and so are the LBP fixpoints. It will be revealed in the following Section, that not only the fixpoints, but any intermediate result is rational, since the messages itself are integer-valued.

[19]Here, $\mathcal{H}_{\boldsymbol{\mu}}[\boldsymbol{X}_v]$ is the entropy of a vertex v, computed from the marginals in $\boldsymbol{\mu}$; and $\mathcal{I}_{\boldsymbol{\mu}}[\boldsymbol{X}_v, \boldsymbol{X}_u]$ is the mutual information between vertices v and u, based on their marginals in $\boldsymbol{\mu}$.

4.3.2 Message Passing Algorithms

In what follows, we focus our discussion on plain belief propagation. Nevertheless, the exact same reasoning applies to the Shafer-Shenoy junction tree messages. Message passing algorithms are well suited to compute the rational marginals and the integer partition function of base-$\mathbb{N}$ exponential families, without requiring any real-valued computation.

Lemma 4.1 (Message Integrality [170, 171]) *Let $\boldsymbol{\theta} \in \mathbb{N}^d$ be an integer parameter vector, and let $p_{\boldsymbol{\theta}}$ be the density of a base-$\mathbb{N}$ model. Suppose that all messages are initialized with natural numbers. Any message computed via the update*

$$\boldsymbol{m}_{u\to v}(\boldsymbol{x}_v) = \sum_{\boldsymbol{x}_u \in \mathcal{X}_u} \psi_{vu}(\boldsymbol{x}_v, \boldsymbol{x}_u) \prod_{w \in \mathcal{N}(u)\setminus\{v\}} \boldsymbol{m}_{w\to u}(\boldsymbol{x}_u) \ ,$$

for vertices $u, v, w \in V$, or via the Shafer-Shenoy update

$$\boldsymbol{m}_{U\to C}(\boldsymbol{x}_{C\setminus U}) = \sum_{\boldsymbol{x}_{U\setminus C} \in \mathcal{X}_{U\setminus C}} \psi_U(\boldsymbol{x}_U) \prod_{W \in \mathcal{N}(U)\setminus\{C\}} m_{W\to U}(\boldsymbol{x}_{U\setminus W}) \ , \tag{4.2}$$

for cliques $U, V, W \in \overline{\mathcal{C}}(G)$, are integer.

Proof. ψ is integer-valued in base-$\mathbb{N}$ models with integer parameters, and sums of products of integers are integer, too. ■

Hence, LBP is already stated without any real-valued computation, whenever the underlying potential function is integer-valued. Nevertheless, recall that the integer width of a CPU is constrained by its word size ω, e.g., $\omega = 64$ in current 64 bit workstation CPUs. This means that $2^\omega - 1$ is the largest, natively representable, unsigned integer. The default behavior for values larger than $2^\omega - 1$ is to wrap them around, which corresponds to integer arithmetic modulo 2^ω. Instead, we may simply concatenate multiple words together via software libraries[20] for arbitrary precision arithmetic. If we are willing to use such a library, we are done here and could proceed to the topic of integer parameter estimation. Nevertheless, on small systems, such software libraries can incur an inacceptable performance overhead, similar to software emulated floating-point arithmetic. We hence assume that our system is too weak to emulate integers with arbitrary precision and take a closer look at the avoidance of overflows.

4.3.3 Bit-Length Propagation

The main reason for overflows is the summation of "large" numbers in (4.2)—the value of $\psi_{vu}(\boldsymbol{x}_v, \boldsymbol{x}_u) \prod_{w\in\mathcal{N}(u)\setminus\{v\}} \boldsymbol{m}_{w\to u}(\boldsymbol{x}_u)$ can quickly exceed $2^\omega - 1$ when the magnitude of the parameters or the number of neighbors is large enough. Simply ignoring overflows will destroy the semantics of the messages, rendering the resulting marginals unusable. An investigation of practical BP implementations reveals, that numerical instabilities also

[20] An open source implementation of arbitrary precision integer and rational arithmetic can be found here: `https://gmplib.org`.

appear in the ordinary floating-point version. To compensate for numerical issues, BP messages are often computed in the log-domain, i.e., $\log \boldsymbol{m}_{u\to v}$ is computed instead of $\boldsymbol{m}_{u\to v}$. But first, it is still required to exponentiate the log messages before summation—which still incurs overflows; and second, taking logarithms is not possible with integer arithmetic. Nevertheless, the base-2 logarithm can be bounded efficiently via a simple, integer based operation, which helps us to avoid this particular source of overflows.

Definition 4.2 (Bit-Length) *Let $n \in \mathbb{N}$ be any integer. The bit-length of n, i.e., the minimum number of bits required for the base-2 binary representation of n, is*

$$\mathrm{bl}(n) = \lfloor \log_2(n) \rfloor + 1 \, .$$

The bit-length of a positive integer is moreover equivalent to the position of the leftmost non-zero bit in the corresponding binary representation. Real-world processing units often support a hardware operation for the determination of the first non-zero bit. We can use this efficient operation to bound the base-2 logarithm.

Corollary 4.2 (Bit-Length Bound on the Logarithm) *Let $n \in \mathbb{N}$ be any integer. The base-2 logarithm is bounded by the bit-length.*

$$\mathrm{bl}(n) - 1 \leq \log_2(n) < \mathrm{bl}(n)$$

Based on this relation, we define new, approximate, messages, whose magnitude is logarithmic in the magnitude of the original messages [171]. Hence, overflows can be mostly avoided.

Definition 4.3 (Bit-Length Messages) *Let $\boldsymbol{\theta}$ be the parameter of a base-b exponential family. The variates $\boldsymbol{\beta}_{u\to v}(\boldsymbol{x}_v)$, computed via*

$$\boldsymbol{\beta}_{u\to v}(\boldsymbol{x}_v) = \mathrm{bl}\left(\sum_{\boldsymbol{x}_u \in \mathcal{X}_u} b^{\langle \boldsymbol{\theta}_{vu}, \phi(\boldsymbol{x}_{vu}) \rangle} \prod_{w \in \mathcal{N}(u)\setminus\{v\}} b^{\boldsymbol{\beta}_{w\to u}(\boldsymbol{x}_u)} \right) \tag{4.3}$$

are called bit-length messages.

Before we discuss how these messages can avoid overflows, let us analyze the resulting marginal probabilities. Any bit-length message is an upper bound on the corresponding true log-message. We "invert" the bit-length via base-2 exponentiation. To simplify notation in the following, let $M_u^f(x) = \prod_{w\in\mathcal{N}(u)} 2^{f_{w\to u}(x)}$, where f could be any type of message. In addition, let $M_{uv}^f(x) = M_u^f(x)/2^{f_{v\to u}(x)}$. The approximate vertex marginals are computed according to (2.8), i.e.,

$$\hat{p}_v(\boldsymbol{x}_v) = \frac{M_v^{\boldsymbol{\beta}}(\boldsymbol{x}_v)}{\sum_{x' \in \mathcal{X}_v} M_v^{\boldsymbol{\beta}}(x')} \, ,$$

and the edge marginals via

$$\hat{p}_{uv}(\boldsymbol{x}_{uv}) = \frac{2^{\boldsymbol{\theta}_{uv=\boldsymbol{x}_{uv}}} M^{\boldsymbol{\beta}}_{uv}(\boldsymbol{x}_u) M^{\boldsymbol{\beta}}_{vu}(\boldsymbol{x}_v)}{\sum_{x' \in \mathcal{X}_u} \sum_{y' \in \mathcal{X}_v} 2^{\boldsymbol{\theta}_{uv=x'y'}} M^{\boldsymbol{\beta}}_{uv}(x') M^{\boldsymbol{\beta}}_{vu}(y')} .$$

Although bl is not too far from $\log_2$, the error in the resulting marginals is hard to quantify, since multiple errors accumulate during message propagation. Our result on the error are hence rather existential. We show that under certain conditions, the expected divergence is small.

As a first step, let us analyze the difference between the true $\log_2$-messages and our bit-length messages.

Lemma 4.2 (Message Errors) *Let $E_{uv}(x) = \boldsymbol{\beta}_{u\to v}(x) - \log_2 \boldsymbol{m}_{u\to v}(x)$ be the message propagation error. Then,*

$$1 - \varepsilon_{uv}(x) \le E_{uv}(x) \le 1 - \varepsilon_{uv}(x) + \max_{y \in \mathcal{X}_u} \sum_{w \in \mathcal{N}(u)\setminus\{v\}} E_{wu}(y) ,$$

where $\varepsilon_{uv}(x)$ is the flooring error, inherent in the bit-length computation of $\boldsymbol{\beta}_{u\to v}(x)$.

Proof. Let ε be the fractional part of a positive real z, defined via $\lfloor z \rfloor = z - \varepsilon$, i.e., the flooring error. Moreover, let $\varepsilon_{uv}(x)$ be the corresponding flooring error, of the flooring involved in the bit-length computation of $\boldsymbol{\beta}_{u\to v}(x)$. We can then rewrite the message propagation error as

$$\begin{aligned} E_{uv}(x) &= \mathrm{bl} \sum_{y \in \mathcal{X}_u} 2^{\langle \boldsymbol{\theta}_{vu}, \phi_{vu}(x,y)\rangle + \log_2 M^{\boldsymbol{\beta}}_{uv}(y)} - \log_2 \sum_{y \in \mathcal{X}_u} 2^{\langle \boldsymbol{\theta}_{vu}, \phi_{vu}(x,y)\rangle + \log_2 M^{\log_2 m}_{uv}(y)} \\ &= 1 - \varepsilon_{uv}(x) + \log_2 \frac{\sum_{y \in \mathcal{X}_u} 2^{\langle \boldsymbol{\theta}_{vu}, \phi_{vu}(x,y)\rangle + \log_2 M^{\boldsymbol{\beta}}_{uv}(y)}}{\sum_{y \in \mathcal{X}_u} 2^{\langle \boldsymbol{\theta}_{vu}, \phi_{vu}(x,y)\rangle + \log_2 M^{\log_2 m}_{uv}(y)}} , \end{aligned}$$

with

$$\begin{aligned} \log_2 M^{\boldsymbol{\beta}}_{uv}(y) &= \sum_{w \in \mathcal{N}(u)\setminus\{v\}} \boldsymbol{\beta}_{w\to u}(y) = \sum_{w \in \mathcal{N}(u)\setminus\{v\}} \log_2 \boldsymbol{m}_{w\to u}(y) + E_{wu}(y) \\ &= \log_2 M^{\log_2 m}_{uv}(y) + \sum_{w \in \mathcal{N}(u)\setminus\{v\}} E_{wu}(y) . \end{aligned}$$

Combining this with the above and applying Hölder's inequality, yields

$$\begin{aligned} E_{uv}(x) &= 1 - \varepsilon_{uv}(x) + \log_2 \frac{\sum_{y \in \mathcal{X}_u} 2^{\langle \boldsymbol{\theta}_{vu}, \phi_{vu}(x,y)\rangle + \log_2 M^{\log_2 m}_{uv}(y) + \sum_{w \in \mathcal{N}(u)\setminus\{v\}} E_{wu}(y)}}{\sum_{y \in \mathcal{X}_u} 2^{\langle \boldsymbol{\theta}_{vu}, \phi_{vu}(x,y)\rangle + \log_2 M^{\log_2 m}_{uv}(y)}} \\ &\le 1 - \varepsilon_{uv}(x) + \max_{y \in \mathcal{X}_u} \sum_{w \in \mathcal{N}(u)\setminus\{v\}} E_{wu}(y) . \end{aligned}$$

Moreover, positivity of the error can be shown via induction over the order of message computation, assuming that all messages are initialized by 1. In this case, the error of the first computed message arises from Corollary 4.2, i.e., the magnitude of the bit-length message $\boldsymbol{\beta}_{u\to v}$ will be larger than the corresponding ordinary message $m_{u\to v}$. Hence, all upcoming bit-length messages will have an increased magnitude as well. The error E_{uv} must thus be positive (≥ 0). ■

In addition, we can establish a connection between the difference in marginals and the message propagation error.

Lemma 4.3 (Marginal Errors) *Let $E_{uv}(x) = \boldsymbol{\beta}_{u\to v}(x) - \log_2 \boldsymbol{m}_{u\to v}(x)$ be the message propagation error. The differences of* log *marginals computed from the true and the bit-length messages, respectively, are bounded via*

$$\log p_v(x) - \log \hat{p}_v(x) \leq \left(\max_{x'\in\mathcal{X}_v} \sum_{u\in\mathcal{N}(v)} E_{uv}(x') \right) - \sum_{u\in\mathcal{N}(v)} E_{uv}(x) \ ,$$

and

$$\begin{aligned}\log p_{vu}(x,y) - \log \hat{p}_{vu}(x,y) \leq & \left(\max_{x'y'\in\mathcal{X}_{vu}} \sum_{w\in\mathcal{N}(v)\setminus\{u\}} E_{wv}(x') + \sum_{w\in\mathcal{N}(u)\setminus\{v\}} E_{wu}(y') \right) - \\ & \left(\sum_{w\in\mathcal{N}(v)\setminus\{u\}} E_{wv}(x) \sum_{w\in\mathcal{N}(u)\setminus\{v\}} E_{wu}(y) \right) \ .\end{aligned}$$

Proof. We start from the definition of BP vertex marginals and apply some algebra.

$$\begin{aligned}\log p_v(x) - \log \hat{p}_v(x) &= \log \frac{M_v^{\log_2 m}(x)}{\sum_{x'\in\mathcal{X}_v} M_v^{\log_2 m}(x')} - \log \frac{M_v^{\boldsymbol{\beta}}(x)}{\sum_{x'\in\mathcal{X}_v} M_v^{\boldsymbol{\beta}}(x')} \\ &= \log \frac{\sum_{x'\in\mathcal{X}_v} M_v^{\log_2 m}(x') 2^{\sum_{u\in\mathcal{N}(v)} E_{uv}(x')}}{\sum_{x'\in\mathcal{X}_v} M_v^{\log_2 m}(x')} - \sum_{u\in\mathcal{N}(v)} E_{uv}(x) \\ &\leq \left(\max_{x'\in\mathcal{X}_v} \sum_{u\in\mathcal{N}(v)} E_{uv}(x') \right) - \sum_{u\in\mathcal{N}(v)} E_{uv}(x)\end{aligned}$$

Again, the inequality follows from Hölder's inequality. The bound for the edge marginals follows from exactly the same reasoning, and is hence omitted. ■

We can now combine the above lemmas to devise the following theorem.

Theorem 4.3 (Expected Kullback-Leibler Divergence) *Assume that the conditional independence structure $G = (V, E)$ is a tree. Suppose that the rounding errors, involved in each bit-length computation, are realizations of a uniform random variable ε. Moreover, assume that all rounding errors incurred in the message computation are* *independent of each other. Let further s be the largest vertex state space, l the length of the longest path in G, and let $r = \max_{v \in V} |\mathcal{N}(v)|$ be the maximum vertex degree. Then*

$$\mathbb{E}[\mathrm{KL}[\, p_v \parallel \hat{p}_v \,]] \leq r \sum_{i=0}^{l-1} (r-1)^i - \frac{l-1}{s+1} \,,$$

where p_v are the BP marginals, $\hat{p}_v$ are the marginals computed from bit-length messages, and the expectation is taken w.r.t. the random rounding error ε.

Proof. By the definition of Kullback-Leibler divergence, and Lemma 4.3, we have

$$\begin{aligned}
\mathbb{E}\left[\sum_{x \in \mathcal{X}_v} p_v(x) \log_2 \frac{p_v(x)}{\hat{p}_v(x)}\right] &\leq \mathbb{E}\left[\sum_{x \in \mathcal{X}_v} p_v(x) \left(\left(\max_{x' \in \mathcal{X}_v} \sum_{u \in \mathcal{N}(v)} E_{uv}(x')\right) - \sum_{u \in \mathcal{N}(v)} E_{uv}(x)\right)\right] \\
&\leq \sum_{x \in \mathcal{X}_v} p_v(x) \left(\sum_{u \in \mathcal{N}(v)} \mathbb{E}\left[\left(\max_{x' \in \mathcal{X}_v} E_{uv}(x')\right) - E_{uv}(x)\right]\right) \,. \qquad (4.4)
\end{aligned}$$

To establish the latter inequality, we made use of $\max_{x'} \sum_u E_{uv}(x') \leq \sum_u \max_{x'} E_{uv}(x')$, where the value can only increase if we maximize the terms of the sum individually. By Lemma 4.2, $E_{uv}(x) \geq 1 - \varepsilon_{uv}(x)$, which simplifies the above inequality to

$$\begin{aligned}
&\leq \sum_{x \in \mathcal{X}_v} p_v(x) \left(\sum_{u \in \mathcal{N}(v)} \mathbb{E}\left[\max_{x' \in \mathcal{X}_v} E_{uv}(x')\right] - 1 + \mathbb{E}\left[\varepsilon_{uv}(x)\right] \right) \\
&\leq -\frac{|\mathcal{N}(v)|}{2} + \underbrace{\sum_{u \in \mathcal{N}(v)} \mathbb{E}\left[\max_{x' \in \mathcal{X}_v} E_{uv}(x')\right]}_{\text{(I)}} \,. \qquad (4.5)
\end{aligned}$$

The second step results from uniform and independent rounding errors, i.e., $\mathbb{E}\left[\varepsilon_{uv}(x)\right] = 1/2$. Applying the upper bound from Lemma 4.2, we arrive at

$$\begin{aligned}
&\leq -\frac{|\mathcal{N}(v)|}{2} + \sum_{u \in \mathcal{N}(v)} \mathbb{E}\left[\max_{x' \in \mathcal{X}_v} 1 - \varepsilon_{uv}(x') + \max_{x'' \in \mathcal{X}_u} \sum_{w \in \mathcal{N}(u) \setminus \{v\}} E_{wu}(x'')\right] \\
&= -\frac{|\mathcal{N}(v)|}{2} + \sum_{u \in \mathcal{N}(v)} 1 - \mathbb{E}\left[\min_{x' \in \mathcal{X}_v} \varepsilon_{uv}(x')\right] + \mathbb{E}\left[\max_{x'' \in \mathcal{X}_u} \sum_{w \in \mathcal{N}(u) \setminus \{v\}} E_{wu}(x'')\right] \,.
\end{aligned}$$

For M uniform random variables $\boldsymbol{U}_i$ on $[0;1]$, we have $\mathbb{E}[\max_{i=1}^{M} \boldsymbol{U}_i] = 1/(M+1)$. Together with the max-sum inequality, known from (4.4), we rewrite the bound to

$$\leq \frac{|\mathcal{N}(v)|}{2} - \frac{|\mathcal{N}(v)|}{|\mathcal{X}_v| + 1} + \sum_{u \in \mathcal{N}(v)} \underbrace{\sum_{w \in \mathcal{N}(u) \setminus \{v\}} \mathbb{E}\left[\max_{x'' \in \mathcal{X}_u} E_{wu}(x'')\right]}_{\text{(II)}} \,. \qquad (4.6)$$

Comparing (4.5) and (4.6), we see that the terms (I) and (II) are similar. And indeed, by applying the very same reasoning to (II) that we already applied to (I) yields

$$\begin{aligned}\leq\ & \frac{|\mathcal{N}(v)|}{2} - \frac{|\mathcal{N}(v)|}{|\mathcal{X}_v|+1} + |\mathcal{N}(v)|(|\mathcal{N}(u)|-1) - \frac{|\mathcal{N}(v)|(|\mathcal{N}(u)|-1)}{|\mathcal{X}_u|+1} \\ +\ & \sum_{u\in\mathcal{N}(v)} \sum_{w\in\mathcal{N}(u)\setminus\{v\}} \underbrace{\sum_{h\in\mathcal{N}(w)\setminus\{u\}} \mathbb{E}\left[\max_{x\in\mathcal{X}_w} E_{hw}(x)\right]}_{\text{(III)}} .\end{aligned}$$

Again, we may proceed with unrolling term (III). Given that l is the length (number of vertices) of the longest path in the underlying graphical structure, we may apply the above reasoning at most l times until we hit at a leaf vertex. Collecting alike terms and using the bounds on $|\mathcal{N}(v)|$ and $|\mathcal{X}_v|$ finally yields

$$\leq \frac{r}{2} + r\sum_{i=1}^{l-1}(r-1)^i - \frac{l-1}{s+1} ,$$

where we used the fact that all inner vertices of the path have at least two neighbors, and that the magnitude of the $l-1$ negative terms is at least $1/(s+1)$. ■

The same reasoning can be applied to higher-order marginals as well. Independence and uniform distribution of rounding errors are merely technical assumptions. We may choose any other distribution for which expectation and expected minimum of ε can be computed in closed-form. Nevertheless, the theorem tells us how the error of our approximate messages is influenced by the graphical structure and the state space. When the state space increases, the negative part of the bound approaches 0, i.e., the influence of the state space size disappears. On the other hand, the influence of the conditional independence structure is unbounded in the sense that larger structures always imply larger errors. The extreme cases are chains and stars. For chain structures of length $l = n$, we have $r = 2$ and the error is dominated by the number of terms in the sum. For star graphs, we have $l = 3$ and the bound is dominated by the large degree of the center vertex. If G contains a single edge, the error at each vertex is strictly less then $^1/_2$. In case of other graphs, the bound depends crucially on both l and r.

Despite these facts, one has to keep in mind that the integer exponential family is designed for ultra-low-power devices—a setting in which large graphical structures might not even fit into the main memory.

If the method is applied to loopy graphs, the variable l can essentially be replaced by the number of iterations for which we run the message passing algorithm. Since we cannot assume the existence of leaf vertices in loopy graphs, the summation has l terms instead of $l-1$. But since running BP on loopy graphs is anyhow highly approximate, bounding the KL between LBP and BLprop marginals might not be meaningful.

A generic analysis of how error is propagated by BP has been conducted in [102]. Their measure of error, the dynamic range of potential functions, does not cover the KL between arbitrary marginals. Hence, we conducted the analysis above.

Algorithm 4.1: Computation of Bit-Length Messages with Sparse Integer Representation

```
input $(u, v)$, $x$
output $\boldsymbol{\beta}_{u \to v}(x)$
 1: $\boldsymbol{B} \leftarrow \emptyset$          // this structure stores the non-zero bits of the message
 2: for $y \in \mathcal{X}_u$ do
 3:     $i \leftarrow \langle \boldsymbol{\theta}_{vu}, \phi(x, y) \rangle + \sum_{w \in \mathcal{N}(u) \setminus \{v\}} \boldsymbol{\beta}_{w \to u}(y)$
 4:     while $\boldsymbol{B}$ contains $i$ do
 5:         Remove $i$ from $\boldsymbol{B}$
 6:         $i \leftarrow i + 1$
 7:     end while
 8:     Insert $i$ into $\boldsymbol{B}$
 9: end for
10: return $(\max \boldsymbol{B}) + 1$
```

4.3.4 Computing Bit-Length Messages

Bit-length messages have computationally convenient properties. In fact, we will now see how to employ an efficient sparse data structure to exploit that all summands in (4.3) are powers of two. The main *bit-length propagation* (BLprop) algorithm is mostly identical to LBP (Algorithm 2.1), whereas the ordinary messages are replaced by $\boldsymbol{\beta}_{u \to v}(x)$, and ϵ has to be provided as a rational number to check for convergence. Our algorithm for the actual message computation is shown in Algorithm 4.1. The most important property of this algorithm is the use of a sorted list structure to represent the message. While this might sound odd at first, each message $\boldsymbol{\beta}_{u \to v}(x)$ is the bit-length of a sum of $|\mathcal{X}_v|$ powers of two. Hence, the binary representation of the sum that corresponds to $\boldsymbol{\beta}_{u \to v}(x)$ has at most $|\mathcal{X}_v|$ bits set to 1. Storing the bit positions in a sorted list allows us to access the highest bit, which represents the bit-length of the sum—and hence $\boldsymbol{\beta}_{u \to v}(x)$—in time $\mathcal{O}(1)$. For the list, we need $\mathcal{O}(|\mathcal{X}_v|)$ temporary storage for the computation of arbitrary large messages. Note that a plain 64 bit integer will overflow soon if the number of neighbors or the magnitude of incoming bit-lengths is large. The sparse integer representation of the sum via a list cannot overflow, as long as the underlying type is large enough to hold the *position* of a bit. As an example, if the underlying integer type has 16 bits, the list allows us to represent any number whose binary representation requires less than 2^{16} bits. Without the list structure, only messages of to 16 bits would be representable. The list itself can be discarded after computation, and only its largest element is propagated.

Theorem 4.4 (Correctness of Message Computation) *Algorithm 4.1 computes the bit-length message $\boldsymbol{\beta}_{u \to v}(\boldsymbol{x}_v)$ for a base-2 exponential family.*

Proof. By definition,

$$\boldsymbol{\beta}_{u\to v}(\boldsymbol{x}_v) = \mathrm{bl} \sum_{\boldsymbol{x}_u \in \mathcal{X}_u} 2^{\langle \boldsymbol{\theta}_{vu}, \phi(\boldsymbol{x}_{vu})\rangle + \sum_{w\in\mathcal{N}(u)\setminus\{v\}} \boldsymbol{\beta}_{w\to u}(\boldsymbol{x}_u)} \ .$$

$\boldsymbol{\beta}_{u\to v}(\boldsymbol{x}_v)$ is the bit-length of a sum of powers of 2. In line 1 we allocate a list structure (or a tree, or a hash map) $\boldsymbol{B}$, to store the sparse integer representation of the message. Here, sparsity is meant w.r.t. to the binary encoding of the message it self. The actual summation takes place in lines 2–9. In line 3, we load the exponent i of the current summand, and lines 4–8 represent raw binary addition of 2^i to the current sum which is stored in $\boldsymbol{B}$. Checking if i is already contained in the list takes at most $\mathcal{O}(\log|\mathcal{X}_u|)$ steps via binary search in the sorted list. Since the summation is over $|\mathcal{X}_u|$ terms, the final outcome can only have $|\mathcal{X}_u|$ non-zero bits. Even if the result of this summation is a 1 million bit number, only $\mathcal{O}(|\mathcal{X}_u|)$ storage is required during message computation. In the last line, we return 1 plus the position of the most significant bit stored in $\boldsymbol{B}$, which is the bit-length of a binary number. Accessing the position of the most significant bit is performed in constant time by using a pointer to the end of the list. ■

Note that at the end of each message computation, +1 is added to the most significant bit position. This leads to an increased message magnitude. To see this, consider a simple chain model with n binary state vertices and a zero parameter vector $\boldsymbol{\theta} = \boldsymbol{0}$. Any outgoing message $\boldsymbol{\beta}_{1\to 2}(x)$ from vertex 1 to vertex 2 has the value 1. The message $\boldsymbol{\beta}_{2\to 3}(x)$ has the value 2, and so on, until the last message $\boldsymbol{\beta}_{(n-1)\to n}(x)$ is $n-1$. This behavior is not problematic due to our sparse message representation, but it shows that the magnitude of messages depends on the size of the graph. Without our sparse representation, the bit-length propagation would be restricted to small graphs and parameters with small magnitude.

The above procedure allows for nearly arbitrary large messages, but it does not guarantee a seamless computation of marginal probabilities. Recall that (vertex) marginals are computed via

$$p_v(x) = \frac{2^{\sum_{u\in\mathcal{N}(v)} \boldsymbol{\beta}_{u\to v}(x)}}{\sum_{y\in\mathcal{X}_v} 2^{\sum_{u\in\mathcal{N}(v)} \boldsymbol{\beta}_{u\to v}(y)}} \ .$$

While we can easily compute the (possibly very large) denominator via the sparse integer representation that we used in Algorithm 4.1, the result cannot be processed natively on the CPU due to the fixed word-size ω. In order to allow for native subsequent processing of the marginals (e.g., in another application without sparse integer support), we can shift numerator and denominator to the right until both fit into native CPU registers with, say, $\omega = 16$ bit word-size.

Definition 4.4 (Right Bit-Shifts) *Let $a, b \in \mathbb{N}$. The right bit-shift operation $a \gg b$ is defined as*

$$a \gg b = \left\lfloor \frac{a}{2^b} \right\rfloor \ .$$

Whenever a b bit sparse integer a has to be transferred to native program code, we store $a \gg (b-\omega)$ instead whenever a does not fit into a native ω bit register. While this procedure introduces a large error into any *single* number, shifting two numbers a and c by the same amount of bits b leaves their *ratio* almost intact, i.e., $a/c \approx (a \gg b)/(c \gg b)$. The following lemma formalizes this observation in the context of probability mass functions.

Lemma 4.4 (Probability Shifting) *Let $\boldsymbol{a}$ be some vector of integers, such that $p(x) = 2^{\boldsymbol{a}_x}/\sum_{y\in\mathcal{X}} 2^{\boldsymbol{a}_y}$ for some random variable $\boldsymbol{X}$ with state space $\mathcal{X}$. Let further ω be the word-size of the underlying CPU, and $b = \operatorname{bl}\sum_{y\in\mathcal{X}} 2^{\boldsymbol{a}_y}$. If $b > \omega$ and*

$$\hat{p}(x) = \frac{2^{\boldsymbol{a}_x} \gg (b-\omega)}{\left(\sum_{y\in\mathcal{X}} 2^{\boldsymbol{a}_y}\right) \gg (b-\omega)} ,$$

is a shifted version of p, then

$$|p(x) - \hat{p}(x)| < 2^{-\omega+1} ,$$

i.e., the error introduced by the shifting is bounded and exponentially small in the word-size.

Proof. First, assume $(2^{\boldsymbol{a}_x} \gg (b-\omega)) = 0$. In this case,

$$|p(x) - \hat{p}(x)| = p(x) = \frac{2^{\boldsymbol{a}_x}}{\sum_{y\in\mathcal{X}} 2^{\boldsymbol{a}_y}} < \frac{2^{b-\omega}}{\sum_{y\in\mathcal{X}} 2^{\boldsymbol{a}_y}} \leq 2^{-\omega+1},$$

because $2^{\boldsymbol{a}_x} < 2^{b-\omega}$ and $\sum_{y\in\mathcal{X}} 2^{\boldsymbol{a}_y} \geq 2^{b-1}$. Now, let $(2^{\boldsymbol{a}_x} \gg (b-\omega)) > 0$. In this case, according to Definition 4.4, we have $(2^{\boldsymbol{a}_x} \gg (b-\omega)) = 2^{\boldsymbol{a}_x-b+\omega}$. Thus,

$$|p(x) - \hat{p}(x)| = \frac{2^{\boldsymbol{a}_x-b+\omega}}{\left(\sum_{y\in\mathcal{X}} 2^{\boldsymbol{a}_y}\right) \gg (b-\omega)} - \frac{2^{\boldsymbol{a}_x}}{\sum_{y\in\mathcal{X}} 2^{\boldsymbol{a}_y}} , \tag{4.7}$$

where the $|\cdot|$ disappears because

$$\underbrace{\left(\left(\sum_{y\in\mathcal{X}} 2^{\boldsymbol{a}_y}\right) \gg (b-\omega)\right)}_{\lfloor c\rfloor} = \left\lfloor 2^{-b+\omega}\sum_{y\in\mathcal{X}} 2^{\boldsymbol{a}_y}\right\rfloor \leq \underbrace{2^{-b+\omega}\left(\sum_{y\in\mathcal{X}} 2^{\boldsymbol{a}_y}\right)}_{c} .$$

Multiplying $p(x)$ in (4.7) with $1 = 2^{-b+\omega}/2^{-b+\omega}$, and observing that $\lfloor c\rfloor > c-1$, yields

$$|p(x) - \hat{p}(x)| = \frac{2^{\boldsymbol{a}_x-b+\omega}}{\lfloor c\rfloor} - \frac{2^{\boldsymbol{a}_x-b+\omega}}{c} \leq \frac{2^{\boldsymbol{a}_x-b+\omega}}{(c-1)c} \leq \frac{2^{\boldsymbol{a}_x-b+\omega}}{(2^{b-1}-1)2^{b-1}} \leq 2^{\boldsymbol{a}_x-3b+3+\omega} .$$

Finally, the lemma follows from $\forall x \in \mathcal{X} : \boldsymbol{a}_x < b$ and $b \geq \omega+1$ (due to integrality of b and ω). ■

It is thus rather safe to pass shifted marginals to native code. An orthogonal approach is the restriction of the parameter space to the set $\{0, 1, \ldots, k-1\}^d \subset \mathbb{N}^d$. This prevents the parameters and the messages from becoming arbitrary large—a topic that we will revisit in Section 4.4.

4.3.5 Gibbs Sampling

Before we discuss different ways of learning integer parameters, we want to point out that inference via integer-valued Gibbs sampling is also possible. Recall that random number generation is inherently integer-valued. Generating random samples from a discrete state space can hence be done with integer arithmetic by, e.g., inversion sampling with rational probabilities. Reviewing the Gibbs sampler (Algorithm 2.4), we see that a familiar problem could arise.

$$p(\boldsymbol{x}_v \mid \boldsymbol{x}_{\mathcal{N}(v)}) = \frac{\prod_{C\in\mathcal{C}(v)} b^{\langle \boldsymbol{\theta}_C, \phi_C(\boldsymbol{x}_v, \boldsymbol{x}_{\mathcal{N}(v)})\rangle}}{\sum_{y\in\mathcal{X}_v} \prod_{C\in\mathcal{C}(v)} b^{\langle \boldsymbol{\theta}_C, \phi_C(y, \boldsymbol{x}_{\mathcal{N}(v)})\rangle}} \tag{4.8}$$

More precisely, the summation in the denominator may lead to overflows in the native integer arithmetic. However, nearly the complete discussion about overflows from the previous section applies also to Gibbs sampling. That is, if the native word-size does not suffice to compute the denominator of (4.8), we can employ the sparse integer representation together with right bit-shifts. This allows us to apply integer-valued Gibbs sampling with exponentially small errors (cf. Lemma 4.4). Nevertheless, Gibbs sampling is slow compared to message passing, due to the large number of samples which must be discarded to achieve independence, and the inherently sequential nature of MCMC algorithms.

4.4 Integer Parameter Estimation

While probabilistic inference provides information that we need to compute or to approximate the marginals and the partition function, updating the parameters of an integer model is a demanding task on its own. The major problem in the course of integer parameter learning via first-order methods is the inherently fractional parameter update $\boldsymbol{\theta}_i^t = \boldsymbol{\theta}_i^{t-1} - \kappa\partial\ell_b(\boldsymbol{\theta};\mathcal{D})/\partial\boldsymbol{\theta}_i$ (2.22). Different ways to circumvent this problem will be devised and discussed in this Section.

Let us start by reviewing the negative, average, log-likelihood in base-b form,

$$\ell_b(\boldsymbol{\theta};\mathcal{D}) = -\left\langle \boldsymbol{\theta}, \frac{1}{|\mathcal{D}|}\sum_{\boldsymbol{x}\in\mathcal{D}} \phi(\boldsymbol{x}) \right\rangle + \log_b Z(\boldsymbol{\theta}) ,$$

together with its partial derivatives

$$\frac{\partial\ell_b(\boldsymbol{\theta};\mathcal{D})}{\partial\boldsymbol{\theta}_i} = \hat{\boldsymbol{\mu}}_i - \tilde{\boldsymbol{\mu}}_i = \hat{p}_{b,\boldsymbol{\theta}}(\phi(\boldsymbol{x})_i = 1) - \frac{1}{|\mathcal{D}|}\sum_{\boldsymbol{x}\in\mathcal{D}} \phi(\boldsymbol{x})_i .$$

Therein, the negative term is the empirical expectation of the i-th sufficient statistic. It is denoted in this particular way to make it obvious that it is a rational number. The other term is the corresponding marginal probability, computed via probabilistic inference. As known from the previous Section, this is a rational number as well, whenever integer-valued inference procedures are used, e.g., our BLprop algorithm.

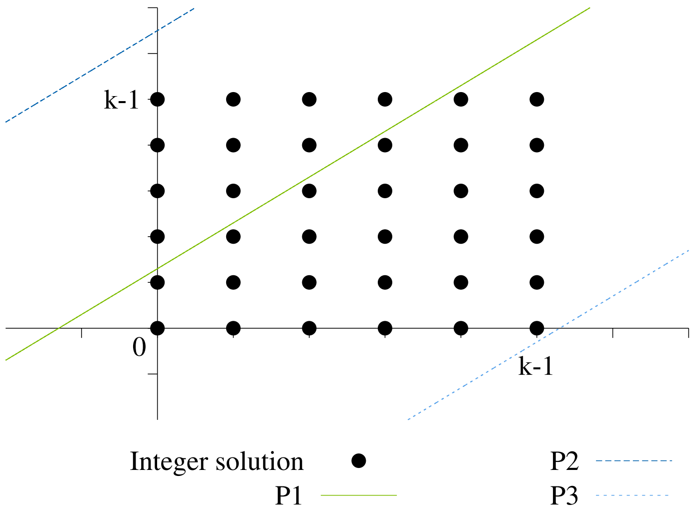

Figure 4.1: Exemplary parameter space. Solid points indicate integer solutions, i.e., elements of the set $\{0, 1, \dots, k-1\}^2$. Lines represent shift-invariant maximum likelihood solutions of three different synthetic estimation problems. The solid line (P1) crosses the integer parameter space and at least five integer solutions are equally close to it. The upper dashed line (P2) is far from the set of possible solutions; k has to be increased in order to find a parameter with reasonable quality. The lower dashed line (P3) does also not cross the integer grid. However, the integer solution at $(0, k-1)$ is close and corresponds to a model with high likelihood.

Similar to the simulation of real-valued arithmetic, real-world compute architectures cannot cope with the full integer space $\mathbb{N}$. Instead, the space of possible unsigned integers is restricted by the architecture's word-size ω. One could employ the sparse integer structure the we proposed for the BLprop algorithm. However, in BLprop, the number of non-zero bits was small, constant, and known in advance, which allows for an efficient implementation. When arbitrary integers have to be stored, the sparse structure would be as inefficient as storing a dense matrix in a sparse matrix format. As explained in Section 4.3.2, one could always increase the range of representable numbers via appropriate software libraries, at the cost of additional resources. But here, we like to investigate what can be achieved with the available hardware. Instead of optimizing the parameters over $\{0, 1, \dots, 2^\omega - 1\}^d$, we will find it convenient to optimize over the even more restrictive space $[k-1]^d = \{0, 1, \dots, k-1\}^d$ with $1 \leq k \leq 2^\omega$. On the one hand, limiting the parameter bit-width restricts the magnitude of the potential function, and hence, prevents overflows during message and marginal computation. On the other hand, memory requirements can be reduced by storing $\boldsymbol{\theta}$ as, say, 8 bit unsigned integers instead of the native ω bit representation.

The optimal parameter vector might indeed lie outside of $[k-1]^d$—a simplification of this situation is depicted in Fig. 4.1. Therein, each line represents a set of optimal solutions, specified via $\boldsymbol{\theta}^* + \alpha \boldsymbol{c}$ for some optimal $\boldsymbol{\theta}^*$, some constant vector $\boldsymbol{c} = (c, c, \dots, c)^\top$, and all real scalars $\alpha \in \mathbb{R}$. Due to shift-invariance (Lemma 2.4), all points on each line are equivalent to the corresponding $\boldsymbol{\theta}^*$. The topmost line is far from the integer grid and the nearest integer solution in $[k-1]^d$ will have a low likelihood—k has to be increased in order to find an integer solution that is close to a maximum likelihood estimate. The solid line crosses the set of feasible points. Multiple equally good integer solution can be found. Due to shift-invariance, the five integer solutions which are closest to the solid line will have the same quality in terms of objective value. Considering Theorem 4.2, the quality of those solutions should be quite good. Despite the fact that the third line does not cross the feasible set, integer parameters can be found that are still close to an optimal fractional solution.

By relaxing the integrality constraint, one could invoke the Karush-Kuhn-Tucker and Slater's conditions, to declare the existence of an optimal solution in $[k-1]^d$. However, optimizing and refining relaxations of (4.4) via standard mixed-integer non-linear programming techniques, like outer-approximation, branch-and-bound, or cutting plane algorithms [111, 26], is not feasible with integer-only arithmetic. Standard relaxation-based techniques make use of rounding, intermediate non-integer solutions, and convex hulls, respectively—each of which cannot be natively represented via integer-only arithmetic. Instead, we propose a special regularization, whose corresponding proximal operator enforces integrality. But beforehand, we present an approximation to the closed-form solution for tree-structured models.

4.4.1 Tree-Structured Models

As explained in Section 2.3.1, the optimal vertex and edge parameters of tree-structured models can be computed in closed-form. In case of base-b models, this closed-form reads as follows:

$$\boldsymbol{\theta}^*_{v=x} = \log_b \tilde{\boldsymbol{\mu}}_{v=x} \qquad \text{and} \qquad \boldsymbol{\theta}^*_{vu=xy} = \log_b \frac{\tilde{\boldsymbol{\mu}}_{vu=xy}}{\tilde{\boldsymbol{\mu}}_{v=x}\tilde{\boldsymbol{\mu}}_{u=y}} \,. \tag{4.9}$$

We will of course focus on $b = 2$ to apply the bit-length-based lower and upper bounds on $\log_2$, and to derive an upper bound on the optimal parameters [170]. Let $f_i = \sum_{\boldsymbol{x}\in\mathcal{D}} \phi(\boldsymbol{x})_i$, then $\tilde{\boldsymbol{\mu}}_i = f_i/|\mathcal{D}|$ and thus

$$\boldsymbol{\theta}^*_{v=x} = \log_2 f_{v=x} - \log_2 |\mathcal{D}| \leq \mathrm{bl}\, f_{v=x} - \mathrm{bl}\, |\mathcal{D}| + 1 = \hat{\boldsymbol{\theta}}_{v=x} \,, \tag{4.10}$$

and

$$\begin{aligned} \boldsymbol{\theta}^*{}_{vu=xy} &= \log_2(f_{vu=xy}|\mathcal{D}|) - \log_2(f_{v=x}f_{u=y}) \\ &\leq \mathrm{bl}(f_{vu=xy}|\mathcal{D}|) - \mathrm{bl}(f_{v=x}f_{u=y}) + 1 = \hat{\boldsymbol{\theta}}_{vu=xy} \,. \end{aligned} \tag{4.11}$$

The $+1$ arises from the lower bound $\mathrm{bl}(z) - 1 \leq \log_2(z)$. We devise a specialized version of Theorem 4.2 to show that this approximation is not too far from the optimal parameters.

Corollary 4.3 (Tree-Structured Integer Approximation) *Consider a discrete, tree-structured exponential family with binary overcomplete sufficient statistic. Let* $\boldsymbol{\theta}^*$ *be computed via* (4.9) *and* $\hat{\boldsymbol{\theta}}$ *via* (4.10) *and* (4.11). *Moreover, let* $\boldsymbol{c} = (1, 1, \dots, 1, 1)^\top$ *be the vector of* 1*s, and* $\boldsymbol{\varepsilon} = \boldsymbol{\theta}^* - (\hat{\boldsymbol{\theta}} - \boldsymbol{c})$. *Then*

$$\ell(\hat{\boldsymbol{\theta}}; \mathcal{D}) - \ell(\boldsymbol{\theta}^*; \mathcal{D}) \leq 2\|\boldsymbol{\epsilon}\|_2 |\overline{\mathcal{C}}(G)| \ ,$$

Proof. Due to shift-invariance, $\ell(\hat{\boldsymbol{\theta}}; \mathcal{D}) = \ell(\hat{\boldsymbol{\theta}} - \boldsymbol{c}; \mathcal{D})$, so we may prove the inequality for $\hat{\boldsymbol{\theta}} - \boldsymbol{c}$ instead of $\hat{\boldsymbol{\theta}}$. For the vertex parameters, we have

$$\begin{aligned}
\boldsymbol{\theta}^*_{v=x} - (\hat{\boldsymbol{\theta}}_{v=x} - 1) &= (\log_2 f_{v=x} - \log_2 |\mathcal{D}|) - (\operatorname{bl} f_{v=x} - \operatorname{bl} |\mathcal{D}| + 1) + 1 \\
&\leq (\log_2 f_{v=x} - \lfloor \log_2 f_{v=x} \rfloor - 1) - (\log_2 |\mathcal{D}| - \lfloor \log_2 |\mathcal{D}| \rfloor - 1) \\
&= \epsilon_{v=x} - \epsilon_{|\mathcal{D}|} \ ,
\end{aligned}$$

where ϵ_z denotes the flooring error $\log_z - \lfloor \log_2 z \rfloor$. And for the edges,

$$\begin{aligned}
\boldsymbol{\theta}^*_{vu=xy} - (\hat{\boldsymbol{\theta}}_{vu=xy} - 1) = &\log_2(f_{vu=xy}|\mathcal{D}|) - \log_2(f_{v=x} f_{u=y}) - \\
&(\operatorname{bl}(f_{vu=xy}|\mathcal{D}|) - \operatorname{bl}(f_{v=x} f_{u=y}) + 1) \\
= &\epsilon_{vu=xy,|\mathcal{D}|} - \epsilon_{v=x,u=y} \ .
\end{aligned}$$

Since any flooring error ϵ_z is always in $[0; 1)$, it follows that $\|\boldsymbol{\varepsilon}\|_\infty < 1$ and we can apply the proof of Theorem 4.2 with $\boldsymbol{\varepsilon} = \boldsymbol{\theta}^* - (\hat{\boldsymbol{\theta}} - \boldsymbol{c})$ to derive the corollary. ■

Our closed-form parameters $\hat{\boldsymbol{\theta}}$ could be negative. While in principle, the integer model supports weights from $\mathbb{Z}$, we may prefer to save the sign bit on systems which anyhow have a low word-size. One can employ the shift-invariance lemma once more, to find equivalent parameters in the positive orthant, i.e., we subtract $c = \min_{i=1}^d \hat{\boldsymbol{\theta}}_i$ from each parameter. The final closed-form integer parameters for tree-structured models are then $\bar{\boldsymbol{\theta}}_i = \hat{\boldsymbol{\theta}}_i - c$ for $1 \leq i \leq d$.

We will now investigate how parameters can be estimated in case of loopy graphs.

4.4.2 The Integer Gradient Descent Method

When closed-form solutions are not available, we have to conduct numerical methods. Standard techniques from integer non-linear programming inherently require real-valued arithmetic. We derive a new, integer first-order method to estimate the parameters without the need for real-valued arithmetic.

Optimizing the log-likelihood over $\mathbb{N}$ corresponds to constrained optimization. In fact, we can write the problem as ordinary optimization over $\mathbb{R}^d$ subject to an integrality constraint, e.g., $\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \ell(\boldsymbol{\theta}; \mathcal{D})$ s.t. $\boldsymbol{\theta} \in \mathbb{N}^d$. Instead of handling the constraint explicitly, we will establish a regularization which penalizes the distance to the next integer solution. Moreover, we show that for appropriate choices of stepsize and regularization weight, the result in any iteration is purely integer. This allows us to rely on convergence results from numerical optimization while eliminating the need for floating-point arithmetic.

To get an idea of why such a procedure is possible, consider the l_1-regularization. Whenever the regularization weight λ is "large enough", any solution will be mapped to the zero vector $\mathbf{0}$. Here, we will exploit this behavior, in that we choose the regularization weight such that the corresponding proximal operator will map any fractional solution to the nearest integer solution. Indeed, a larger regularization parameter implicitly removes importance of the log-likelihood. However, when we assume that integer parameters are all we can handle, it does not matter if better fractional solutions exist.

We will first propose the integer regularization and derive the convergence of the corresponding proximal method. Afterwards, we derive the proximal operator. Finally, we will show that the corresponding algorithm does not involve any real-valued arithmetic.

Integer Regularization and Convergence

We define the following regularization, which penalizes any parameter proportional to its absolute distance to the nearest integer.

Definition 4.5 (Integer Regularization) *Let $\boldsymbol{\theta}$ be a parameter vector. The integer regularization which penalizes the distance from any parameter $\boldsymbol{\theta}_i$ to the nearest integer value is given by*

$$R_{int}(\boldsymbol{\theta}) = \sum_{i=1}^{d} \rho_{int}(\boldsymbol{\theta}_i)$$

with

$$\rho_{int}(\boldsymbol{\theta}_i) = 1 - |1 - 2(\lceil \boldsymbol{\theta}_i \rceil - \boldsymbol{\theta}_i)| \ . \tag{4.12}$$

The regularization is visualized in Fig. 4.2, left. It assigns maximum penalty to each integer multiple of $1/2$. At integers, the penalty is zero. Clearly, $R_{\mathtt{int}}(\boldsymbol{\theta})$ is non-convex.

One might or even should expect, that this would harm the correctness and convergence results of proximal optimization schemes. Luckily, it can be shown that proximal algorithms with non-convex regularization will converge to a critical point of the composite objective function (in our case $\ell(\boldsymbol{\theta}; \mathcal{D}) + \lambda R_{\mathtt{int}}(\boldsymbol{\theta})$), whenever it satisfies the Kurdyka-Łojasiewicz property [9, 11, 25].

Definition 4.6 (Kurdyka-Łojasiewicz Property) *Let $\epsilon > 0$, and let $\varphi : [0, \epsilon) \to \mathbb{R}_+$ be a concave, continuous function with $\varphi(0) = 0$, differentiable on $(0; \epsilon)$, and with positive derivative on $(0; \epsilon)$. Let f be a function, subdifferentiable at $\boldsymbol{z}' \in \operatorname{dom} \partial f$[21]. If, for all $\boldsymbol{z}$ in a neighborhood $\mathcal{N}(\boldsymbol{z}')$, with $f(\boldsymbol{z}') < f(\boldsymbol{z}) < f(\boldsymbol{z}') + \epsilon$, the function $h : \mathbb{R}^d \to \mathbb{R}$ with*

$$h_{\boldsymbol{z}'}(\boldsymbol{z}) = \frac{\partial \varphi(x)}{\partial x} \Big|_{f(\boldsymbol{z}) - f(\boldsymbol{z}')}$$

satisfies

$$h_{\boldsymbol{z}'}(\boldsymbol{z}) \min_{\boldsymbol{g} \in \partial f(\boldsymbol{z})} \|\boldsymbol{g}\|_2 \geq 1 \ , \tag{4.13}$$

then, the function f has the Kurdyka-Łojasiewicz property at $\boldsymbol{z}'$.

[21] Here, $\partial f(\boldsymbol{z})$ is the subdifferential of f at $\boldsymbol{z}$, and $\operatorname{dom} \partial f$ contains all subdifferentiable points $\boldsymbol{z}$ with $\partial f(\boldsymbol{z}) \neq \emptyset$. See, e.g., [149, 25].

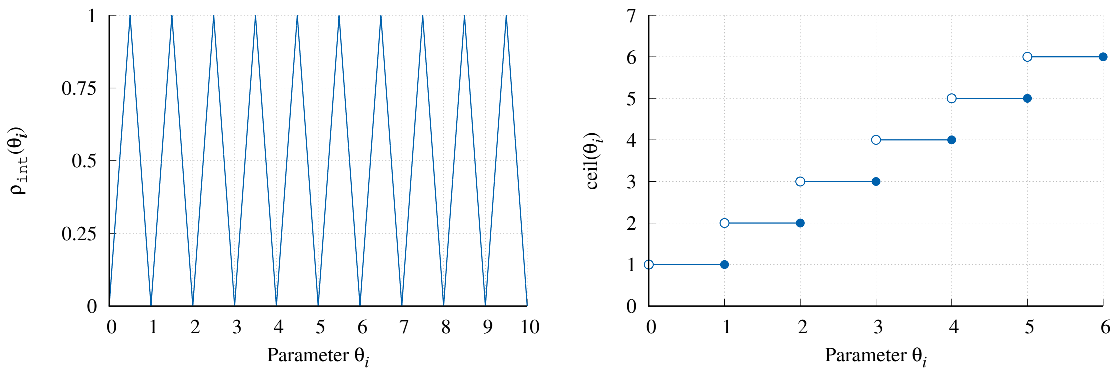

Figure 4.2: Left: Plot of the integer regularization $\rho_{\mathtt{int}}(\boldsymbol{\theta}_i)$ on $[0; 10]$. Any $\boldsymbol{\theta}_i \notin \mathbb{N}$ is penalized proportionally to its absolute distance to the nearest integer. Right: Visualization of lower semicontinuousness of $\lceil\cdot\rceil$. The filled circles represent values of $\lceil\cdot\rceil$ at discontinuities, i.e., left sided limit points.

The rather cumbersome form of the above definition stems from the fact that we have to work with multimodal functions, like $R_{\mathtt{int}}$, and the fact that the minimum of ℓ is > 0.

One can get an intuition for the above definition by assuming that the subdifferential contains just a single point. In this case, $\boldsymbol{g}$ is the gradient $\nabla f(\boldsymbol{z})$, and (4.13) guarantees that there exists a scale function φ, such that the scaled norm of the gradient at any point $\boldsymbol{z}$ is greater than 1, no matter how close $\boldsymbol{z}$ is to $\boldsymbol{z}'$.

Before we verify that our composite objective indeed has this property, let us clarify some prerequisites.

Definition 4.7 (Lower Semicontinuity) *A function* $f : \mathbb{R}^d \to (-\infty; +\infty]$ *is lower semicontinuous at* z_0*, if*

$$\liminf_{z \to z_0} f(z) \geq f(z_0) \ .$$

Continuous functions are lower semicontinuous.

The negative average log-likelihood (2.18) is continuous. $R_{\mathtt{int}}$ is the composition of affine functions and $|\cdot|$ (both continuous), with $\lceil\cdot\rceil$, which is lower semicontinuous (cf. Fig. 4.2, right). Their composition has no discontinuities (cf. Fig. 4.2, left), and is thus continuous. Lower semicontinuous functions satisfy the Kurdyka-Łojasiewicz property at any non-critical point in $\mathbb{R}^n$ ([10], Remark 4(b)). We only have to check whether a lower semicontinuous function satisfies the property at its critical points. Here, a subdifferentiable point $\boldsymbol{z}$ of a function f is *critical*, whenever $\mathbf{0} \in \partial f(\boldsymbol{z})$.

Any real-analytic function satisfies the Kurdyka-Łojasiewicz (KL) property [10] everywhere on its domain. While the log-likelihood falls in this class, $\rho_{\mathtt{int}}(z)$, is not analytic at any $z \in \mathbb{N}$. We hence need a broader characterization of KL-functions.

It can be shown, that functions with bounded domain, definable in some order-minimal (o-minimal) structure over $\mathbb{R}$, obey the Kurdyka-Łojasiewicz property [131, 24]. O-minimal structures are a principled way to define families of subsets of $\mathbb{R}^d$, which are closed under various operations. A discussion of basics and advanced results on this kind of model theoretic framework can be found in [216, 217, 59]. Real functions which contain exp and log are definable in the o-minimal structure $(\mathbb{R}_{\text{an}}, \exp)$ [214, 233]. Moreover, sums, products, inverse, and composition of definable functions are again definable in the same structure. Classical statements from real analysis also hold true in the o-minimal setting, e.g., the mean value theorem, L'Hospital's rule, and the implicit function theorem. Without going deeper into this branch of model theory, we state the relevant theorem.

Theorem 4.5 (O-Minimality and Kurdyka-Łojasiewicz Functions) *Let $U \subset \mathbb{R}^d$ and $f : K \to \mathbb{R}$ be an either smooth or non-smooth and lower semicontinuous function, definable in an o-minimal structure over $\mathbb{R}$. Suppose w.l.o.g. that $f(\boldsymbol{\theta}) > 0$ for all $\boldsymbol{\theta} \in K$, and that the function φ (from Definition 4.6) is definable in the same o-minimal structure as f. Then, f is a Kurdyka-Łojasiewicz function.*

A proof for non-smooth, lower semicontinuous functions f can be found in [24]. The corresponding result for differentiable functions f can be found in [131]. Therein, the concavity of φ—which is required by Definition 4.6—is not derived. It is, however, always possible to construct a concave φ—a corresponding proof can be found in [10], Theorem 14.

Having said all this, we can devise our main result.

Theorem 4.6 (Kurdyka-Łojasiewicz Integer Regularization) *The function $\ell(\boldsymbol{\theta}; \mathcal{D}) + \lambda R_{\mathtt{int}}(\boldsymbol{\theta})$, i.e., the sum of* (2.18) *and* (4.12), *is a Kurdyka-Łojasiewicz function on $U = [0; k-1]^d \subset \mathbb{R}^d$.*

Proof. Lower semicontinuity of ℓ and $R_{\mathtt{int}}$ follows directly from their corresponding definitions. Moreover, since both functions are lower bounded by 0, their sum $\ell(\boldsymbol{\theta}; \mathcal{D}) + \lambda R_{\mathtt{int}}(\boldsymbol{\theta})$ is also lower semicontinuous. Now, let us investigate the definability of

$$\ell(\boldsymbol{\theta}; \mathcal{D}) = \left(\log \sum_{\boldsymbol{x} \in \mathcal{X}} \exp(\langle \boldsymbol{\theta}, \phi(\boldsymbol{x}) \rangle) \right) - \langle \boldsymbol{\theta}, \tilde{\boldsymbol{\mu}} \rangle \ .$$

W.l.o.g., let $\mathcal{X} \subset \mathbb{R}^n$ be some discrete subset of $\mathbb{R}^n$. The states $\boldsymbol{x}$ of $\mathcal{X}$ are treated as arbitrary but fixed constants within any o-minimal structure. This implies that the vectors $\phi(\boldsymbol{x})$ are constants too (and not actual functions of $\boldsymbol{x}$), and we do not have to bother with the definability of ϕ. Each inner product $\langle \boldsymbol{\theta}, \phi(\boldsymbol{x}) \rangle$ is hence a sum of monomials, and as such definable in any (reasonable) structure over $\mathbb{R}$. The remaining part of ℓ is a log-sum-exp function with a summation over the finite, constant set $\mathcal{X}$. Altogether, ℓ on the set $[0; k-1]^d \subset \mathbb{R}^d$ is definable in the o-minimal structure $(\mathbb{R}_{\text{an}}, \exp)$. Details on $(\mathbb{R}_{\text{an}}, \exp)$ and examples of definable functions can be found in [214, 131, 162]. More on definability of functions within o-minimal structures can be found in [114].

In case of $R_{\mathtt{int}}$, the reasoning is slightly different, since $|\cdot|$ and $\lceil \cdot \rfloor$ are not (native) operations in $(\mathbb{R}_{\text{an}}, \exp)$. We will instead establish that $R_{\mathtt{int}}$ is component-wise semi-algebraic, which implies its definability. One can see in Fig. 4.2 (left), that the set

$F = \{(z, \rho_{\texttt{int}}(z)) : z \in [0; k-1]\}$, is a union of line segments in $[0; k-1] \times [0; 1]$. Namely the lines between the points $(x, 0)$ and $(x + (1/2), 1)$, and the lines between $(x + (1/2), 1)$ and $(x + 1, 0)$, for each $x \in \{0, 1, \dots, k-2\}$. The corresponding lines in $\mathbb{R}^2$ are given by $f_j(x, y) = 2(x - j) - y$ and $g_j(x, y) = 2(1 + j - x) - y$. Then, $F = (\cup_{i=0}^{k-2}\{(x, y) : f_i(x) = 0\} \cup_{i=0}^{k-2} \{(x, y) : g_i(x) = 0\}) \cap ([0; k-1] \times [0; 1])$, which shows that F is a semialgebraic set. Consequently, $\rho_{\texttt{int}}(\boldsymbol{\theta}_i)$ is a semialgebraic function, and hence definable in $(\mathbb{R}_{\text{an}}, \exp)$. Finally, $\ell(\boldsymbol{\theta}; \mathcal{D}) + \lambda R_{\texttt{int}}(\boldsymbol{\theta})$ is lower semicontinuous and definable in the o-minimal structure $(\mathbb{R}_{\text{an}}, \exp)$. According to Theorem 4.5, $\ell(\boldsymbol{\theta}; \mathcal{D}) + \lambda R_{\texttt{int}}(\boldsymbol{\theta})$ must be a Kurdyka-Łojasiewicz function. ■

At a first glace, the language of o-minimal structures might seem unreasonably powerful. However, several natural functions, which are based on non-elementary and improper integrals, are not definable in $(\mathbb{R}_{\text{an}}, \exp)$. Among them, the gamma function $\Gamma(z)$ on the interval $(0, +\infty)$, the (scaled) error function $\operatorname{erf}(z) = \int_0^\infty \exp(-z^2)$, and the Riemann zeta function on $(1, +\infty)$. Proofs of these statements can be found in [215]. Moreover, any non-constant periodic function (like $\sin(z)$) is not definable in any o-minimal structure. However, on finite compact subsets of their domains, all of the above examples are definable.

We may now invoke the following theorem (cf. Theorem 3.1 in [25]).

Theorem 4.7 (Convergence of the Alternating Proximal Method [25]) *Suppose $f : \mathbb{R}^d \to \mathbb{R}$ is a continuous differentiable function with Lipschitz continuous gradient and $g : \mathbb{R}^d \to \mathbb{R}$ is a (proper) lower semicontinuous function with $\inf_{\mathbb{R}^d}(f+g) > -\infty$. Let the bounded sequence $\boldsymbol{z}^1, \boldsymbol{z}^2, \dots$ be generated by a Gauss-Seidel block proximal method, i.e., $\forall C \in \overline{\mathcal{C}}(G) : \boldsymbol{z}_C^{i+1} = \operatorname{prox}_g^{1/L_C}(\boldsymbol{z}_C^i - \frac{1}{L_C}\nabla f(\boldsymbol{z}^i)_C)$ with block Lipschitz constant L_C. If $f + g$ is a Kurdyka-Łojasiewicz function, then the sequence $\boldsymbol{z}^1, \boldsymbol{z}^2, \dots$ will converge to a critical point $\boldsymbol{z}^*$, i.e., $\mathbf{0} \in \partial(f+g)(\boldsymbol{z}^*)$.*

Note, that being definable in an o-minimal structure is not sufficient, since the functions in question must be differentiable and lower semicontinuous, respectively, and f must have a Lipschitz continuous gradient. However, these requirements are fulfilled by our objective function, and we may state this result as a corollary.

Corollary 4.4 (Convergence of Integer Estimation) *The block-wise proximal minimization of $f(\boldsymbol{\theta}) = \ell(\boldsymbol{\theta}; \mathcal{D}) + \lambda R_{\texttt{int}}(\boldsymbol{\theta})$ over $U = [0; k-1]^d$ will either converge to a critical point of f or to a boundary point of U.*

Proof. The boundedness of the sequence, as required by Theorem 4.7, results from the fact that we are optimizing over U. Since U is a nonempty, closed, and convex subset of $\mathbb{R}^d$, we may apply Euclidean projection to keep solutions within U. Hence, if the proximal algorithm converges to a solution which lies on the boundary of U, it might not be a critical point. We may, however, choose k large enough to exclude this pathological case (cf. Fig. 4.1)—recall that the lower bound $\mathbf{0}$ on the parameter space is not restrictive due to shift-invariance. According to Theorem 4.7, any non-boundary solution must be critical point of $\ell(\boldsymbol{\theta}; \mathcal{D}) + \lambda R_{\texttt{int}}(\boldsymbol{\theta})$. ■

The Integer Proximal Operator

Corollary 4.4 asserts that the proximal optimization of the integer regularized log-likelihood has a well defined result: whenever it outputs a non-boundary point, we found a critical point of $\ell(\boldsymbol{\theta};\mathcal{D}) + \lambda R_{\mathtt{int}}(\boldsymbol{\theta})$. Hence, it is worth to derive a closed-form for $\operatorname{prox}_{\lambda R_{\mathtt{int}}}$ with $\lambda R_{\mathtt{int}}$. In what follows, κ is the step size of the underlying first-order method and λ is the regularization weight.

Theorem 4.8 (Closed-Form Proximal Operator) *The proximal operator that corresponds to the integer regularization* (4.12) *with* $\lambda = 1/(4\kappa)$*, has a closed-form that is given by*

$$\operatorname{prox}_{(4\kappa)^{-1}R_{int}}(\boldsymbol{\theta})_i = \begin{cases} \lfloor \boldsymbol{\theta}_i \rfloor & ,\boldsymbol{\theta}_i \in (\lfloor \boldsymbol{\theta}_i \rfloor - \frac{1}{2}; \lfloor \boldsymbol{\theta}_i \rfloor + \frac{1}{2}] \\ \lceil \boldsymbol{\theta}_i \rceil & ,\boldsymbol{\theta}_i \in (\lceil \boldsymbol{\theta}_i \rceil - \frac{1}{2}; \lceil \boldsymbol{\theta}_i \rceil + \frac{1}{2}] \end{cases} . \tag{4.14}$$

The image of $\mathbb{R}^d_+$ *under* $\operatorname{prox}_{(4\kappa)^{-1}R_{int}}$ *is* $\mathbb{N}^d$.

Proof. According to Definition 4.5 and the description of proximal operators in Section 2.3.3, the closed-form is a solution to the following optimization problem:

$$\begin{aligned} \operatorname{prox}^{\kappa}_{\lambda R_{\mathtt{int}}}(\boldsymbol{\theta}) &= \operatorname*{arg\,min}_{\boldsymbol{\gamma}\in\mathbb{R}^d} \lambda R_{\mathtt{int}}(\boldsymbol{\gamma}) + \frac{1}{2\kappa}\|\boldsymbol{\theta} - \boldsymbol{\gamma}\|_2^2 \\ &= \operatorname*{arg\,min}_{\boldsymbol{\gamma}\in\mathbb{R}^d} \lambda \left(\sum_{i=1}^d \rho_{\mathtt{int}}(\boldsymbol{\gamma}_i)\right) + \frac{1}{2\kappa}\|\boldsymbol{\theta} - \boldsymbol{\gamma}\|_2^2 \\ &= \operatorname*{arg\,min}_{\boldsymbol{\gamma}\in\mathbb{R}^d} \sum_{i=1}^d \underbrace{\lambda - \lambda|1 - 2(\lceil \boldsymbol{\gamma}_i \rceil - \boldsymbol{\gamma}_i)| + \frac{1}{2\kappa}(\boldsymbol{\theta}_i - \boldsymbol{\gamma}_i)^2}_{g_{\lambda,\kappa}(\boldsymbol{\gamma}_i,\boldsymbol{\theta}_i)} . \end{aligned} \tag{4.15}$$

Due to its sum form, $\operatorname{prox}_{\lambda R_{\mathtt{int}}}$ is separable among the dimensions of $\boldsymbol{\theta}$, and we may solve the above problem for each dimension separately. Let us derive the minimizer of (4.15). Observe that $|\cdot|$ is not differentiable at 0 and $\lceil\cdot\rceil$ is not differentiable at any $x \in \mathbb{N}$. To reduce the inconveniences that arise through the composition of non-smooth functions, we conduct a piecewise analysis of (4.15). Fix an arbitrary integer $z \in \mathbb{N}$ and consider the open interval $I_z = (z - 1/2; z + 1/2)$. Denote values $x \in I_z$ by $x = z + \epsilon_x$ with $\epsilon_x \in (-1/2; +1/2)$. The partial subdifferential w.r.t. $x \in I_z$ is

$$\begin{aligned} \partial g_{\lambda,\kappa}(x,y) &= \partial(\lambda - \lambda|1 - 2(\lceil x \rceil - x)| + \frac{1}{2\kappa}(y-x)^2) \\ &= \partial(2\lambda|x - z| + \frac{1}{2\kappa}(y-x)^2) = \begin{cases} 2\lambda\operatorname{sgn}(x - z) + \frac{1}{\kappa}(x - y) & ,\epsilon_x \neq 0 \\ [-2\lambda; 2\lambda] + \frac{1}{\kappa}(x-y) & ,\epsilon_x = 0 \end{cases} \end{aligned}$$

The above holds for all real numbers x with $\epsilon_x \in [0; 1/2)$. (Sub)derivatives at points with $\epsilon_x = \pm 1/2$ are, however, not required, since they are local maxima of $g_{\lambda,\kappa}(x,y)$. By resubstitution of $\boldsymbol{\gamma}_i$ and $\boldsymbol{\theta}_i$, we get the optimality condition

$$0 \in [-2\kappa\lambda; 2\kappa\lambda] + \boldsymbol{\gamma}_i - \boldsymbol{\theta}_i , \tag{4.16}$$

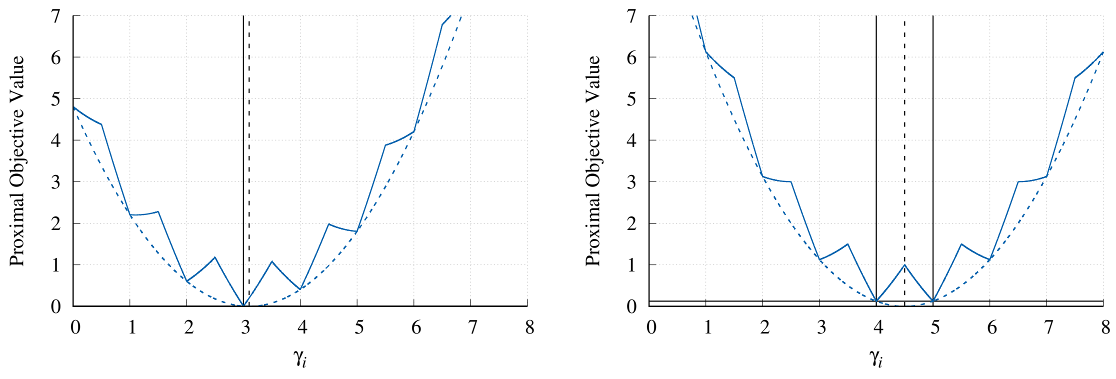

Figure 4.3: Left: Exemplary proximal problem (4.15) for $\rho_{\texttt{int}}(\boldsymbol{\theta}_i)$ with $\boldsymbol{\theta}_i = 3.1$ and $\lambda = 1/(4\kappa)$. The solid curve is the objective function value (4.15) for $\boldsymbol{\gamma}_i \in [0; 8]$, and its minimum is indicated by the solid vertical line. The dashed curve shows the value of the quadratic term $(\boldsymbol{\theta}_i - \boldsymbol{\gamma}_i)^2$, and the dashed vertical line indicates the position of its minimum. The horizontal line shows the function value of the optimal $\boldsymbol{\gamma}_i$, which is $\boldsymbol{\gamma}_i^* = 3$. Right: The same plot for the situation $\boldsymbol{\theta}_i = 4.5$—two different $\boldsymbol{\gamma}_i$ values are globally optimal, namely $\boldsymbol{\gamma}_i^* \in \{4, 5\}$.

at any integer $\boldsymbol{\gamma}_i$. We can now devise the closed-form, based on the above relation. Setting $\boldsymbol{\gamma}_i$ to $\lfloor\boldsymbol{\theta}_i\rfloor$ or $\lceil\boldsymbol{\theta}_i\rceil$ in (4.16), reveals under which conditions these choices are optimal. Hence, whenever $\boldsymbol{\theta}_i \in [\lfloor\boldsymbol{\theta}_i\rfloor - 2\kappa\lambda; \lfloor\boldsymbol{\theta}_i\rfloor + 2\kappa\lambda]$, it is optimal to set $\boldsymbol{\gamma}_i$ to $\lfloor\boldsymbol{\theta}_i\rfloor$. And whenever $\boldsymbol{\theta}_i \in [\lceil\boldsymbol{\theta}_i\rceil - 2\kappa\lambda; \lceil\boldsymbol{\theta}_i\rceil + 2\kappa\lambda]$, it is optimal to choose $\lceil\boldsymbol{\theta}_i\rceil$. Now, plugging $\lambda = 1/(4\kappa)$ into these conditions results in a complete coverage of the real line. At the boundaries of the intervals, i.e., at integer multiples of $1/2$, both neighboring integers are optimal, as depicted in the right plot of Figure 4.3. W.l.o.g., we choose the value that is closer to zero. Thus,

$$\operatorname{prox}_{(4\kappa)^{-1}R_{\texttt{int}}}(\boldsymbol{\theta})_i = \begin{cases} \lfloor\boldsymbol{\theta}_i\rfloor & ,\boldsymbol{\theta}_i \in (\lfloor\boldsymbol{\theta}_i\rfloor - \frac{1}{2}; \lfloor\boldsymbol{\theta}_i\rfloor + \frac{1}{2}] \\ \lceil\boldsymbol{\theta}_i\rceil & ,\boldsymbol{\theta}_i \in (\lceil\boldsymbol{\theta}_i\rceil - \frac{1}{2}; \lceil\boldsymbol{\theta}_i\rceil + \frac{1}{2}] \end{cases} .$$

■

$\lambda = 1/(4\kappa)$ is not the only choice which results in integer parameters, but it is the smallest λ that achieves this. Let us see what happens if we depart from $\lambda = 1/(4\kappa)$. On the one hand, $\lambda > 1/(4\kappa)$ will still produce pure integer solutions, but downweights the importance of the log-likelihood. From this point of view, the utility of driving λ beyond $1/(4\kappa)$ is questionable. However, empirical evaluation shows that a slightly stronger integer regularization with $\lambda = 3/(8\kappa) > 1/(4\kappa)$ gives superior results in practice. This can be explained as follows: if $\boldsymbol{\theta}_i$ is already an integer, the regularization will update the value to $\boldsymbol{\theta}_i \pm 1$ only if the gradient step is large enough, e.g., outside of $\boldsymbol{\theta}_i \pm 1/2$. This can be too strict in many settings. By choosing a larger λ values, say $\lambda = 3/(8\kappa)$, the optimal regions for $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_i \pm 1$ will overlap (cf. (4.16)). Hence, we are free to tighten the region in which we will stay at $\boldsymbol{\theta}_i$ to $\boldsymbol{\theta}_i \pm 1/4$, which reduces the rigidity of the proximal gradient update.

On the other hand, whenever $\lambda < 1/(4\kappa)$, there will be cases in which $\boldsymbol{\theta}_i$ is neither in $[\lfloor\boldsymbol{\theta}_i\rfloor - 2\kappa\lambda; \lfloor\boldsymbol{\theta}_i\rfloor + 2\kappa\lambda]$ nor in $[\lceil\boldsymbol{\theta}_i\rceil - 2\kappa\lambda; \lceil\boldsymbol{\theta}_i\rceil + 2\kappa\lambda]$—the regions in which it is optimal to set $\boldsymbol{\theta}_i$ to an integer would not touch each other. That would result in fractional solutions whenever the update falls in this void region. While this observation is not relevant in the setting of constraint arithmetic, the proposed technique can be used for general mixed-integer programming as well. Moreover, this allows us to investigate meaningful regularization paths for increasing values of λ. We regard this direction as future work.

One could imagine a theorem similar to Theorem 3.2, stating that for an appropriate choice of lambda, the method will detect the "true" integer parameters while bounding the distance to the parameter that generated the data. However, in the context of restricted arithmetic, we are not really interested if and which dimensions of the true parameter vector are actually integer. Instead, we employ the regularization to have a principled and theoretically sound way of generating integer solutions with well-defined properties, i.e., being critical points of the regularized log-likelihood.

4.4.3 Learning with Integer Arithmetic

Based on the insights that we gained about exponential families under arithmetic constraints, we devise a learning procedure that requires no real-valued computation. The corresponding pseudocode is shown in Algorithm 4.2. Therein and in what follows, λ is fixed to $(4\kappa)^{-1}$. Nevertheless, as explained in the previous Section, other choices, e.g., $\lambda = 3/(8\kappa)$, are possible. We begin in line 1 by initializing the sum of all sufficient statistics, parameters, and temporary variables. In each iteration of the main loop, rational marginals are computed via bit-length propagation or another integer-based inference technique. Lines 5–10 correspond to a coordinate-wise application of the proximal operator for the block of parameters that corresponds to the current clique C. Let us explain how $\operatorname{prox}_{(4\kappa)^{-1}R_{\mathtt{int}}}(\boldsymbol{\theta}_j - \kappa\nabla\ell(\boldsymbol{\theta};\mathcal{D})_j)$ (4.14) is implemented. To this end, we first check which new values are possible for $\boldsymbol{\theta}_j$ after the update. The image of $\nabla\ell(\boldsymbol{\theta};\mathcal{D})_j$ is $(-1;1)$—it is the difference of empirical and inferred marginal probabilities. In accordance to Theorem 4.7, our algorithm updates the parameters blockwise, where each block is constituted by one clique. This requires the clique-wise Lipschitz constants which we derived in Lemma 2.7. For each block, the Lipschitz constant is < 1 which implies that any block-wise stepsize κ_C that satisfies $\kappa_C \leq 1 < 1/L_C$ suffices to guarantee convergence. Algorithm 4.2 hence uses $\kappa_C = 1$ to ensure that the gradient steps will be within $(-1;1)$ which facilitates integer proximal steps.

Now, the integer proximal operator will map the gradient step to the nearest integer. Possible new parameter values are $\boldsymbol{\theta}_j - 1, \boldsymbol{\theta}_j, \boldsymbol{\theta}_j + 1$, where the new value is $\boldsymbol{\theta}_j - 1$, if and only if $\kappa\nabla\ell(\boldsymbol{\theta};\mathcal{D})_j \geq 1/2$. The new value is instead $\boldsymbol{\theta}_j + 1$, if and only if $\kappa\nabla\ell(\boldsymbol{\theta};\mathcal{D})_j < -1/2$. Let us write the rational marginal $\boldsymbol{\mu}_j$ as $\boldsymbol{a}_j/\boldsymbol{b}_j$. We can then rewrite the condition for

Algorithm 4.2: Integer Parameter Estimation via Integer Gradient Descent

```
input Data set $\mathcal{D}$, max. weight $k$, max. iterations $I$, desired precision $\varepsilon \in \mathbb{Q}$
output Integer parameters $\boldsymbol{\theta}^* \in \mathbb{N}^d$
 1: $\boldsymbol{f} \leftarrow \sum_{\boldsymbol{x}\in\mathcal{D}} \phi(\boldsymbol{x})$;  $\epsilon \leftarrow \infty$;  $\epsilon^* = \infty$;  $\boldsymbol{\theta} \leftarrow \mathbf{0}$;  $\boldsymbol{\theta}^* = \mathbf{0}$ ;  $i \leftarrow 1$
 2: Compute rational marginals $\boldsymbol{\mu}$ with $\boldsymbol{\mu}_i = {}^{\boldsymbol{a}_i}/_{\boldsymbol{b}_i} \in \mathbb{Q}$ // via, e.g., Algorithm 2.1 +
    Algorithm 4.1
 3: repeat
 4:   for $C \in \overline{\mathcal{C}}(G)$ do
 5:     for $j \in [d] : \exists \boldsymbol{x} \in \mathcal{X}_C : j \equiv \{C = \boldsymbol{x}\}$ do
 6:       if $\left(2\boldsymbol{f}_j\boldsymbol{b}_j - 2\boldsymbol{a}_j|\mathcal{D}| > \boldsymbol{b}_j|\mathcal{D}|\right)$ and $(\boldsymbol{\theta}_j + 1 \leq k - 1)$ then
 7:         $\boldsymbol{\theta}_j \leftarrow \boldsymbol{\theta}_j + 1$
 8:       else if $\left(2\boldsymbol{a}_j|\mathcal{D}| - 2\boldsymbol{f}_j\boldsymbol{b}_j \geq \boldsymbol{b}_j|\mathcal{D}|\right)$ and $(\boldsymbol{\theta}_j \geq 1)$ then
 9:         $\boldsymbol{\theta}_j \leftarrow \boldsymbol{\theta}_j - 1$
10:       end if
11:     end for
12:     Recompute rational marginals $\boldsymbol{\mu}$ with $\boldsymbol{\mu}_i = {}^{\boldsymbol{a}_i}/_{\boldsymbol{b}_i} \in \mathbb{Q}$
13:     $\epsilon \leftarrow \|\boldsymbol{\mu} - {}^{\boldsymbol{f}}/_{|\mathcal{D}|}\|_\infty$
14:     if $\epsilon < \epsilon^*$ then
15:       $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}$ ;  $\epsilon^* \leftarrow \epsilon$
16:     end if
17:   end for
18: until $\epsilon^* < \varepsilon$ or $i > I$
```

$\boldsymbol{\theta}_j - 1$ as

$$\kappa\nabla\ell(\boldsymbol{\theta};\mathcal{D})_j = \boldsymbol{\mu} - \tilde{\boldsymbol{\mu}} \geq \frac{1}{2} \Leftrightarrow 2\left(\frac{\boldsymbol{a}_j}{\boldsymbol{b}_j} - \frac{1}{|\mathcal{D}|}\sum_{\boldsymbol{x}\in\mathcal{D}} \phi(\boldsymbol{x})_j\right) \geq 1$$
$$\Leftrightarrow 2\boldsymbol{a}_j|\mathcal{D}| - 2\boldsymbol{b}_j\boldsymbol{f}_j \geq \boldsymbol{b}_j|\mathcal{D}|$$

with $\boldsymbol{f} = \sum_{\boldsymbol{x}\in\mathcal{D}} \phi(\boldsymbol{x})$, which is the condition in line 8. The corresponding condition for $\boldsymbol{\theta}_j + 1$ in line 6 is derived likewise. Both conditions are extended by boundary checks, to guarantee that $\boldsymbol{\theta} \in [k-1]^d$. By applying the equality $a/b - c/d = (ad - bc)/(bd)$, and the equivalence $a/b > c/d \Leftrightarrow ad > bc$, we can compute the l_∞-norm of the gradient (line 13) with integer-valued arithmetic. Finally, the same rules apply when we evaluate the termination criterion in line 18. We may conclude that we achieved the goal of native integer parameter estimation.

Lemma 4.5 (Integer-Only Learning) *Algorithm 4.2 estimates integer parameters of a discrete base-2 exponential family member with binary, overcomplete sufficient statistics. Running the algorithm does not involve any real-valued computation.*

Our algorithm terminates after a maximum number of iterations I has been reached, or when the gradient's uniform norm is below a prescribed threshold ε. We know that any optimal parameter $\boldsymbol{\theta}^*$ has gradient norm 0, and due to Lipschitz continuity of the gradient, we know further that

$$\|\nabla\ell(\boldsymbol{\theta};\mathcal{D}) - \nabla\ell(\boldsymbol{\theta}^*;\mathcal{D})\|_2 \leq 2|\overline{\mathcal{C}}(G)|\|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|_2 \, .$$

This implies that $\|\nabla\ell(\boldsymbol{\theta};\mathcal{D})\|_\infty < \varepsilon$ is a necessary condition for the case that we are close to the true optimizer, i.e., $\|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|_2 < \varepsilon/(2|\overline{\mathcal{C}}(G)|)$ [171].

As indicated in [221], considering the gradient norm as a quality measure during parameter estimation can compensate for errors which are induced via approximate probabilistic inference. E.g., our bit-length approximation and probability shifting (Lemma 4.4) both introduce errors into our marginals $\hat{\boldsymbol{\mu}}$. These defective marginals could be closer to $\tilde{\boldsymbol{\mu}}$ than the exact marginals $\boldsymbol{\mu}$, which are actually encoded by $\boldsymbol{\theta}$. Taking the norm of the gradient $\|\hat{\boldsymbol{\mu}} - \tilde{\boldsymbol{\mu}}\|$ to guide the optimization procedure may hence compensate for the fact that our approximated marginals might be far from $\boldsymbol{\mu}$. This statement is indeed true for any parameter learning that is based on approximate inference.

4.4.4 Alternative Integer-Valued Estimation Procedures

Algorithm 4.2 performs block-wise parameter updates, but coordinate-wise updates are conceivable, too. Different strategies for deciding which coordinates have to be updated lead to different methods. Obvious choices are cyclic updates, where a window of l parameters is updated in each iteration, and random updates, where a random subset of parameters is updated in each iteration. Convergence analysis for different scenarios can be found in [210, 155, 18].

An extreme case of randomized parameters updates are evolutionary algorithms (EA) [28]. We denote these methods as "extreme", because information about the gradient is not used at all. Such methods are also called zero-order methods. Here, we will shortly explain parameter learning via the so-called (1+1)-EA. For ease of exposition, assume that $k = 2^K$ is a power of two and consider the binary representation $\mathrm{bin}(\boldsymbol{\theta})$ of the parameter vector $\boldsymbol{\theta} \in [k-1]^d$. The length of $\mathrm{bin}(\boldsymbol{\theta})$ is hence dK, i.e., $\mathrm{bin}(\boldsymbol{\theta}) \in \{0,1\}^{dK}$. In its simplest form, the (1+1)-EA works directly on this binary representation, as shown in Algorithm 4.3. In contrast to the integer gradient descent, parameter updates are performed randomly. For each bit of the binary representation of the parameter vector, we sample a Bernoulli random variable $\boldsymbol{Y}$ with $p(\boldsymbol{Y} = 1) = q$. If the sample is 1, we flip the corresponding bit. Afterwards, the quality of the resulting parameter vector is measured via its gradient norm. In each iteration i, the probability to jump to any arbitrary point in $[k-1]^d$ is strictly greater 0, since $q \in (0;1)$. The procedure may hence find a globally optimal solution in any iteration. The expected number of iterations until the EA generates a globally optimal solution depends, however, heavily on q and the problem itself. Common choices are $q = 1/(dK)$ and $q = \log(dK)/(dK)$. The convergence of evolutionary algorithms under different choices of q and different problem types has been studied [21, 57]. In general, one should expect a number of steps that is exponential in the length of the binary representation of $\boldsymbol{\theta}$. There are, however,

Algorithm 4.3: Integer Parameter Estimation via (1+1)-EA

```
input Data set $\mathcal{D}$, iterations $I$, precision $\varepsilon \in \mathbb{Q}$, mutation probability $q \in (0;1)$
output Integer parameters $\boldsymbol{\theta}^* \in \mathbb{N}^d$
1: $\boldsymbol{f} \leftarrow \sum_{\boldsymbol{x}\in\mathcal{D}} \phi(\boldsymbol{x})$; $\epsilon \leftarrow \infty$; $\epsilon^* = \infty$; $\boldsymbol{\theta} \leftarrow \boldsymbol{0}$; $\boldsymbol{\theta}^* = \boldsymbol{0}$ ; $i \leftarrow 1$
2: repeat
3:   for $j = 1$ to $dK$ do
4:     $z \sim$ Bernoulli($q$)
5:     if $z = 1$ then
6:       $\text{bin}(\boldsymbol{\theta})_j \leftarrow 1 - \text{bin}(\boldsymbol{\theta})_j$
7:     end if
8:   end for
9:   Compute rational marginals $\boldsymbol{\mu}$ with $\boldsymbol{\mu}_i = {}^{\boldsymbol{a}_i}/_{\boldsymbol{b}_i} \in \mathbb{Q}$
10:  $\epsilon \leftarrow \|\boldsymbol{\mu} - {}^{\boldsymbol{f}}/_{|\mathcal{D}|}\|_\infty$
11:  if $\epsilon < \epsilon^*$ then
12:    $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}$ ; $\epsilon^* \leftarrow \epsilon$
13:  else
14:    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^*$
15:  end if
16: until $\epsilon^* < \varepsilon$ or $i > I$
```

results on randomized zero-order methods for convex and non-convex problems, which indicate that the expected number of iterations can be much lower.

4.5 Experimental Demonstration

Based on basic principles of the Bethe approximation and non-smooth non-convex optimization, we derived a subclass of exponential families in which probabilistic inference and parameter estimation does not require any floating-point arithmetic. Our results are independent of the specific conditional independence structure and allow us to specify an upper bound on the largest representable model parameter to control the memory consumption per parameter. We conduct a series of experiments to demonstrate this behavior on synthetic and real-world data. More precisely, we are interested in answering the following questions empirically:

Q1 Can we observe a decreased runtime?

Q2 Can we observe a decreased memory consumption?

Q3 Are marginals computed by bit-length propagation similar to those computed by loopy belief propagation?

Q4 How does the quality of undirected integer models compare to the ordinary Markov random field?

We explained at the beginning of this chapter that integer arithmetic usually requires less clock-cycles than the corresponding floating-point operations. Our undirected integer models should hence exhibit a decreased runtime compared to the ordinary MRF. However, loopy belief propagation in ordinary MRFs uses message normalization. This suppresses small changes in the messages which in turn helps to enforce convergence. In bit-length propagation, such a normalization is not possible. Hence, the runtime per training iteration of the integer model can exceed the runtime of the ordinary model whenever the ordinary loopy belief propagation converges significantly faster. We quantify the runtime to investigate this issue and to find an answer to **Q1**.

To answer the second question, we measure the memory consumption of the model by consulting its proportion of non-zero components, a.k.a. the NNZ-Ratio. In addition, we will discuss effects on the memory consumption which are implied by the upper bound on the parameter values.

Question **Q3** arises from Theorem 4.3. Moreover, the error of the inference procedure is a function of the maximal degree and the longest path in the model. We should thus expect that simpler graphical structures yield better results.

The last question is motivated by Theorem 4.2, which tells us that we should expect to find a good integer solution whenever the true parameter is not too far from being integer. In addition, Theorem 4.7 guarantees that our integer gradient descent finds a critical point of the integer regularized log-likelihood. When all Theorems which are presented in this chapter hold simultaneously, we should expect a reasonable learning results. As we did in Chapter 3, we consider the normalized squared Euclidean distance—equivalent to the mean squared error—between our estimated marginals $\hat{\boldsymbol{\mu}}$ and the empirical marginals $\tilde{\boldsymbol{\mu}}$ to investigate the quality of different models and to find an answer to question **Q4**.

To summarize, our evaluation incorporates

- the mean squared error between estimated and empirical marginals: $\mathrm{MSE}(\boldsymbol{\theta}) = \frac{1}{d}\sum_{i=1}^{d}(\hat{\boldsymbol{\mu}} - \tilde{\boldsymbol{\mu}})^2$ to quantify the general quality of a model,
- the relative number of non-zero components of $\|\boldsymbol{\theta}\|_0/d$, to measure the memory consumption of a model,
- and the runtime per training iteration in milliseconds, to measure the computational complexity.

4.5.1 Setup

We know from the experimental evaluation of Chapter 3 that a regularized model is slightly faster and uses less memory than an unregularized model. Hence, to have a realistic comparison, we employ an l_1-regularized MRF with $\lambda = 0.01$ as baseline model. Its parameters are estimated via accelerated first-order optimization. This model is compared to an integer undirected model whose parameters are learned via integer gradient descent. In summary, we use the following models:

M1 Regularized discrete state Markov random field with accelerated first-order optimization

M2 Integer Markov random field with Integer Gradient Descent optimization

In case of model **M2**, we consider various upper bounds on the maximum model parameter $k \in \{2, 4, 8, 16, 32\}$ which restricts the estimated model parameters to the space $\{0, 1, \ldots, k-1\}^d$. Data from different sources is considered to demonstrate the robustness of integer models.

Each experiment is performed on an Intel Xeon E5-2697 v2 system. We provide a docker image with our own C++ implementation of models **M1** and **M2** for download at `https://sfb876.tu-dortmund.de/px`.

Synthetic Data

We conduct a series of experiments on synthetic data to incorporate several degrees of parameter magnitude and integrality. For simplicity, all synthesized models contain only pairwise factors ψ_{vu}. The number of variables is fixed to $n = 100$ and the number of states per variable is fixed to 2.

We consider the same synthetic graphs that we used in Chapter 3, namely chain, grid, star, and full (Fig. 3.8), as conditional independence structures. Theorem 4.3 tells us that the graphical structure has a direct impact on the integer probabilistic inference. The graphs are hence chosen to cover different levels of complexity. In the simplest structure (chain), each vertex has degree at most 2 and the longest path has length n. The star structure exhibits one vertex with relatively high degree, while the longest path has length 3. Grid structures consist of multiple loops, obey a vertex degree which is independent of n, and also contain long paths. Finally, fully connected structures represent the worst-case for degree, path length, and loopyness.

The model parameters are sampled independently from a Gaussian with mean 0 and standard deviation $\sigma \in \{1, 2, 4, 8\}$, to simulate various parameter magnitudes.

Finally, we inject integrality into the random model parameters. Recall that Theorem 4.2 ensures that the integer approximation error will be smaller the more integer parameters are contained in the true model parameter. After drawing independent Gaussian parameters as described above, we draw a Bernoulli random variable $\boldsymbol{B}$ with parameter $q \in [0; 1]$. Whenever $\boldsymbol{B} = 1$, the fractional part of the random parameter will be discarded, which mimics a flooring operation. Note that the parameter q allows us to control the integrality of the synthesized parameters in a compact manner. If $q = 0$, all parameters are independent realizations of a standard normal random variable, which has probability density zero of being an integer. By increasing q, the fraction of integer parameter increases. When $q = 1$, all parameters are integers. In our experiments, we consider $q \in \{0, 0.25, 0.5, 0.75, 1\}$ to study the effects of different integrality levels.

Eventually, 1000 samples are generated from each model via Gibbs sampling (Algorithm 2.4). During data generation, the first 100 samples are discarded. Between consecutive samples, all variables are resampled 16 times in a round-robin fashion to enforce independence of consecutive samples.

Table 4.2: Summary of real-world data sets.

Name	**# variables** (n)	**# readings**	**# data points** (N)
D1 INSIGHT	2367	316632468	1608
D2 VaVeL	4988	3934145	516
D3 Intel Lab	56	2313682	432

Real-world Data

In the second set of experiments, we estimate the model parameters on the same data that we used in the real-world experiments of Section 3.6.1, namely:

D1 INSIGHT—Dublin City SCATS Data[22]

D2 VaVeL—Warsaw City Mobile Network Cell Data[23]

D3 Intel Lab—Temperature and Humidity Data[24]

Even in large, wired sensor networks, integrating the machine learning model into the sensors which actually measure the data can have benefits in terms of reliability and privacy, since no data would leave the actual network.

In contrast to the experiments from the previous chapter, we do not use the temporal information. Instead of combining the data from a whole day into a single training instance, we generate twelve training instances per day which leads to a slight increase in the data set size. Information on these data sets is summarized in Table 4.2. For each real-world data set, we choose the optimal tree structure as conditional independence structure, computed via the Chow-Liu algorithm (cf. Section 2.3.4).

4.5.2 Results

Results on the synthetic data sets are presented in Figures 4.5–4.8, and results on the real-world data sets are shown in Figures 4.10–4.11. The black solid circles represent results for model **M1**, while the other symbols correspond to results for model **M2**—the common key of all plots is shown in Fig. 4.4. In total, the results represent 2435 learning runs. Error bars are ommited in favor of readability. Instead, we provide the empirical worst-case variances in Table 4.3. Except for the runtime on the full structure, all variances are rather low. The high variance runtime can be explained with the convergence of bit-length propagation. If some of the repetitions of our experiments converge fast while others do not, a large variances arises. Nevertheless, the variance of

[22]Section 4.1.1 in `http://www.insight-ict.eu/sites/default/files/deliverables/D5-1.pdf`
[23]Section 3.15 in `http://www.vavel-project.eu/sites/default/files/VaVeL_D1_3.pdf`
[24]`http://db.csail.mit.edu/labdata/labdata.html`

Table 4.3: Empirical upper bounds on the variances of each experiment on synthetic data.

Name	$\tilde{\mathbb{V}}^*$[MSE]	$\tilde{\mathbb{V}}^*$[NNZ Ratio]	$\tilde{\mathbb{V}}^*$[millis/iter]
Chain	3.67366233175944e-06	0.00851341699826563	0.0197278642126124
Star	5.886993309304e-06	0.0683236404448525	0.00845444081641737
Grid	5.51239300507056e-06	0.0101987654320987	1.43933385201495
Full	1.12903489689042e-05	0.0130796720742782	67.3147063826381

M1	●	M2, k=4	▼	M2, k=16	⬟
M2, k=2	▲	M2, k=8	◆	M2, k=32	🟩

Figure 4.4: Key for Figures 4.5 to 4.7 to indicate the maximum integer parameter.

67.314 corresponds to a standard deviation of ≈ 8 milliseconds per iteration—a value which may still be regarded as low in practice.

We explain and discuss the results w.r.t. to the questions stated at the beginning of this Section.

Q1 Can we observe a decreased runtime?

The main motivation for integer models lies in the fact that some system architectures do not provide hardware for floating-point arithmetic. In this case, floating-point arithmetic has to be emulated in software, which involves large performance penalties. However, even if a floating-point unit is available, integer arithmetic can be used to speed up inference and learning. Here, we compare the runtime of models **M1** and **M2** on an Intel Xeon E5 server CPU. Corresponding results can be found in the runtime plots of Figures 4.7–4.11). And indeed, on all tree-structured models (chain,star,insight,intel, and vavel), we observe speedups between $4\times$ and $10\times$ for the integer models compared to **M1**.

On the other structures, it seems that the integer models are outperformed by the plain MRF. However, a closer inspection of the results shows, that the reason lies in the convergence behavior of both algorithms. The loopy belief propagation of the **M1** model converges in about 10 iterations. In contrast, the bit-length propagation does not converge in some cases on the grid structure and in all cases on the fully connected structure—due to the missing message normalization. Since we enforce the termination after updating all bit-length messages 100 times, BLprop computes about ten times more messages than LBP. Having this in mind, it is clear that computation of bit-length messages is more than twice as fast as ordinary loopy BP. The fact that BLprop might not converge can easily avoided in practice, by reducing the maximum number of iterations—additional experiments revealed that reducing the maximum iteration count to 25 has almost no impact on the marginal quality. Moreover, LBP might not converge as well, but this did not happen during our experiments.

It is no surprise that the runtime is almost independent of the integrality of the ground truth model. However, considering the runtime as a function of the standard deviation, we can observe a speedup for both, **M1** and **M2** models. There is also no noteworthy difference in the runtime of **M2** models with different maximum parameter bounds.

In addition to our results on synthetic data, the speedup on real-world data is even larger. This is also due to the fact that our real-world data sets exhibit a larger state space than the synthetic binary data.

Altogether, integer models exhibit the expected speedup over ordinary, floating-point-based MRFs. By transitivity, we conclude that **M2** models must also outperform emulated floating-point arithmetic on devices which do not provide floating-point hardware. This conjecture will be confirmed in Chapter 6.

Q2 Can we observe a reduced memory consumption?

To investigate the memory consumption, we consult the NNZ-Ratio plots in Figures 4.5, 4.6, and 4.9–4.11. Regarding the NNZ-Ratio as a function of the parameter integrality (Figures 4.5), we see that there is a constant gap between the NNZ-Ratio of **M1** and **M2** models in favor of the latter. Indeed, increasing the regularization weight of the **M1** model would result in more sparsity. However, the best regularization weight is a priori unknown, while the integer parameter estimation algorithm is parameter free. Instead, the degree of sparsity discovered by **M2** models is due to the fact that 0 is an integer. There is no clear order of **M2** models w.r.t. the maximum parameter value k. E.g., the NNZ-ratio plots show that models with $k = 8$ are most dense in case of chain graphs and most sparse in case of grid structures. Similar effects can be observed for other values of k as well.

Treating the sparsity as a function of the standard deviation shows a clear linear dependence between both—the higher the standard deviation, the higher the sparsity. In all cases, **M2** models exhibit a higher sparsity than **M1** models. Again, there is no order among models with different k values on the synthetic data. However, our experiments on real-world data show, that models with low k value tend to be more sparse than models with a large k value.

Overall, our proposed integer models are inherently sparse without providing an explicit regularization weight, which implies a reduced memory consumption.

Q3 Are BLprop's marginals similar to those computed by LBP?

We are now interested in the quality of our integer models. First, we investigate the behavior of bit-length propagation. To avoid side-effects, we restrict ourselves to tree-structured models on which LBP delivers the correct marginals. No parameters are learned in this experiment. Instead, we take the random parameters with full integrality ($q = 1$) which we used to generate the synthetic data.

On these models, we run BLprop and LBP, and store the computed marginals. The results are shown in Fig. 4.8. Each cross indicates a pair of LBP marginal and BLprop marginal, and the straight line indicates the region of zero-error, i.e., any cross which lies exactly on the line represents a marginal in which LBP and BLprop agree. The mean squared error over all marginals is reported in the title of each plot. We know from Theorem 4.3 that the expected error between the marginal densities of both methods depends on the length of the longest path (l) and the largest vertex degree (r). The results in Fig. 4.8 suggest, that l has a larger impact on the approximation error than r. Moreover, the error shrinks with increasing σ, which reflects the intuition that small differences in the marginals cannot be captured well by integer parameters. In contrast, marginal densities which are far from the uniform distribution—and thus have low entropy—can be represented with much smaller MSE.

An artifact of practical implementations can be seen in the plot for the star structure with $\sigma^2 = 8$. Due to high standard deviation and parameter shifting into the positive orthant, the magnitude of the parameters becomes large—in this particular case, we have $\max_i^d \boldsymbol{\theta}_i = 43$.Together with the high max-degree of the star graph ($r = 99$), this leads to overflows in the double precision arithmetic, which forces our implementation to output the uniform distribution, i.e., $\hat{p}(\boldsymbol{x}_{vu}) = 1/4$ for our synthetic data. Since the bit-length propagation makes use of the sparse integer data structure, arbitrary large messages can be represented and no overflow occurs. Note that a standard multi-precision floating-point library would avoid the overflow problem in BP but at the cost of large performance penalties. In contrast, our sparse integer data structure does not suffer from performance penalties since the number of non-zero bits is constant ($|\mathcal{X}_v| = 2, \forall v \in V$).

These results show, that the BLprop marginals are quite close to the LBP results, and that BLprop can be even more robust than LBP w.r.t. numerical issues.

Q4 How does the quality of learned integer models compare to the ordinary MRF models?

Integer regularization prevents our learning procedure from selecting fractional solutions. In this last experiment, we assess the quality of the overall learning result. The first results on synthetic data are shown in Fig. 4.5. Therein, we see that the induced integrality has only a minor effect on the mean squared error. **M1** models yield the lowest MSE on all synthetic structures. The star structure seems to be slightly harder for **M2** models. On the other structures, the quality of integer models can be very close to that of ordinary MRFs. While the MSE of the best **M2** model is about twice as large as the MSE of the corresponding **M1** model, all models deliver a small error regarding its absolute magnitude. As can be seen in Table 4.3, the MSE results are quite stable—the estimated variance around each point does not exceed 1.2×10^{-5}. The most restricted **M2** models ($k = 2$) deliver the lowest performance on average. However, recall that $k = 2$ implies that the model parameters are actually binary, i.e., $\boldsymbol{\theta} \in \{0, 1\}^d$. Having this in mind, it is rather surprising that they yield the best **M2** performance on the fully connected structure. There is no clear order among the other **M2** models. The optimal k value is application specific. The MSE values of **M2** models with $k > 2$ are close to each other.

When we plot the MSE as a function of the standard deviation (Fig. 4.6)—while averaging over the various levels of integrality—shows, that the MSE tends to decrease with an increasing σ. This effect is stronger on the loopy structures grid and full but can also be observed on the chain structure. Again, difference between **M2** models with different k values are small. In all experiments on real-world data, we observe that the MSE degrades nicely with an increasing value of k and converges at about $k = 8$. Recall that this implies that only 3 bits are required for each non-zero parameter—this alone makes a $20\times$ reduction in memory required to store the model parameter $\boldsymbol{\theta}$, in addition to the increased sparsity discussed above. Indeed, the ordinary MRF model exhibits a superior small error, but when the underlying device is highly restricted, integer models offer a reasonable way of trading quality against resource consumption.

4.6 Discussion

Resource constraint systems can have highly limited arithmetic capabilities. In this chapter, we investigated the inherent arithmetic requirements of exponential family members. More precisely, we analyzed the effects of restricting *the parameter space and any computation* to the natural numbers, i.e., $\boldsymbol{\theta} \in \mathbb{N}_0^d$. Understanding the impact of low-precision arithmetic is an active research area in machine learning, but most aspects have not been studied rigorously. Many empirical results are furthermore not strict in the sense that computations are still carried out via floating-point arithmetic. Here, we instead assumed that the underlying hardware has no floating-point coprocessor, which implies that software emulation of floating-point arithmetic is required whenever real-valued arithmetic cannot be avoided.

We started by proving that the exponential family may alternatively be written w.r.t. to any base b (instead of exp). Choosing $b = 2$ turned out to have various appealing properties. To restrict the parameters to the integer space, we defined a non-universal integer reparametrization and analyzed its reparametrization error. Intuitively, the error depends on the size of the conditional independence structure and the involved rounding errors. We then derived a new, message passing algorithm for probabilistic inference in base-2 models, called bit-length propagation. In contrast to ordinary belief propagation, only the bit-length of a message is propagated, and the corresponding computation can be carried out fully in the integer domain. We provided an upper bound on the expected Kullback-Leibler divergence between LBP and BLprop marginals w.r.t. the conditional independence structure. The longest path and the largest vertex degree were identified to have the largest impact on the approximation error. Internally, the BLprop algorithm makes use of a sparse integer data structure to represent arbitrary large messages. In contrast to general multi-precision libraries, the non-zero bits are stored in a sorted list structure. Since the number of summands in each message is $|\mathcal{X}_v|$, the number of non-zero bits per message is upper bounded by $|\mathcal{X}_v|$. This allows an efficient computation of the bit-length of arbitrary large messages via our sparse integer representation. After discussing the inference, we derived methods for the actual integer parameter estimation.

The first method is based on a combination of the integer reparametrization and the closed-form maximum likelihood estimator for tree-structured models. Secondly, we derived a new optimization technique, called *integer gradient descent*. Therein, the most important idea was to phrase the problem as ordinary parameter estimation over $\mathbb{R}^d$ while employing a new, non-smooth, non-convex integer regularization which enforces the output of each iteration to be integer. Due to non-smoothness, proximal optimization techniques were used. We derived a closed-form of the corresponding proximal operator for our integer regularization which is required to solve the proximal sub-problem in each training iteration efficiently. Based on o-minimality and Kurdyka-Łojasiewicz functions, we proved that proximal alternating linearized minimization algorithms will converge to critical or boundary points of the learning problem. The result is a completely new algorithm for solving optimization problems over the integers.

In addition to the theoretical derivation and analysis of our new methods, extensive experiments on synthetic and real-world data showed their empirical effectiveness. In terms of inference, we first assessed the quality of bit-length propagation and compared the results to marginals computed via ordinary loopy belief propagation. It turned our that marginal densities with low entropy are easier to infer than marginals which are close to uniform. On several different structures BLprop succeeded to approximate the marginals with low error. We then assessed the full learning procedure, including bit-length propagation and integer gradient descent to estimate the parameters on several data sets. Our results show, that small mean squared errors can be achieved, even when we restrict the word-size of each model parameter to three bits. In terms of memory and computation time, integer undirected models enjoy $\approx 10\times$ speedup and $\approx 20\times$ memory reduction, compared to the ordinary MRF. We can conclude that integer models are extremely well suited for small, resource-constrained devices.

Almost all theoretical and empirical contributions in this chapter are new. Our parameter estimation for tree-structured models was first published in [170], however, a weaker inference algorithm without sparse integer data structure and without probability shifting was used. First results for learning and inference in loopy structures were presented in [171], but the optimization procedure employed therein was fully heuristic. Since the focus of this work lies in the estimation of probability densities, we measured the quality of our methods in terms of marginal densities. However, additional results which investigate the classification accuracy of integer models can be found in [170, 171]. There, it is shown that integer model can achieve almost state-of-the-art classification performance in named entity recognition and synthetic classification tasks. Additional experimental results on the prediction of smartphone usage can be found in [174].

Our experiments revealed, that convergence of bit-length propagation can be an issue on loopy conditional independence structures. This problem is inherent to message passing algorithms. Moreover, quality guarantees for LBP cannot be made on most loopy graphs. This motivates the question if we can find approximate inference procedures which do not rely on message propagation, while providing quality guarantees on all conditional independence structures, and are in addition well-suited for resource-constrained systems. We devote the next chapter to shed some light on this issue.

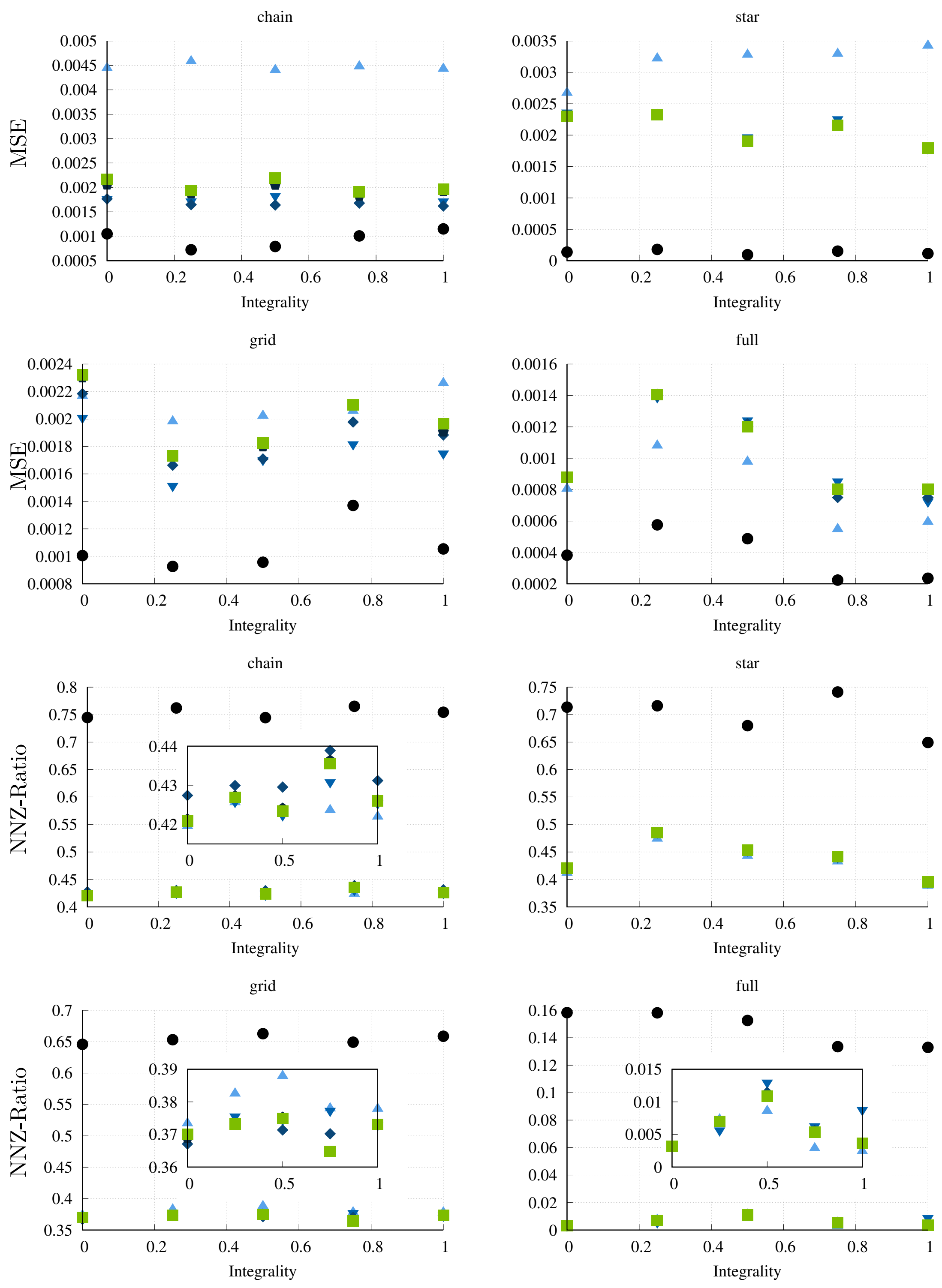

Figure 4.5: Average MSE in estimated marginals (y-axis, first two rows), and average number of non-zero parameters (y-axis, last two rows), as a function of the parameter integrality (x-axis). The results are averaged over multiple runs and different standard deviations $\sigma \in \{1, 2, 4, 8\}$. Different colors indicate different decay types (see Fig. 4.4).

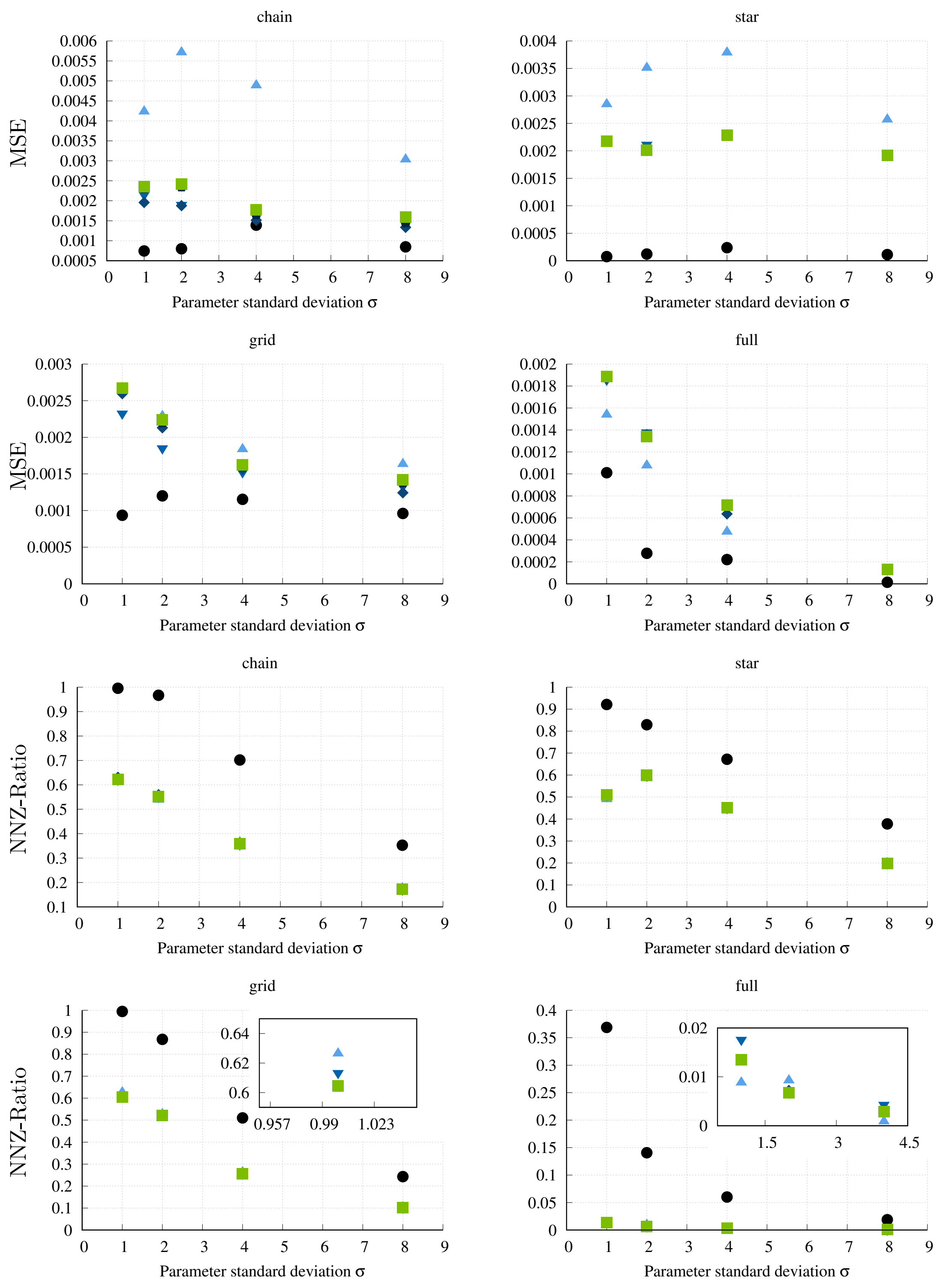

Figure 4.6: Average MSE in estimated marginals (y-axis, first two rows), and average number of non-zero parameters (y-axis, last two rows), as a function of the standard deviation (x-axis). The results are averaged over multiple runs and different levels of integrality $\{0, 1/4, 1/2, 3/4, 1\}$. Different colors indicate different decay types (see Fig. 4.4).

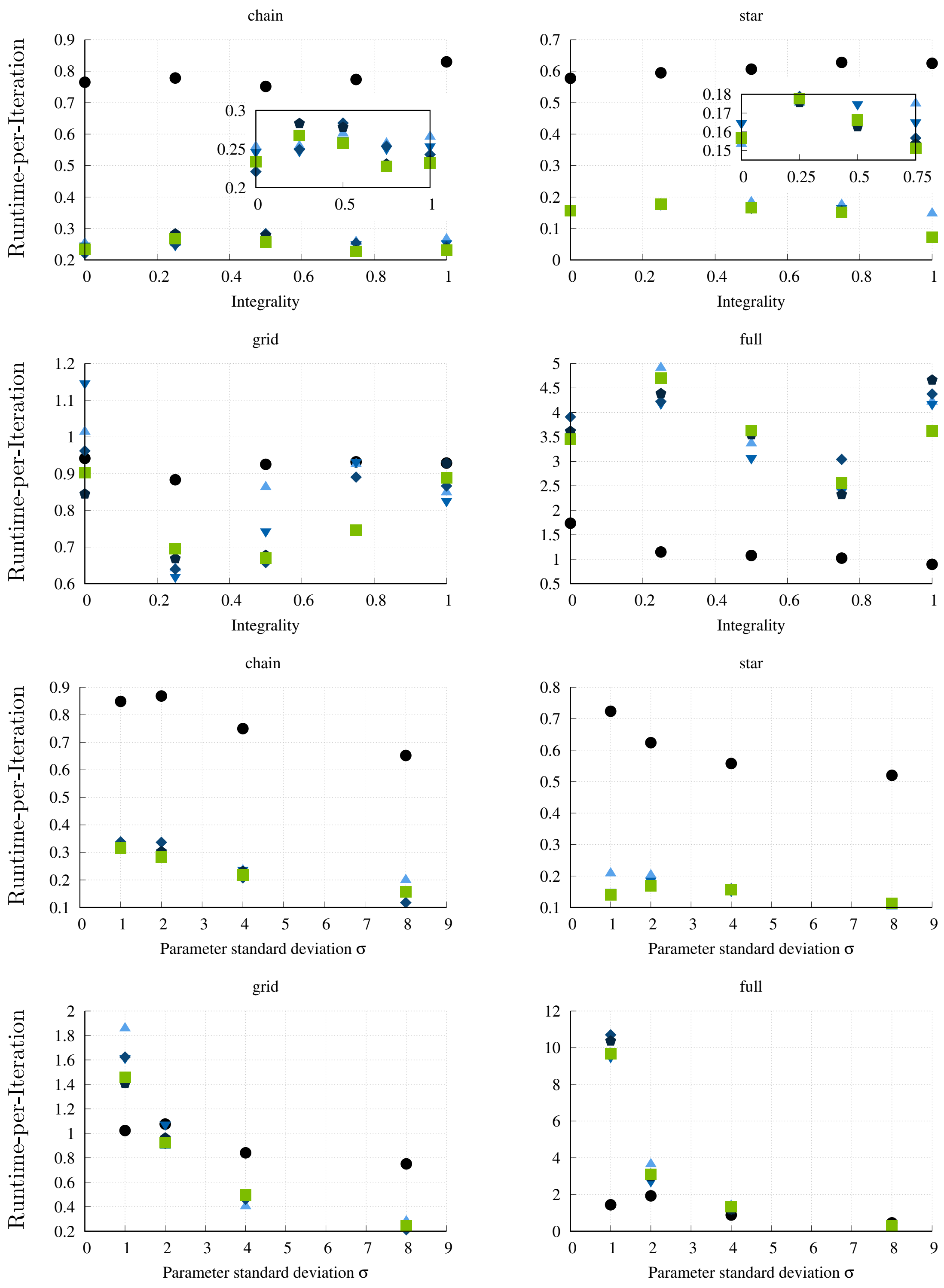

Figure 4.7: Average runtime per training iteration (milliseconds, y-axis) as a function of the parameter integrality (x-axis, first two rows), and as a function of the standard deviation (x-axis, last two rows). The results are averaged over multiple runs. Different colors indicate different decay types (see Fig. 4.4).

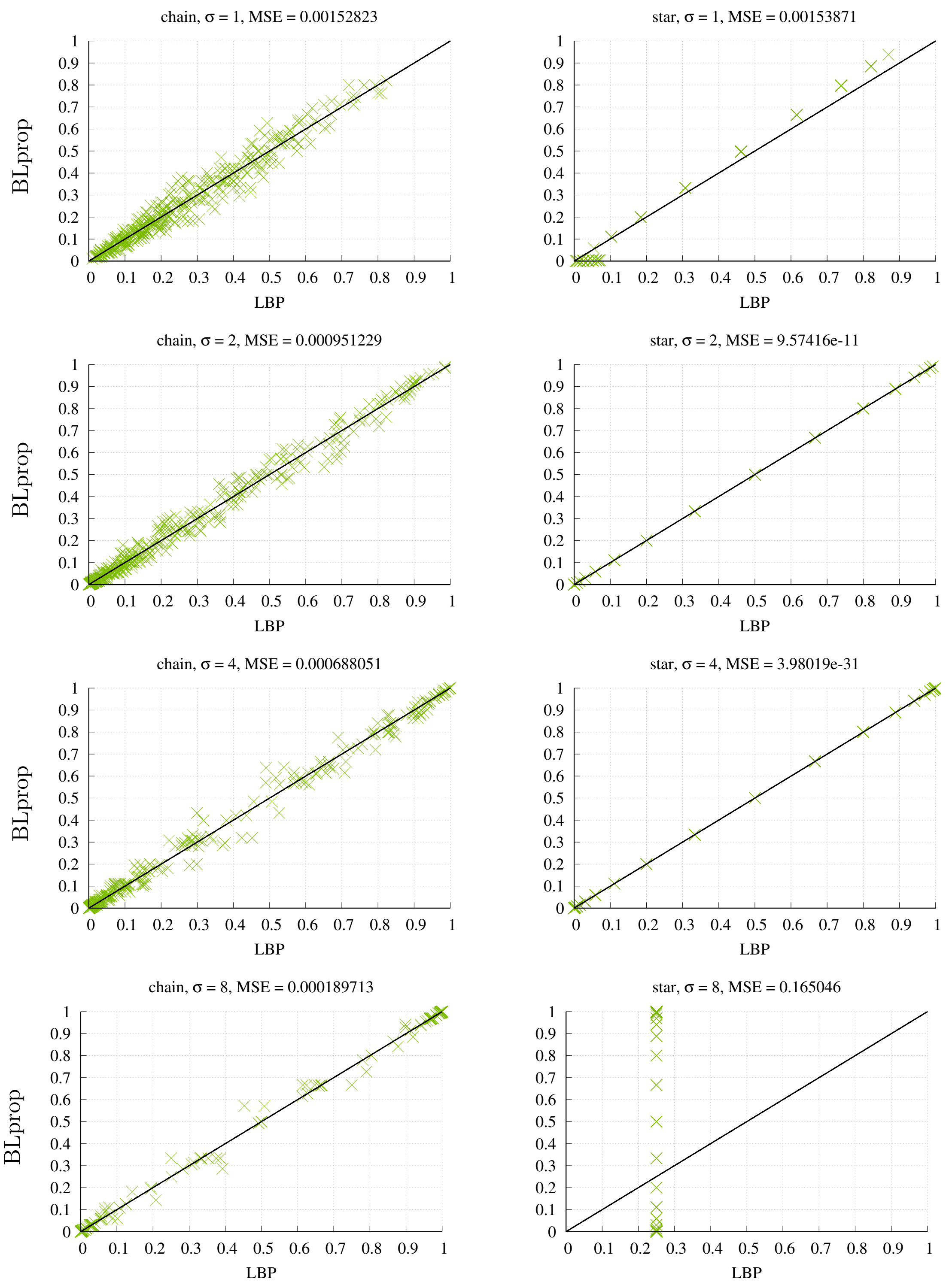

Figure 4.8: Marginals computed on the synthetic chain and star structures with integer parameters ($q = 1$) by bit-length propagation and loopy belief propagation. Each plot shows the marginals computed from a single run of LBP and BLprop for the same random parameters with $\sigma \in \{1, 2, 4, 8\}$ (from top to bottom).

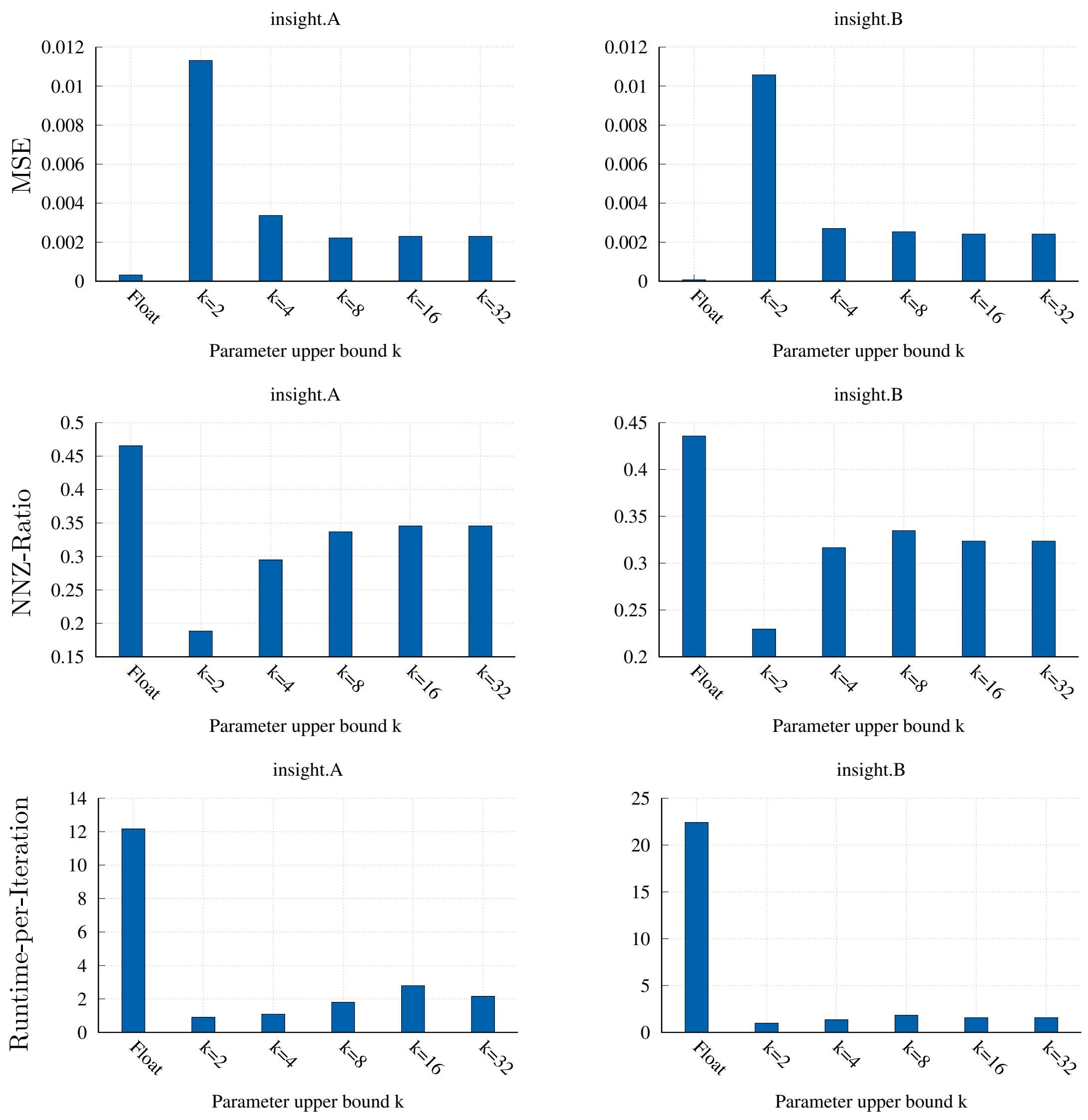

Figure 4.9: Experimental results on the INSIGHT data. Average MSE between estimated and empirical marginals (first row), average number of non-zero parameters (second row), and average runtime per training iteration (last row), as a function of the parameter upper bound k (x-axis).

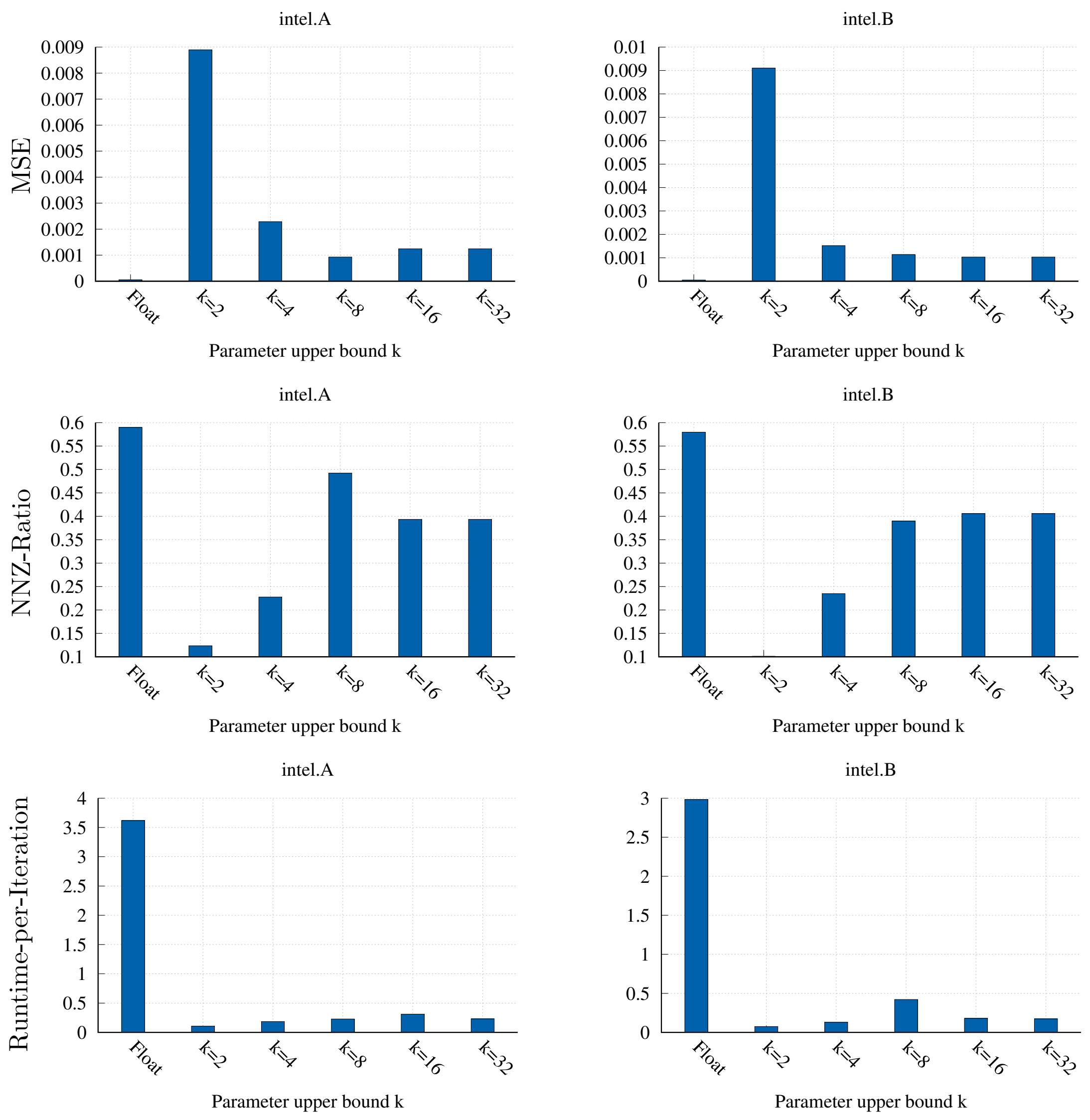

Figure 4.10: Experimental results on the Intel Lab data. Average MSE between estimated and empirical marginals (first row), average number of non-zero parameters (second row), and average runtime per training iteration (last row), as a function of the parameter upper bound k (x-axis).

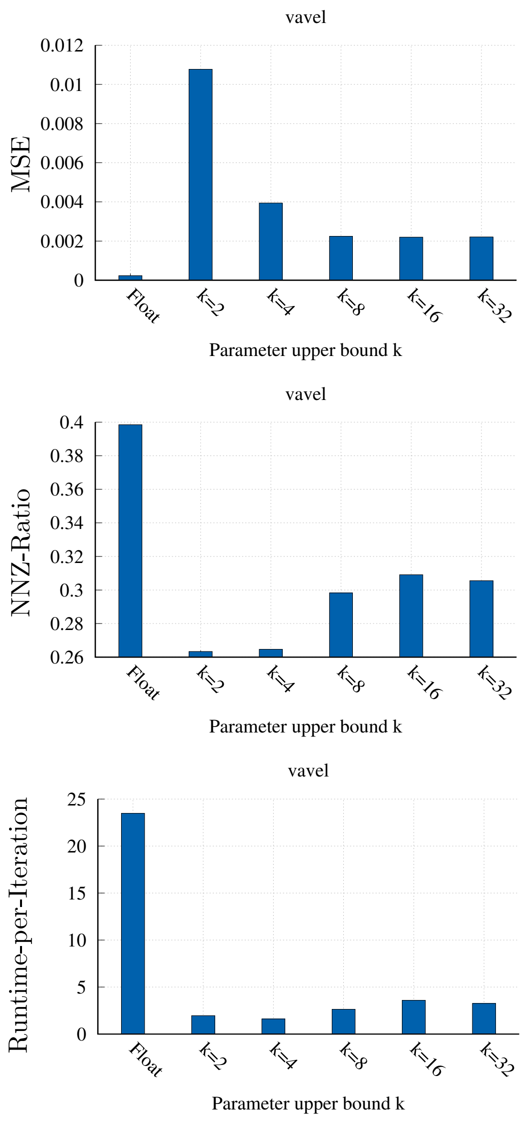

Figure 4.11: Experimental results on the VaVeL data. Average MSE between estimated and empirical marginals (first row), average number of non-zero parameters (second row), and average runtime per training iteration (last row), as a function of the parameter upper bound k (x-axis).

5 Computation and Quality Constraints

Beside memory and arithmetic requirements of exponential family models, demanding computational challenges arise through the $\#\mathbf{P}$-hardness of probabilistic inference. This problem arises especially in exponential family densities of discrete random variables, since many relevant real-valued exponential family members are easy to normalize via closed-form integrals (cf. Section 2.2.2). Up to now, we leveraged this issue by assuming that either the tree-width is small enough to run probabilistic inference on the junction tree, or that an approximation via loopy belief propagation suffices. It is, however, not unlikely that resource-constrained systems have to comply with certain quality constraints, especially when they are autonomous. When we encounter situations that require tight bounds on the approximation error of probabilistic inference, we cannot rely on variational techniques [227] like TRW-BP, mean field or the Bethe approximation. Some of these technique indeed deliver bounds, but the error cannot be quantified without making assumptions that go beyond the ordinary variational principle. An overview is provided in Table 5.1. MCMC sampling techniques may be conducted, but the number of samples we have to sacrifice in order to generate a set of independent data points might be either unknown, or too large to be applicable in the context of resource constraints. Assuming that the clock speed of the underlying system is low, say, 16 MHz, methods that throw the results of intermediate steps away might not be the best choice.

Approximate and exact inference methods that are specialized for certain model types arose in the last decades. While quite promising results have been achieved for (simplified) Ising models (with reparametrization (3.2), right), even these simple models are hard to approximate whenever the inverse temperature β is strictly positive [107]. As described in Section 2.2.2, exact inference can be done in polynomial time on tree structures. In addition also some planar models [191], and so-called matroid Ising models [78], allow for exact inference in polynomial time.

Nevertheless, efficient exact methods which work for any structure do not exist. Approximate techniques that do not sacrifice parts of the conditional independence structure are rare. There is a class of sampling-based methods which provide error bounds on their estimate, and they further do not have the uneconomic property that intermediate samples have to be thrown away. For these methods, the underlying sampling procedures differ from the techniques that we know from Section 2.2.2. Samples from the target measure $\mathbb{P}$ are not generated directly. Instead, MAP states of altered densities are computed, which can in turn be used to approximate statistics (like marginals, or the partition function) of the underlying random variable. There are two well established

Table 5.1: State-of-the-art methods to exactly or approximately compute the partition function of an undirected model with graph $G = (V, E)$, $n = |V|$, $m = |E|$. MCMC-based methods are omitted. Here, I is the number of iterations until convergence. $L = \max_{v \in V} |\mathcal{X}_v|$ and $r = \max_{v \in V} |\mathcal{N}_v|$ are the largest vertex domain and vertex degree, respectively. w is the tree-width of G and d is the dimension of the parameter space. k is the degree of a polynomial approximation to the potential function. N is the number of samples for the stochastic quadrature.

Algorithm	**Time Complexity**	**Quality**
JT [134]	$\mathcal{O}(L^w)$	Exact
MF [230]	$\mathcal{O}(InLr)$	Lower bound
LBP [245, 92]	$\mathcal{O}(ImL^2r)$	Local minimum of Bethe free energy
Log-supermodular LBP [183, 184]	$\mathcal{O}(ImL^2r)$	Lower bound
TRW [226]	$\mathcal{O}(ImL^2r + m \log n)$	Upper bound
WISH [63]	$\mathcal{O}(n \log(n/\zeta)) \times \text{TIME(MAP)}$	$(16, \zeta)$-approx
SQM [172]	$N\mathcal{O}\left(k^4 + (k+1)n + \lvert\overline{\mathcal{C}}(G)\rvert\right)$ with precomputed permutations and partitions, cf. Theorem 5.6	$(1 \pm \epsilon, 1 - \delta)$-approx when k and N satisfy Theorem 5.3

ways how the density is modified.

First, one may alter the density by randomly perturbing the parameter vector [163, 89, 90, 161, 119]. In short, one draws a per-instance noise $\boldsymbol{\epsilon}(\boldsymbol{x})$ from an extreme-value distribution $\mathbb{Q}$ and computes the perturbed MAP state

$$\boldsymbol{x}^*_{\text{PAM}} = \arg\max_{\boldsymbol{x} \in \mathcal{X}} \psi(\boldsymbol{x}) + \boldsymbol{\epsilon}(\boldsymbol{x}) \ .$$

It can be shown that for appropriate choices of $\mathbb{Q}$, $\boldsymbol{x}^*_{\text{PAM}}$ is a sample from $\mathbb{P}_{\boldsymbol{\theta}}$. This technique is also called *perturb-and-MAP* (PAM).

Second, the density can be modified by imposing hashing-based random constraints on the state space $\mathcal{X}$ [63, 64, 98]. Computing the MAP values of multiple randomly constrained state spaces, allows one to model the energy landscape (the possible values of the potential function $\psi(\boldsymbol{x})$) which can in turn serve to estimate tight bounds on the partition function. The algorithm that builds the foundation of these techniques is called weighted integrals and sums by hashing (WISH).

Due to the evaluation of MAP queries, the above methods perform a series of calls to an **NP**-oracle, and are hence not suitable for constrained systems with low clock speed. We will nonetheless have a closer look at these kind of methods, since it turns

Algorithm 5.1: Weighted Integrals and Sums by Hashing

```
input  Parameter $\boldsymbol{\theta} \in \mathbb{R}^d$, failure probability $\delta \in (0;1)$, constant $\alpha \in \mathbb{R}$
output  Approximate partition function $\hat{Z}_{\mathrm{WISH}}(\boldsymbol{\theta})$
1: $T \leftarrow \lceil \ln(n) \ln(1/\delta)/\alpha \rceil$
2: for $i = 0$ to $n$ do
3:    for $t = 1$ to $T$ do
4:       Sample uniform $\boldsymbol{A} \in \{0,1\}^{m \times n}$ and $\boldsymbol{b} \in \{0,1\}^m$
5:       $\boldsymbol{w}_i^t \leftarrow \arg\max_{\boldsymbol{x} \in \{\boldsymbol{y} \in \mathcal{X} : \boldsymbol{A}\boldsymbol{y} = \boldsymbol{b} \mod 2\}} \psi(\boldsymbol{x})$
6:    end for
7:    $\boldsymbol{M}_i \leftarrow \mathrm{Median}(\boldsymbol{w})$
8: end for
9: $\hat{Z}_{\mathrm{WISH}}(\boldsymbol{\theta}) \leftarrow \boldsymbol{M}_0 + \sum_{i=1}^{n} 2^{i-1} \boldsymbol{M}_i$
```

out that their underlying concept can be interpreted as a particular type of numeric integration, namely a left Riemann sum. This fact is of exceptional importance, since it shows us a way to approximate the partition function without touching the conditional independence structure; as opposed to the first type of MAP-based methods in which the complexity of the correct noise distribution depends on the conditional independence structure. Hence, approximating the noise distribution to ease the computation alters the structure in an implicit manner.

We will review the WISH approach and explain the corresponding error bound. Afterwards, we will present our own technique, the *stochastic quadrature method* (SQM), which is also based on numerical integration and randomization [172]. But in contrast to the above method, we employ a polynomial approximation [185] of the potential function. Rather than relying on **NP**-hard MAP queries, the proposed methods relies on sampling from a low-dimensional space. The stochastic quadrature method allows us, to devise error bounds while maintaining a low resource consumption, by partitioning the problem into a two-step procedure. Assuming that the conditional independence structure is known, we can precompute the first step, which depends on the parameter vector only through an upper bound on $\|\boldsymbol{\theta}\|_q$ ($q \in \{1,2\}$). The result can then be used to perform the second step in the context of parameter estimation, approximation of the partition function, or approximation of marginal probabilities.

5.1 Integration, Hashing and Optimization

Only a few approximate inference techniques allow us to quantify the approximation error via non-trivial bounds. Weighted integrals and sums by hashing, is a technique that relies on randomized hashing to compute a stochastic partitioning of the high dimensional state space $\mathcal{X}$ [63]. The hashing method was developed to study the dependence between the number of solutions of a problem and the hardness of a combinatorial search [213]. In what follows, we assume for ease of exposition that $\mathcal{X} = \{0,1\}^n$. Consider the partition function $Z(\boldsymbol{\theta}) = \int_{\mathcal{X}} \psi \, \mathrm{d}\, \nu$. Instead of integrating ψ over $\mathcal{X}$, one may integrate $G(z) = |\{\boldsymbol{x} \in \mathcal{X} : \psi(\boldsymbol{x}) \geq z\}|$ over $\mathbb{R}_+$, where $G(z)$ is the number of instances whose potential is at least z. The partition function may then be rewritten as $Z(\boldsymbol{\theta}) = \int_0^{+\infty} G(z)\,\mathrm{d}\,z$. Let the instances $\boldsymbol{x} \in \mathcal{X}$ be sorted in ascending order w.r.t. to their potential-function value, i.e., $\psi(\boldsymbol{x}^1) \leq \psi(\boldsymbol{x}^2) \leq \cdots \leq \psi(\boldsymbol{x}^{2^n})$ where $\boldsymbol{x}^{2^n} = \boldsymbol{x}^*$ is the MAP state. Hence, integrating G over the interval $I = [0, \psi(\boldsymbol{x}^{2^n})]$ yields Z. We will now approximate this integral by a Riemann sum. To this end, we partition I into $n+1$ subintervals,

$$\{[0,\psi(\boldsymbol{x}^1)], [\psi(\boldsymbol{x}^1),\psi(\boldsymbol{x}^2)], \dots, [\psi(\boldsymbol{x}^{2^{n-1}}),\psi(\boldsymbol{x}^{2^n})]\} = \{[\boldsymbol{b}_{n+1},\boldsymbol{b}_n], [\boldsymbol{b}_n,\boldsymbol{b}_{n-1}], \dots, [\boldsymbol{b}_1,\boldsymbol{b}_0]\},$$

and construct the corresponding left Riemann sum

$$Z_{\text{left}}(\boldsymbol{\theta}) = \sum_{i=0}^{n} G(\boldsymbol{b}_{i+1})(\boldsymbol{b}_i - \boldsymbol{b}_{i+1}) = \sum_{i=0}^{n} 2^i(\boldsymbol{b}_i - \boldsymbol{b}_{i+1}) = \boldsymbol{b}_0 + \sum_{i=1}^{n} 2^{i-1}\boldsymbol{b}_i \;.$$

Note how the partition is constructed such that the values of G at the subinterval endpoints are powers of 2. Moreover, the area under the curve of G for the i-th subinterval lies within $2^i(\boldsymbol{b}_i - \boldsymbol{b}_{i+1})$ and $2^{i+1}(\boldsymbol{b}_i - \boldsymbol{b}_{i+1})$. Hence, $Z_{\text{left}}(\boldsymbol{\theta})$ is a 2-approximation to $Z(\boldsymbol{\theta})$. The values $\boldsymbol{b}_i$ are indeed unknown (except for $\boldsymbol{b}_{n+1} = 0$). However, any q-approximation to the $\boldsymbol{b}_i$ values would yield a $2q$-approximation to $Z(\boldsymbol{\theta})$. Luckily, it turns out that a random hashing of instances into buckets of appropriate size allows us to estimate the $\boldsymbol{b}_i$ values. Let us declare some fundamental terms.

Definition 5.1 (Pairwise Independence) *Let $\mathcal{F} = \{f : \{0,1\}^n \to \{0,1\}^m\}$ be a family of Boolean functions, and let $\boldsymbol{F}$ be a discrete random variable with state space $\mathcal{F}$ and $p(\boldsymbol{F} = f) = 1/|\mathcal{F}|$. $\mathcal{F}$ is called pairwise independent, if*

- $\forall \boldsymbol{x} \in \{0,1\}^n$, $p(\boldsymbol{F}(\boldsymbol{x}) = y) = 2^{-m}$,
- $\forall \boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n : \boldsymbol{x} \neq \boldsymbol{y} \Rightarrow \boldsymbol{F}(\boldsymbol{x}) \perp\!\!\!\perp \boldsymbol{F}(\boldsymbol{y})$.

Pairwise independent hash functions can be implemented efficiently via affine maps in $\mathbb{F}(2)$ (arithmetic mod-2).

Lemma 5.1 (Pairwise Independent Hash Functions [63]) *Suppose $\boldsymbol{A} \in \{0,1\}^{m\times n}$ and $\boldsymbol{b} \in \{0,1\}^m$. The family $\mathcal{A} = \{f_{\boldsymbol{A},\boldsymbol{b}} : \{0,1\}^n \to \{0,1\}^m\}$ with $f : \boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ mod 2 is a family of pairwise independent hash functions.*

Now, the algorithm consists in repeatedly hashing instances into buckets of size 2^i, and computing the MAP value restricted to this bucket. $\boldsymbol{b}_i$ may then be estimated by the median MAP value of the corresponding bucket. We provide the pseudocode of this procedure in Algorithm 5.1. Its output is a constant factor approximation to $Z(\boldsymbol{\theta})$.

Theorem 5.1 (WISH Error Bound [63]) *For any $\delta > 0$ and positive constant $\alpha \leq 0.0042$, Algorithm 5.1 makes $\Theta(n \log \frac{n}{\delta})$ MAP queries. The output $\hat{Z}_{WISH}(\boldsymbol{\theta})$ is a 16-approximation to $Z(\boldsymbol{\theta})$ with probability at least $(1-\delta)$.*

To prove the above theorem, one has to apply the Chebyshev and Chernoff inequalities, to show that the $\boldsymbol{M}_i$, computed by Algorithm 5.1, are within $[\boldsymbol{b}_{i-c}, \boldsymbol{b}_{i+c}]$ with high probability—this part of the proof uses the pairwise independence of the employed hash functions (Lemma 5.1). The success probability for the $\boldsymbol{M}_i$ lying in the correct interval increases exponentially as a function of αT. The proof is completed by finding appropriate values for c, α, T, combined with the observation that knowing the true $\boldsymbol{b}_i$ values would result in a 2-approximation. A detailed derivation can be found in [63]. An extension, called SPARSE-WISH, achieves the same approximation guarantee while employing much shorter constraints [64]. In practice, the corresponding constrained MAP problems are easier to solve, but are indeed still **NP**-hard.

In contrast to LBP, JT or even Gibbs sampling, WISH is oblivious w.r.t. the conditional independence structure. It does neither make assumptions or restrictions on G, nor does it harm the structure in any way. In fact, the structure does not appear explicitly in any WISH-related computation. However, access to an **NP**-oracle is required. In the rest of this chapter, we devise a method which makes also no changes, assumptions, or restrictions on G. Instead of putting random constraints on $\mathcal{X}$, we approximate ψ by a polynomial, to find an inference algorithm with error bounds—the raw computational complexity being still exponential though.

5.2 Quadrature

Whenever integrating a function f is not tractable, one may resort to numerical methods in order to approximate the definite integral $I[f] = \int_l^u f(z)\,\mathrm{d}\,z$. For instance, the Riemann sum, known from the previous Section, may be used to produce different approximations, based on the points where the integrand is evaluated. In the derivation of WISH, the integrand was evaluated at the left endpoint of the interval, which leads to the left Riemann sum. Other common approaches include the right Riemann sum, the middle sum or the trapezoidal rule. However, any of these choices requires the determination of the interval end points, and hence, would lead to a WISH-like algorithm—including the need to perform MAP inference on a resource-constrained system.

A different way of performing numeric integration are general quadrature rules. There, the basic idea is to replace the integrand f by an approximation $h \approx f$, that admits tractable integration. It turns out, that choosing $h = h_k$ to be a degree-k Chebyshev polynomial approximation of f, delivers highly accurate results, due to the equioscillation property implied by near-minimax optimality. The general quadrature procedure can be summarized as

$$I[f] = \int_l^u f(x)\mathrm{d}x \approx \int_l^u h_k(x)\mathrm{d}x = \sum_{i=0}^{k} w_i f(x_i) = I_k[f] \tag{5.1}$$

where w_i are certain coefficients and x_i are certain abscissae in $[l, u]$ (all to be determined) [142]. Depending on the choice of interpolation points and different kinds of orthogonality properties, Chebyshev polynomial based quadrature rules are termed Gauss-Chebyshev quadrature, Fejér quadrature or Clenshaw-Curtis quadrature [42].

An upper bound on the approximation error of the quadrature rule can be derived from the upper bound on the polynomial approximation error (2.8). Assuming the terminology and conditions of Theorem 2.8, the error of (5.1) is bounded by

$$|I[f] - I_{\zeta}[f]| \leq \frac{32V_k}{15k\pi 2K(2K-1)\dots(2K+1-k)} \, . \tag{5.2}$$

Derivation and bounds for various special cases can be found in [142, 237] and references therein.

This bound holds if we integrate over a closed interval $[l,u]$. When we transfer the general quadrature to the partition function of a discrete exponential family member, the analogue to (5.1) is

$$Z(\boldsymbol{\theta}) = \int_{\mathcal{X}} \psi(\boldsymbol{x}) \,\mathrm{d}\,\nu(\boldsymbol{x}) = \sum_{\boldsymbol{x}\in\mathcal{X}} \exp(\langle\boldsymbol{\theta},\phi(\boldsymbol{x})\rangle) \approx \sum_{\boldsymbol{x}\in\mathcal{X}} \hat{\exp}^k_{\boldsymbol{\zeta}}(\langle\boldsymbol{\theta},\phi(\boldsymbol{x})\rangle) = \hat{Z}^k_{\boldsymbol{\zeta}}(\boldsymbol{\theta}) \, . \tag{5.3}$$

Here, $\hat{\exp}^k_{\boldsymbol{\zeta}}$ is a degree-k polynomial approximation with coefficients $\boldsymbol{\zeta} \in \mathbb{R}^{k+1}$, to the exponential function over the interval $[l;u]$ with $l = \min_{\boldsymbol{x}\in\mathcal{X}}\langle\boldsymbol{\theta},\phi(\boldsymbol{x})\rangle$ and $u = \max_{\boldsymbol{x}\in\mathcal{X}}\langle\boldsymbol{\theta},\phi(\boldsymbol{x})\rangle$. Note that the (discrete) integration is not carried out over $[l;u]$—only a finite set of values from $[l;u]$ will appear as an argument of exp and the error bound of the general quadrature (5.2) does not apply, because in the worst-case, the quantity $\langle\boldsymbol{\theta},\phi(\boldsymbol{x})\rangle$ could hit a maximum of the error function of the underlying polynomial approximation for each $\boldsymbol{x}$. Hence, when ε is the worst-case error of the polynomial approximation, then

$$|Z(\boldsymbol{\theta}) - \hat{Z}^k_{\boldsymbol{\zeta}}(\boldsymbol{\theta})| = \left| \sum_{\boldsymbol{x}\in\mathcal{X}} \exp(\langle\boldsymbol{\theta},\phi(\boldsymbol{x})\rangle) - \hat{\exp}^k_{\boldsymbol{\zeta}}(\langle\boldsymbol{\theta},\phi(\boldsymbol{x})\rangle) \right| \leq \varepsilon|\mathcal{X}| \, . \tag{5.4}$$

Note that the range $[l;u]$ on which we approximate exp has to be known. Computing the exact interval $[l^*;u^*] = [\langle\boldsymbol{\theta},\phi(\boldsymbol{x}^{\min})\rangle;\langle\boldsymbol{\theta},\phi(\boldsymbol{x}^{\max})\rangle]$ is **NP**-hard, since $\boldsymbol{x}^{\max} = \boldsymbol{x}^*$ is the MAP state of the model, and $\boldsymbol{x}^{\min}$ is the MAP state of the same model with negated parameters, i.e., $\boldsymbol{x}^{\min} = \arg\min_{\boldsymbol{x}\in\mathcal{X}} p_{-\boldsymbol{\theta}}(\boldsymbol{x})$. The interval $[l^*;u^*]$ can indeed be approximated without solving the exact MAP. For Chebyshev polynomials, we know from (2.8) that the approximation error is related to the norm of the k-th derivative of the function we are trying to approximate. Due to $\partial\exp(z)/\partial z = \exp(z)$ and monotonicity of exp, we shall not overestimate the upper limit u too much. This would harm the tightness of the polynomial error bound. In case of a given $\boldsymbol{\theta}$, techniques for the approximation of maximum a posteriori states (e.g., LP-relaxations; cf. Section 2.2.2) deliver bounds on the MAP value, which can in turn be used to approximate the interval $[l^*;u^*]$. When the parameters are to be estimated, in case of binary sufficient statistics, $\langle\boldsymbol{\theta},\phi(\boldsymbol{x})\rangle \leq \|\boldsymbol{\theta}\|_1$ follows from Hölder's inequality. We may thus apply regularization based techniques (e.g., [240]) for parameter estimation, to guarantee that $\|\boldsymbol{\theta}\|_1$ and thus $\langle\boldsymbol{\theta},\phi(\boldsymbol{x})\rangle$ will not exceed any predefined upper bound B.

Considering (5.3) again, we see that $\hat{Z}^k_{\boldsymbol{\zeta}}(\boldsymbol{\theta})$ involves a summation over $\mathcal{X}$, i.e., over exponentially many objects. It is still unclear, how this approach could facilitate inference on a resource-constrained system. To shed some light on this issue, we discover a new property of sufficient statistics, before we devise the actual algorithm and its error bound.

5.3 Integrable Sufficient Statistics

Data enters an exponential family model only through the sufficient statistic ϕ. The actual functional form of ϕ determines how density is assigned to the various possible realizations of a random variable. Moreover, we know that it encodes the conditional independence structure, and hence which joint realizations of variables have an indivisible impact on the density. Also, other important properties of sufficient statistics, like overcompleteness or minimality, have shown to directly influence the type of convexity (strict vs. non-strict) of the objective function, and hence the identifiability of the estimation problem. Overcompleteness allows us in addition, to derive properties like shift-invariance, and a straightforward derivation of the Lipschitz constant—due to the fact that for all $\boldsymbol{x} \in \mathcal{X}$:

$$\|\phi(\boldsymbol{x})\|_1 = |\overline{\mathcal{C}}(G)|, \qquad \|\phi(\boldsymbol{x})\|_2 = \sqrt{|\overline{\mathcal{C}}(G)|}, \quad \text{and} \quad \|\phi(\boldsymbol{x})\|_\infty = 1,$$

whenever $\boldsymbol{X}$ is discrete and ϕ is overcomplete. We will now investigate a new property, which will become important when we derive the stochastic quadrature method.

Definition 5.2 (k-integrable Sufficient Statistics [172]) *The sufficient statistic $\phi : \mathcal{X} \to \mathbb{R}^d$ is called k-integrable, if the function $\chi_\phi^k : [d]^k \to \mathbb{R}$, with*

$$\chi_\phi^k(\boldsymbol{j}) = \int_{\mathcal{X}} \prod_{i=1}^{k} \phi(\boldsymbol{x})_{\boldsymbol{j}_i} \,\mathrm{d}\,\nu(\boldsymbol{x})$$

admits a polynomial time computable closed-form expression for all $\boldsymbol{j} \in [d]^k$.

At a first glace, k-integrability seems to be a rather theoretical feature. Moreover, it is unclear which, or if any, reasonable sufficient statistic has this property. We will now see that both points depend crucially on the type of random variable.

5.3.1 Discrete Random Variables

Suppose that $\boldsymbol{\theta}$ is a discrete random variable with state space $\mathcal{X}$, and binary, overcomplete ϕ. Let us interpret the values of χ_ϕ^k. First, recall that there is a one-to-one correspondence between each index $j \in [d]$ of the sufficient statistic $\phi(\boldsymbol{x})$ and a specific joint state $\boldsymbol{y} \in \mathcal{X}_C$ of a particular clique $C \in \overline{\mathcal{C}}(G)$. Consequently, any vector $\boldsymbol{j} \in [d]^k$ corresponds to an index tuple $\boldsymbol{j} = (\boldsymbol{j}_1, \boldsymbol{j}_2, \ldots, \boldsymbol{j}_k)$ that can be interpreted as a tuple of clauses:

$$\boldsymbol{j} = (\boldsymbol{j}_1, \boldsymbol{j}_2, \ldots, \boldsymbol{j}_k) \equiv (\boldsymbol{X}_{C_1} = \boldsymbol{y}_{C_1}, \boldsymbol{X}_{C_2} = \boldsymbol{y}_{C_2}, \ldots, \boldsymbol{X}_{C_k} = \boldsymbol{y}_{C_k}) \,. \tag{5.5}$$

Here, $C_i \in \overline{\mathcal{C}}(G)$, $\boldsymbol{y}_{C_i} \in \mathcal{X}_{C_i}$, and $\boldsymbol{X}_{C_i} = \boldsymbol{y}_i$ is the clause that corresponds to the i-th index in the index tuple $\boldsymbol{j}$. Since ϕ is binary, the product $\prod_{i=1}^{k} \phi(\boldsymbol{x})_{\boldsymbol{j}_i}$ will evaluate to 0, whenever at least one of the $\phi(\boldsymbol{x})_{\boldsymbol{j}_i}$ is 0, and 1 otherwise. We may hence interpret the product over components of $\phi(\boldsymbol{x})$ as the conjunction of the corresponding clauses[25], i.e.,

$$\prod_{i=1}^{k} \phi(\boldsymbol{x})_{\boldsymbol{j}_i} \equiv (\boldsymbol{x}_{C_1} = \boldsymbol{y}_{C_1} \wedge \boldsymbol{x}_{C_2} = \boldsymbol{y}_{C_2} \wedge \cdots \wedge \boldsymbol{x}_{C_k} = \boldsymbol{y}_{C_k}) = \varphi_{\boldsymbol{j}}(\boldsymbol{x}) \,. \tag{5.6}$$

The discrete version of the function $\chi_{\phi}^{k}(\boldsymbol{j})$ from Definition 5.2 is the summation of the above expression over all $\boldsymbol{x} \in \mathcal{X}$. Hence, $\chi_{\phi}^{k}(\boldsymbol{j})$ counts the number of configurations $\boldsymbol{x} \in \mathcal{X}$ which satisfy the formula $\varphi_{\boldsymbol{j}}$. Based on this interpretation of index tuple, we devise the following result.

Lemma 5.2 (k-integrability of Discrete MRF Models [172]) *The binary, overcomplete sufficient statistic (Definition 2.6) of an n-dimensional discrete random variable, is k-integrable for all $k \in \mathbb{N}$.*

Proof. Let us fix an arbitrary $\boldsymbol{j} \in [d]^k$ and consider the corresponding logical expression (5.6). Obviously, $\varphi_{\boldsymbol{j}}$ is not satisfiable, whenever it contains contradicting clauses, e.g., $\boldsymbol{X}_{C_i} = \boldsymbol{y}_{C_i} \wedge \boldsymbol{X}_{C_j} = \boldsymbol{y}_{C_j}$ where $C_i = C_j$ are the same clique, but $\boldsymbol{y}_{C_i} \neq \boldsymbol{y}_{C_j}$ are two different states. In this case, no $\boldsymbol{x} \in \mathcal{X}$ can satisfy $\varphi_{\boldsymbol{j}}$ and hence $\chi_{\phi}^{k}(\boldsymbol{j}) = 0$. We will call such index tuples non-realizable, otherwise realizable. Checking if a tuple is realizable requires $k|C_{\max}|$ steps—for each of the k indices in the tuple, we write the, at most $|C_{\max}| = \max_{C \in \overline{\mathcal{C}}(G)} |C|$, states into an array of length n, and check, if they contradict with the state we have seen so far, if any.

Now, recall that $|\mathcal{X}| = \prod_{v \in V} |\mathcal{X}_v|$. Obviously, $\sum_{\boldsymbol{x} \in \mathcal{X}} 1 = |\mathcal{X}|$. When we fix the state $x \in \mathcal{X}_u$ of an arbitrary variable $u \in V$, then $\sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{1}_{\{\boldsymbol{x}_u = x\}} = |\mathcal{X}|/|\mathcal{X}_u|$. To see this, observe that the full joint state space size is no longer $\prod_{v \in V} |\mathcal{X}_v|$, but instead $\prod_{v \in V \setminus \{u\}} |\mathcal{X}_v| = |\mathcal{X}|/|\mathcal{X}_u|$—the value of u is fixed. When we fix not only the state of a single vertex u, but of a vertex subset $U \subseteq V$, then $\sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{1}_{\{\boldsymbol{x}_U = \boldsymbol{y}\}} = |\mathcal{X}|/|\mathcal{X}_U|$.

Suppose that $\boldsymbol{j}$ is realizable. Then $\boldsymbol{j} \in [d]^k$ fixes the states of the vertices in $W = \bigcup_{i=1}^{k} C_i$ to some value $\boldsymbol{y}_W$. Hence, $\mathbb{1}_{\{\boldsymbol{x}_W = \boldsymbol{y}_W\}} \equiv \varphi_{\boldsymbol{j}}(\boldsymbol{x})$, and thus

$$\chi_{\phi}^{k}(\boldsymbol{j}) = \sum_{\boldsymbol{x} \in \mathcal{X}} \prod_{i=1}^{k} \phi(\boldsymbol{x})_{\boldsymbol{j}_i} = \sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{1}_{\{\boldsymbol{x}_W = \boldsymbol{y}_W\}} = \frac{|\mathcal{X}|}{|\mathcal{X}_W|} \,,$$

for any realizable index tuple $\boldsymbol{j}$. So finally,

$$\chi_{\phi}^{k}(\boldsymbol{j}) = \begin{cases} \frac{|\mathcal{X}|}{|\mathcal{X}_{\bigcup_{i=1}^{k} C_{\boldsymbol{j}_i}}|} & , \boldsymbol{j} \text{ is realizable} \\ 0 & , \text{ otherwise} \,. \end{cases}$$

[25]We use the symbol $\equiv$ to indicate the equivalence between arithmetic expressions and expressions in propositional logic.

We did not make any assumptions on $\boldsymbol{j}$ and k, which implies that the binary, overcomplete ϕ is k-integrable. ■

Note, that this result is quite general. It covers all exponential family members for discrete random variable—for those, an overcomplete, binary sufficient statistic always exists.

5.3.2 Continuous Random Variables

One could apply the k-integrability result for discrete random variables directly to continuous state spaces via discretization. E.g., by using indicator functions that tell us if a continuous realization lies within a specific interval. Moreover, it turns out that the sufficient statistic of common exponential family densities can be denoted in terms of elementary functions which have elementary antiderivatives. In fact, the sufficient statistics of the Gaussian, Bernoulli, Poisson, exponential, Pareto, Weibull, chi-squared, log-normal, beta, gamma, inverse gamma, and Dirichlet densities, consist only of terms of the form $1/x^c$, x^c, and $\log(x)^c$ with $c \in \mathbb{N}, x \in \mathbb{R}$. For each of them being integrable to a closed-form function, and any finite product of them being integrable to a closed-form, too.

Lemma 5.3 (Continuous k-integrability) *Let $\boldsymbol{X}$ be an n-dimensional continuous random vector whose dimensions have support on $(0; u_i]$ for $1 \le i \le n$ and $u_i \in \mathbb{R}_+$. Any real-valued sufficient statistic $\phi(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}^d$, whose coordinate-wise statistics $\phi(\boldsymbol{x})_i$ are of the form*

$$\phi(\boldsymbol{x})_i = \boldsymbol{x}_j^c, \text{ or } \phi(\boldsymbol{x})_i = \tfrac{1}{\boldsymbol{x}_j^c}, \text{ or } \phi(\boldsymbol{x})_i = \log(\boldsymbol{x}_j)^c,$$

for some $c \in \mathbb{N}, j \in [n]$, is k-integrable for all $k \in \mathbb{N}$. The log *may be taken w.r.t. any base.*

Proof. Each of the coordinate-wise statistics which are mentioned in the lemma, have closed-form integrals for any $c \in \mathbb{N}$. Now, in the course of integrating the product $\int_{\mathcal{X}} \prod_{i=1}^k \phi(\boldsymbol{x})_{\boldsymbol{j}_i} \,\mathrm{d}\,\nu(\boldsymbol{x})$ for some fixed $\boldsymbol{j} \in [d]^k$, we can encounter the following situations. First, products of the same coordinate-wise statistics, say $\phi(\boldsymbol{x})_{\boldsymbol{j}_l}$, yield a statistic of the same functional form, but with a power of $2c$ instead of c. Hence, those products have a closed-form integral. The integral of the product of different coordinate-wise statistics $\phi(\boldsymbol{x})_{\boldsymbol{j}_l}$ and $\phi(\boldsymbol{x})_{\boldsymbol{j}_h}$ $(l, h \in [k], l \neq k)$, which access different components of $\boldsymbol{x}$, say $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$, is simply the product of the corresponding integrals, i.e., $\int\int \phi(\boldsymbol{x})_{\boldsymbol{j}_l}\phi(\boldsymbol{x})_{\boldsymbol{j}_h} \,\mathrm{d}\,\boldsymbol{x}_a \,\mathrm{d}\,\boldsymbol{x}_b = \left(\int \phi(\boldsymbol{x})_{\boldsymbol{j}_l} \,\mathrm{d}\,\boldsymbol{x}_a\right)\left(\int \phi(\boldsymbol{x})_{\boldsymbol{j}_h} \,\mathrm{d}\,\boldsymbol{x}_b\right)$. The last case is the integration of the product of statistics $\phi(\boldsymbol{x})_{\boldsymbol{j}_l}$ and $\phi(\boldsymbol{x})_{\boldsymbol{j}_h}$ (with $l \neq h$) which involve the same variable $\boldsymbol{x}_a$. Any such integral can be written in the form

$$\int x^{c_1} \log(x)^{c_2} \,\mathrm{d}\, x \,,$$

where $c_1 \in \mathbb{Z}$ and $c_2 \in \mathbb{N}$. Then, for $c_1 \neq -1$,

$$\int x^{c_1} \log(x)^{c_2} \,\mathrm{d}\, x = \frac{x^{c_1+1}\log(x)^{c_2}}{c_1+1} - \frac{n}{m+1}\int x^{c_1}\log(x)^{c_2-1} \,\mathrm{d}\, x \,.$$

For any fixed sufficient statistic as defined in the lemma, c_2 is a known integer constant, and we may repeat the unrolling of the integral until we arrive at a summation over a constant number (c_2) of term. ■

Details on the integration of elementary functions can be found in [29]. One may extend the above lemma to include sufficient statistics of the type $|x-m|^c$, which appears in the density of the Laplace distribution. Closed-form expressions for integrals of such functions exist. However, we excluded them here due to the cumbersome notation which arises when products of such functions are integrated.

Based on the fact that all discrete and various continuous exponential family densities have k-integrable statistics, we derive a new inference method that exploits k-integrability.

5.4 Stochastic Quadrature Method

The term stochastic quadrature is slightly ambiguous, since a former body of work [81, 82, 83, 159] refers to stochastic quadrature as a specific form of Monte Carlo integration. Therein, N points from an n-dimensional space $\mathcal{X}$ are sampled, and the integral $Z(\boldsymbol{\theta}) = \int_{\mathcal{X}} \psi \, \mathrm{d}\,\nu$ is then approximated by summing and reweighting the samples. The term stochastic refers to the way how we approximate the integral. In contrast, the stochastic quadrature method that we present in the sequel approximates the integral via a general quadrature rule (5.1), e.g., by approximating the integrand by a polynomial. Due to nice algebraic properties of polynomials and k-integrability, the expression for the partition function can be simplified. This part is deterministic. In a second step, the simplified expression is approximated via sampling, which gives rise to the term stochastic. Therefore, we refer to our approach as stochastic quadrature method, while the former is called Monte Carlo integration. Our technique is conceptually related to WISH, in that WISH employs a deterministic approximation of the integral in terms of a Riemann sum, while the boundaries of the intervals are subject to a stochastic approximation through random constraints.

To actually derive the SQM, let us define a specific probability density function over index tuples $\boldsymbol{j} \in [d]^i$ of length $0 \leq i \leq k$. For ease of notation, we assume that indices of $(k+1)$-dimensional objects start at 0.

Definition 5.3 (Index Tuple Density) *Suppose $\phi : \mathcal{X} \to \mathbb{R}_+^d$ is a non-negative, k-integrable, sufficient statistic. Let $\mathbf{c}, \boldsymbol{q} \in \mathbb{R}^{k+1}$ with $\boldsymbol{q}_i = \|\chi_\phi^i\|_1$ and $\tau = \langle |\mathbf{c}|, \boldsymbol{q} \rangle$, where $\|\chi_\phi^i\|_1$ denotes the 1-norm of the function χ_ϕ^i, and $|\mathbf{c}|$ denotes the element-wise absolute value of $\mathbf{c}$, i.e.,*

$$\|\chi_\phi^i\|_1 = \sum_{\boldsymbol{j} \in [d]^i} |\chi_\phi^i(\boldsymbol{j})| \qquad \text{and} \qquad |\mathbf{c}| = \begin{pmatrix} |\mathbf{c}_1| \\ \vdots \\ |\mathbf{c}_k| \end{pmatrix} .$$

Let further $(\boldsymbol{J}, I)$ be the discrete random variable with state space $[d]^k \times ([k] \cup \{0\})$ and *joint density* $p_{\boldsymbol{c},\phi}(\boldsymbol{J}=\boldsymbol{j}, I=i) = p_{\phi}(\boldsymbol{J}=\boldsymbol{j} \mid I=i) p_{\boldsymbol{c},\phi}(I=i)$ *with*

$$p_{\boldsymbol{c},\phi}(I=i) = \frac{|\boldsymbol{c}_i| \boldsymbol{q}_i}{\tau} \qquad \text{and} \qquad p_{\phi}(\boldsymbol{J}=\boldsymbol{j} \mid I=i) = \frac{\chi^i_{\phi}(\boldsymbol{j})}{\boldsymbol{q}_i} \; .$$

We call $p_{\boldsymbol{c},\phi}$ *the tuple density of* $(\boldsymbol{J}, I)$ *with parameter* $(\boldsymbol{c},\phi)$*.*

Now, based on the tuple density, we define the random variable which is the core of SQM.

Definition 5.4 (1-Stochastic Quadrature) *Let* $\boldsymbol{J}$ *be a random index tuple of random length* I*, having joint tuple density* $p_{\boldsymbol{c},\phi}$*. The random variable*

$$\hat{Z}^k_{\boldsymbol{J},I}(\boldsymbol{\theta}) = \tau \operatorname{sgn}(\boldsymbol{c}_I) \prod_{r=0}^{I} \boldsymbol{\theta}_{\boldsymbol{J}_r}$$

with τ *as defined in Definition 5.3, is called 1-SQM.*

We will now see that the above random variable is closely related to the quadrature approximation to $Z(\boldsymbol{\theta})$ (5.3).

Theorem 5.2 (Unbiasedness of SQM) *Let the parameter* $\boldsymbol{c}$ *of the tuple density, be equal to the coefficient vector* $\boldsymbol{\zeta}$ *of a degree-*k *approximation to* exp *over some interval. Then,* $\hat{Z}^k_{\boldsymbol{J},I}(\boldsymbol{\theta})$ *is an unbiased estimator for* $\hat{Z}^k_{\boldsymbol{\zeta}}(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}\in\mathcal{X}} \widehat{\exp}^k_{\boldsymbol{\zeta}}(\langle \boldsymbol{\theta}, \phi(\boldsymbol{x})\rangle)$*.*

Proof. It follows from Definitions 7.3, 5.3, and 5.4, that

$$\begin{aligned}
\mathbb{E}\left[\hat{Z}^k_{\boldsymbol{J},I}(\boldsymbol{\theta})\right] &= \sum_{i=0}^{k} \sum_{\boldsymbol{j}\in[d]^k} \mathbb{P}_{\boldsymbol{\zeta},\phi}(\boldsymbol{J}=\boldsymbol{j}, I=i) \tau \operatorname{sgn}(\boldsymbol{\zeta}_i) \prod_{r=0}^{i} \boldsymbol{\theta}_{\boldsymbol{j}_r} \\
&= \sum_{i=0}^{k} \boldsymbol{\zeta}_i \sum_{\boldsymbol{j}\in[d]^i} \left(\prod_{r=0}^{i} \boldsymbol{\theta}_{\boldsymbol{j}_r}\right) \sum_{\boldsymbol{x}\in\mathcal{X}} \left(\prod_{r=0}^{i} \phi(\boldsymbol{x})_{\boldsymbol{j}_r}\right) \\
&= \sum_{\boldsymbol{x}\in\mathcal{X}} \sum_{i=0}^{k} \boldsymbol{\zeta}_i \langle \boldsymbol{\theta}, \phi(\boldsymbol{x})\rangle^i \; = \; \hat{Z}^k_{\boldsymbol{\zeta}}(\boldsymbol{\theta}) \; ,
\end{aligned}$$

where the last equality stems from the fact that

$$\langle \boldsymbol{\theta}, \phi(\boldsymbol{x})\rangle^i = \sum_{j_1=1}^{d} \sum_{j_2=1}^{d} \cdots \sum_{j_i=1}^{d} \prod_{l=0}^{i} \boldsymbol{\theta}_{j_l} \prod_{r=0}^{i} \phi(\boldsymbol{x})_{j_r} \; .$$

■

The theorem implies a simple Monte Carlo procedure, called N-SQM or simply SQM, shown in Algorithm 5.2. The output is unbiased and we will see in Section 5.4.1, that the probability of large deviations from $\hat{Z}^k_{\boldsymbol{\zeta}}(\boldsymbol{\theta})$ decreases exponentially with an increasing number of samples N. However, the algorithm expects $p_{\boldsymbol{\zeta},\phi}$ and τ as an input. While the computation of $\boldsymbol{\zeta}$ is rather efficient—$\mathcal{O}(k \log k)$ via DCT or $\mathcal{O}(k^2)$ via (2.44)—the complexity of computing $\|\chi^i_{\phi}\|_1$ (and thus τ) is unclear and we will study this topic in Section 5.4.2.

Algorithm 5.2: Stochastic Quadrature Method (N-SQM)

```
input $\boldsymbol{\theta}$, $k$, $\boldsymbol{\zeta}$, $N$
output Approximate partition function $\hat{Z}_{\boldsymbol{\zeta}}^{N,k}(\boldsymbol{\theta})$
1: $S \leftarrow 0$
2: for $l = 1$ to $N$ do
3:     $(\boldsymbol{j}, i) \sim \mathbb{P}_{\boldsymbol{\zeta},\phi}$
4:     $S \leftarrow S + \hat{Z}_{\boldsymbol{j},i}^{k}(\boldsymbol{\theta})$
5: end for
6: $\hat{Z}_{\boldsymbol{\zeta}}^{N,k}(\boldsymbol{\theta}) \leftarrow \frac{1}{N} S$
```

5.4.1 Approximation Error and Sample Complexity

We will now analyze the error of Algorithm 5.2. When a random variable is bounded in $[l; u]$, so is its variance bounded by $1/4(l-u)^2$. Any Monte Carlo estimate of the partition function via independent uniform sampling over $\mathcal{X}$, has a variance in $\mathcal{O}(|\mathcal{X}|^2(\psi(\boldsymbol{x}^{\max}) - \psi(\boldsymbol{x}^{\min}))^2)$ (recall that the underlying random variable is $|\mathcal{X}|\psi(\boldsymbol{X})$). On the other hand, the random variable $\hat{Z}_{\boldsymbol{J},I}^{k}(\boldsymbol{\theta})$—the core of SQM—has its variance in $\mathcal{O}(\tau^2\|\boldsymbol{\theta}\|_\infty^{2k'})$, where k' is 1 when $\|\boldsymbol{\theta}\|_\infty < 1$ and $k' = k$, otherwise. While it's not obvious which of the two estimators has the lower variance, the term $(\psi(\boldsymbol{x}^{\max}) - \psi(\boldsymbol{x}^{\min}))^2$ is hard to control, while $\|\boldsymbol{\theta}\|_\infty$ may be controlled directly via regularization.

Having this in mind, we derive a probabilistic bound on the relative error of SQM, and show how it is related to the number of samples N and the polynomial degree k. A major difference between plain Monte Carlo estimation and SQM is, that the former is a randomized procedure for the estimation of $Z(\boldsymbol{\theta})$, whereas SQM is a randomized procedure for the estimation of $\hat{Z}_{\boldsymbol{\zeta}}^{k}(\boldsymbol{\theta})$. The error bounds of $\hat{Z}_{\boldsymbol{\zeta}}^{k}(\boldsymbol{\theta})$ and the sampling procedure have to be chained to get a bound on the overall error. Since Chebyshev approximations are near-minimax optimal, and tight error bounds of such approximations are known, we restrict ourselves to those when we derive the error bound of SQM. However, different choices of polynomial approximations will lead to different error bounds.

Theorem 5.3 (SQM Error Bound [172]) *Let $\boldsymbol{\zeta}$ be the coefficient vector of a degree-k Chebyshev approximation to* exp *on $[l; u] = [-\|\boldsymbol{\theta}\|_1; +\|\boldsymbol{\theta}\|_1]$ with worst-case error ε. Let $\hat{Z}_{\boldsymbol{\zeta}}^{N,k}(\boldsymbol{\theta})$ be the output of Algorithm 5.2. Furthermore, let $\delta \in (0, 1]$, $\epsilon > 0$, $N = (\log 2/\delta)\tau^2 2\|\boldsymbol{\theta}\|_\infty^{2k'}\varepsilon^{-2}|\mathcal{X}|^{-2}$, with $(k-1)\,k! \geq 8\exp(2\|\boldsymbol{\theta}\|_1)/(\pi\epsilon)$, and $k' = 1$ if $\|\boldsymbol{\theta}\|_\infty < 1$ or otherwise $k' = k$. Then,*

$$\mathbb{P}[|\hat{Z}_{\boldsymbol{\zeta}}^{N,k}(\boldsymbol{\theta}) - Z(\boldsymbol{\theta})| < \epsilon Z(\boldsymbol{\theta})] \;\geq\; 1 - \delta\,. \tag{5.7}$$

Proof. By Hoeffding's inequality [94], for N independent samples from a random variable $\boldsymbol{Y}$, bounded in $[a; b]$, we have

$$\mathbb{P}\left[\left|\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{y}^i - \mathbb{E}[\boldsymbol{Y}]\right| \geq t\right] \leq 2\exp\left(-\frac{2Nt^2}{(b-a)^2}\right)\,.$$

Setting $t = \varepsilon|\mathcal{X}|$ in combination with (5.4), Theorem 5.2 and the triangle inequality, we get

$$\begin{aligned}\mathbb{P}[|\hat{Z}_{\boldsymbol{\zeta}}^{N,k}(\boldsymbol{\theta}) - \hat{Z}_{\boldsymbol{\zeta}}^{k}(\boldsymbol{\theta})| \geq \varepsilon|\mathcal{X}|] &\leq 2\exp\left(-N\frac{\varepsilon^2|\mathcal{X}|^2}{\tau^2 2\|\boldsymbol{\theta}\|_\infty^{2k'}}\right) = \delta \\ \Rightarrow \mathbb{P}[|\hat{Z}_{\boldsymbol{\zeta}}^{N,k}(\boldsymbol{\theta}) - \hat{Z}_{\boldsymbol{\zeta}}^{k}(\boldsymbol{\theta})| + |\hat{Z}_{\boldsymbol{\zeta}}^{k}(\boldsymbol{\theta}) - Z(\boldsymbol{\theta})| \geq 2\varepsilon|\mathcal{X}|] &\leq \delta \\ \Rightarrow \mathbb{P}[|\hat{Z}_{\boldsymbol{\zeta}}^{N,k}(\boldsymbol{\theta}) - Z(\boldsymbol{\theta})| \geq 2\varepsilon|\mathcal{X}|] &\leq \delta \,. \end{aligned} \tag{5.8}$$

Now, we apply the error bound for Chebyshev approximations (Theorem 2.8) and Hölder's inequality [95].

$$\varepsilon 2|\mathcal{X}| \leq \frac{4\exp\|\boldsymbol{\theta}\|_1}{\pi\,(k-1)\,k!}2|\mathcal{X}| \leq \epsilon\frac{|\mathcal{X}|}{\exp\|\boldsymbol{\theta}\|_1} < \epsilon Z(\boldsymbol{\theta}) \tag{5.9}$$

The rightmost inequality, known as naive mean field lower bound [227], follows from (2.13). More precisely,

$$\log Z(\boldsymbol{\theta}) = \sup_{\boldsymbol{\mu}\in\mathcal{M}} \langle\boldsymbol{\theta},\boldsymbol{\mu}\rangle + \mathcal{H}(\boldsymbol{\mu}) \geq \langle\boldsymbol{\theta},\boldsymbol{\mu}\rangle + \mathcal{H}(\boldsymbol{\mu}) \quad \forall\boldsymbol{\mu}\in\mathcal{M}(G)\,,$$

where $\mathcal{M}(G)$ is the marginal polytope (2.12) and $\mathcal{H}(\boldsymbol{\mu})$ is the entropy of the density implied by $\boldsymbol{\mu}$. Since the inequality is valid for all $\boldsymbol{\mu}\in\mathcal{M}(G)$, we may choose the fully factorized density with uniform marginals. Hence,

$$\mathcal{H}(\boldsymbol{\mu}) = -\sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x})\log p(\boldsymbol{x}) = \sum_{\boldsymbol{x}\in\mathcal{X}} \frac{1}{\prod_{v\in V}|\mathcal{X}_v|}\log\prod_{v\in V}|\mathcal{X}_v| = \log|\mathcal{X}|\,.$$

Combining this with $\langle\boldsymbol{\theta},\boldsymbol{\mu}\rangle > -\|\boldsymbol{\theta}\|_1$ implies $\log Z(\boldsymbol{\theta}) > -\|\boldsymbol{\theta}\|_1 + \log|\mathcal{X}|$ and thus $Z(\boldsymbol{\theta}) > |\mathcal{X}|/\exp\|\boldsymbol{\theta}\|_1$, which explains the last inequality in (5.9). The statement of the theorem is then derived by plugging (5.9) into the probability that is complementary to (5.8). ■

So if $\|\boldsymbol{\theta}\|_\infty < 1$, a small number of samples will suffice to achieve low error with high probability. Moreover, any choice of N and k implies a particular $(\varepsilon,\epsilon,\delta)$-triple, where ε is the worst-case error of the polynomial approximation, ϵ is the relative error of SQM and $1-\delta$ is the probability that this error is not exceeded. We may thus tailor the choice of N and k to the underlying resource-constrained system, and get an appropriate error bound. In addition, when computational resources are allocated dynamically, we may stop the algorithm after any number of samples N'—the procedure may hence be interpreted as anytime algorithm.

Considering the total complexity of different randomized methods, one can see that in plain Monte Carlo integration, all the complexity goes into the generation of samples. In WISH, the randomization is realized via Bernoulli random variables, where the i-th variable represents the event that the instance with 2^i-th strongest potential survives the random constraints. Due to low variance in each Bernoulli variable, the number of samples required for WISH is moderate compared to vanilla Monte Carlo integration or

SQM. On the other hand, significant computational effort goes into the computation of constrained MAP states. The method presented here, SQM, requires the computation of a polynomial approximation and the normalization of the tuple density, which we discussed in the previous Section. The number of samples required to achieve a specific relative error can be controlled via $\|\boldsymbol{\theta}\|_\infty$. When we investigate the SQM sampling itself, it turns out that structural properties of $p_{\boldsymbol{\zeta},\phi}$ lead to efficient sampling schemes.

5.4.2 Normalizing the Tuple Density

Recall that the index tuple density (Definition 5.3) is normalized via $\boldsymbol{q}_i = \|\chi_\phi^i\|_1$ and $\tau = \langle |\boldsymbol{c}|, \boldsymbol{q}\rangle$ (Definition 5.3). Reviewing the definition of $\|\chi_\phi^i\|_1$,

$$\|\chi_\phi^i\|_1 = \sum_{\boldsymbol{j}\in[d]^i} |\chi_\phi^i(\boldsymbol{j})| \;, \tag{5.10}$$

we see that a naive computation of τ requires $\mathcal{O}(d^k)$ steps where k is the degree of our polynomial approximation. The first thing to note is that if $\|\boldsymbol{\theta}\|_1$ is assumed to be upper bounded by a constant B (which can be guaranteed via regularized parameter estimation), then, the upper bound on $\langle \boldsymbol{\theta}, \phi(\boldsymbol{x})\rangle$ is constant, which implies that the approximation interval $[l;u]$ is constant, and thus k is constant. Following this reasoning, $\mathcal{O}(d^k)$ would be a polynomial expression. But if the norm of $\boldsymbol{\theta}$ is not assumed to be constant, then $\mathcal{O}(d^k)$ is indeed exponential. However, we can remove the dependence on d when we restrict ourselves to discrete models, by relying on the invariance of χ_ϕ^i against permutation of $\boldsymbol{j}$, and the independence between χ_ϕ^i and the actual state assignment of realizable index tuples.

In what follows, we explain how these properties can lead to an improved procedure for normalizing the tuple density. We will group the elements of $[d]^i$ into equivalence classes and use them to rewrite and simplify the summation in (5.10). In this context, it is important to recall the semantic of index tuples $\boldsymbol{j}$ from (5.5). In addition to index tuples, we will also work with clique tuples. We say that a clique tuple $\boldsymbol{C} = (\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_k)$ corresponds to the index tuple $\boldsymbol{j} = (\boldsymbol{j}_1, \boldsymbol{j}_2, \ldots, \boldsymbol{j}_k)$, if $\boldsymbol{j} \equiv (\boldsymbol{X}_{\boldsymbol{C}_1} = \boldsymbol{y}_{\boldsymbol{C}_1}, \boldsymbol{X}_{\boldsymbol{C}_2} = \boldsymbol{y}_{\boldsymbol{C}_2}, \ldots, \boldsymbol{X}_{\boldsymbol{C}_k} = \boldsymbol{y}_{\boldsymbol{C}_k})$. We then write $\boldsymbol{C} = \boldsymbol{C}(\boldsymbol{j})$. To support the upcoming derivations, we first define the equivalence classes of index tuples and clique tuples.

Definition 5.5 (Sub-Alphabets) *Let $\mathcal{A}$ be some set of objects or symbols—$\mathcal{A}$ is an* alphabet*—and let $\mathcal{P}(\mathcal{A})$ be its power set. The set $\mathcal{P}(\mathcal{A}, n) \subseteq \mathcal{P}(\mathcal{A})$ contains all subsets of $\mathcal{A}$ with at most n elements, i.e.,*

$$\mathcal{P}(\mathcal{A}, n) = \{S \in \mathcal{P}(\mathcal{A}) \mid |S| \leq n\} \;.$$

The size of $\mathcal{P}(\mathcal{A}, n)$ is thus

$$|\mathcal{P}(\mathcal{A}, n)| = \sum_{i=1}^{n} \binom{|\mathcal{A}|}{i} \;.$$

Definition 5.6 (Tuple Classes) *Let $i \in \mathbb{N}$, and denote the clique tuple that corresponds to an index tuple $\boldsymbol{j} \in [d]^i$ by $\boldsymbol{C}(\boldsymbol{j}) \in \overline{\mathcal{C}}(G)^i$. Two or more index tuples $\boldsymbol{j}, \boldsymbol{j}'$ may correspond to the same clique tuple, i.e., $\boldsymbol{C}(\boldsymbol{j}) = \boldsymbol{C}(\boldsymbol{j}')$. The equivalence class of all index tuples that correspond to the same clique tuple is denoted by*

$$[\![\boldsymbol{j}]\!] = \{\boldsymbol{j}' \in [d]^i \mid \boldsymbol{C}(\boldsymbol{j}) = \boldsymbol{C}(\boldsymbol{j}')\} \,.$$

Similarly, two or more clique tuples $\boldsymbol{C}, \boldsymbol{C}'$ may correspond to the same set of cliques. The equivalence class of clique tuples that correspond to the same set of cliques is denoted by

$$[\![\boldsymbol{C}]\!] = \left\{ \boldsymbol{C}' \in \overline{\mathcal{C}}(G)^i \,\middle|\, \bigcup_{c \in \boldsymbol{C}} \{c\} = \bigcup_{c' \in \boldsymbol{C}} \{c'\} \right\} \,.$$

Combining both, the equivalence class of all index tuples, whose corresponding clique tuples are in the same equivalence class, is denoted by

$$[\![\boldsymbol{j}]\!]^* = \{\boldsymbol{j}' \in [d]^i \mid \boldsymbol{C}(\boldsymbol{j}') \in [\![\boldsymbol{C}(\boldsymbol{j})]\!]\} \,.$$

Note that all members of a specific clique tuple equivalence class $[\![\boldsymbol{C}]\!]$ are determined by a unique set of cliques which is used to construct the tuples. Hence, we identify each class $[\![\boldsymbol{C}]\!]$ with this unique set of cliques and treat each $[\![\boldsymbol{C}]\!]$ as a member $\mathcal{P}(\overline{\mathcal{C}}(G), i)$. Moreover, there are $|\mathcal{P}(\overline{\mathcal{C}}(G), i)|$ distinct clique tuple equivalence classes.

In the remainder, it will be important to know how large these equivalence classes are.

Lemma 5.4 (Counting Tuples) *Let $i, j \in \mathbb{N}$, $\boldsymbol{j} \in [d]^j$, $\boldsymbol{C} \in \overline{\mathcal{C}}(G)^i$, and consider the equivalence classes defined above. Then,*

$$|[\![\boldsymbol{j}]\!]| = \prod_{l=1}^{i} |\mathcal{X}_{\boldsymbol{C}(\boldsymbol{j})_l}|, \quad |[\![\boldsymbol{C}]\!]| = h(\boldsymbol{C})! \left\{ \begin{matrix} i \\ h(\boldsymbol{C}) \end{matrix} \right\}, \quad \textit{and} \quad |[\![\boldsymbol{j}]\!]^*| = |[\![\boldsymbol{C}(\boldsymbol{j})]\!]| |[\![\boldsymbol{j}]\!]|$$

where $h(\boldsymbol{C})$ is the number of distinct cliques which appear in the tuple $\boldsymbol{C}$, $n!$ is the factorial, and $\{n\ k\}^\top$ is the Stirling number of second kind.

Proof. Considering the semantic of index tuples (5.5), the number of index tuples that correspond to the same clique tuple is equal to the number of joint state assignments to all cliques in the tuple. The number of such assignments is the product of the state spaces $\prod_{l=1}^{i} |\mathcal{X}_{\boldsymbol{C}(\boldsymbol{j})_l}|$, which establishes the statement for $|[\![\boldsymbol{j}]\!]|$. Second, the number of permutations of l objects is $l! = l(l-1)(l-2)\ldots 2$. Here, we are interested in the number of ways how $h(\boldsymbol{C}) \leq i$ objects—$h(\boldsymbol{C})$ distinct cliques—can be distributed to i different places. For example, when we consider the cliques A and B and the clique tuple (A, A, B), its equivalence class $[\![(A, A, B)]\!]$ contains the tuples (A, A, B), (A, B, A), (A, B, B), (B, B, A), (B, A, B), and (B, A, A). Counting such multicombinations is a well known combinatorial enumeration problem, equivalent to the total number of surjective functions from $[i]$ to $[h(\boldsymbol{C})]$. The resulting number is $h(\boldsymbol{C})!\{i\ h(\boldsymbol{C})\}^\top$, where the factorial accounts for the number of permutations of distinct cliques. The second factor is the Stirling number of second kind $\{n\ k\}^\top$, which counts the number of ways to partition a set of n elements into k subsets. This establishes the second equality. Lastly, each clique tuple in the equivalence class $[\![\boldsymbol{C}]\!]$ corresponds to the same number of indices, namely $|[\![\boldsymbol{j}]\!]|$. Hence, the total number of indices covered by the equivalence class $|[\![\boldsymbol{j}]\!]^*|$ is the product of the sizes $|[\![\boldsymbol{C}]\!]|$ and $|[\![\boldsymbol{j}]\!]|$, which completes the proof. ■

It will also be helpful to define equivalence classes of index tuples w.r.t. some k-integrable sufficient statistics.

Definition 5.7 (Tuple Classes and k-integrability) *Let ϕ be a k-integrable sufficient statistic, $i \in \mathbb{N}$, and $\boldsymbol{j} \in [d]^i$. The equivalence class of all index tuples which correspond to the same clique tuple and have non-zero χ^i_ϕ-value is denoted by*

$$[\![\boldsymbol{j}]\!]_\phi = \{\boldsymbol{j}' \in [d]^i \mid \boldsymbol{j}' \in [\![\boldsymbol{j}]\!] \wedge \chi^i_\phi(\boldsymbol{j}') \neq 0\} \ .$$

The corresponding extension to equivalence classes of clique tuples, is denoted by

$$[\![\boldsymbol{j}]\!]^*_\phi = \{\boldsymbol{j}' \in [d]^i \mid \boldsymbol{j}' \in [\![\boldsymbol{j}]\!]^* \wedge \chi^i_\phi(\boldsymbol{j}') \neq 0\} \ .$$

Lemma 5.5 (Counting Realizable Tuples) *Suppose ϕ is the binary, overcomplete sufficient statistic. Then,*

$$|[\![\boldsymbol{j}]\!]_\phi| = |\mathcal{X}_{\boldsymbol{C}(\boldsymbol{j})}|, \quad \textit{and} \quad |[\![\boldsymbol{j}]\!]^*_\phi| = |[\![\boldsymbol{C}(\boldsymbol{j})]\!]||[\![\boldsymbol{j}]\!]_\phi| \ ,$$

with $\mathcal{X}_{\boldsymbol{C}(\boldsymbol{j})} = \mathcal{X}_{\cup_{l=1}^i \boldsymbol{C}(\boldsymbol{j})_l}$ and $\boldsymbol{C}(\boldsymbol{j}) \in \overline{\mathcal{C}}(G)^i$.

Proof. According to Lemma 5.2, we have $\chi^i_\phi(\boldsymbol{j}) = 0$ whenever an index tuple is not realizable. I.e., at least two indices in the tuple imply different assignments to the same variable (cf. proof of Lemma 5.2). For the clique tuple $\boldsymbol{C}(\boldsymbol{j})$, observe that $\mathcal{X}_{\boldsymbol{C}(\boldsymbol{j})}$ is the full joint state space of all unique variables in $\boldsymbol{C}$. If $\boldsymbol{C}$ contains cliques that share some vertices, those vertices must have the same state if and only if $\boldsymbol{j}$ is realizable. Thus, there cannot be more than $|\mathcal{X}_{\boldsymbol{C}(\boldsymbol{j})}|$ distinct realizable index tuples. The second equality is then a direct consequence of Lemma 5.4. ■

Now, we have gathered all terms and definitions to devise an improved procedure for the normalization of the index tuple density.

Theorem 5.4 (Tuple Density Normalization) *Suppose ϕ is the binary, overcomplete sufficient statistic. The conditional index tuple density $p_{\boldsymbol{\zeta},\phi}(\boldsymbol{J} = \boldsymbol{j} \mid I = i)$ (Definition 5.3) can be normalized in $\mathcal{O}(k^3)$ steps. More precisely,*

$$\|\chi^i_\phi\|_1 = |\mathcal{X}| \sum_{l=0}^{i} \left\{ \begin{matrix} i \\ l \end{matrix} \right\} \binom{|\overline{\mathcal{C}}(G)|}{l} l! \ . \tag{5.11}$$

Proof. The χ^i_ϕ value of non-realizable index tuples is 0, hence, it suffices to do the summation only over realizable tuples. Reviewing Lemma 5.2, we see that the value $\chi^i_\phi(\boldsymbol{j})$ of any realizable index tuple depends only on the associated clique tuple $\boldsymbol{C}(\boldsymbol{j})$. This implies that all tuples from the same equivalence class $[\![\boldsymbol{j}]\!]_\phi$ will contribute equally to the sum. We can thus rewrite the summation over all index tuples $[d]^i$ in terms of a summation over all index tuples $\overline{\mathcal{C}}(G)^i$ and multiply each summand by the size of the corresponding equivalence class:

$$\|\chi^i_\phi\|_1 = \sum_{\boldsymbol{j}\in[d]^i} |\chi^i_\phi(\boldsymbol{j})| = \sum_{\boldsymbol{C}\in\overline{\mathcal{C}}(G)^i} |[\![\boldsymbol{j}(\boldsymbol{C})]\!]_\phi|\chi^i_\phi(\boldsymbol{j}(\boldsymbol{C}))\ ,$$

where $\boldsymbol{j}(\boldsymbol{C})$ is an arbitrary index tuple associated with clique tuple $\boldsymbol{C}$.

Furthermore, note that χ^i_ϕ is permutation invariant, e.g., for any fixed indices $a, b, c, \dots$ and any permutation σ, we have $\chi^i_\phi(a, b, c, \dots) = \chi^i_\phi(\sigma(a, b, c, \dots))$. In other words, χ^i will yield the same function value for all $\boldsymbol{j}$ which correspond to clique tuples $\boldsymbol{C}$ that are in the same equivalence class $[\![\boldsymbol{C}]\!]$. We hence sum only over the elements of $\mathcal{P}(\overline{\mathcal{C}}(G), i)$ and multiply each term in the sum by the size of the corresponding equivalence class:

$$\|\chi^i_\phi\|_1 = \sum_{\boldsymbol{C}\in\overline{\mathcal{C}}(G)^i} |[\![\boldsymbol{j}(\boldsymbol{C})]\!]_\phi|\chi^i_\phi(\boldsymbol{j}(\boldsymbol{C})) = \sum_{[\![\boldsymbol{C}]\!]\in\mathcal{P}(\overline{\mathcal{C}}(G),i)} |[\![\boldsymbol{C}]\!]||[\![\boldsymbol{j}(\boldsymbol{C})]\!]_\phi|\chi^i_\phi([\![\boldsymbol{j}(\boldsymbol{C})]\!]_\phi)\ .$$

By Lemma 5.2, $\chi^i_\phi([\![\boldsymbol{j}(\boldsymbol{C})]\!]_\phi) = |\mathcal{X}|/|\mathcal{X}_{[\![\boldsymbol{C}]\!]}| = |\mathcal{X}|/|[\![\boldsymbol{j}(\boldsymbol{C})]\!]_\phi|$. Plugging this into the equation above and invoking Lemmas 5.4 and 5.5 to compute the sizes of equivalence classes, yields

$$\|\chi^i_\phi\|_1 = |\mathcal{X}| \sum_{[\![\boldsymbol{C}]\!]\in\mathcal{P}(\overline{\mathcal{C}}(G),i)} h(\boldsymbol{C})! \begin{Bmatrix} i \\ h(\boldsymbol{C}) \end{Bmatrix}\ .$$

Here, $\boldsymbol{C}$ is an arbitrary member of the equivalence class $[\![\boldsymbol{C}]\!]$, and $h(\boldsymbol{C})$ is the number of distinct cliques which appear in the tuple $\boldsymbol{C}$. We finally partition the summation over $\mathcal{P}(\overline{\mathcal{C}}(G), i)$, into i separate summations, each over those members of $\mathcal{P}(\overline{\mathcal{C}}(G), i)$ which have size $1 \le l \le i$. Hence, $h(\boldsymbol{C}) = l$ in each of these separate summations:

$$\|\chi^i_\phi\|_1 = |\mathcal{X}| \sum_{l=0}^{i} \sum_{[\![\boldsymbol{C}]\!]\in\mathcal{P}(\overline{\mathcal{C}}(G),i):h(\boldsymbol{C})=l} \begin{Bmatrix} i \\ l \end{Bmatrix} l!\ .$$

(5.11) follows from the fact that number of terms in each inner sum can be computed via binomial coefficients. Note that $\{0\ 0\}^\top = 1$, $0! = 1$, and that the empty set is contained in $\mathcal{P}(\overline{\mathcal{C}}(G), 0)$. The runtime reported in the theorem follows from the fact that each of the Stirling numbers that appear in the formula can be computed in $\mathcal{O}(i^2)$ steps, and indeed $i \le k$. ■

The complexity $\mathcal{O}(k^3)$ provided in the theorem is several orders of magnitude lower than that of the naive summation, i.e., $\mathcal{O}(d^k)$. Since we need the normalization for all k tuple lengths, τ can be computed in $\mathcal{O}(k^4)$ steps.

Figure 5.1: Ordering of classes and tuples. $\boldsymbol{C}^r$ denotes the r-th clique tuple, $[\![\boldsymbol{C}]\!]^l$ denotes the l-th equivalence class of clique tuples, and $[\![\boldsymbol{j}]\!]^{*l}$ denotes the equivalence class of index tuples which is induced by the l-th equivalence class of clique tuples. First row: Equivalence classes of clique tuples. Second row: Clique tuples. Third row: Index tuples.

$[\![\boldsymbol{C}]\!]^1$				$[\![\boldsymbol{C}]\!]^2$				$\dots$
$\boldsymbol{C}^1$		$\dots$	$\boldsymbol{C}^{\lvert[\![\boldsymbol{C}]\!]^1\rvert}$	$\boldsymbol{C}^{\lvert[\![\boldsymbol{C}]\!]^1\rvert+1}$		$\dots$	$\boldsymbol{C}^{\lvert[\![\boldsymbol{C}]\!]^1\rvert+\lvert[\![\boldsymbol{C}]\!]^2\rvert}$	$\dots$
$\boldsymbol{j}^1$	$\boldsymbol{j}^2$	$\dots$ $\boldsymbol{j}^{\lvert[\![\boldsymbol{j}]\!]^{*1}\rvert-1}$	$\boldsymbol{j}^{\lvert[\![\boldsymbol{j}]\!]^{*1}\rvert}$	$\boldsymbol{j}^{\lvert[\![\boldsymbol{j}]\!]^{*1}\rvert+1}$	$\boldsymbol{j}^{\lvert[\![\boldsymbol{j}]\!]^{*1}\rvert+2}$	$\dots$	$\boldsymbol{j}^{\lvert[\![\boldsymbol{j}]\!]^{*1}\rvert+\lvert[\![\boldsymbol{j}]\!]^{*2}\rvert}$	$\dots$

Even more important, τ (and each $\|\chi_\phi^i\|_1$) does not (explicitly) depend on $\boldsymbol{\theta}$. There is indeed an implicit dependence, since $\|\boldsymbol{\theta}\|_1$ influences the polynomial coefficients which appear in the computation of τ. However, if we compute the polynomial approximation for some parameter with norm $B = \|\boldsymbol{\theta}\|_1$, the same coefficients are valid for any parameter in the ball $\{\boldsymbol{\theta} \in \mathbb{R}^d \mid \|\boldsymbol{\theta}\|_1 \leq B\}$. In the context of resource-constrained systems, it is reasonable to assume that we precompute τ and the $\|\chi_\phi^i\|_1$ on an unconstrained system, and deploy the values on a resource-constrained device. The weak system may in turn use these values to run Algorithm 5.2 for any $\boldsymbol{\theta}$ in $\{\boldsymbol{\theta} \in \mathbb{R}^d \mid \|\boldsymbol{\theta}\|_1 \leq B\}$.

5.4.3 Index Tuple Sampling

Finally, we shall discuss the complexity of the SQM sampling step. Drawing a random tuple length according to $p_{\boldsymbol{\zeta}}(I = i)$ is easy, since the state space of I is small, e.g., $0 \leq I \leq k$, but sampling from the tuple density $p_{\boldsymbol{\zeta},\phi}(\boldsymbol{J} = \boldsymbol{j} \mid I = i)$ can be more involved (cf. Definition 5.3). We may indeed employ rejection sampling (Algorithm 2.3), since the dimensionality of the tuple space is rather low. However, rejection sampling has the unwanted side effect that several samples will be rejected and hence, resources will be wasted. We will now show how rejections can be avoided by introducing a specific ordering of index tuples $\boldsymbol{j}^1, \boldsymbol{j}^2, \dots, \boldsymbol{j}^{d^i}$.

To this end, let $\prec$ be an any arbitrary but fixed strict total ordering on the equivalence classes of clique tuples. I.e., $\forall \boldsymbol{A}, \boldsymbol{B} \in \mathcal{P}(\overline{\mathcal{C}}(G), i)$ with $\boldsymbol{A} \neq \boldsymbol{B}$, either $[\![\boldsymbol{A}]\!] \prec [\![\boldsymbol{B}]\!]$ or $[\![\boldsymbol{A}]\!] \prec [\![\boldsymbol{B}]\!]$—by definition, each element of $\mathcal{P}(\overline{\mathcal{C}}(G), i)$ corresponds to a unique equivalence class. This order induces an order on clique tuples and index tuples, i.e., $\boldsymbol{j}, \boldsymbol{j}' \in [d]^i$, $\boldsymbol{j} \leq \boldsymbol{j}' \Leftrightarrow [\![\boldsymbol{C}(\boldsymbol{j})]\!] \preceq [\![\boldsymbol{C}(\boldsymbol{j}')]\!]$. The situation is depicted in Fig. 5.1. Within each equivalence class, we assume that tuples are ordered lexicographically.

For any fixed i, inversion sampling of j then consists of drawing a uniform random number u in $(0;1)$, and finding the smallest $L \in \mathbb{N}$, such that

$$\sum_{l=1}^{L} p_{\boldsymbol{\zeta},\phi}(\boldsymbol{J} = \boldsymbol{j}^l \mid I = i) = \sum_{\boldsymbol{j} \in [d]^i : \boldsymbol{j} \preceq \boldsymbol{j}^L} p_{\boldsymbol{\zeta},\phi}(\boldsymbol{J} = \boldsymbol{j} \mid I = i) > u\ .$$

The worst-case complexity of the above summation is $\mathcal{O}(d^k)$, which can be prohibitively

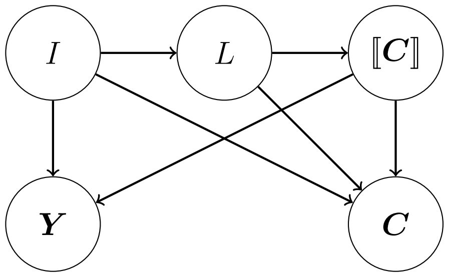

Figure 5.2: Directed graphical model for the factorization of the tuple density $p_{\boldsymbol{\zeta},\phi}(\boldsymbol{J} = \boldsymbol{j}, I = i)$. Any index tuple $\boldsymbol{j}$ can be identified with some pair $(\boldsymbol{C}, \boldsymbol{y})$ of clique tuple and state tuple.

expensive whenever the dimension d of the model is large. Based on the equivalence classes that we exploited already for the normalization of the tuple density, we derive a factorization of $p_{\boldsymbol{\zeta},\phi}(\boldsymbol{J} = \boldsymbol{j} \mid I = i)$ which in turn implies an efficient stagewise sampling procedure.

Theorem 5.5 (Tuple Density Factorization) *Suppose ϕ is the binary, overcomplete sufficient statistic. Suppose that $\boldsymbol{j}$ denotes the joint assignment $(\boldsymbol{C}_{\boldsymbol{j}_1} = \boldsymbol{y}_{\boldsymbol{j}_1}, \boldsymbol{C}_{\boldsymbol{j}_2} = \boldsymbol{y}_{\boldsymbol{j}_2}, \dots, \boldsymbol{C}_{\boldsymbol{j}_i} = \boldsymbol{y}_{\boldsymbol{j}_i})$. The tuple density factorizes:*

$$p_{\boldsymbol{\zeta},\phi}(\boldsymbol{J} = \boldsymbol{j} \mid I = i) = p(\boldsymbol{C} \mid [\![\boldsymbol{C}]\!], l, i) p(\boldsymbol{y} \mid [\![\boldsymbol{C}]\!], i) p([\![\boldsymbol{C}]\!] \mid l) p(l \mid i)$$

with

$$p(l \mid i) = \frac{\left\{ {i \atop l} \right\} \binom{|\overline{\mathcal{C}}(G)|}{l} l!}{\sum_{h=0}^{i} \left\{ {i \atop h} \right\} \binom{|\overline{\mathcal{C}}(G)|}{h} h!} \qquad p([\![\boldsymbol{C}]\!] \mid l) = \frac{1}{\binom{|\overline{\mathcal{C}}(G)|}{l}}$$

$$p(\boldsymbol{C} \mid [\![\boldsymbol{C}]\!], l, i) = \frac{1}{\left\{ {i \atop l} \right\} l!} \qquad p(\boldsymbol{y} \mid [\![\boldsymbol{C}]\!], i) = \begin{cases} \frac{1}{|\mathcal{X}_{[\![\boldsymbol{C}]\!]}|} & , \boldsymbol{y} \in \mathcal{X}_{[\![\boldsymbol{C}]\!]} \\ 0 & , \textit{ otherwise} \end{cases},$$

where l denotes the number of distinct cliques in the clique tuple $\boldsymbol{C}$, $[\![\boldsymbol{C}]\!]$ denotes the equivalence class that contains $\boldsymbol{C}$, and $\boldsymbol{y}$ is the joint state of all cliques in the tuple $\boldsymbol{C}$.

Proof. If $\boldsymbol{j}$ is not realizable, then $\boldsymbol{y} \notin \mathcal{X}_{[\![\boldsymbol{C}]\!]}$ and both $p_{\boldsymbol{\zeta},\phi}(\boldsymbol{J} = \boldsymbol{j} \mid I = i) = 0$ and the proposed factorization will assign mass 0. Now, assume that $\boldsymbol{j}$ is realizable and hence $\boldsymbol{y} \in \mathcal{X}_{[\![\boldsymbol{C}]\!]}$. By definition of the above quantities:

$$p(\boldsymbol{C} \mid [\![\boldsymbol{C}]\!], l, i) p(\boldsymbol{y} \mid [\![\boldsymbol{C}]\!], i) p([\![\boldsymbol{C}]\!] \mid l) p(l \mid i) = \frac{1}{|\mathcal{X}_{[\![\boldsymbol{C}]\!]}| \sum_{h=0}^{i} \left\{ {i \atop h} \right\} \binom{|\overline{\mathcal{C}}(G)|}{h} h!} \,. \tag{5.12}$$

Algorithm 5.3: Fast Index Tuple Sampler for Discrete State Models

```
input Tuple length $i$
output Sample $\boldsymbol{j} \mid I = i$ from $\mathbb{P}_{\boldsymbol{\zeta},\phi}$
1: $l \sim p(l \mid i)$ // See Theorem 5.5
2: $a \sim \mathbb{U}(1; \textbf{binom}(|\overline{\mathcal{C}}(G)|, l))$
3: $b \sim \mathbb{U}(1; \textbf{Stirling2}(i, l) \times \textbf{factorial}(l))$
4: $[\![\boldsymbol{C}]\!] \leftarrow$ compute $a$-th $l$-combination of $\{1, 2, \dots, |\overline{\mathcal{C}}(G)|\}$ // e.g., via [32]
5: $\boldsymbol{C} \leftarrow$ compute $b$-th composition of $\{1, 2, \dots, i\}$ with $l$ subsets // e.g., via [60]
6: $S \leftarrow \bigcup_{h=1}^{i} \boldsymbol{C}_h$
7: $c \sim \mathbb{U}(1; \prod_{v \in S} |\mathcal{X}_v|)$
8: $\boldsymbol{y} \leftarrow$ compute $c$-th joint state of all vertices in $S$
9: return $\boldsymbol{j}$ that corresponds to the joint assignment $\boldsymbol{C} = \boldsymbol{y}$
```

According to Lemmma 5.2, we have $\chi^i_\phi(\boldsymbol{j}) = |\mathcal{X}|/|\mathcal{X}_{\bigcup_{i=1}^k C_{j_i}}|$ for any realizable $\boldsymbol{j}$. The denominator will be the same for all clique tuples $\boldsymbol{C}$ from the same equivalence class $[\![\boldsymbol{C}]\!]$, since those share the same variables. Hence, $\chi^i_\phi(\boldsymbol{j}) = |\mathcal{X}|/|\mathcal{X}_{[\![\boldsymbol{C}]\!]}|$. Multiplication of (5.12) by $1 = |\mathcal{X}|/|\mathcal{X}|$ hence yields

$$p(\boldsymbol{C} \mid [\![\boldsymbol{C}]\!], l, i) p(\boldsymbol{y} \mid [\![\boldsymbol{C}]\!], i) p([\![\boldsymbol{C}]\!] \mid l) p(l \mid i) = \frac{\chi^i_\phi(\boldsymbol{j})}{|\mathcal{X}| \sum_{h=0}^{i} \left\{ \begin{matrix} i \\ h \end{matrix} \right\} \binom{|\overline{\mathcal{C}}(G)|}{h} h!} . \tag{5.13}$$

According to Definition 5.3, $p_{\boldsymbol{\zeta},\phi}(\boldsymbol{J} = \boldsymbol{j} \mid I = i) = \chi^i_\phi(\boldsymbol{j})/\|\chi^i_\phi\|_1$ for any realizable $\boldsymbol{j}$. Thus, the theorem follows by plugging the result of Theorem 5.4 into (5.13). ■

While the proof is rather simple, it is not obvious how to come up with this factorization. The key lies in the strict ordering, shown in Fig. 5.1. The idea is to first draw the equivalence class $[\![\boldsymbol{C}]\!]$, then a uniform member $\boldsymbol{C}$ of this class, then a uniform (realizable) joint state $\boldsymbol{y}$ of all cliques in $\boldsymbol{C}$. Notice that the sampling steps for $[\![\boldsymbol{C}]\!]$, $\boldsymbol{C}$ and $\boldsymbol{y}$ are uniform, while the probability mass of the number l of distinct cliques that will appear in the tuple is a function of l. In any case, the normalization is computed by the number of corresponding combinatorial objects.

The resulting factorization is shown in Fig. 5.2, and the implied sampling algorithm is shown in Algorithm 5.3. We will now investigate its complexity.

Theorem 5.6 (Complexity of Tuple Sampling) *Algorithm 5.3 samples an index tuple $\boldsymbol{j}$ of given length l from $\mathbb{P}_{\boldsymbol{\zeta},\phi}$ in*

$$\mathcal{O}(k^4 + (k+1)n + \{k\ l\}^\top + l! + |\overline{\mathcal{C}}(G)|)$$

steps. When permutations and partitions are precomputed, the runtime reduces to

$$\mathcal{O}(k^4 + (k+1)n + |\overline{\mathcal{C}}(G)|)$$

per sample. Here, k is the polynomial degree, l is the number of distinct cliques in the generated tuple, and $n = |V|$.

Proof. We analyze the complexity of each step separately and assume that uniform random numbers can be drawn in $\mathcal{O}(1)$. To draw a sample according to $p(l \mid i)$, we employ inversion sampling. Computing each probability requires to evaluate the Stirling number of second kind (complexity $\mathcal{O}(l^2)$), the binomial coefficient (complexity $\mathcal{O}(l)$), and the factorial (also $\mathcal{O}(l)$). These probabilities have to be computed for each $1 \leq l \leq i$. Since i is at most k, the total runtime for the probability computation is $\mathcal{O}(k^3)$. Drawing an actual inversion sample from $\{1, 2, \ldots, i\}$ hence requires $\mathcal{O}(k^4)$ steps. Whenever multiple samples must be drawn, we can reuse the probabilities. Thus, any subsequent sample requires only $\mathcal{O}(k)$ steps. Moreover, the terms that appear in the uniform sampling steps in lines 2 and 3 have already be computed for the first step. Consequently, their complexity is $\mathcal{O}(1)$.

We explained earlier that each clique equivalence class corresponds to a subset of l cliques. The algorithm presented in [32] is used to directly generate the a-th clique combination (in lexicographic order). The runtime of this algorithm is equal to the largest element in the combination, and the complexity of line 4 is hence $\mathcal{O}(|\mathcal{C}(G)|)$.

In contrast, we are not aware of an algorithm to generate the b-th composition of $\{1, 2, \ldots, i\}$ with l subsets directly. Instead, we generate such compositions by first computing an unordered partition of i elements into exactly l blocks, followed by a particular l-permutation.

As an example, let $[\![\boldsymbol{C}]\!] = \{A, B, C\}$ and $i = 5$ (the tuple length). Hence, $l = 3$ cliques must be assigned to $i = 5$ places. After determining the tuple, say $(1, 2, 2, 3, 1)$, a permutation of (A, B, C) must be determined to find the actual clique tuple. in case of the identity permutation, the resulting tuple would be (A, B, B, C, A). However, another permutation would lead to (C, A, A, B, C)—there are $3! = 6$ of such permutation in total.

We employ the algorithm from [60] (Section 5.2.2) to generate all unordered partitions of i elements into exactly l blocks. There are $\{i\ l\}^\top$ such partitions and the algorithm from [60] requires $\mathcal{O}(1)$ steps to generate the successor of any partition. Another algorithm from [60] can be used to generate all l-permutations in $\mathcal{O}(l!)$ steps. However, there are algorithms which do not require to store all permutations but generate the q-th permutation (for any q) directly—such procedures require $\mathcal{O}(l^2)$ steps. In total, it requires $\mathcal{O}(\{i\ l\}^\top + l!)$ steps (and memory) to precompute all partitions and permutations—any subsequent execution of line 5 (for the same combination of i and l) will take $\mathcal{O}(1)$ steps. Since clique tuples $\boldsymbol{C}$ may involve all $|V| = n$ vertices, the complexity of lines 6 and 7 are $\mathcal{O}(kn)$ and $\mathcal{O}(n)$, respectively. The joint state $\boldsymbol{y}$, computed in line 8, is the c-th element of the product space $\bigotimes_{v \in S} \mathcal{X}_v$. Converting c into the particular state $\boldsymbol{y}$ is done by a series of $|S|$ subtractions and divisions. As explained above, S may contain all vertices, and line 8 can thus be computed in $\mathcal{O}(n)$ steps. By employing an array of offsets to access the first parameter index of any clique, the conversion of the pair $(\boldsymbol{C}, \boldsymbol{y})$ to the index tuple $\boldsymbol{j}$ is done in $\mathcal{O}(k)$ steps. Combining these insights yields the statement of the theorem. ■

Thus, we found a Monte Carlo algorithm to sample from $\mathbb{P}_{\boldsymbol{\zeta},\phi}$ without any rejection step. Since the algorithm does not use any Markov chain, the generated samples are truly independent. Any number of samples can thus be generated in parallel. Because no data has to be exchanged, the overall runtime scales linearly with the number of processors.

This is a superior property compared to MCMC methods, in which sampling cannot be parallelized and consecutive samples are not independent.

Moreover, the theorem tells us how the complexity of stochastic quadrature is related to the graphical structure and the polynomial degree. The runtime is independent of the parameter dimension d and the state space sizes. In contrast, the runtime of loopy belief propagation and similar variational techniques is at least quadratic in the vertex state space sizes (cf. Table 5.1), which renders models with large vertex state spaces intractable. SQM can thus be a valuable alternative to those inference methods, even on not strongly constrained systems.

5.5 Parameter Learning and SQM

Now, after the fundamental theoretical properties and algorithmic concepts of SQM have been clarified, we discuss its application to actual probabilistic models. This includes the estimation of marginal densities and the log-likelihood function of exponential family members.

5.5.1 Marginal Inference

To facilitate SQM-based parameter estimation via numerical first-order methods, we have to compute the gradient of the log-partition function, which corresponds to the expected sufficient statistic $\nabla \log Z(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})]$. For discrete $\boldsymbol{X}$, $\mathbb{E}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})_i] = \mathbb{E}_{\boldsymbol{\theta}}[\phi(\boldsymbol{X})_{C=\boldsymbol{x}}] = p_{\boldsymbol{\theta}}(\boldsymbol{X}_C = \boldsymbol{x})$, for some $C \in \overline{\mathcal{C}}(G)$ and some $\boldsymbol{x} \in \mathcal{X}_C$. Recall that for any subset $U \subseteq V$ of variables, and any joint state $\boldsymbol{x}_U$, the marginal density is

$$p_{\boldsymbol{\theta}}(\boldsymbol{X}_U = \boldsymbol{x}_U) = \sum_{\boldsymbol{x}_{V\setminus U} \in \mathcal{X}_{V\setminus U}} p_{\boldsymbol{\theta}}(\boldsymbol{x}_U, \boldsymbol{x}_{V\setminus U}) = \frac{1}{Z(\boldsymbol{\theta})} \sum_{\boldsymbol{x}_{V\setminus U} \in \mathcal{X}_{V\setminus U}} \psi(\boldsymbol{x}_U, \boldsymbol{x}_{V\setminus U}) \,.$$

Especially the sum on the right hand side of the equation is reminiscent of the partition function. In fact, it can be interpreted as the partition function of another model with state space $\mathcal{X}_{V\setminus U}$. It is this sum that will be approximated via SQM to estimate the marginal. To formalize this idea, we provide adjusted definitions of the SQM core concepts. First, we adapt the notion of k-integrability to marginal densities.

Definition 5.8 (Marginal k-integrability) *Let $G = (V, E)$ be the conditional independence structure of a random variable $\boldsymbol{X}$. The sufficient statistic $\phi : \mathcal{X} \to \mathbb{R}^d$ is called marginally k-integrable, if the function $\chi^k_\phi : [d]^k \to \mathbb{R}$, with*

$$\chi^k_{\phi,U}(\boldsymbol{j}, \boldsymbol{x}_U) = \int_{\mathcal{X}_{V\setminus U}} \prod_{i=1}^{k} \phi(\boldsymbol{x}_U, \boldsymbol{x}_{V\setminus U})_{\boldsymbol{j}_i} \,\mathrm{d}\,\nu(\boldsymbol{x}_{V\setminus U})$$

admits a polynomial time computable closed-form expression for all $\boldsymbol{j} \in [d]^k$, for all $U \subseteq V$, and for all $\boldsymbol{x}_U \in \mathcal{X}_{V\setminus U}$.

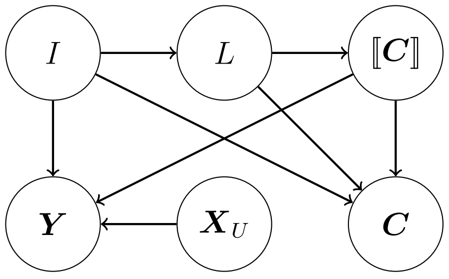

Figure 5.3: Directed graphical model for the factorization of the marginal tuple density $p_{\boldsymbol{\zeta},\phi}(\boldsymbol{J} = \boldsymbol{j}, \boldsymbol{X}_U = \boldsymbol{x}_U, I = i)$. Any index tuple $\boldsymbol{j}$ can be identified with some pair $(\boldsymbol{C}, \boldsymbol{y})$ of clique tuple and state tuple.

The general result on k-integrability for discrete models carries indeed over to the marginal case.

Corollary 5.1 (Marginal k-integrability of Overcomplete Statistics) *The binary, overcomplete sufficient statistic is marginally k-integrable for all $k \in \mathbb{N}$.*

Proof. There are two noteworthy deviations from the proof of Lemma 5.2. First, in addition to the states of the variables in $\cup_{l=1}^{i} \boldsymbol{C}_l$, the states of the variables in U are fixed. This reduces the total number of summands from $|\mathcal{X}|$ to $|\mathcal{X}_{V\setminus U}|$. To simplify notation, we define $|\mathcal{X}_\emptyset| = 1$. Moreover, in this reduced state space, the formula

$$\varphi_{\boldsymbol{j}}(\boldsymbol{x}_U, \boldsymbol{x}_{V\setminus U}) = (\boldsymbol{x}_{C_1} = \boldsymbol{y}_{C_1} \wedge \boldsymbol{x}_{C_2} = \boldsymbol{y}_{C_2} \wedge \cdots \wedge \boldsymbol{x}_{C_k} = \boldsymbol{y}_{C_k}) \equiv \prod_{i=1}^{k} \phi(\boldsymbol{x})_{\boldsymbol{j}_i} \tag{5.14}$$

might not be specifiable, even when $\boldsymbol{j}$ itself is realizable. Keeping this in mind, the same reasoning that we used to prove Lemma 5.2 leads to the closed form

$$\chi^{k}_{\phi,U}(\boldsymbol{j}, \boldsymbol{x}_U) = \begin{cases} \frac{|\mathcal{X}|}{|\mathcal{X}_{(\cup_{i=1}^{k} C_{\boldsymbol{j}_i})\setminus U}||\mathcal{X}_U|} & , \boldsymbol{j} \text{ is realizable} \wedge \ \nexists v \in U : \boldsymbol{x}_v \neq \boldsymbol{y}_v \\ 0 & , \text{ otherwise ,} \end{cases}$$

where, $\boldsymbol{y}$ is the state implied by $\boldsymbol{j}$ (5.14), and we applied $\left|\mathcal{X}_{V\setminus U}\right| = \prod_{v\in V\setminus U} |\mathcal{X}_v| = |\mathcal{X}|/\prod_{v\in U} |\mathcal{X}_v| = |\mathcal{X}|/|\mathcal{X}_U|$. ■

Based on marginal k-integrability, and Theorem 5.5, we present the factorization of the corresponding marginal tuple density as shown in Fig. 5.3.

Corollary 5.2 (Marginal Tuple Density Factorization) *Suppose ϕ is the binary, overcomplete sufficient statistic (Definition 2.6). Suppose that $\boldsymbol{j}$ denotes the joint assignment $(\boldsymbol{C}_{\boldsymbol{j}_1} = \boldsymbol{y}_{\boldsymbol{j}_1}, \boldsymbol{C}_{\boldsymbol{j}_2} = \boldsymbol{y}_{\boldsymbol{j}_2}, \ldots, \boldsymbol{C}_{\boldsymbol{j}_i} = \boldsymbol{y}_{\boldsymbol{j}_i})$. The marginal tuple density factorizes:*

$$p_{\boldsymbol{\zeta},\phi}(\boldsymbol{J} = \boldsymbol{j}, I = i, \boldsymbol{X}_U = \boldsymbol{x}_U) = p(\boldsymbol{C} \mid [\![\boldsymbol{C}]\!], l, i) p(\boldsymbol{y}, \boldsymbol{x}_U \mid [\![\boldsymbol{C}]\!], i) p([\![\boldsymbol{C}]\!] \mid l) p(l \mid i) p(i)$$

where $p(l \mid i)$*,* $p(\llbracket \boldsymbol{C} \rrbracket \mid l)$*, and* $p(\boldsymbol{C} \mid \llbracket \boldsymbol{C} \rrbracket, l, i)$ *are given by Theorem 5.5, and*

$$p(\boldsymbol{y}, \boldsymbol{x}_U \mid \llbracket \boldsymbol{C} \rrbracket, i) = \begin{cases} \frac{1}{|\mathcal{X}_{\llbracket \boldsymbol{C} \rrbracket \cup U}|} & , \boldsymbol{y} \in \mathcal{X}_{\llbracket \boldsymbol{C} \rrbracket} \wedge \nexists v \in U : \boldsymbol{x}_v \neq \boldsymbol{y}_v \\ 0 & , \textit{otherwise} . \end{cases}$$

The ordinary tuple density $p_{\boldsymbol{\zeta},\phi}(\boldsymbol{J} = \boldsymbol{j}, I = i)$ and the marginal tuple density $p_{\boldsymbol{\zeta},\phi}(\boldsymbol{J} = \boldsymbol{j}, \boldsymbol{X}_U = \boldsymbol{x}_U, I = i)$ differ only in the factor $p(\boldsymbol{y}, \boldsymbol{x}_U \mid \llbracket \boldsymbol{C} \rrbracket, i)$. We may hence use their quotient as importance weight to convert SQM samples for the partition function into SQM samples for marginal probabilities. We have

$$\begin{aligned} &\mathbb{E}_{\boldsymbol{J},I}\left[\frac{p(\boldsymbol{y}(\boldsymbol{J}), \boldsymbol{x}_U \mid \llbracket \boldsymbol{C}(\boldsymbol{J}) \rrbracket, I)}{p(\boldsymbol{y}(\boldsymbol{J}) \mid \llbracket \boldsymbol{C}(\boldsymbol{J}) \rrbracket, I)} \hat{Z}^k_{\boldsymbol{J},I}(\boldsymbol{\theta})\right] \\ =&\sum_{i=0}^{k} \sum_{\boldsymbol{j} \in [d]^i} p_{\boldsymbol{\zeta},\phi}(\boldsymbol{J} = \boldsymbol{j}, I = i) \underbrace{\frac{p(\boldsymbol{y}(\boldsymbol{j}), \boldsymbol{x}_U \mid \llbracket \boldsymbol{C}(\boldsymbol{j}) \rrbracket, i)}{p(\boldsymbol{y}(\boldsymbol{j}) \mid \llbracket \boldsymbol{C}(\boldsymbol{j}) \rrbracket, i)}}_{w_{\boldsymbol{j},i,U}} \hat{Z}^k_{\boldsymbol{j},i}(\boldsymbol{\theta}) \\ =&\sum_{i=0}^{k} \sum_{\boldsymbol{j} \in [d]^i} p_{\boldsymbol{\zeta},\phi}(\boldsymbol{J} = \boldsymbol{j}, I = i, \boldsymbol{X}_U = \boldsymbol{x}_U) \hat{Z}^k_{\boldsymbol{j},i}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{J},I,\boldsymbol{X}_U = \boldsymbol{x}_U}\left[\hat{Z}^k_{\boldsymbol{J},I}(\boldsymbol{\theta})\right] , \end{aligned}$$

with importance weight $w_{\boldsymbol{j},i,U}$.

Now, we gathered all parts which are required for marginal inference. The corresponding inference procedure is provided in Algorithm 5.4. While the main idea is to perform d separate runs of Algorithm 5.2, such a naive approach would result in an unnecessary high runtime. Instead, we make use of Corollary 5.2 to propose an importance sampling approach, in which each SQM sample is shared among all marginals. For each marginal $p(\boldsymbol{X}_C = \boldsymbol{x}_C)$, we validate if the pair $(\boldsymbol{j}, i)$ that is sampled in line 3 agrees with the assignment $\boldsymbol{x}_C$ (line 6)—otherwise, its marginal tuple density is zero. If they agree, we reweigh the sample, perform the summation and count the number of successes in lines 7 and 8. In lines 13–17, the estimated sums are normalized and written to $\hat{\boldsymbol{\mu}}$.

5.5.2 Parameter Estimation

Based on Algorithm 5.4, we can readily compute the likelihood's gradient $\nabla \ell(\boldsymbol{\theta}; \mathcal{D})$, and employ any first-order method to estimate the parameters. The most essential component is hence available and we may proceed to conduct some experiments. To investigate the progress of learning, we can look at the gradient norm or the objective function. Due to its functional form, the negative log-likelihood,

$$\ell(\boldsymbol{\theta}; \mathcal{D}) = -\langle \boldsymbol{\theta}, \tilde{\boldsymbol{\mu}} \rangle + \log Z(\boldsymbol{\theta}) ,$$

inherits its complexity from the computation of the log-partition function $\log Z(\boldsymbol{\theta})$. We will hence close the theoretical part of this chapter, by translating the SQM error bound from Theorem 5.3 to an error bound on the negative log-likelihood.

Algorithm 5.4: SQM Marginal Inference with Shared Sampling

```
input $\boldsymbol{\theta}$, $k$, $\boldsymbol{\zeta}$, $N$
output Approximate marginals $\hat{\boldsymbol{\mu}}$
 1: $\boldsymbol{S} \leftarrow \mathbf{0}$, $\boldsymbol{m} \leftarrow \mathbf{0}$, done $\leftarrow$ false
 2: while $\exists i : \boldsymbol{m}_i < N$ do
 3:   $(\boldsymbol{j}, i) \sim \mathbb{P}_{\boldsymbol{\zeta},\phi,C}(\cdot, \boldsymbol{x}_C)$
 4:   for $C \in \overline{\mathcal{C}}(G)$ do
 5:     for $\boldsymbol{x}_C \in \mathcal{X}_C$ do
 6:       if agree$(\boldsymbol{j}, i, C, \boldsymbol{x}) \wedge \boldsymbol{m}_i < N$ then
 7:         $\boldsymbol{S}_l \leftarrow \boldsymbol{S}_l + \hat{Z}^k_{\boldsymbol{j},i}(\boldsymbol{\theta}) \frac{p(\boldsymbol{y}(\boldsymbol{j}), \boldsymbol{x}_C | [\![\boldsymbol{C}(\boldsymbol{j})]\!], i)}{p(\boldsymbol{y}(\boldsymbol{j}) | [\![\boldsymbol{C}(\boldsymbol{j})]\!], i)}$
 8:         $\boldsymbol{m}_i \leftarrow \boldsymbol{m}_i + 1$
 9:       end if
10:     end for
11:   end for
12: end while
13: for $C \in \overline{\mathcal{C}}(G)$ do
14:   for $\boldsymbol{x}_C \in \mathcal{X}_C$ do
15:     $\hat{\boldsymbol{\mu}}_{C=\boldsymbol{x}_C} \leftarrow \frac{\boldsymbol{S}_{C=\boldsymbol{x}_C}}{\sum_{\boldsymbol{x}_C \in \mathcal{X}_C} \boldsymbol{S}_{C=\boldsymbol{x}_C}}$
16:   end for
17: end for
```

Theorem 5.7 (SQM Error for $\log Z(\boldsymbol{\theta})$) *Assume that the preconditions of Theorem 5.3 hold. Whenever the outcome $\hat{Z}^{N,k}_{\boldsymbol{\zeta}}(\boldsymbol{\theta})$ of Algorithm 5.2 is positive, then*

$$\mathbb{P}[|\log \hat{Z}^{N,k}_{\boldsymbol{\zeta}}(\boldsymbol{\theta}) - \log Z(\boldsymbol{\theta})| < \epsilon Z(\boldsymbol{\theta}) / \min\{\hat{Z}^{N,k}_{\boldsymbol{\zeta}}(\boldsymbol{\theta}), Z(\boldsymbol{\theta})\}] \quad \geq \quad 1 - \delta \; .$$

Proof. Let $l = \min\{\hat{Z}^{N,k}_{\boldsymbol{\zeta}}(\boldsymbol{\theta}), Z(\boldsymbol{\theta})\}$ and $u = \max\{\hat{Z}^{N,k}_{\boldsymbol{\zeta}}(\boldsymbol{\theta}), Z(\boldsymbol{\theta})\}$. Applying the mean value theorem to the logarithm, there is $\xi \in [l, u]$ such that $\xi|\log l - \log u| = |l - u|$. By (5.7),

$$\mathbb{P}[|\hat{Z}^{N,k}_{\boldsymbol{\zeta}}(\boldsymbol{\theta}) - Z(\boldsymbol{\theta})| < \epsilon Z(\boldsymbol{\theta})] \geq 1 - \delta \; ,$$

and hence

$$\mathbb{P}[\xi|\log \hat{Z}^{N,k}_{\boldsymbol{\zeta}}(\boldsymbol{\theta}) - \log Z(\boldsymbol{\theta})| < \epsilon Z(\boldsymbol{\theta})] \geq 1 - \delta \; .$$

Dividing by ξ and using $\xi \geq l$ implies the desired result. ■

We will now conduct a series of experiments to demonstrate the various theoretical properties of SQM that have been discussed so far.

5.6 Experimental Demonstration

Fundamental principles of approximation theory allow us to construct a stochastic quadrature-based approximation to the partition function. Moreover, we apply the same technique to approximate the marginal densities of exponential family members.

In contrast to variational inference techniques, the quality of our proposed method is independent of the graphical structure—even in the presence of circular conditional independences. Theorem 5.3 shows that the error is influenced by three factors: the degree of the underlying polynomial approximation, the norm of the parameter vector, and the number of Monte Carlo samples. We will now conduct a series of experiments to demonstrate this behavior. More precisely, we are interested in answering the following questions empirically:

Q1 How does SQM respond to different parameter norms?

Q2 How does SQM respond to different polynomial degrees?

Q3 How does SQM respond to different sample sizes?

Q4 What are the resource requirements of SQM?

We know from Theorems 5.3 and 5.7 under which circumstances the error of SQM is bounded. However, the theorem provides no intuition about the specific growth behavior. We will hence evaluate the absolute approximation error of the log-partition function, e.g., $|\log(\hat{\mathrm{Z}}) - \log(\mathrm{Z})|$ where Z is the value of the exact partition function at $\boldsymbol{\theta}$ and $\hat{Z}$ is an approximation to Z. In addition, we investigate SQM-based parameter learning. In accordance with our experiments in the previous chapters, we evaluate the mean squared error of estimated marginals, i.e., $\|\hat{\boldsymbol{\mu}} - \tilde{\boldsymbol{\mu}}\|_2^2/d$, to measure the quality of the learned model. Both quantities are investigated in the course of answering **Q1**–**Q3**. For **Q4**, we consider a theoretical review of the memory requirement as well as various runtime measurements (in milliseconds).

5.6.1 Setup

Our baseline in this set of experiments is an ordinary Markov random field in which probabilistic inference is carried out by loopy belief propagation—corresponding to the Bethe approximation of the partition function. LBP has unbounded error on general loopy graphs, but is known to perform extra ordinary well in practice. This baseline is compared to our proposed Monte Carlo method N-SQM as provided in Algorithm 5.2. The error of N-SQM comprises the error of the underlying polynomial approximation as well as the sampling error. To understand their interplay, we consider an exact quadrature method in which the expectation $\mathbb{E}[\hat{Z}^k_{\boldsymbol{J},I}(\boldsymbol{\theta})]$ is evaluated exactly. Due to high complexity, this method is not reasonable in general, but it helps us to reveal the error introduced by the sampling step. In summary, we use the following models:

M1 Discrete state Markov random field with loopy belief propagation

M2 Discrete state Markov random field with exact quadrature

M3 Discrete state Markov random field with stochastic quadrature

Table 5.2: Parameter dimensions for models on synthetic data sets.

Dimension	**Chain**	**Star**	**Grid**	**Full**
$n = 16$	60	60	96	480
$n = 100$	396	396	720	19800

For **M2** and **M3** models, we consider various polynomial degrees in $\{2, 3, \ldots, 9\}$. The polynomial coefficients of SQM are estimated via the Remez exchange algorithm (Algorithm 2.5). Each experiment is performed on an Intel Xeon E5-2697 v2 system. We provide a docker image with our own C++ implementation of models **M1**–**M3** for download at `https://sfb876.tu-dortmund.de/px`.

Synthetic Data

Experiments on synthetic data help us to study the effects of the parameter norm on the approximation error. For simplicity, all synthesized models contain only pairwise factors ψ_{vu}. The number of variables is fixed to $n = 16$ when we estimate the partition function, and fixed to $n = 100$ when we estimate the marginal densities. In any case, the number of states per variable is 2.

Chain, star, grid, and fully connected graphs, shown in Fig. 3.8, serve as synthetic conditional independence structures. The theoretical insights that we gained in this chapter suggest, that the conditional independence structure has no direct impact on the approximation error of SQM. However, models with different graphs can have different dimensions, and the model dimension has an indirect impact on the parameter norm. For our data generating process, each parameter is drawn independently from a Gaussian distribution with mean 0 and variance $\sigma^2 \in \{10^{-6}, 10^{-5}, \ldots, 10^{0}, 10^{1}\}$, to simulate various parameter norms. The relation between parameter variance σ^2, model dimension d, and norm $\|\boldsymbol{\theta}\|_2$ is established via the following equation.

$$\mathbb{E}\left[\|\boldsymbol{\theta}\|_2^2\right] = \mathbb{E}\left[\sum_{i=1}^{d} \boldsymbol{\theta}_i^2\right] = d\sigma^2$$

It hence suffices to investigate how the parameter variance affects the approximation quality. Moreover, we provide the dimensions of our synthetic models in Table 5.2.

Eventually, 1000 samples are generated from each model via Gibbs sampling (Algorithm 2.4). During data generation, the first 100 samples are discarded. Between consecutive samples, all variables are resampled 16 times in a round-robin fashion to enforce independence of consecutive samples.

Table 5.3: Empirical upper bounds on the variances of each experiment on synthetic data.

Name	$\tilde{\mathbb{V}}^*$[MSE]	$\tilde{\mathbb{V}}^*$[seconds/iter]
Chain	0.000565661040653599	3281.22
Star	0.0005950736234665	1607.77
Grid	0.0006882348319504	23.86
Full	0.0011911645839424	13721.40

Real-world Data

In the second set of experiments, we estimate the model parameters on the intel data set that we already know from the previous chapters. Experiments on the other two real-world data sets were omitted, in favor of more synthetic experiments.

D1 Intel Lab—Temperature and Humidity Data[26]

As in Chapter 4, we do not use the temporal information and generate twelve training instances per day. Information on the data set can be found in Table 4.2. We choose the optimal tree structure as conditional independence structure, computed via the Chow-Liu algorithm (cf. Section 2.3.4).

5.6.2 Results

Results on the synthetic data sets are presented in Figures 5.5–5.8, and results on the real-world data sets are shown in Figures 5.9 and 5.10. The black solid circles represent results for model **M1**, while the other symbols and colors correspond to results for models **M2** and **M3**—the common key of all plots is shown in Fig. 5.4. In total, the results represent 9140 experiments. The error bars of the learning results (Figures 5.7–5.10) are ommited to enhance the readability. Instead, we provide the empirical worst-case variances in Table 5.3. While the MSE variances are reasonably low, the runtime variances are quite high, whereby the highest variance corresponds to a standard deviation of 117 seconds per training iteration. Due to the Monte Carlo nature of SQM, its runtime is deterministic. But recall that Algorithm 5.4 uses shared samples to estimate all marginals simultaneously. Since not every sample is valid for every marginal, the algorithm has to wait until N valid samples have been observed for each marginal. The runtime of SQM marginal inference is hence a random variable which depends on the number of marginals, i.e., the model dimension.

We now explain and discuss the results w.r.t. to the questions stated at the beginning of this Section.

[26]`http://db.csail.mit.edu/labdata/labdata.html`

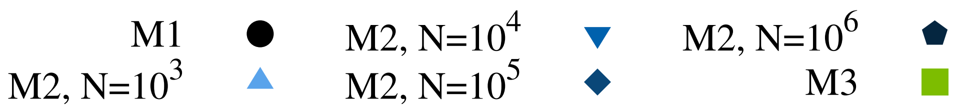

Figure 5.4: Key for Figures 5.5–5.10 to indicate the inference algorithm and the number of SQM samples.

Q1 How does SQM respond to different parameter norms?

It is well known that the conditional independence structure has a strong impact on the complexity of exact inference, and on the quality of approximate inference techniques like belief propagation or naive mean field. It is thus an important insight from Theorem 5.3 that the approximation error of SQM does not depend on the specific structure. Instead, the error depends on the norm of the parameter vector.

The first set of results can be found in Figures 5.5 and 5.6. On all four synthetic structures, the average error increases with increasing standard deviation. Up to $\sigma = 1$, different choices of N and k result in different approximation errors. However, for $\sigma = 10$, the error is almost constant w.r.t. degree and sample number. Interestingly, Figures 5.5 and 5.6 reveal that the error introduced by the our sampling procedure is relatively larger when the parameter norm is small. Quadrature-based inference in which the expectation is computed exactly (rightmost/green bar in each group) achieves zero error when σ is small. In case of larger standard deviations, the approximation error is mostly determined by the polynomial approximation and not by the sampling procedure. In all cases, the error of loopy belief propagation was below 10^{-2} so that the corresponding bars cannot be seen in the figures.

Similar observations can be made when we measure the error in terms of the mean squared error of estimated marginal probabilities (Fig. 5.7). For $\sigma < 10^{-2}$, the error is small and exhibits super-linear growth, starting from $\sigma = 10^{-2}$. With $\sigma = 10^0$, the growth rate decreases. The results of the corresponding experiment with real-world data are shown in Fig. 5.9. Therein, we cannot control the parameter variance and instead employ l_1-regularization to control the parameter norm. For small λ values, some SQM instances can achieve a small error. Stronger regularization implies smaller parameter norms, which eases the SQM inference. On the other hand, strong regularization downweights the likelihood. The MSE hence increases, which overrules the effect of the reduced SQM error.

To sum up, the error increases sublinearly whenever we increase the parameter norm, e.g., a ten times higher σ results in an additive error.

Q2 How does SQM respond to different polynomial degrees?

The error of uniform polynomial approximations depends on the width of the approximated domain. In our case, this width is determined by the parameter norm. For any fixed width, Theorem 2.8 tells us that increasing the polynomial degree should reduce the approximation error.

First, we consider the approximation of the log-partition function (Figures 5.5 and 5.6). Results for the deterministic quadrature[27] (rightmost/green bar in each grouped bar plot) show, that in case of small standard deviations ($\sigma \leq 0.1$), the error is almost zero for all polynomial degrees larger than 1. For $\sigma = 1$, the error decreases whenever the polynomial degree is increased. If the parameter norm is large ($\sigma = 10$), increasing the degree has almost no effect on the error. The picture is similar when we consider the stochastic quadrature. However, increasing the degree can lead to a strongly increased error whenever the number of samples is too small. This behavior can be observed across all graphical structures. Results for $\sigma = 0.1$ on the fully connected structure show, that the error can shrink to nearly zero when we increase the degree sufficiently. The sampling procedure itself approximates the exact quadrature very well. Especially for larger norms, the additional error that is introduced by the sampling procedure is negligible, compared to the error of the polynomial approximation.

When we consider the MSE instead (last two rows of Fig. 5.7), the error strictly increases when we increase the polynomial degree while keeping all other quantities fixed. This behavior is predicted by Theorem 5.3, which tells us that the number of samples has to be chosen w.r.t. to the degree. Looking at the results for the 10×10 grid structure in the bottom left plot of Fig. 5.7, we see that for $N = 1000$ samples, a polynomial of degree $k = 4$ has a larger error than a polynomial of degree $k = 2$. However, increasing the sample size to $N \geq 10000$, the degree 4 polynomial achieves a lower error than the degree two polynomial with 1000 samples. In this particular set of experiments, degree $k = 2$ with $N = 100000$ samples always achieved the lowest MSE. The same observations can me made in the corresponding experiment on real-world (Fig. 5.10).

In summary, our results suggest that small polynomial degrees suffice for a practical approximation, as long as the sample size is large enough. A linearly decreased error can be observed when we increase the polynomial degree and the number of samples accordingly.

Q3 How does SQM respond to different sample sizes?

We conducted experiments with four sample sizes. With the exception of a few outliers, increasing the sample size leads to a smaller approximation error. In Figures 5.5 and 5.6, we see that $N = 1000$ samples are not enough when the polynomial approximation is cubic or higher and the parameter norm is rather small (≤ 0.1). Even in the quadratic case, the approximation error of SQM with $N = 1000$ samples stands out. When the parameter norm is large ($\sigma \geq 1$), the error of the polynomial approximation dominates and the impact of the number of samples is negligible. This observation is independent of the underlying conditional independence structure.

The parameter learning experiments on synthetic data (Fig. 5.7) show, that increasing the number of samples leads to better marginal estimates. But increasing the samples beyond a certain point does not incur much gain. Marginal approximations with $N = 10^5$

[27]Only available for chain, star and grid structures.

and $N = 10^6$ samples are often close to each other, while smaller sample sizes result in substantially higher mean squared errors. This relation is confirmed by results on real-world data (Figures 5.9 and 5.10).

All experiments show, that the improvement from taking more samples is sublinear. That is, taking ten times more samples improves the quality by a small additive constant.

Q4 What are the resource requirements of SQM?

It can be seen from Algorithms 5.2 and 5.4, that the memory requirement is linear in the number of approximated quantities. E.g., if only the partition function is approximated, we need memory space for the estimated partition function and temporary storage for the SQM samples. When the sampling is performed sequentially, only one sample needs to be stored at a time. The memory requirement is hence lower than that of loopy belief propagation, which requires to store $2|E||\mathcal{X}_v|$ messages (where $\mathcal{X}_v$ is the vertex state space).

The runtime of SQM is indeed higher than that of belief propagation. This can be considered as the price for the theoretical error guarantees of SQM. Corresponding results can be found in Figures 5.8, 5.9, and 5.10. Clearly, the SQM runtime scales almost perfectly linear with the number of samples—taking ten times more samples increases the runtime by a factor of 10. On the other hand, the SQM algorithm can be trivially parallelized due to the independence of Monte Carlo samples—taking ten times more CPUs reduces the runtime by a factor of 10. The parameter norm has only an indirect impact on the runtime, in that larger norms require a larger polynomial degrees, which in turn increase the time to generate random index tuples. Keeping all other quantities constant, increasing the parameter norm has no effect on the runtime. Moreover, the computational complexity of many subroutines, like the estimation of polynomial coefficients, sampling a single index tuple, and checking the realizability of an index tuple, depend on the polynomial degree. Figure 5.8 shows, that the runtime of SQM is a linear function of the degree—doubling the degree almost doubles the runtime.

Especially the results on real-world data show, that in case of relatively small polynomial degrees and sample sizes, the runtime of SQM is reasonably low. However, guaranteeing a small error by using large polynomial degrees and large sample sizes indeed implies a large runtime.

5.7 Discussion

Resource constraint systems can have highly limited computational capabilities, like low clock rates, and no hardware accelerators of any kind. But state-of-the-art machine learning relies on the availability of high clock rates, high parallelism, and vector arithmetic accelerators. To overcome this discrepancy, we need approximation techniques that (1) exhibit moderate hardware demands while (2) guaranteeing a reasonable quality. While the first point can be satisfied be various techniques, the second point is often neglected due to the heuristic nature of many existing methods. Even if a method is derived from theoretically sound principles, the approximation error is often unbounded. In case of undirected models, the most demanding step of parameter learning is the probabilistic inference—the central research object of this chapter.

We started by reviewing existing inference techniques and their complexity. Most approximate inference techniques work by approximating the conditional independence structure. As a result, the approximation error is hard to control if at all possible. Consequently, it is advantageous to keep the structure intact and find other avenues to approximate inference. Due to the specific form of the partition function, it is amenable to numerical integration. We reviewed the WISH algorithm as approximate inference technique with error guarantees. In fact, WISH delivers an (ϵ, δ)-approximation, based on left Riemann sums and pairwise independent hash functions. The method keeps the conditional independence structure intact and provides error bounds, but requires multiple queries to an **NP**-oracle, and is hence not suitable for highly restricted systems. However, the technique motivated the investigation of other approximate integration methods.

We hence investigated the class of quadrature techniques. The underlying idea is to approximate an intractable integral by replacing the integrand by an appropriate polynomial approximation. Based on known error bounds for polynomial approximations, we showed that the quadrature technique could in principle be applied to estimate the partition function of exponential families. Nevertheless, it was not clear that this procedure would yield any computational benefit. To this end, we introduced the notion of integrable sufficient statistics. It was shown that the statistics of discrete and continuous state space models satisfy this property. Based on these insights, we introduced the stochastic quadrature method which is a stochastic reinterpretation of a quadrature approximation to the partition function. We proved that the corresponding random quantity is an unbiased estimator to the deterministic quadrature. Moreover, we provided a probabilistic upper bound on the approximation error w.r.t. the true partition function value. We identified the norm of the parameter vector, the degree of the underlying polynomial approximation, and the number of samples as crucial factors for the approximation error. The upper bound was later extended into an upper bound on the approximation error of the log-partition function.

After establishing these intriguing theoretical properties, we derived a Monte Carlo algorithm that actually outputs the SQM approximation. In the corresponding derivation, we made heavy use of equivalence classes of state spaces, and counted their total number via combinatorial enumerative problems. Based on these insights, we discovered a factorization of the density of the underlying random variable as well as a fast sampling algorithm. Finally, we extended our SQM approximation to the partition function into an algorithm for the approximation of marginal probabilities to facilitate SQM based parameter learning. The marginal approximation is based on random variables whose density differs from the ordinary SQM tuple density. We therefore calculated the optimal importance weights in order to devise a shared sampling procedure for approximating all marginal probabilities of a discrete state space model simultaneously.

To round our theoretical insights and gain some practical intuition about how our new methods perform, we conducted extensive experiments on synthetic and real-world data. Among various graphical structures, the error increases sublinearly whenever the parameter norm increases. Surprisingly, it turned out that rather small polynomial degrees suffice for a good approximation, as long as the sample size is large enough. A linearly decreased error can be observed whenever the polynomial degree and the number of samples increases. The improvement from taking more samples is in general sublinear. The memory requirements are linear in the number of computed marginals and the runtime of SQM is a linear function of the polynomial degree and number of samples. In almost all experiments, loopy belief propagation was faster than SQM and delivered a lower error. Nevertheless, this behavior cannot be guaranteed for all conditional independence structures, while the behavior of SQM is guaranteed. It depends on the actual application if the slightly worse but trustworthy SQM results are to be preferred over LBP.

The basic idea of SQM, the tuple density, the error bound (Theorem 5.3), and the k-integrability of discrete MRFs (Lemma 5.2) have been published before in [172]. All results related to marginal inference, fast normalization of the tuple density, and fast sampling from the tuple density are new. Moreover, compared to [172], new experiments have been conducted which led to new insights on the error and the runtime behavior.

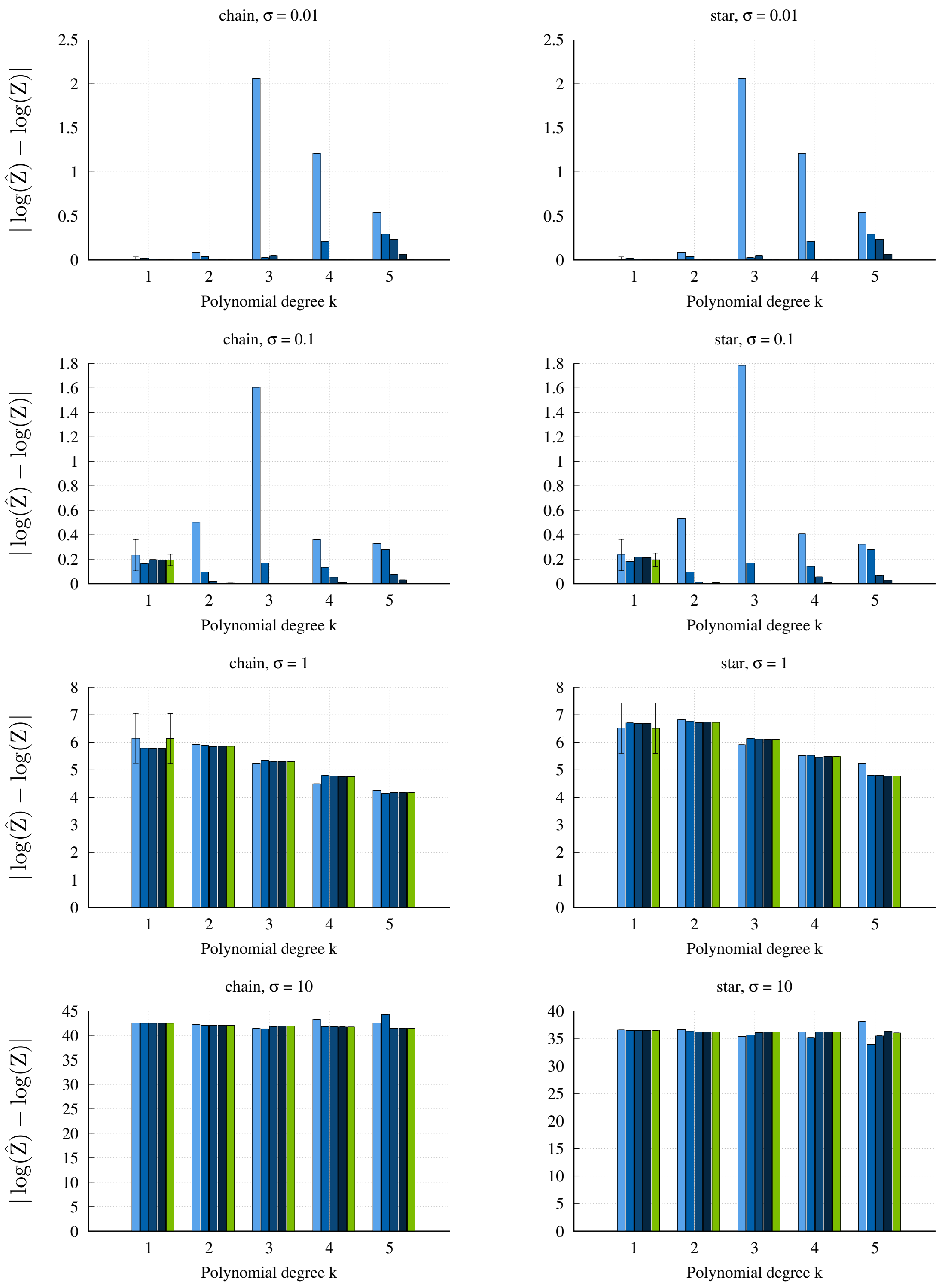

Figure 5.5: Average approximation error for the log-partition function on chain and star structures. Results for five SQM variants and five polynomial degrees. From left to right: 10^3-SQM, 10^4-SQM, 10^5-SQM, 10^6-SQM, exact SQM (see Fig. 5.4). Results are averaged over 10 repetitions.

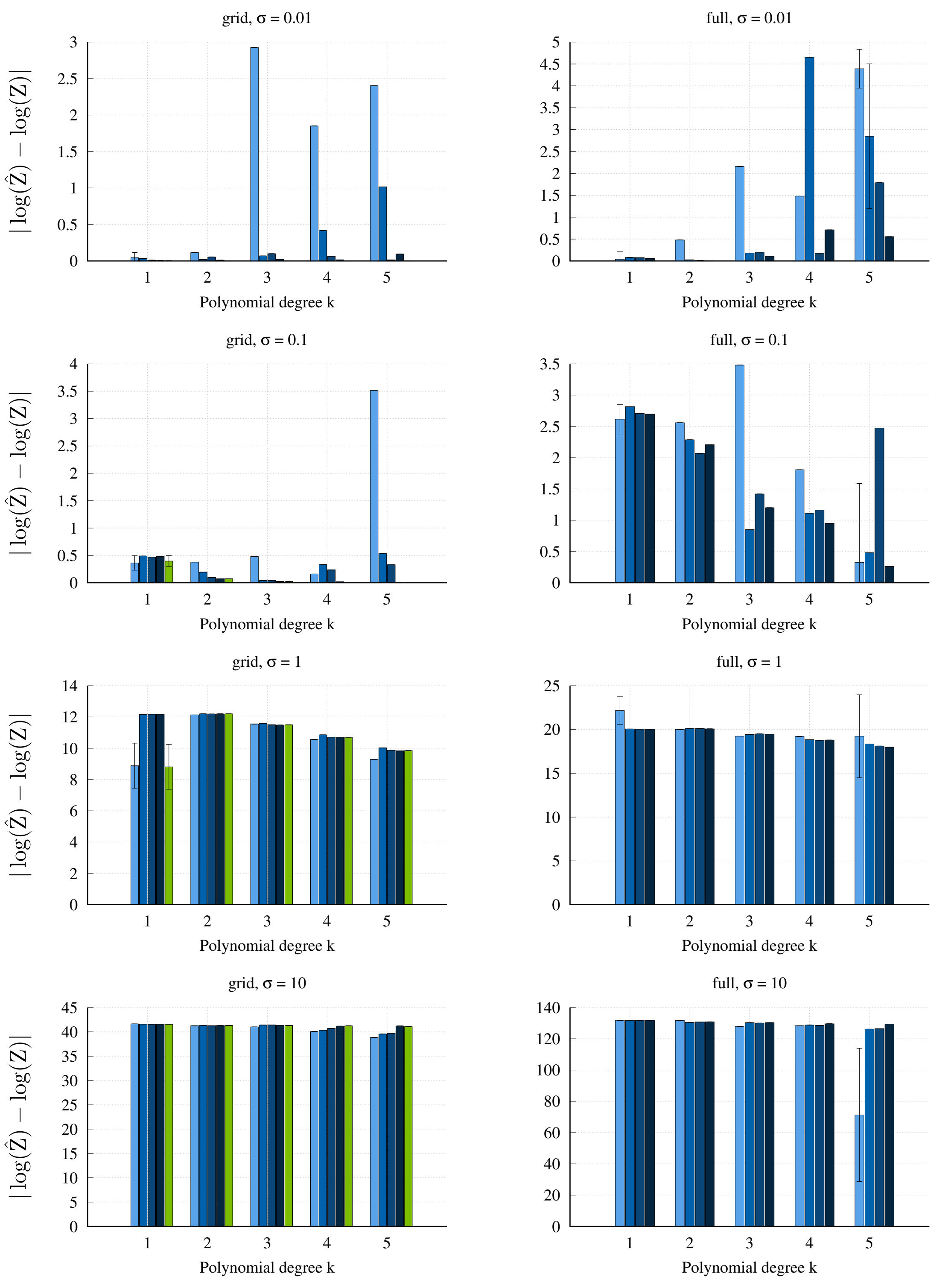

Figure 5.6: Average approximation error for the log-partition function on grid and fully connected structures. Results for five SQM variants and five polynomial degrees. From left to right: 10^3-SQM, 10^4-SQM, 10^5-SQM, 10^6-SQM, exact SQM (see Fig. 5.4). Results are averaged over 10 repetitions.

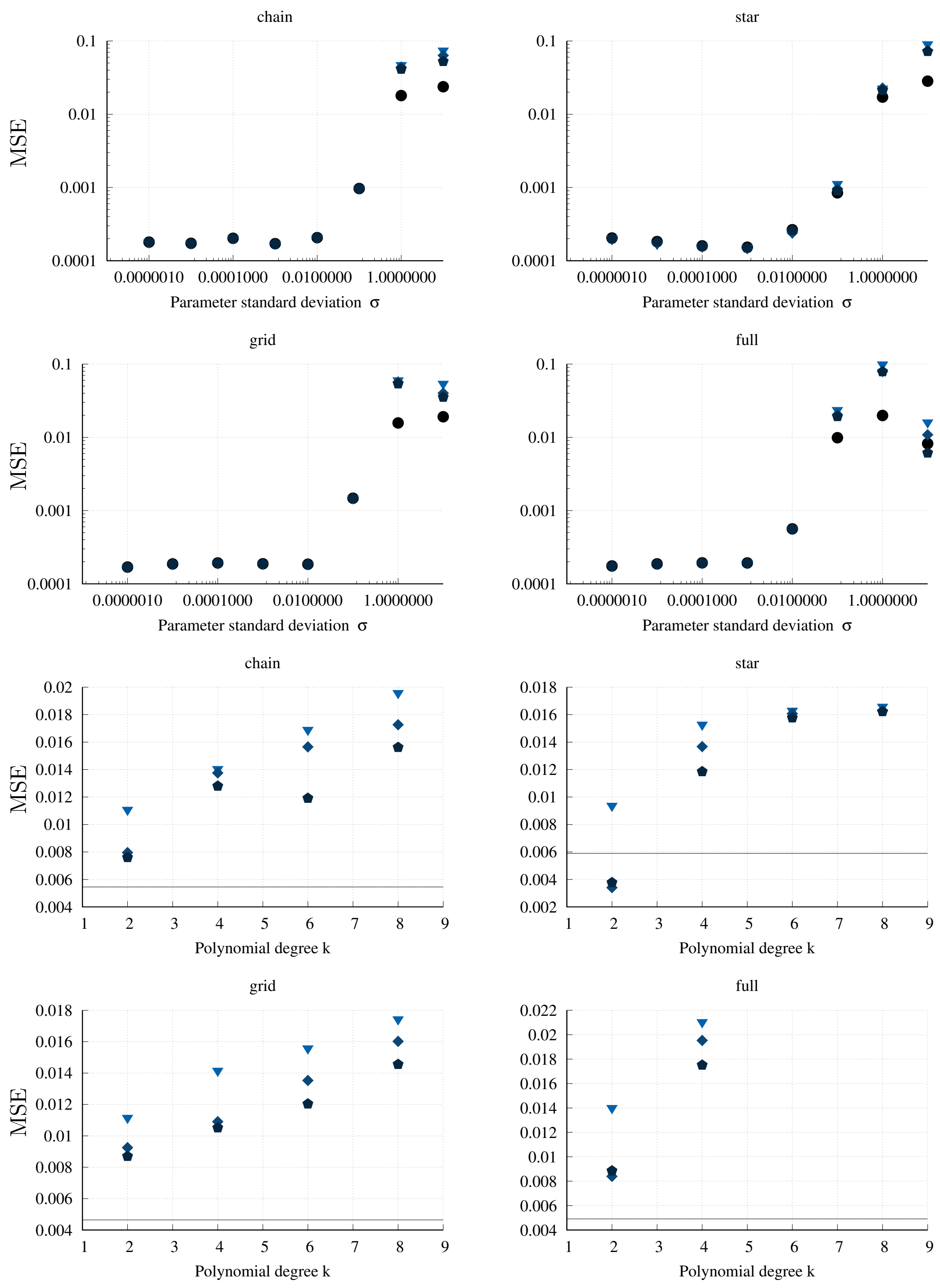

Figure 5.7: Average MSE in estimated marginals (y-axis), as a function of the standard deviation (x-axis, first two rows), and as a function of the polynomial degree (x-axis, last two rows). The results are averaged over multiple runs and various degrees. Different colors indicate different decay types (see Fig. 5.4). Straight lines indicate constant **M1** results.

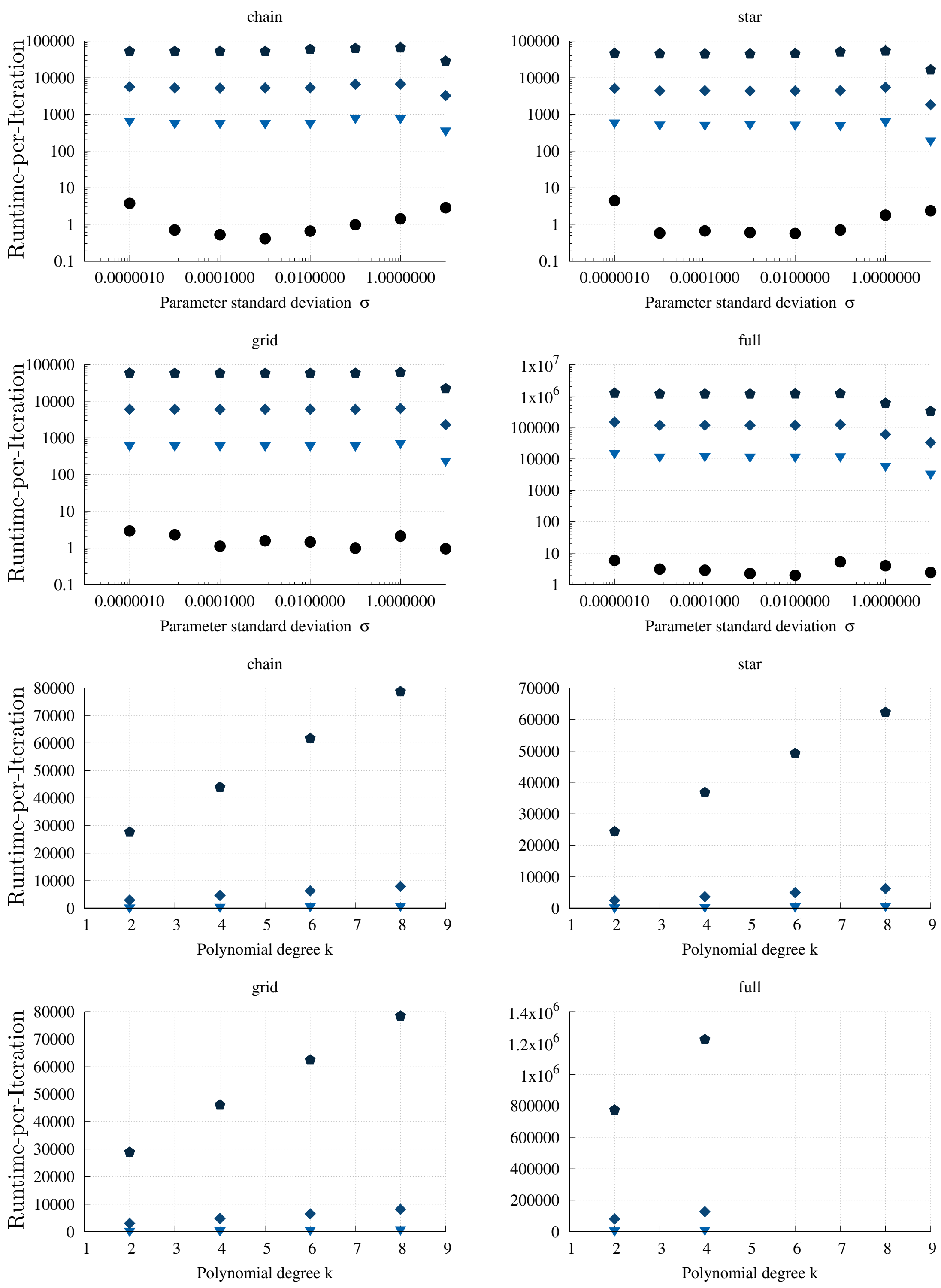

Figure 5.8: Average runtime per iteration in milliseconds (y-axis), as a function of the standard deviation (x-axis, first two rows), and as a function of the polynomial degree (x-axis, last two rows). The results are averaged over multiple runs and various standard deviations. Different colors indicate different decay types (see Fig. 5.4).

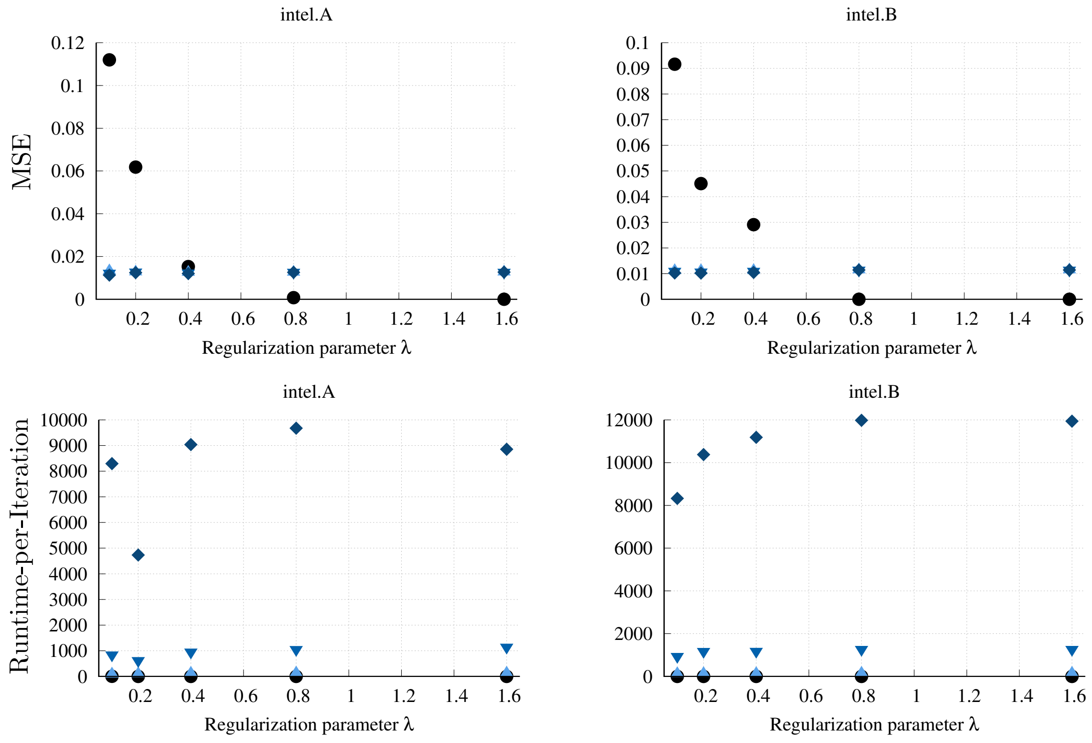

Figure 5.9: Experimental results on the Intel Lab data. Average MSE between estimated and empirical marginals (first row), and average runtime per training iteration (last row), as a function of the regularization parameter λ (x-axis), averaged over various polynomial degrees.

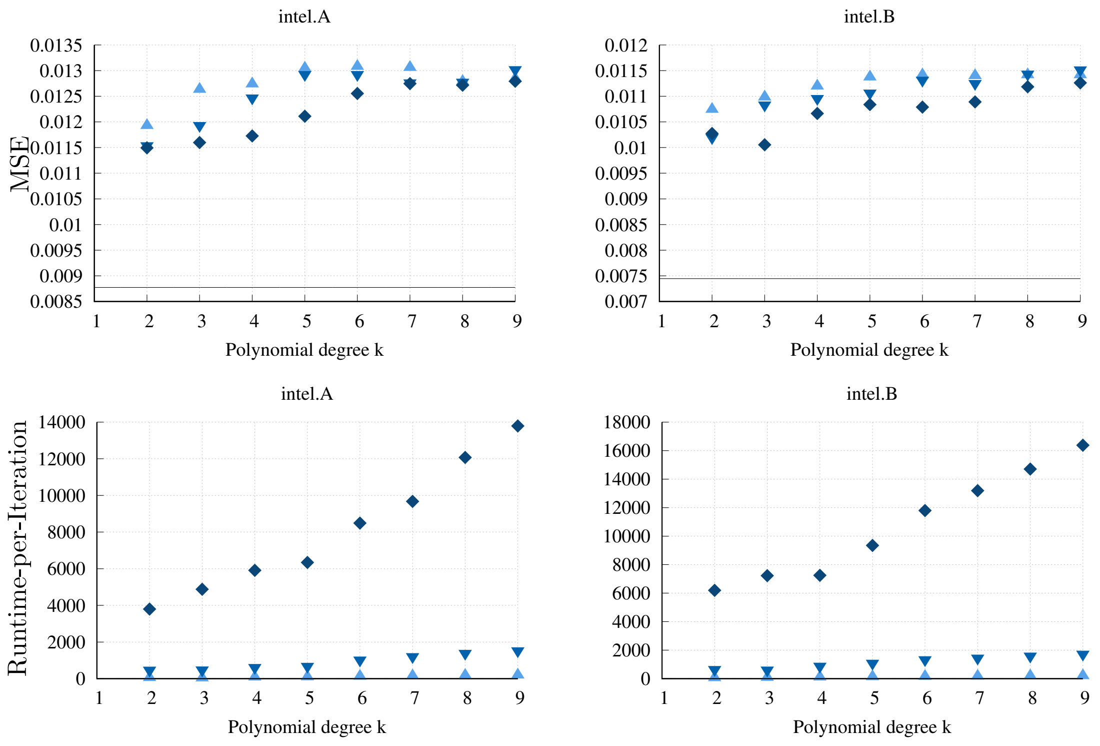

Figure 5.10: Experimental results on the Intel Lab data. Average MSE between estimated and empirical marginals (first row), and average runtime per training iteration (last row), as a function of the polynomial degree (x-axis), averaged over various regularization weights. Straight lines indicate constant **M1** results.

6 Conclusion

Theoretical and empirical results from Chapters 3–5 showed, that our techniques can reduce the memory consumption, arithmetic requirements, and computational complexity of ordinary exponential family models. The conditional independence structure is kept intact, no new assumptions are introduced, and our proposed methods arise from the very definition of the exponential family. Reparametrization, integer models, and the stochastic quadrature are inherently connected via regularization, which allows us to transfer resource constraints to constraints or costs on the model's parameter space. More detailed discussions of related work can be found at the beginning of each chapter, while a discussion of our results can be found at the end of each Chapter.

Here, we conclude this work by discussing our findings in the context of the Texas Instruments MSP430-FR5x microcontroller unit. The system is representative for state-of-the-art ultra-low-power architectures. It contains a 16 bit CPU with reduced instruction set at 16 MHz clock rate. At runtime, it consumes 118 $^{\mu\text{A}}/_{\text{MHz}}$ (with 500 nA standby). It provides 256 kB of non-volatile ferroelectric random-access memory and 8 kB static random-access memory. Common applications of such systems are metering, wearable electronics, sensor management, logging, home automation, watches, fitness accessories and other mobile consumer electronics. The utility of machine learning and probabilistic models on such systems depends on the number of variables that can be processed, and the runtime of learning and inference algorithms. Specifically, we investigate if our proposed methods increase the size of models that can be learned on the MSP430-FR5x MCU.

Our sparse reparametrization is designed to allow more spatio-temporal models to fit into the main memory of a resource-constrained system. Let us review the experimental results from Chapter 3 (shown in Fig. 3.10–3.17) in the context of the MSP430-FR5x. Regularized models enjoy a superior sparsity, compared to ordinary maximum likelihood estimates. If the regularization is too strong, the model quality begins to decrease. Our proposed reparametrization approach damps this effect by transferring information from non-zero parameters to parameters which are 0. Thus, parameters may be zero without sacrificing the model's quality. Exemplary results are summarized in Table 6.1. The table contains the average sizes in kilobyte of estimated parameter vectors, supposing a 32 bit floating-point representation, for models without regularization, with l_1-regularization, and with l_1-regularized reparametrization. Models on synthetic data use the inverse exponential decay and models on real-world data the rational decay. Bold entries indicate that the parameter vector would fit into the main memory of our ultra-low-power architecture, where we assume that the inference algorithm, the optimization algorithm and additional variables consume ≈ 50 kB. First of all, we see that our regularization approach doubles the number of models which fit onto the MCU. Secondly,

Table 6.1: Memory consumption (in kilobyte) of $\boldsymbol{\theta}$ (32 bit floating-point) without regularization, with l_1-regularization, and with l_1-regularized reparametrization (inverse exponential decay for synthetic data and rational decay for real-world data). Results shown are for $\lambda = 0.64$ and $T = 32$, averaged over all redundancy levels. Bold models would fit into the main memory of the MSP430-FR5x MCU, assuming 50 kB program code.

Data	d	**None**	l_1**-Reg.**	**Reparam.**
Chain	1066.68	4266.72	247.52	**202.08**
Star	1084.0	4336.0	**201.6**	**197.44**
Grid	1037.8	4151.2	277.92	**199.04**
Full	843.8	3375.2	257.92	**181.6**
Insight	2662025.0	10648100.0	5940.0	7130.0
Intel	62150.0	248600.0	**142.0**	430.0
Vavel	5610650.0	22442600.0	90196.0	30508.0

we know from the results in Chapter 3 that the plain l_1-regularized models sacrifice the conditional independence structure and hence exhibit a substantially higher mean squared error than the reparametrized models. Thus, although two l_1-models would in principle fit into the systems memory, they are unusable in practice due to their low quality.

The second major obstacle when we try to learn undirected models on the MSP430-FR5x is the missing floating-point hardware. It is indeed possible to emulate 32 bit and 64 bit floating-point arithmetic in software, but this is accompanied by a very high computational overhead. Our integer undirected model which we presented in Chapter 4 solves this issue. We provide inference and optimization algorithms which work only on the integer domain—they do not require any floating-point arithmetic. Moreover, our theoretical derivations are valid for any subset $\{1, 2, \ldots, k-1\}$ of the non-negative integers, which implies that the results are also valid for small word-sizes. The experimental results in Figures 4.5–4.11 show, that small values of k suffice to provide a practical model quality. Hence, we may choose the native 16 bit integers as underlying data type for our integer undirected model. Corresponding runtime results are summarized in Tables 6.2 and 6.3. The values in columns three and four represent the average runtime in milliseconds of one iteration of loopy belief propagation and one iteration of bit-length propagation executed on the MSP430-FR5x MCU. Table 6.2 contains LBP results for emulated 64 bit floating-point arithmetic, and Table 6.3 contains the corresponding results for emulated 32 bit arithmetic[28]. BLprop was executed with the MCU's native word-size of 16 bit. Both sets of results are unambiguous: compared to 64 bit emula-

[28] 16 bit floating-point emulation is not supported by the MCU.

Table 6.2: Runtime in milliseconds of one iteration of message passing for LBP (64 bit floating-point), generation of one SQM sample (polynomial degree 2, 64 bit floating-point), and one iteration of BLprop (16 bit integer), on an MSP430-FR5x microcontroller unit.

Data	$\lvert E\rvert$	**LBP** (1 iter)	**BLprop** (1 iter)	**SQM** (1 sample)
Chain	15	4843.4	19.0	773.8
Star	15	4744.3	19.0	774.0
Grid	24	7713.9	29.5	1081.3
Full	120	40422.6	141.2	4704.2

Table 6.3: Runtime in milliseconds of one iteration of message passing for LBP (32 bit floating-point), generation of one SQM sample (polynomial degree 2, 32 bit floating-point), and one iteration of BLprop (16 bit integer), on an MSP430-FR5x microcontroller unit.

Data	$\lvert E\rvert$	**LBP** (1 iter)	**BLprop** (1 iter)	**SQM** (1 sample)
Chain	15	1156.2	19.0	350.3
Star	15	1140.4	19.0	393.1
Grid	24	1838.1	29.5	445.3
Full	120	9642.1	141.2	1549.7

tion, our integer models are at least 250 times faster. In case of 32 bit emulation, the speedup is still 60. Having the results from Section 4.5.2 in mind, we see that the benefit of our integer models is higher the weaker the underlying computational architecture is. Moreover, we know from Section 4.5.2 that good integer parameters can be found, even when the parameters of the data generating process are far from integrality. Thus, one may indeed expect to observe the same behavior in practice. As explained in Chapter 4, the speedup goes hand in hand with a reduced energy consumption. When we assume that our microcontroller is battery powered, using integer models can hence significantly extend the uptime of our system.

Reparametrization and integer models may be combined, e.g., by choosing binary decay matrices. However, both rely on approximate message passing algorithms to perform probabilistic inference. Such algorithms have several limitations like unknown convergence behavior on general cyclic graphs, and an unbounded approximation error. In certain settings, these properties may be undesirable and one has to resort to exact algorithms or approximation algorithms with error bounds. Those algorithms, like the junction tree algorithm or WISH, have an exponential complexity and are hence not well-suited for small systems. Our stochastic quadrature method, presented in Chapter 5, combines polynomial approximation with Monte Carlo sampling for fast approximate inference. Moreover, the approximation error is bounded and depends on the norm of the parameter vector $\boldsymbol{\theta}$. Our results from Section 5.6 show, that small polynomial degrees and moderate sample numbers are often enough to achieve practical error rates. Exemplary results on the MSP430-FR5x can be found in the fifth column of Tables 6.2 and 6.3. We measured the average runtime to generate a single SQM sample for a polynomial degree of 2. The runtime is in between LBP and BLprop, which, in contrast to SQM, do not provide any guarantees on general loopy graphs. In practical applications, several samples have to be drawn. When we assume a sample size of 100, the accumulated runtime would be within $1/2$ and 8 minutes. This could indeed be prohibitive for some applications. Nevertheless, Theorem 5.3 allows us to find an upper bound on the error that depends on the number of samples. For any fixed model, we may thus adjust the sample size to find a reasonable tradeoff between resource consumption and quality. In contrast, the error of message passing on loopy graphs may oscillate, and a functional relation between the LBP error and the number of iterations is unknown. Moreover, SQM admits that several computational steps may be precomputed on a strong system. In the above example, $\boldsymbol{\theta}$ was known and we precomputed τ, the $\|\chi^i_\phi\|_1$ values, and the polynomial coefficients $\boldsymbol{\zeta}$. These values are valid for all parameters whose norm does not exceed $B = \|\boldsymbol{\theta}\|$. It is thus possible to reuse the same precomputed values during training, if the optimization procedure guarantees that the parameter norm does not exceed B, e.g., via regularization.

To sum up, our techniques reduced the memory consumption, accelerated inference and learning, and offer several ways to trade quality against resource consumption. Hence, a wider range of probabilistic models can be learned and applied on resource-constrained systems. This was the first investigation of undirected exponential family models on ULP devices. We hope that our results can guide the future development of ubiquitous machine learning systems and provide inspiration for future research. We close this work by pointing out some possible future directions.

6.1 Future Directions

One new direction for reparametrized models is the derivation of specialized inference algorithms like those known for dynamic Bayesian networks [22]. Since the message computation schedule of loopy belief propagation is almost arbitrary, one could perform a layer-wise forward-backward scheme which only stores the message of a single layer at a time. The simple piecewise linear form of our reparametrization may also be exploited to propose new kinds of messages in which the parameter decoding is performed implicitly. Another direction treats the reparametrization itself. For now, we used predefined decay types. However, decay matrices could be learned during parameter estimation, like filter kernels in convolutional neural networks [79]. Such an approach would render the optimization problem non-convex, but theoretically well-defined optimization procedures may still be found [10, 25]. Moreover, we predefined the particular functional form of our reparametrization, but an automated derivation of functional dependencies between parameters is also conceivable. First approaches include automatic parameter tying, based on k-means clustering [40], as well as automatic parameter tying via random hash functions.

In the field of mixed-integer programming (MIP), most algorithms rely on relaxations and do not consider any form of integer regularization to find critical integer solutions. It is hence interesting to understand the benefits and limitations of our integer gradient descent method in the general MIP setting. In machine learning, several heuristic approaches for learning quantized parameters of neural networks are known. However, most of them have no theoretical justification [116]. It will therefore improve the field when such techniques would be reviewed in the light of our integer gradient descent and BLprop algorithms. Other directions include integer approximations to distributions with infinite support, like the Poisson distribution, rational random variables, and integer tree-reweighted belief propagation with rational edge appearance probabilities. Also the combination of spatio-temporal and integer models is interesting, and first experiments have already been conducted on smartphone usage data [174].

Our SQM runtime results on the microcontroller unit revealed that the method still suffers from a high runtime. Due to the complexity of the inference problem, it should be clear that we cannot find arbitrary good solutions in arbitrary short time. However, we expect that a substantial proportion of this runtime is due to the emulation of floating-point arithmetic. Assuming that the true model parameters are integer, an integer SQM is conceivable. In fact, the values of $|\chi^i_\phi|$ $(\forall i)$ are integer by definition, and any polynomial can be approximated well by another polynomial with rational coefficients. Based on these observations, integer SQM could reduce the runtime issues of ordinary SQM on resource-constrained systems. Moreover, we showed which types of continuous sufficient statistics are k-integrable, but our results have not yet been connected to known continuous distributions. Discrete and continuous statistics could be combined to yield an SQM-based inference procedure for random variables with discrete and continuous components. In Section 5.2 we explained that existing error bounds for the general quadrature do not apply to our method, due to the non-uniform sampling of the approximation interval $[l; u]$ during discrete integration. Including knowledge about the energy landscape can lead to non-uniform polynomial approximations, in which highly dense regions exhibit a lower error than regions that contain only a few instances. Finally, we used the norm of the parameter vector as an upper bound to the inner product via Hölder's inequality. By using additional knowledge about the parameter's prior distribution, the norm could be removed from the bound, which would be interesting especially in a Bayesian setting.

7 Appendix

7.1 Basic Probability and Information Theory

For reasons of clarity and completeness, we give short rigorous definitions of random variables and expected values. An extensive introduction to probability theory can be found in [41].

Definition 7.1 (Probability Measure) *Let $\mathcal{Q}$ be some ground set and $\Sigma \subseteq \mathcal{P}(\mathcal{Q})$ a subset of its power set. The pair $(\mathcal{Q}, \Sigma)$ is a measurable space if $\mathcal{Q} \in \Sigma$, $\overline{\mathcal{Q}} = \emptyset$, and Σ is closed under complementation and countable union—Σ is then a σ-algebra relative to $\mathcal{Q}$. A function $\nu : \Sigma \to \mathbb{R} \cup \{+\infty\}$ is a measure, if it is non-negative, with $\nu(\emptyset) = 0$, and for any sequence of disjoint sets $S_1, S_2, \dots \in \Sigma$, it holds that $\nu(\cup_{i=1}^{\infty} S_i) = \sum_{i=1}^{\infty} \nu(S_i)$. If further $\nu : \Sigma \to [0; 1]$ and $\nu(\mathcal{Q}) = 1$, then ν is a probability measure. A function $f : \mathcal{A} \to \mathcal{B}$ is measurable, if $(\mathcal{A}, \Gamma)$ and $(\mathcal{B}, \Lambda)$ are measurable spaces and the preimage of any $S \in \Lambda$ is contained in Γ, i.e., $f^{-1}(S) = \{a \in \mathcal{A} \mid f(a) \in S\} \in \Gamma$.*

Measures allow us to assign non-negative numbers to sets, in a well-defined way. Intuitively, Σ is the space of all measurable combinations of atoms from $\mathcal{Q}$. If one thinks of real-world events as sets of atomic situations in which the event occurs, its (probability) measure may loosely be interpreted as a fraction of possible situations in which an event may occur. Based on these very basic concepts, we define a special type of measurable transformation which is known as random variable.

Definition 7.2 (Random Variable) *Let $(\mathcal{Q}, \Sigma)$ and $(\mathcal{X}, \Lambda)$ be two measurable spaces with probability measure $\mathbb{P}$ on $(\mathcal{Q}, \Sigma)$. A random variable $\boldsymbol{X}$ is a measurable function from $\mathcal{Q}$ to $\mathcal{X}$. $\mathcal{Q}$ is called reference space and $\mathcal{X}$ is called state space of $\boldsymbol{X}$. If the state space is finite or countable infinite, $\boldsymbol{X}$ is a discrete random variable. Whenever $\mathcal{X}$ is uncountable, $\boldsymbol{X}$ is a continuous random variable. Moreover, we declare the existence of a base measure ν on $(\mathcal{X}, \Lambda)$, such that the pushforward measure $\mathbb{P} \circ \boldsymbol{X}^{-1}$ is absolutely continuous w.r.t. ν. This means that for any $S \in \Lambda : \nu(S) = 0 \Rightarrow (\mathbb{P} \circ \boldsymbol{X}^{-1})(S) = 0$, i.e., $\mathbb{P} \circ \boldsymbol{X}^{-1}$ is dominated by ν.*

For any random variable $\boldsymbol{X}$, the notation $\mathbb{P}(\boldsymbol{X} \in S)$ for some $S \subseteq \mathcal{X}$ is an abbreviation for $\mathbb{P}(\boldsymbol{X}^{-1}(S)) = \mathbb{P}(\{a \in \mathcal{Q} \mid \boldsymbol{X}(a) \in S\})$, i.e., the "size" (measure) of the set that contains all atoms a whose image is in S. Whenever $S = \{\boldsymbol{x}\}$ is a singleton set, we write $\mathbb{P}(\boldsymbol{X} = \boldsymbol{x})$ instead of $\mathbb{P}(\boldsymbol{X} \in S)$. The intuition is, that a random variable $\boldsymbol{X}$ can take different values $\boldsymbol{x}$, also called realizations, from the state space $\mathcal{X}$. The number of times a particular $\boldsymbol{x}$ will be seen whenever we look at $\boldsymbol{X}$ is controlled by the probability measure $\mathbb{P}$. We write $\boldsymbol{x} \sim \mathbb{P}$ to indicate that the chances to observe a fixed $\boldsymbol{x}$ are controlled by $\mathbb{P}$—we also say that $\boldsymbol{x}$ was sampled from $\mathbb{P}$.

Various types of measures and σ-algebras are studied in measure theory. Here, we restrict ourselves to the cases which are relevant for inference in exponential families. We set the Lebesgue measure as the base measure of continuous random variables, i.e., the measure of any $S \in \Lambda$ is

$$\nu(S) = \inf \left\{ \sum_{i=1}^{\infty} \text{length}(I_i) \mid I_1, I_2, \ldots \text{ open intervals in } \mathbb{R} \wedge S \subseteq \bigcup_{i=1}^{\infty} I_i \right\}$$

and Λ is a Lebesgue σ-algebra. That is, Λ contains all subsets of $\mathbb{R}$ such that $\nu(R) = \nu(R \cap S) + \nu(R \cap \overline{S})$, $\forall R \in \mathbb{R}$. It follows that the probability of a continuous $\boldsymbol{X}$ taking any particular value $\boldsymbol{x}$ is zero; any $\boldsymbol{x}$ is contained in all open ϵ-intervals around it—$\boldsymbol{x} \in (\boldsymbol{x} - \epsilon; \boldsymbol{x} + \epsilon)$ for all $\epsilon > 0$—and the length of any of these intervals is 2ϵ. The infimum over all these interval lengths must be zero. Thus $\nu(\{\boldsymbol{x}\}) = 0$ and hence $\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}) = \mathbb{P}(\boldsymbol{X}^{-1}(\{\boldsymbol{x}\})) = 0$ by absolute continuity.

Given Definition 7.2, the Radon-Nikodym theorem [41] guarantees the existence of a measurable function $p : \boldsymbol{X} \to [0; \infty)$ with $\mathbb{P}(\boldsymbol{X} \in S) = \int_S p \, \mathrm{d}\, \nu$ almost everywhere. In the context of continuous random variables, p is called probability density function (p.d.f.) of $\boldsymbol{X}$.

When a random variable is discrete, we set the counting measure as its base measure, i.e., $\forall S \in \Lambda : \nu(S) = |S|$ whenever S is finite and $\nu(S) = +\infty$ otherwise. Plugging the counting measure into the Radon-Nikodym theorem shows that the probability measure $\mathbb{P}$ and the probability density function p are equal: $\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}) = \mathbb{P}(\boldsymbol{X}^{-1}(\{\boldsymbol{x}\})) = \int_{\{\boldsymbol{x}\}} p \, \mathrm{d}\, \nu = p(\boldsymbol{x})$, because $\nu(\{\boldsymbol{x}\}) = 1$. In the context of discrete random variables, p and $\mathbb{P}$ are called probability mass function (p.m.f.) of $\boldsymbol{X}$. Whenever it's clear from the context that $\boldsymbol{X}$ is a discrete random variable, we will use $\mathbb{P}$ instead of p.

Expressions involving random variables cannot be evaluated directly due to their random nature. Functions ϕ whose domain is the state space of a random variable $\boldsymbol{X}$ implicitly define a new random variable $\phi(\boldsymbol{X})$ whose state space is the image of ϕ and its reference space is inherited from $\boldsymbol{X}$. Instead of evaluating ϕ directly, it is possible to evaluate its expected value—an integration[29] over all possible states of a random variable.

Definition 7.3 (Expectation) *Let $\boldsymbol{X}$ be a random variable and ϕ any function on its state space. The expected value of $\phi(\boldsymbol{X})$, denoted by $\mathbb{E}[\phi(\boldsymbol{X})]$ is*

$$\mathbb{E}[\phi(\boldsymbol{X})] = \int_{\mathcal{Q}} (\phi \circ \boldsymbol{X}) \, \mathrm{d}\, \mathbb{P} = \int_{\mathcal{X}} \phi \, \mathrm{d}\, \mathbb{P} \circ \boldsymbol{X}^{-1} = \int_{\mathcal{X}} \phi \frac{\mathrm{d}\, \mathbb{P} \circ \boldsymbol{X}^{-1}}{\mathrm{d}\, \nu} \, \mathrm{d}\, \nu = \int_{\mathcal{X}} \phi p \, \mathrm{d}\, \nu \ ,$$

and if $\boldsymbol{X}$ is discrete, the statement reduces to

$$\mathbb{E}[\phi(\boldsymbol{X})] = \sum_{\boldsymbol{x} \in \mathcal{X}} \phi(\boldsymbol{x}) \mathbb{P}(\boldsymbol{X} = \boldsymbol{x}) \ .$$

[29]Whenever an integration is carried out over all variables, we suppress the arguments of the involved functions and use the fully functional notation (cf. [41]).

The raw expectation of $\boldsymbol{X}$ is achieved by setting ϕ to the identity function.

Note that different probability densities can lead to the same expectation, i.e., $p \neq q$ but $\mathbb{E}_p[\phi(\boldsymbol{X})] = \mathbb{E}_q[\phi(\boldsymbol{X})]$.

When we observe processes from nature and interpret them as random variables, we do most often encounter situations in which not one but multiple random variables are of interest. In particular, we want to understand how multiple variables interact and influence each other.

Definition 7.4 (Marginalization) *Let $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{W}$ be random variables with joint measure $\mathbb{P}$, and joint density p—their σ-algebras are sub-σ-algebras of $\Sigma_{\boldsymbol{X},\boldsymbol{Y},\boldsymbol{W}}$ and the underlying product reference measure*[30] *is ν. The specific marginal densities $p_{\boldsymbol{X}}$, $p_{\boldsymbol{Y}}$, and $p_{\boldsymbol{W}}$ are discovered by integrating the remaining variables out, i.e.,*

$$p_{\boldsymbol{X}}(\boldsymbol{x}) = \int_{\mathcal{Y}} \int_{\mathcal{W}} p(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}) \, \mathrm{d}\, \nu_{\boldsymbol{Y}}(\boldsymbol{y}) \nu_{\boldsymbol{W}}(\boldsymbol{w}) \; .$$

Again, we get a plain summation in case of discrete random variables:

$$\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}) = \sum_{\boldsymbol{y} \in \mathcal{Y}} \sum_{\boldsymbol{w} \in \mathcal{W}} \mathbb{P}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}) \; .$$

Any subset of variables may be integrated out to get the marginal probability of the remaining variables.

Based on marginalization, we may "extract" the isolated behavior of any random variable from the joint density of the system in which it exists. It is often not desired to remove the dependence via full marginalization, because the behavior of a random variable $\boldsymbol{X}$ might change if some related event $\boldsymbol{Y} \in S$ has occurred.

Definition 7.5 (Conditional Probability) *The conditional density of $\boldsymbol{X} = \boldsymbol{x}$ given $\boldsymbol{Y} = \boldsymbol{y}$ is defined as the joint density of $\boldsymbol{x}$ and $\boldsymbol{y}$ relative to the marginal density of $\boldsymbol{y}$:*

$$p_{\boldsymbol{X}}(\boldsymbol{x} \mid \boldsymbol{Y} = \boldsymbol{y}) = \frac{p(\boldsymbol{x}, \boldsymbol{y})}{p_{\boldsymbol{Y}}(\boldsymbol{y})} \; .$$

In this case, we say that the density of $\boldsymbol{X}$ is conditioned on $\boldsymbol{Y} = \boldsymbol{y}$. Note that $\boldsymbol{X}$ cannot be conditioned on $\boldsymbol{Y} = \boldsymbol{y}$ whenever $p_{\boldsymbol{Y}}(\boldsymbol{y}) = 0$. Finally, $\boldsymbol{X}$ and $\boldsymbol{Y}$ are independent if and only if $\forall \boldsymbol{x} \in \mathcal{X} : \forall \boldsymbol{y} \in \mathcal{Y} : p(\boldsymbol{x}, \boldsymbol{y}) = p_{\boldsymbol{X}}(\boldsymbol{x}) p_{\boldsymbol{Y}}(\boldsymbol{y})$, denoted by $\boldsymbol{X} \perp\!\!\!\perp \boldsymbol{Y}$. Equivalently, $\boldsymbol{X}$ and $\boldsymbol{Y}$ are independent when their sub-σ algebras are independent.

Similarly, $\boldsymbol{X}$ and $\boldsymbol{Y}$ are independent given $\boldsymbol{W} = \boldsymbol{w}$ if and only if $\forall \boldsymbol{x} \in \mathcal{X} : \forall \boldsymbol{y} \in \mathcal{Y} : p_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{W} = \boldsymbol{w}) = p_{\boldsymbol{X}}(\boldsymbol{x} \mid \boldsymbol{W} = \boldsymbol{w}) p_{\boldsymbol{Y}}(\boldsymbol{y} \mid \boldsymbol{W} = \boldsymbol{w})$, denoted by $\boldsymbol{X} \perp\!\!\!\perp \boldsymbol{Y} \mid \boldsymbol{W} = \boldsymbol{w}$.

[30] Both, the counting measure on σ-algebras with countable reference sets, and the Lebesgue measure on Lebesgue σ-algebras are σ-finite. Hence, product measures are uniquely determined by Fubini's theorem.

For ease of exposition, multiple random variables are often collected together to form a random variable, also called random field, whose components are indexed by a set $V = [n]$. Here, $[n]$ denotes the set of the first n positive integers $[n] = \{1, 2, \ldots, n\}$.

Having all that said, we drop the notion of a reference space. Within this thesis a (multivariate) random variable $\boldsymbol{X}$ is fully specified by its (joint) density and its state space $\mathcal{X}$. Moreover, unless otherwise explicitly stated, $\boldsymbol{X}$ is a discrete random variable and its p.m.f. $\mathbb{P}(\boldsymbol{X} = \boldsymbol{x})$ is abbreviated by $\mathbb{P}(\boldsymbol{x})$ or simply $p(\boldsymbol{x})$ whenever the random variable is known from the context.

7.2 Information Entropy and Related Functionals

Randomness is closely related to the concept of uncertainty. If all outcomes of a random variable are almost equally likely, we are rather uncertain about which particular realization we will see. On the other hand, if a random variable takes only a few values with high probability while other are unlikely, we can be rather certain that one of these values will appear. This intuition is captured by the concept of entropy.

Definition 7.6 (Entropy) *Let $\boldsymbol{X}$ be a random variable with probability measure $\mathbb{P}$ and strictly positive*[31] *density p. The functional $\mathcal{H}_b$, defined by*

$$\mathcal{H}_b[\mathbb{P}] = -\int_{\mathcal{X}} p \log_b p \, \mathrm{d}\,\nu = \mathbb{E}\left[-\log_b p(\boldsymbol{X})\right]$$

is the base-b entropy of $\mathbb{P}$. Whenever the base of the logarithm is not explicitly mentioned, we set $b = e$, i.e., $\mathcal{H} = \mathcal{H}_e$.

Varying the base of the logarithm in the above definition, changes the unit of entropy. In this thesis, we will focus on $\mathcal{H}_e$ and $\mathcal{H}_2$, which yield the units nat and bit, respectively. Choosing a specific base seems neglectable, but we will see in Chapter 4 under which circumstances the base has major implications for resource-constrained systems.

Based on the entropy, we derive a measure that tells us how much certainty we loose, when we try to measure the probability of $\boldsymbol{X}$ with a “wrong” measure $\mathbb{F}$ in place of the correct measure $\mathbb{P}$.

Definition 7.7 (Kullback-Leibler Divergence) *Let $\boldsymbol{X}$ be a random variable with probability measure $\mathbb{P}$, and some alternative probability measure $\mathbb{F}$ on the same σ-algebra with densities p and q, respectively. The functional* KL*, with*

$$\mathrm{KL}[\ \mathbb{P} \parallel \mathbb{F}\ ] = \int_{\mathcal{X}} p \log \frac{p}{q} \, \mathrm{d}\,\nu$$

is the Kullback-Leibler divergence between $\mathbb{P}$ and $\mathbb{F}$.

[31]In this definition, we require that p is strictly positive. Some authors allow $p : \mathcal{X} \to [0; +\infty]$ and set $0 \log 0 = 0$ to define the entropy. However, probability densities considered in this thesis, namely exponential families of densities, cannot map any state to 0 and the convention $0 \log 0 = 0$ is hence not required in the context of this thesis.

Note that KL is not symmetric, i.e., $\mathrm{KL}[\,\mathbb{P} \,\|\, \mathbb{F}\,] \neq \mathrm{KL}[\,\mathbb{F} \,\|\, \mathbb{P}\,]$ unless $\mathbb{P} = \mathbb{F}$. Extending $\mathcal{H}$ and KL to the multivariate case is straightforward—summation or integration must be carried out over the full joint state space of all random variables. If $\mathbb{F}$ is chosen appropriately, KL implies a natural measure for independence between random variables.

Definition 7.8 (Mutual Information) *Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be two random variables with joint probability measure $\mathbb{P}$ and joint density p. Let further $\mathbb{F}$ be the probability measure in which $\boldsymbol{X}$ and $\boldsymbol{Y}$ are independent but have the same marginals as in $\mathbb{P}$, i.e., $q(\boldsymbol{x}, \boldsymbol{y}) = p_{\boldsymbol{X}}(\boldsymbol{x}) p_{\boldsymbol{Y}}(\boldsymbol{y})$. The mutual information between $\boldsymbol{X}$ and $\boldsymbol{Y}$ is*

$$\mathcal{I}[\boldsymbol{X}, \boldsymbol{Y}] = \mathrm{KL}[\,\mathbb{P} \,\|\, \mathbb{F}\,] = \int_{\mathcal{X}} \int_{\mathcal{Y}} p(\boldsymbol{x}, \boldsymbol{y}) \log \frac{p(\boldsymbol{x}, \boldsymbol{y})}{p_{\boldsymbol{X}}(\boldsymbol{x}) p_{\boldsymbol{Y}}(\boldsymbol{y})} \,\mathrm{d}\,\nu_{\boldsymbol{X}}(\boldsymbol{x}) \nu_{\boldsymbol{Y}}(\boldsymbol{y}) \;.$$

Moreover, $\mathcal{I}[\boldsymbol{X}, \boldsymbol{Y}] = 0 \Leftrightarrow \boldsymbol{X} \perp\!\!\!\perp \boldsymbol{Y}$.

Bibliography

[1] Pooyan Abouzar, David G. Michelson, and Maziyar Hamdi. Rssi-based distributed self-localization for wireless sensor networks used in precision agriculture. *IEEE Transactions on Wireless Communications*, 15(10):6638–6650, 2016.

[2] Alekh Agarwal, Sahand Negahban, and Martin J. Wainwright. Fast global convergence of gradient methods for high-dimensional statistical recovery. *The Annals of Statistics*, 40(5):2452–2482, 10 2012.

[3] Babak Ahmadi, Kristian Kersting, Martin Mladenov, and Sriraam Natarajan. Exploiting symmetries for scaling loopy belief propagation and relational training. *Machine Learning*, 92(1):91–132, 2013.

[4] Srinivas M. Aji and Robert J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

[5] Isaac Amundson, Manish Kushwaha, and Xenofon D. Koutsoukos. A method for estimating angular separation in mobile wireless sensor networks. *Journal of Intelligent and Robotic Systems*, 71(3-4):273–286, 2013.

[6] Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1-2):5–43, 2003.

[7] Alexander Artikis, Matthias Weidlich, Francois Schnitzler, Ioannis Boutsis, Thomas Liebig, Nico Piatkowski, Christian Bockermann, Katharina Morik, Vana Kalogeraki, Jakub Marecek, Avigdor Gal, Shie Mannor, Dimitrios Gunopulos, and Dermot Kinane. Heterogeneous stream processing and crowdsourcing for urban traffic management. In Sihem Amer-Yahia, Vassilis Christophides, Anastasios Kementsietsidis, Minos N. Garofalakis, Stratos Idreos, and Vincent Leroy, editors, *Proceedings of the 17th International Conference on Extending Database Technology (EDBT)*, pages 712–723. OpenProceedings.org, 2014.

[8] Kendall Atkinson and Weimin Han. *Theoretical Numerical Analysis*. Springer-Verlag New York, 3rd edition, 2009.

[9] Hedy Attouch and Jérôme Bolte. On the convergence of the proximal algorithm for nonsmooth functions involving analytic features. *Mathematical Programming*, 116(1):5–16, 2009.

[10] Hédy Attouch, Jérôme Bolte, Patrick Redont, and Antoine Soubeyran. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the kurdyka-lojasiewicz inequality. *Mathematics of Operations Research*, 35(2):438–457, 2010.

[11] Hédy Attouch, Jérôme Bolte, and Benar Fux Svaiter. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward-backward splitting, and regularized Gauss-Seidel methods. *Math. Program.*, 137(1-2):91–129, 2013.

[12] Hedy Attouch and Juan Peypouquet. The rate of convergence of nesterov's accelerated forward-backward method is actually faster than $1/k^2$. *SIAM Journal on Optimization*, 26(3):1824–1834, 2016.

[13] Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obo zinski. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, 4(1):1–106, 2012.

[14] Jiatong Bao, Yunyi Jia, Yu Cheng, Hongru Tang, and Ning Xi. Detecting target objects by natural language instructions using an RGB-D camera. *Sensors*, 16(12):2117, 2016.

[15] Jiatong Bao, Yunyi Jia, Yu Cheng, and Ning Xi. Saliency-guided detection of unknown objects in RGB-D indoor scenes. *Sensors*, 15(9):21054–21074, 2015.

[16] Dhruv Batra, Payman Yadollahpour, Abner Guzmán-Rivera, and Gregory Shakhnarovich. Diverse m-best solutions in Markov random fields. In *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V*, pages 1–16, 2012.

[17] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[18] Amir Beck and Luba Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization*, 23(4):2037–2060, 2013.

[19] Anne Berry, S. Jean R. Blair, Pinar Heggernes, and W. Barry Peyton. Maximum cardinality search for computing minimal triangulations of graphs. *Algorithmica*, 39(4):287–298, 2004.

[20] Julian Besag. Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 24(3):179–195, 1975.

[21] Hans-Georg Beyer, Hans-Paul Schwefel, and Ingo Wegener. How to analyse evolutionary algorithms. *Theoretical Computer Science*, 287(1):101–130, 2002.

[22] John Binder, Kevin P. Murphy, and Stuart J. Russell. Space-efficient inference in dynamic probabilistic networks. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI 97, Nagoya, Japan, August 23-29, 1997, 2 Volumes*, pages 1292–1296, 1997.

[23] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[24] Jérôme Bolte, Aris Daniilidis, Adrian S. Lewis, and Masahiro Shiota. Clarke subgradients of stratifiable functions. *SIAM Journal on Optimization*, 18(2):556–572, 2007.

[25] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1-2):459–494, 2014.

[26] Pierre Bonami, Mustafa Kilinç, and Jeff Linderoth. *Algorithms and Software for Convex Mixed Integer Nonlinear Programs*, pages 1–39. Springer New York, New York, NY, 2012.

[27] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, pages 177–187, Paris, France, August 2010. Springer.

[28] Patrick Briest, Dimo Brockhoff, Bastian Degener, Matthias Englert, Christian Gunia, Oliver Heering, Thomas Jansen, Michael Leifhelm, Kai Plociennik, Heiko Röglin, Andrea Schweer, Dirk Sudholt, Stefan Tannenbaum, and Ingo Wegener. The Ising model: Simple evolutionary algorithms as adaptation schemes. In *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference, Birmingham, UK, September 18-22, 2004, Proceedings*, pages 31–40, 2004.

[29] Manuel Bronstein. Integration of elementary functions. *Journal of Symbolic Computation*, 9(2):177–173, 1990.

[30] Luigi Bruno and Patrick Robertson. Observability of path loss parameters in wlan-based simultaneous localization and mapping. In *International Conference on Indoor Positioning and Indoor Navigation, IPIN 2013, Montbeliard, France, October 28-31, 2013*, pages 1–10, 2013.

[31] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pages 535–541, 2006.

[32] B. P. Buckles and M. Lybanon. Algorithm 515: Generation of a vector from the lexicographical index [g6]. *ACM Transactions on Mathematical Software*, 3(2):180–182, June 1977.

[33] Andrei Bulatov and Martin Grohe. The complexity of partition functions. In Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, *Automata, Languages and Programming*, volume 3142 of *Lecture Notes in Computer Science*, pages 294–306. Springer, Heidelberg, Germany, 2004.

[34] Tony Cai, Weidong Liu, and Xi Luo. A constrained ℓ_1 minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association*, 106(494):594–607, 2011.

[35] David Campbell. Is it still big data if it fits in my pocket? In *Proceedings of the VLDB Endowment*, volume 4, page 694, 2011.

[36] Wenlin Chen, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2285–2294, 2015.

[37] Wenlin Chen, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. Compressing convolutional neural networks in the frequency domain. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1475–1484, 2016.

[38] Xianda Chen, Kyung Tae Kim, and Hee Yong Youn. Integration of Markov random field with Markov chain for efficient event detection using wireless sensor network. *Computer Networks*, 108:108–119, 2016.

[39] Jungwook Choi and Rob A. Rutenbar. Video-rate stereo matching using Markov random field TRW-S inference on a hybrid CPU+FPGA computing platform. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(2):385–398, 2016.

[40] Li Chou, Somdeb Sarkhel, Nicholas Ruozzi, and Vibhav Gogate. On parameter tying by quantization. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 3241–3247, 2016.

[41] Yuan Shih Chow and Henry Teicher. *Probability Theory.* Springer, New York, USA, 1997.

[42] C. W. Clenshaw and A. R. Curtis. A method for numerical integration on an automatic computer. *Numerische Mathematik*, 2(1):197–205, 1960.

[43] Adam Coates, Brody Huval, Tao Wang, David J. Wu, Bryan Catanzaro, and Andrew Y. Ng. Deep learning with COTS HPC systems. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 1337–1345, 2013.

[44] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

[45] Intel Corporation. Intel 64 and ia-32 architectures optimization reference manual. In *Order Number: 248966-033*, 2016.

[46] Intel Corporation. Intel Xeon Phi processor: Your path to deeper insight. In *Intel Xeon Phi Product Brief*, 2016.

[47] NVIDIA Corporation. NVIDIA DGX-1 deep learning system. In *Nvidia DGX-1 Data Sheet Apr16*, June 2017.

[48] Andrew Cotter, Nathan Srebro, and Joseph Keshet. A GPU-tailored approach for training kernelized SVMs. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 805–813, 2011.

[49] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3123–3131. Curran Associates, Inc., 2015.

[50] Rina Dechter. *Constraint processing*. Elsevier Morgan Kaufmann, 2003.

[51] James Demma and Peter Athanas. A hardware generator for factor graph applications. In *2014 International Conference on ReConFigurable Computing and FPGAs, ReConFig14, Cancun, Mexico, December 8-10, 2014*, pages 1–8, 2014.

[52] Guanzhong Ding, Chung-Ta King, and Yi-Fan Chung. Consumsense: A framework for physical consuming behavior prediction on smartphones. In *19th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2013, Seoul, Korea, December 15-18, 2013*, pages 182–189, 2013.

[53] Trinh Minh Tri Do, Olivier Dousse, Markus Miettinen, and Daniel Gatica-Perez. A probabilistic kernel method for human mobility prediction with smartphones. *Pervasive and Mobile Computing*, 20:13–28, 2015.

[54] Trinh Minh Tri Do and Daniel Gatica-Perez. Human interaction discovery in smartphone proximity networks. *Personal and Ubiquitous Computing*, 17(3):413–431, 2013.

[55] Trinh Minh Tri Do and Daniel Gatica-Perez. Where and what: Using smartphones to predict next locations and applications in daily life. *Pervasive and Mobile Computing*, 12:79–91, 2014.

[56] Bertrand Douillard, Dieter Fox, and Fabio T. Ramos. A spatio-temporal probabilistic model for multi-sensor object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2402–2408, 2007.

[57] Stefan Droste, Thomas Jansen, and Ingo Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276(1-2):51–81, 2002.

[58] Christophe Dumontier, Franck Luthon, and Jean-Pierre Charras. Real-time DSP implementation for MRF-based video motion detection. *IEEE Transactions on Image Processing*, 8(10):1341–1347, 1999.

[59] Mario J. Edmundo. An introduction to o-minimal structures. *ArXiv Mathematics e-prints*, 2000.

[60] Gideon Ehrlich. Loopless algorithms for generating permutations, combinations, and other combinatorial configurations. *Journal of the ACM*, 20(3):500–513, 1973.

[61] Alberto Elfes. Robot navigation: Integrating perception, environmental constraints and task execution within a probabilistic framework. In *Reasoning with Uncertainty in Robotics, International Workshop, RUR '95, Amsterdam, The Netherlands, December 4-6, 1995, Proceedings*, pages 93–130, 1995.

[62] Alberto Elfes, Samuel Siqueira Bueno, Marcel Bergerman, Ely Carneiro de Paiva, Josué Jr. Guimarães Ramos, and José R. Azinheira. Robotic airships for exploration of planetary bodies with an atmosphere: Autonomy challenges. *Auton. Robots*, 14(2-3):147–164, 2003.

[63] Stefano Ermon, Carla P. Gomes, Ashish Sabharwal, and Bart Selman. Taming the curse of dimensionality: Discrete integration by hashing and optimization. In *Proceedings of the 30th International Conference on Machine Learning*, pages 334–342, 2013.

[64] Stefano Ermon, Carla P. Gomes, Ashish Sabharwal, and Bart Selman. Low-density parity constraints for hashing-based discrete integration. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 271–279, 2014.

[65] Sukru Burc Eryilmaz, Emre Neftci, Siddharth Joshi, SangBum Kim, Matthew BrightSky, Hsiang-Lan Lung, Chung Lam, Gert Cauwenberghs, and H.-S. Philip Wong. Training a probabilistic graphical model with resistive switching electronic synapses. *CoRR*, abs/1609.08686, 2016.

[66] Paul Fearnhead. Exact Bayesian curve fitting and signal segmentation. *IEEE Transactions on Signal Processing*, 53(6):2160–2166, 2005.

[67] Thomas S. Ferguson. *A Course in Large Sample Theory.* Chapman and Hall/CRC, 1st edition, 1996.

[68] Natalia Flerova, Emma Rollon, and Rina Dechter. Bucket and mini-bucket schemes for M best solutions over graphical models. In *Graph Structures for Knowledge Representation and Reasoning - Second International Workshop, GKR 2011, Barcelona, Spain, July 16, 2011. Revised Selected Papers*, pages 91–118, 2011.

[69] W. Fraser. A survey of methods of computing minimax and near-minimax polynomial approximations for functions of a single independent variable. *Journal of the ACM*, 12(3):295–314, July 1965.

[70] Jerome Friedman, Trevor. Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[71] Menachem Fromer and Amir Globerson. An LP view of the m-best MAP problem. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada.*, pages 567–575, 2009.

[72] Enrique Garcia-Ceja, Ramón F. Brena, José C. Carrasco-Jiménez, and Leonardo Garrido. Long-term activity recognition from wristwatch accelerometer data. *Sensors*, 14(12):22500–22524, 2014.

[73] W. Gautschi. Questions of numerical condition related to polynomials. *Studies in Numerical Analysis*, (24):140–177, 1985.

[74] Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[75] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 330–339, 2007.

[76] Fabian Gieseke, Justin Heinermann, Cosmin E. Oancea, and Christian Igel. Buffer k-d trees: Processing massive nearest neighbor queries on GPUs. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 172–180, 2014.

[77] Amir Globerson and Tommi S. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 553–560, 2007.

[78] Leslie Ann Goldberg and Mark Jerrum. A polynomial-time algorithm for estimating the partition function of the ferromagnetic Ising model on a regular matroid. *SIAM Journal on Computing*, 42(3):1132–1157, 2013.

[79] Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016.

[80] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1737–1746, 2015.

[81] Seymour Haber. A modified Monte-Carlo quadrature. *Mathematics of Computation*, 20(95):361–368, 1966.

[82] Seymour Haber. A modified Monte-Carlo Quadrature. ii. *Mathematics of Computation*, 21(99):388–397, 1967.

[83] Seymour Haber. Numerical evaluation of multiple integrals. *SIAM Review*, 12(4):481–526, 1970.

[84] John Michael Hammersley and Peter Clifford. Markov fields on finite graphs and lattices. *Unpublished manuscript*, 1971.

[85] Fang Han and Han Liu. Transition matrix estimation in high dimensional time series. In *Proceedings of the 30th International Conference on Machine Learning*, pages 172–180, 2013.

[86] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* Springer, second edition, 2009.

[87] Wilfred Keith Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[88] Sebastian Haug, Andreas Michaels, Peter Biber, and Jörn Ostermann. Plant classification system for crop /weed discrimination without segmentation. In *IEEE Winter Conference on Applications of Computer Vision, Steamboat Springs, CO, USA, March 24-26, 2014*, pages 1142–1149, 2014.

[89] Tamir Hazan and Tommi S. Jaakkola. On the partition function and random maximum a-posteriori perturbations. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.

[90] Tamir Hazan, Subhransu Maji, and Tommi S. Jaakkola. On sampling from the Gibbs distribution with random maximum a-posteriori perturbations. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 1268–1276, 2013.

[91] Pinar Heggernes. Minimal triangulations of graphs: A survey. *Discrete Mathematics*, 306(3):297–317, 2006.

[92] Tom Heskes. Stable fixed points of loopy belief propagation are local minima of the Bethe free energy. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 343–350, 2002.

[93] Sibylle Hess, Katharina Morik, and Nico Piatkowski. The PRIMPING routine - tiling through proximal alternating linearized minimization. *Data Mining and Knowledge Discovery*, 31(4):1090–1131, 2017.

[94] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[95] Otto Ludwig Hölder. Ueber einen Mittelwerthssatz. *Nachrichten von der Königlichen Gesellschaft der Wissenschaften und der Georg-August-Universität Göttingen*, 2:38–47, 1889.

[96] Jean Honorio. Lipschitz parametrization of probabilistic graphical models. In *UAI 2011, Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, Barcelona, Spain, July 14-17, 2011*, pages 347–354, 2011.

[97] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14, Feb 2014.

[98] Lun-Kai Hsu, Tudor Achim, and Stefano Ermon. Tight variational bounds via random projections and i-projections. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016*, pages 1087–1095, 2016.

[99] Rui Huang, Vladimir Pavlovic, and Dimitris Metaxas. A new spatio-temporal MRF framework for video-based object segmentation. In *The 1st International Workshop on Machine Learning for Vision-based Motion Analysis*, 2008.

[100] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4107–4115, 2016.

[101] Timothy Hunter, Pieter Abbeel, and Alexandre M. Bayen. The path inference filter: Model-based low-latency map matching of probe vehicle data. *IEEE Transactions on Intelligent Transportation Systems*, 15(2):507–529, 2014.

[102] Alexander T. Ihler, John W. Fischer III, and Alan S. Willsky. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, 6:905–936, 2005.

[103] Ernst Ising. Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift für Physik*, 31:253–258, 1925.

[104] Tommi S. Jaakkola and Michael I. Jordan. Computing upper and lower bounds on likelihoods in intractable networks. In *Proceedings of the 12th Annual Conference on Uncertainty in Artificial Intelligence*, pages 340–348, 1996.

[105] Ferris Jabr. Does thinking really hard burn more calories? *Scientific American Mind*, July 2012. https://www.scientificamerican.com/article/thinking-hard-calories, accessed at July 10 2017.

[106] Johan Ludwig William Valdemar Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, 30(1):175–193, 1906.

[107] Mark Jerrum and Alistair Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on Computing*, 22(5):1087–1116, 1993.

[108] Mark Jerrum and Alistair Sinclair. Approximation algorithms for NP-hard problems. chapter The Markov Chain Monte Carlo Method: An Approach to Approximate Counting and Integration, pages 482–520. PWS Publishing Co., Boston, MA, USA, 1997.

[109] Aleksandar Jovicic, Ivan Klimek, Cyril Measson, Tom Richardson, and Lei Zhang. Mobile device positioning using learning and cooperation. In *46th Annual Conference on Information Sciences and Systems, CISS 2012, Princeton, NJ, USA, March 21-23, 2012*, pages 1–6, 2012.

[110] Elliott Ward Cheney Jr. *Introduction to Approximation Theory.* Amer Mathematical Society, 2nd edition, 1966.

[111] Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey. *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art.* Springer-Verlag Berlin Heidelberg, 1st edition, 2010.

[112] Vikash K. Mansinghka, Eric M. Jonas, and Joshua B. Tenenbaum. Stochastic digital circuits for probabilistic inference. Technical report, MIT Computer Science and Artificial Intelligence Laboratory, November 2008.

[113] Leslie Pack Kaelbling. *Learning in Embedded Systems.* MIT Press, 1993.

[114] Tomohiro Kawakami. *G-Manifolds and G-Vector Bundles in the Definable Category*, pages –51. Springer Netherlands, New York, NY, 2002.

[115] Brent Keeth, R. Jacob Baker, Brian Johnson, and Feng Lin. *DRAM Circuit Design: Fundamental and High-Speed Topics.* Wiley-IEEE Press, 2nd edition, 2007.

[116] A. H. Khan and E. L. Hines. Integer-weight neural nets. *Electronics Letters*, 30(15):1237–1238, Jul 1994.

[117] Osama Ullah Khan and David D. Wentzloff. Hardware accelerator for probabilistic inference in 65-nm CMOS. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(3):837–845, 2016.

[118] Byoungjip Kim, Seungwoo Kang, Jin-Young Ha, and Junehwa Song. Visitsense: Sensing place visit patterns from ambient radio on smartphones for targeted mobile ads in shopping malls. *Sensors*, 15(7):17274–17299, 2015.

[119] Carolyn Kim, Ashish Sabharwal, and Stefano Ermon. Exact sampling with integer linear programs and random perturbations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 3248–3254, 2016.

[120] Eric P. Kim, Jungwook Choi, Naresh R. Shanbhag, and Rob A. Rutenbar. Error resilient and energy efficient MRF message-passing-based stereo matching. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(3):897–908, 2016.

[121] Mladen Kolar, Le Song, Amr Ahmed, and Eric P Xing. Estimating time-varying networks. *Annals of Applied Statistics*, 4(1):94–123, 2010.

[122] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009.

[123] Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006.

[124] Vladimir Kolmogorov. A new look at reweighted message passing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(5):919–930, 2015.

[125] Vladimir Kolmogorov. A faster approximation algorithm for the Gibbs partition function. *CoRR*, abs/1608.04223, 2016.

[126] Vladimir Kolmogorov and Martin J. Wainwright. On the optimality of tree-reweighted max-product message-passing. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pages 316–323, 2005.

[127] Hikosaburo Komatsu. A characterization of real analytic functions. *Proceedings of the Japan Academy, Ser. A, Mathematical Sciences*, 36(3):90–93, 1960.

[128] Nikos Komodakis and Nikos Paragios. Beyond loose LP-relaxations: Optimizing MRFs by repairing cycles. In *Computer Vision - ECCV 2008, 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part III*, pages 806–820, 2008.

[129] Bernard Osgood Koopman. On distributions admitting a sufficient statistic. *Transactions of the American Mathematical Society*, 39(3):399–409, 1936.

[130] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[131] Krzysztof Kurdyka. On gradients of functions definable in o-minimal structures. *Annales de l'institut Fourier*, 48(3):769–783, 1998.

[132] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, 2001.

[133] Steffen L. Lauritzen. *Graphical Models.* Oxford University Press, Oxford, UK, 1996.

[134] Steffen L. Lauritzen and David J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50(2):157–224, 1988.

[135] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[136] Youngseol Lee and Sung-Bae Cho. Mobile context inference using two-layered Bayesian networks for smartphones. *Expert Syst. Appl.*, 40(11):4333–4345, 2013.

[137] Thomas Liebig, Nico Piatkowski, Christian Bockermann, and Katharina Morik. Dynamic route planning with real-time traffic predictions. *Information Systems*, 64:258–265, 2017.

[138] Darryl Dexu Lin, Sachin S. Talathi, and V. Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 2849–2858, 2016.

[139] Qiang Liu, Jian Peng, Alexander Ihler, and John Fisher III. Estimating the partition function by discriminance sampling. In *31st Annual Conference on Uncertainty in Artificial Intelligence*, pages 514–522. AUAI Press, 2015.

[140] David Luckham. *The Power of Events - An Introduction to Complex Event Processing in Distributed Enterprise Systems.* Addison Wesley, 2002.

[141] Jan R. Magnus and Heinz Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics.* John Wiley &amp; Sons, 2nd edition, 1999.

[142] J.C. Mason and David C. Handscomb. *Chebyshev polynomials.* Chapman and Hall/CRC, 1st edition, 2002.

[143] Michael May and Lorenza Saitta, editors. *Ubiquitous Knowledge Discovery*, volume 6202 of *Lecture Notes in Artificial Intelligence.* Springer, 2010.

[144] Nicolai Meinshausen and Peter Bühlmann. High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, 34(3):1436–1462, 2006.

[145] Talya Meltzer, Chen Yanover, and Yair Weiss. Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation. In *10th IEEE International Conference on Computer Vision (ICCV 2005), 17-20 October 2005, Beijing, China*, pages 428–435, 2005.

[146] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21:1087–1092, 1953.

[147] Stefan Michaelis, Nico Piatkowski, and Katharina Morik. Predicting next network cell ids for moving users with discriminative and generative models. In *Mobile Data Challenge by Nokia Workshop in conjunction with Int. Conf. on Pervasive Computing*, Newcastle, UK, 2012.

[148] Nikita Mishra, Huazhe Zhang, John D. Lafferty, and Henry Hoffmann. A probabilistic graphical model-based approach for minimizing energy under performance constraints. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '15, pages 267–281, New York, NY, USA, 2015. ACM.

[149] Boris S. Mordukhovich. *Variational Analysis and Generalized Differentiation I.* Springer-Verlag Berlin Heidelberg, 1st edition, 2006.

[150] Kevin Patrick Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning.* PhD thesis, University of California, Berkeley, Fall 2002.

[151] Radford M. Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.

[152] Sahand Negahban, Pradeep Ravikumar, Martin J. Wainwright, and Bin Yu. A unified framework for high-dimensional analysis of m-estimators with decomposable regularizers. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada.*, pages 1348–1356, 2009.

[153] Yurii Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

[154] Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

[155] Yurii Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[156] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course.* Springer US, 2013.

[157] Andrew Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In Carla E. Brodley, editor, *Proceedings of the 21st International Conference on Machine Learning*, ICML 2004, pages 78–86. ACM, 2004.

[158] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization.* Springer, New York, NY, USA, 2nd edition, 2006.

[159] E. Novak and K. Petras. Optimal stochastic quadrature formulas for convex functions. *BIT Numerical Mathematics*, 34(2):288–294, 1994.

[160] Yosihiko Ogata and Masaharu Tanemura. Estimation of interaction potentials of spatial point patterns through the maximum likelihood procedure. *Annals of the Institute of Statistical Mathematics*, 33(1):315–338, 1981.

[161] Francesco Orabona, Tamir Hazan, Anand D. Sarwate, and Tommi S. Jaakkola. On measure concentration of random maximum a-posteriori perturbations. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 432–440, 2014.

[162] Margarita Otero. *A survey on groups definable in o-minimal structures*, volume 2 of *London Mathematical Society Lecture Note Series*, pages 177–206. Cambridge University Press, 2008.

[163] George Papandreou and Alan L. Yuille. Perturb-and-MAP random fields: Using discrete optimization to learn and sample from energy models. In *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, pages 193–200, 2011.

[164] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

[165] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, Burlington, MA, USA, 1988.

[166] François Petitjean and Geoffrey I. Webb. Scaling log-linear analysis to datasets with thousands of variables. In *Proceedings of the 2015 SIAM International Conference on Data Mining, Vancouver, BC, Canada, April 30 - May 2, 2015*, pages 469–477, 2015.

[167] François Petitjean and Geoffrey I. Webb. Scalable learning of graphical models. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 2131–2132, 2016.

[168] Nico Piatkowski, Sangkyun Lee, and Katharina Morik. Spatio-temporal models for sustainability. In Manish Marwah, Naren Ramakrishnan, Mario Berges, and Zico Kolter, editors, *Proceedings of the SustKDD Workshop, Conference on Knowledge Discovery and Data Mining (SIGKDD)*. ACM, 2012.

[169] Nico Piatkowski, Sangkyun Lee, and Katharina Morik. Spatio-temporal random fields: Compressible representation and distributed estimation. *Machine Learning*, 93(1):115–139, 2013.

[170] Nico Piatkowski, Sangkyun Lee, and Katharina Morik. The integer approximation of undirected graphical models. In *Proceedings of the 3rd International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, pages 296–304. SciTePress, 2014.

[171] Nico Piatkowski, Sangkyun Lee, and Katharina Morik. Integer undirected graphical models for resource-constrained systems. *Neurocomputing*, 173, Part 1:9–23, 2016.

[172] Nico Piatkowski and Katharina Morik. Stochastic discrete clenshaw-curtis quadrature. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 3000–3009. JMLR.org, 2016.

[173] Nico Piatkowski and François Schnitzler. Compressible reparametrization of time-variant linear dynamical systems. In *Solving Large Scale Learning Tasks. Challenges and Algorithms - Essays Dedicated to Katharina Morik on the Occasion of Her 60th Birthday*, volume 9580 of *Lecture Notes in Computer Science*, pages 234–250. Springer, 2016.

[174] Nico Piatkowski, Jochen Streicher, Spinczyk Olaf, and Katharina Morik. Open smartphone data for structured mobility and utilization analysis in ubiquitous systems. In Martin Atzmueller, Alvin Chin, Christoph Scholz, and Christoph Trattner, editors, *Mining, Modeling, and Recommending 'Things' in Social Media*, volume 8940 of *Lecture Notes in Computer Science*, pages 116–130. Springer, 2015.

[175] Edwin James George Pitman. Sufficient statistics and intrinsic accuracy. *Mathematical Proceedings of the Cambridge Philosophical Society*, 32:567–579, 1936.

[176] David Pollard. *Convergence of Stochastic Processes*. Springer-Verlag New York, 1984.

[177] G. Potamianos and J. Goutsias. Stochastic approximation algorithms for partition function estimation of Gibbs random fields. *IEEE Transactions on Information Theory*, 43(6):1948–1965, 1997.

[178] M. J. D. Powell. On the maximum errors of polynomial approximations defined by interpolation and by least squares criteria. *The Computer Journal*, 9(4):404, 1967.

[179] L. Qian, J. Fuller, and C. Simpson. A community sensing framework for threat detection in metropolitan area. In *2013 IEEE International Conference on Technologies for Homeland Security (HST)*, pages 259–264, Nov 2013.

[180] Pradeep Ravikumar, Martin J. Wainwright, and John D. Lafferty. High-dimensional Ising model selection using ℓ_1-regularized logistic regression. *Annals of Applied Statistics*, 38(3):1287–1319, 2010.

[181] Ralph Tyrrell Rockafellar. *Convex Analysis.* Convex Analysis. Princeton University Press, 1997.

[182] Sergio Rodrigues de Morais and Alex Aussem. A novel scalable and data efficient feature subset selection algorithm. In Walter Daelemans, Bart Goethals, and Katharina Morik, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 5212 of *Lecture Notes in Computer Science*, pages 298–312. Springer Berlin Heidelberg, 2008.

[183] Nicholas Ruozzi. The Bethe partition function of log-supermodular graphical models. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 117–125. Curran Associates, Inc., 2012.

[184] Nicholas Ruozzi. Beyond log-supermodularity: Lower bounds and the Bethe partition function. In *29th Conference Annual Conference on Uncertainty in Artificial Intelligence*, pages 546–555, Corvallis, OR, USA, 2013. AUAI Press.

[185] Sushant Sachdeva and Nisheeth K. Vishnoi. Faster algorithms via approximation theory. *Foundations and Trends in Theoretical Computer Science*, 9(2):125–210, 2014.

[186] Guy Sagy, Daniel Keren, Izchak Sharfman, and Assaf Schuster. Distributed threshold querying of general functions by a difference of monotonic representation. In *Proceedings of the VLDB Endowment*, volume 4, 2011.

[187] Charbel Sakr, Yongjune Kim, and Naresh Shanbhag. Analytical guarantees on numerical precision of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3007–3016, 2017.

[188] Lauren Samy, Paul M. Macey, and Majid Sarrafzadeh. A gender-aware framework for the daytime detection of obstructive sleep apnea. In *37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2015, Milan, Italy, August 25-29, 2015*, pages 7683–7687, 2015.

[189] Mark W. Schmidt, Alexandru Niculescu-Mizil, and Kevin P. Murphy. Learning graphical model structure using l1-regularization paths. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1278–1283, 2007.

[190] François Schnitzler, Alexander Artikis, Matthias Weidlich, Ioannis Boutsis, Thomas Liebig, Nico Piatkowski, Christian Bockermann, Katharina Morik, Vana Kalogeraki, Jakub Marecek, Avigdor Gal, Shie Mannor, Dermot Kinane, and Dimitrios Gunopulos. Heterogeneous stream processing and crowdsourcing for traffic monitoring: Highlights. In Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo, editors, *European Conference on Machine Learning and Knowledge Discovery in Databases, Part III*, volume 8726 of *Lecture Notes in Computer Science*, pages 520–523. Springer, 2014.

[191] Nicol N. Schraudolph and Dmitry Kamenetsky. Efficient exact inference in planar Ising models. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21, pages 1417–1424, 2008.

[192] Glenn Shafer and Prakash P. Shenoy. Probability propagation. *Annals of Mathematics and Artificial Intelligence*, 2:327–351, 1990.

[193] Lufeng Shi and Jindong Tan. Two-tier target tracking framework in distributed sensor networks. *IJSNet*, 16(1):32–40, 2014.

[194] Le Song, Mladen Kolar, and Eric P Xing. Time-varying dynamic Bayesian networks. *Advances in Neural Information Processing Systems*, 22:1732–1740, 2009.

[195] David Sontag, Talya Meltzer, Amir Globerson, Tommi Jaakkola, and Yair Weiss. Tightening LP relaxations for MAP using message passing. In *Proceedings of the Twenty-Fourth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-08)*, pages 503–510, Corvallis, Oregon, 2008. AUAI Press.

[196] Stephan Stilkerich and Joachim K. Anlauf. High-level design environment for massive parallel VLSI-implementations of statistical signal- and image processing models. In *Proceedings of the 2004 International Symposium on Circuits and Systems, ISCAS 2004, Vancouver, BC, Canada, May 23-26, 2004*, pages 37–40, 2004.

[197] Tzu-Pin Sung and Hsin-Mu Tsai. Real-time traffic light recognition on mobile devices with geometry-based filtering. In *Seventh International Conference on*

Distributed Smart Cameras, ICDSC 2013, October 29 2013-November 1, 2013, Palm Springs, CA, USA, pages 1–7, 2013.

[198] Charles Sutton and Andrew McCallum. An introduction to conditional random fields. *Foundations and Trends in Machine Learning*, 4(4):267–373, 2011.

[199] Istvan Szentandrasi, Adam Herout, and Markéta Dubská. Fast detection and recognition of QR codes in high-resolution images. In *Spring Conference on Computer Graphics, SCCG'12, Smolenice, Slovakia, May 2-4, 2012*, pages 129–136, 2012.

[200] Toyokazu Takagi and Tsutomu Maruyama. Accelerating HMMER search using FPGA. In *19th International Conference on Field Programmable Logic and Applications, FPL 2009, August 31 - September 2, 2009, Prague, Czech Republic*, pages 332–337, 2009.

[201] Micron Technology. DDR3L SDRAM, MT41K512M4, MT41K256M8, MT41K128M16. In *2Gb_DDR3L.pdf; Rev.N07/16EN*, 2015.

[202] Micron Technology. DDR4 SDRAM, MT40A2G4, MT40A1G8, MT40A512M16. In *8gb_ddr4_dram.pdf; Rev.G1/17EN*, 2015.

[203] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996.

[204] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society Series B*, 67(1):91–108, 2005.

[205] Ghada Trabelsi, Philippe Leray, Mounir Ben Ayed, and AdelMohamed Alimi. Dynamic mmhc: A local search algorithm for dynamic Bayesian network structure learning. In Allan Tucker, Frank Höppner, Arno Siebes, and Stephen Swift, editors, *Advances in Intelligent Data Analysis XII*, volume 8207 of *Lecture Notes in Computer Science*, pages 392–403. Springer Berlin Heidelberg, 2013.

[206] Lloyd N. Trefethen. Is Gauss quadrature better than Clenshaw-Curtis? *SIAM Review*, 50(1):67–87, 2008.

[207] Jean-Baptiste Tristan, Joseph Tassarotti, and Guy L. Steele Jr. Efficient training of LDA on a GPU by mean-for-mode estimation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 59–68, 2015.

[208] Sebastian Tschiatschek and Franz Pernkopf. On Bayesian network classifiers with reduced precision parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(4):774–785, 2015.

[209] Sebastian Tschiatschek, Peter Reinprecht, Manfred Mücke, and Franz Pernkopf. Bayesian network classifiers with reduced precision parameters. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2012.

[210] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.

[211] Unknown. Artificial intelligence and go—showdown. *The Economist*, March 2016. https://www.economist.com/news/science-and-technology/21694540-win-or-lose-best-five-battle-contest-another-milestone, accessed at July 10 2017.

[212] Leslie Gabriel Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[213] Leslie Gabriel Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47(3):85–93, 1986.

[214] Lou van den Dries, Angus Macintyre, and David Marker. The elementary theory of restricted analytic fields with exponentiation. *Annals of Mathematics, Second Series*, 140(1):183–205, July 1994.

[215] Lou van den Dries, Angus Macintyre, and David Marker. Logarithmic-exponential power series. *Journal of the London Mathematical Society*, 56(3):417–434, 1997.

[216] Lou van den Dries and Chris Miller. On the real exponential field with restricted analytic functions. *Israel Journal of Mathematics*, 85(1):19–56, 1994.

[217] Lou van den Dries and Chris Miller. Geometric categories and o-minimal structures. *Duke Mathematical Journal*, 84(2):497–540, August 1996.

[218] Maarten van der Heijden and Peter J. F. Lucas. Probabilistic models for smart monitoring. In *Proceedings of CBMS 2012, The 25th IEEE International Symposium on Computer-Based Medical Systems, June 20-22, 2012, Rome, Italy*, pages 1–6, 2012.

[219] Fabian Luis Vargas, Rubem Dutra Ribeiro Fagundes, and D. Barros Júnior. A FPGA-based Viterbi algorithm implementation for speech recognition systems. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2001, 7-11 May, 2001, Salt Palace Convention Center, Salt Lake City, Utah, USA, Proceedings*, pages 1217–1220, 2001.

[220] Benjamin Vigoda, David Reynolds, Jeffrey Bernstein, Theophane Weber, and Bill Bradley. Low power logic for statistical inference. In *Proceedings of the 2010 International Symposium on Low Power Electronics and Design, 2010, Austin, Texas, USA, August 18-20, 2010*, pages 349–354, 2010.

[221] Martin J. Wainwright. Estimating the "wrong" graphical model: Benefits in the computation-limited setting. *Journal of Machine Learning Research*, 7:1829–1859, 2006.

[222] Martin J. Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using l1-constrained quadratic programming (lasso). *IEEE Transactions on Information Theory*, 55(5):2183–2202, 2009.

[223] Martin J. Wainwright, Tommi S. Jaakkola, and Alan S. Willsky. Exact MAP estimates by (hyper)tree agreement. In *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pages 809–816, 2002.

[224] Martin J. Wainwright, Tommi S. Jaakkola, and Alan S. Willsky. Tree-based reparameterization for approximate inference on loopy graphs. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1001–1008. MIT Press, Cambridge, MA, USA, 2002.

[225] Martin J. Wainwright, Tommi S. Jaakkola, and Alan S. Willsky. Tree-reweighted belief propagation algorithms and approximate ML estimation by pseudo-moment matching. In Christopher M. Bishop and Brendan J. Frey, editors, *9th Workshop on Artificial Intelligence and Statistics*. Society for Artificial Intelligence and Statistics, Key West, FL, 2003.

[226] Martin J. Wainwright, Tommi S. Jaakkola, and Alan S. Willsky. A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, 51(7):2313–2335, 2005.

[227] Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008.

[228] Di Wang, Elke A. Rundensteiner, and Richard T. Ellison. Active complex event processing of event streams. In *Procs. of the VLDB Endowment*, volume 4, 2011.

[229] Ingo Wegener. *Complexity Theory—Exploring the Limits of Efficient Algorithms*. Springer, 2005.

[230] Yair Weiss. Comparing the mean field method and belief propagation for approximate inference in MRFs. In M. Opper and D. Saad, editors, *Advanced Mean Field Methods:Theory and Practice*, pages 229–239. MIT Press, Cambridge, MA, USA, 2001.

[231] Yair Weiss and William T. Freeman. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47(2):736–744, 2001.

[232] Yair Weiss, Chen Yanover, and Talya Meltzer. MAP estimation, linear programming and belief propagation with convex free energies. In *Proceedings of the Twenty-Third Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-07)*, pages 416–425, Corvallis, Oregon, 2007. AUAI Press.

[233] A. J. Wilkie. Model completeness results for expansions of the ordered field of real numbers by restricted Pfaffian functions and the exponential function. *Journal of the American Mathematical Society*, 9(4):397–421, 1996.

[234] Anja Wille and Peter Bühlmann. Low-order conditional independence graphs for inferring genetic networks. *Statistical Applications in Genetics and Molecular Biology*, 5(1):1–34, 2006.

[235] Ran Wolff, Kanishka Badhuri, and Hillol Kargupta. A generic local algorithm for mining data streams in large distributed systems. *IEEE Transactions on Knowledge and Data Engineering*, 21(4):465–478, 2009.

[236] Eleanor Wong, Suyash Awate, and Thomas Fletcher. Adaptive sparsity in Gaussian graphical models. In *JMLR W&CP 28*, pages 311–319, 2013.

[237] Shuhuang Xiang, Xiaojun Chen, and Haiyong Wang. Error bounds for approximation in Chebyshev points. *Numerische Mathematik*, 116(3):463–491, 2010.

[238] Zhixiang Eddie Xu, Kilian Q. Weinberger, and Olivier Chapelle. The greedy miser: Learning under test-time budgets. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.

[239] Xiang Xuan and Kevin Murphy. Modeling changing dependency structure in multivariate time series. In *Proceedings of the 24th International Conference on Machine Learning*, pages 1055–1062. ACM, 2007.

[240] Eunho Yang, Aurelie C. Lozano, and Pradeep Ravikumar. Elementary estimators for graphical models. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2159–2167, 2014.

[241] Eunho Yang and Pradeep Ravikumar. On the use of variational inference for learning discrete graphical model. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 1009–1016, 2011.

[242] Chen Yanover, Talya Meltzer, and Yair Weiss. Linear programming relaxations and belief propagation - an empirical study. *Journal of Machine Learning Research*, 7:1887–1907, 2006.

[243] Chen Yanover and Yair Weiss. Finding the m most probable configurations using loopy belief propagation. In S. Thrun, L. K. Saul, and P. B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 289–296. MIT Press, 2004.

[244] J. Yarkony, C. Fowlkes, and A. Ihler. Covering trees and lower-bounds on quadratic assignment. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 887–894, June 2010.

[245] Jonathan S Yedidia, William T. Freeman, and Yair Weiss. Generalized belief propagation. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 689–695. MIT Press, 2001.

[246] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Exploring artificial intelligence in the new millennium. chapter Understanding Belief Propagation and its Generalizations, pages 239–269. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[247] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.

[248] Zhaozheng Yin and Robert Collins. Belief propagation in a 3D spatio-temporal MRF for moving object detection. *IEEE Computer Vision and Pattern Recognition*, 2007.

[249] H. Zhang, J. Liu, and N. Kato. Threshold tuning-based wearable sensor fault detection for reliable medical monitoring using Bayesian network model. *IEEE Systems Journal*, PP(99):1–11, 2016.

[250] Shuheng Zhou, John D. Lafferty, and Larry A. Wasserman. Time varying undirected graphs. *Machine Learning*, 80(2–3):295–319, 2010.

[251] Shuheng Zhou, Philipp Rütimann, Min Xu, and Peter Bühlmann. High-dimensional covariance estimation based on Gaussian graphical models. *Journal of Machine Learning Research*, 12:2975–3026, 2011.