March 2019

Learning with Aggregate Data

TAO SUN

LEARNING WITH AGGREGATE DATA

A Dissertation Presented

by

TAO SUN

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2019

College of Information and Computer Sciences

LEARNING WITH AGGREGATE DATA

A Dissertation Presented

by

TAO SUN

Approved as to style and content by:

Daniel R. Sheldon, Chair

David Jensen, Member

Brendan O'Connor, Member

Subhransu Maji, Member

Patrick Flaherty, Member

James Allan, Chair of Department
College of Information and Computer Sciences

ACKNOWLEDGMENTS

I am very fortunate to have been advised by Dan Sheldon. Dan leads me to the broader machine learning research community from my non-CS background. Dan has always been very supportive, encouraging, and understanding. He also holds me to high standards, both academically and morally. I have learned many lessons that will not materialize in this thesis but will guide my day-to-day work in my research position.

I want to express my gratitude to the faculty members I have collaborated with at UMass. Many thanks to Brendan O'Connor and David Jensen, for providing refreshing perspectives on the collaborated projects.

I would also like to thank my committee members for sending me their feedbacks. These inputs help me to position my work better in the specific domains and think from a bigger picture.

I am also very grateful to Tom Dietterich, my academic advisor at Oregon State University (OSU), and Prasad Tadepalli, the graduate student advisor at OSU. Tom showcased to me how to think and do research that has long-term impacts on academics, industry, and the earth. Prasad guided me to discover my research interests.

I would like to individually recognize Steve Li, a fantastic friend, for daily discussions and exchange of ideas that inspired me to build my work habits, technical skills, and research interests. And Liping Liu for collaborating and instructing on my software design, and Xiaojian Wu, Bo Liu for being role models for dedication to research.

MLDS is a great lab that I am proud to be part of. I appreciate the helps and discussions from: Kevin Winner, Garrett Bernstein, Jeffrey Geevarghese, Roy Adams,

Annamalai Natarajan, Steve Li, Juston Moore, Aaron Schein, Dan Sheldon, Ben Marlin, Brendan O'Connor, and Hanna Wallach.

Many thanks to the UMass CICS administrative staff, especially Leeanne Leclerc, and our grant and contract coordinator Michele Roberts, for being so patient and dedicated that make my PhD program running smoothly.

Last, I am tremendously grateful for my family. My parents provided immense help taking care of my pregnant wife and my young daughter during my last months in the PhD program so that I could be absolutely focused. Thank my wife Yanqin for her many years of support and love and bearing with me my many frustrations. Also, thank my lovely daughter Olivia for decorating my everyday lives. Love you all!

ABSTRACT

LEARNING WITH AGGREGATE DATA

FEBRUARY 2019

TAO SUN

B.Sc., ZHENGZHOU UNIVERSITY

M.Sc., TSINGHUA UNIVERSITY

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Daniel R. Sheldon

Various real-world applications involve directly dealing with aggregate data. In this work, we study *Learning with Aggregate Data* from several perspectives and try to address their combinatorial challenges.

At first, we study the problem of learning in *Collective Graphical Models (CGMs)*, where only noisy aggregate observations are available. Inference in CGMs is NP-hard and we proposed an approximate inference algorithm. By solving the inference problems, we are empowered to build large-scale bird migration models, and models for human mobility under the differential privacy setting.

Secondly, we consider problems given bags of instances and bag-level aggregate supervisions. Specifically, we study the US presidential election and try to build a model to understand the voting preferences of either individuals or demographic groups. The data consists of characteristic individuals from the US Census as well as

voting tallies for each voting precinct. We proposed a fully probabilistic *Learning with Label Proportions (LLPs)* model with exact inference to build an instance-level model.

Thirdly, we study *distribution regression*. It has similar problem setting to LLPs but builds bag-level models. We experimentally evaluated different algorithms on three tasks, and identified key factors in problem settings that impact the choice of algorithm.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Aggregate data is ubiquitous in many reseach domains and in different forms. But there is no consensus on which models to use for what types of data. In this thesis, I will present our development of new models and algorithms with real-world use cases, and recommend models that are naturally fit for two typical forms of aggregate data.

The first form of aggregate data we will study are noisy sufficient statistics. In ecological studies, researchers use count statistics (e.g., the number of animals trapped, seen, heard, or otherwise detected) to estimate species abundance (Seber et al., 1973), estimate species distributions (Hefley and Hooten, 2016), infer arrivals, departures, and population size of transient species (Winner et al., 2015), estimate death rates of moths and butterflies (Zonneveld, 1991), reconstruct bird migration routes (Sheldon et al., 2007), etc. These count statistics turn out to be sufficient statistics. In clinical, census, human mobility and other information-sensitive domains, data publishers anonymize their data before release in order to protect individual privacy, the anonymized data is usually in the form of noisy sufficient statistics (Fredrikson et al., 2014; McSherry and Talwar, 2007; de Montjoye et al., 2013).

We recommend solving these problems using Collective Graphical Models (CGMs) (Sheldon and Dietterich, 2011), which directly work on noisy sufficient statistics. In this work, we designed efficient inference algorithms that make large-scale real-world applications possible. We demonstrated the potential of CGMs through two applications: bird migration and collective human mobility.

The second form of aggregate data we will study consists of bags of instances and bag-level aggregate supervisions. Computer vision researchers classify point clouds and each cloud is a set of permutation-invariant points (Qi et al., 2017a). Cosmologists estimate the red-shift of a galaxy cluster based on photometric features of its galaxy members (Connolly et al., 1995; Zaheer et al., 2017). Social scientists try to understand voting preferences of specific demographical groups based on their voting tallies and characteristics of their voters (Flaxman et al., 2015; Sun et al., 2017).

Depending on the intent, a modeler can train either an instance-level or a group-level model given these data. We can train an instance-level model to predict individual voting preference, or predict which part of an object a point belongs to in a point cloud. We can also train a group-level model to estimate the entropy of a bag of samples, or estimate the mass of a galaxy cluster. In this work, we designed a fully probabilistic instance-level model that performs exact marginal inference. We also investigated into how different problem settings impact the choice of a group-level model.

1.1 Synthesis of Different Parts

In this section, we will synthesize the different parts of the thesis, and provide a general guidance on how to choose models given aggregate data of different forms.

Chapter 2 is based on Collective Graphical Models (CGMs) proposed by Sheldon and Dietterich (2011). CGMs are used when individual data is hard to collect because we are unable to identify individuals, or when there are privacy concerns, etc. Instead, noisy aggregate statistics are readily available. We focus on observed noisy *sufficient statistics* of individuals that summarize all their information and have the same data-generating mechanism as individuals. Given these sufficient statistics, we could use CGMs to infer individual behavior such as the transition probability $p_{t,t+1}(A, B)$ of a bird from location A to B from time t to $t + 1$.

Chapter 3 and Chapter 4 deal with a completely different form of aggregate data. They work on bags of instances and have bag-level aggregate supervisions. The setting is similar to standard supervised learning but instead of having one supervision for one data instance, it has one supervision for a bag of instances. This supervision summarizes the property of the entire bag. Chapter 3 falls into the learning framework of Learning with Label Proportions (LLPs), that tries to learn an instance-level model for predicting individual supervisions in a bag. For LLPs, the aggregate supervision is in the form of proportion of instances that belong to each class. In Chapter 3, we work on a close alternative that the aggregate supervision is the count statistic of how many individuals for each class. Our probabilistic LLP model exploits the aggregation mechanism for developing an efficient inference procedure. Chapter 4 falls into the learning framework of distribution regression, that tries to learn a bag-level model for predicting a characteristic of the entire bag. Distribution regression skips the intermediate step of inferring individual supervisions, and is more scalable to large bag sizes. Distribution regression also takes richer forms of aggregate supervision, such as class counts, class labels, proportions, scalars, and vectors. Moreover, new task is flexible in defining its own mechanism of aggregating individual supervisions, such as "logical OR" (as in multi-instance learning), max pooling, etc. Depending on the goal of building either an instance-level model or a bag-level model, the readers may choose LLPs or distribution regression for the task at hand.

Next, we will discuss the assumptions, capabilities and limitations of each model.

CGMs are compact graphical models for the sufficient statistics of a population. The compact graphical representation is derived by marginalizing away all *i.i.d.* individuals. The *i.i.d.* assumption is handy for deriving the objective function that we will optimize, but the assumption itself may not be accurate in practice. For example, a bird rarely migrates alone, so the independence assumption breaks. However, since aggregate data is the first-class citizen in CGMs, the aggregation mechanism cancels

out most randomness in individuals, and massages the non-independence among individuals. Admitting these imperfections, CGMs are straightforward and scalable solutions for learning with noisy aggregate observations.

It is also worth mentioning that the larger the population size is, the more accurate our CGM inference algorithms will be. We empirically demonstrate this in the experiment section. This suggests CGMs will not work well for very small populations, which may not be of concern for most real-world applications.

The theoretical underpinning for LLPs and distribution regression (Szabó et al., 2015) assumes a two-stage sampling procedure: first sample *i.i.d.* distributions for bags $P_b \sim \mathbb{P}$ from a meta distribution $\mathbb{P}$, then sample *i.i.d.* instances for a bag $\mathbf{x}_{bi} \sim P_b$ from the bag's distribution P_b. At test time, we also assume test bags are sampled from the same distribution as training bags, and test instances within a bag are sampled from the same distribution as training instances. In practice, however, these assumptions may be violated. For example, many voting precincts – as training bags – are geographically adjacent and therefore dependent. Moreover, we may want to use distribution regression for predicting voting preferences of demographic subgroups in election (Flaxman et al., 2015; Sun et al., 2017), but "ecological fallacy" may come into play and cause problems. Fallacy means we incorrectly deduce inference of an individual or a sub-population from the inference of the whole population. For our problem, the distribution of test bags of only males will differ from the distribution of training bags of both males and females. Consequently, direct predictions of those test bags based on models learned from a different distribution will be inaccurate. In practice, a test bag should be more similar to the training bags, so that the generalization error is small. Last, the prediction of a highly fine-grained bag may deviate a lot from the ground truth. An example of highly fine-grained bag may include females of age > 65, with high school education, and lives in LA. To the best

of our knowledge, for distribution regression we are unaware of any theoretical analysis about how much the deviation will be from the ground truth.

For LLPs, since we are learning an instance-level model, there is no need to deduce inference of an individual from the inference of population. When we aggregate individual supervisions to obtain an aggregate-level supervision, it may also be interesting to see whether anything occurs at the aggregate level that adds to the understanding of the task. Moreover, LLPs assumes the aggregate supervisions take the form of class counts or class proportions. When this assumption is violated, existing LLP algorithms, including the state-of-the-art Alternating Mean Map (AMM) (Patrini et al., 2014) and our probabilistic LLP (Sun et al., 2017), will not work. We shall resort to alternative models, such as multi-instance learning when the aggregate supervision for a bag is the logical OR of individual labels, or learning with mutual-label constraints when an explicit relationship between individual class labels is provided for each bag. The latter includes, for example, all individuals in a bag have the same categorization (Kumar and Rowley, 2007), or when a bag contains at most one instance from each class (Kuncheva, 2010).

Next, I will describe my contributions of the thesis to each problem setting.

- My contributions to CGMs: I co-formulated (with Dan) the approximate MAP inference of CGMs as an optimization problem, co-proposed (with Dan) a message passing algorithm for the approximate MAP inference, which is more efficient than all alternatives when the work was published. I also proposed a new application: collective human mobility under the differential privacy setting. My work advanced the state-of-the-art by tapping into the real-world sized applications that were previously intractable.

- My contributions to LLPs: I co-formulated (with Dan) the first fully probabilistic version of LLPs that performs exact marginal inference using cardinality

potentials. I applied this algorithm to the 2016 US presidential election for predicting voting preferences of both individuals and demographic subgroups.

- My contributions to distribution regression: I experimentally evaluated different algorithms for distribution regression on three tasks. I identified distribution regression as a two-step procedure: for each bag, first build a bag-level embedding, then regress to bag-level supervision. I also identified three key factors that have major impact on the performances of different algorithms, and suggested the choice of algorithms given different problem settings. From experiment results, I also highlighted the competitiveness of several quantization based approaches, which are largely ignored in the distribution regression literature.

Next, we will discuss aggregate data problems in a big picture and existing studies, and also lay positions of our work in context.

Aggregate data is widely used in many application domains, but to the best of our knowledge, there is not a specialized survey discussing what models are more amenable to what forms of aggregate data. My goal of this thesis is to fill the gap and provide general guidance on how to deal with aggregate data.

Garcıa-Garcıa and Williamson (2011) discussed multi-instance learning and label proportions as two examples in a spectrum of problems of varied "degrees of supervision". Hernández-González et al. (2016) focused on weakly supervised learning (WSL) and incorporated instance-label relationship as one axis of their WSL taxonomy (the other two axes are whether we have supervision in the learning stage and whether we have supervision in the prediction stage.). For the instance-label relationship, they specified three types: single instance with multiple labels (SIML), a set of instances with a single label (MISL), and multi-instance multi-label (MIML). Our basic settings for LLPs and distribution regression are exactly MISL. These works provide perspectives on how to recognize and differentiate aggregate data problems with supervision. But none of them address aggregate count statistics as we did in CGMs. Also, for the

problems with aggregate supervisions, they did not recommend what models are available and suitable for what types of tasks.

There are also rich studies in social, biological, and statistical sciences that focus on meta-analysis of aggregate data. They target summarizing evidence across studies of the same problem to investigate whether there is consistent or inconsistent evidence supporting specific input-output relationships and to examine sources of heterogeneity in those studies that might produce differences in finding (Blettner et al., 1999). In this thesis, we use already prepared aggregate data and we don't have control over how the data is collected and aggregated, such as bird counts and voting tallies.

It is also worth discussing whether ecological fallacy (EF) applies. EF means we incorrectly deduce individual-level inferences from population-level inference. EF does not apply to CGMs. CGM algorithms operate on compact graphical representation of a population. This representation is derived by marginalizing away all individual variables yet keeps the same data-generating mechanism as individuals. Therefore, the two levels of inferences are intrinsically the same thing. For example, the migration probabilities of an individual bird between locations are the same as population-level migration probabilities between the same locations obtained from CGM algorithms. EF does apply to distribution regression but not to LLPs. Distribution regression is a group-level model and its inference does not automatically generalize to individuals. For example, a voting precinct that prefers Democrats to Republican does not necessarily indicate a voter in this precinct has the same voting preference. LLPs is an instance-level model. There is no need to deduce individual-level inferences from population-level inference since individual-level inferences are obtained directly from the model.

Next, we will discuss the sensitivity of each model to violation of its assumptions. This is critical to understanding the limit of each model. The three general assumptions we will consider (but may not apply all to each model) are:

- How sensitive are estimates to spillover (non-*i.i.d.* or non-exchangable instances)?

- Heterogeneity: how sensitive are estimates to heterogeneity in the underlying generative mechanism? How bad can it be before it substantially biases the estimates?

- Measurement error: how sensitive are the estimates to measurement error?

For CGMs, our algorithms work on the sufficient statistics of a population, which would cancel out most randomness in individuals and ease non-independence among individuals. So to some extent it's not very sensitive to spillovers. For sensitivity to heterogeneity in the underlying generative mechanism, at first for our two real-world applications of bird migration and human mobility, we need to choose a spatiotemporal granularity for aggregating individual observations for inferring the migration/commute routes. We could aggregate *many* individuals to move from macro cell A to macro cell B, or we can further partition each macro cell to several smaller cells so that the migration between smaller cells are more detailed and informative. The moves between macro cells A and B would hide differences in migration patterns that the moves between finer cells would otherwise expose. From this perspective, CGMs are more sensitive to heterogeneity in the underlying generative mechanism given finer-grained data aggregation (such as two essentially indistinguishable bird species that migrate with different patterns), and less sensitive to coarse-grained data aggregation. However, coarser data aggregation will introduce more biases from fine-grained ground truths in the estimates, because delicate details are buried in aggregation. For measurement error, we run experiments for a synthetic human mobility task: assuming observations were corrupted by Laplace noises that mimic systematic measurement error, our goal was to recover true move patterns from noisy observations. The results showed that more measurement errors – represented by larger Laplace noises – make it harder to recover ground truth, which is quite intuitive.

For distribution regression, theoretical results assume bags are *i.i.d.* and instances within a bag are *i.i.d.*. In practice, however, these assumptions are usually violated.

For example, contiguous voting regions or family groups are dependent. We can choose data carefully for training models such as using non-contiguous regions for the election task or choosing point clouds from diverse classes for the point cloud classification task. But for some applications we don't have much control: the number of voting regions is only a few and researchers prefer to use them all for training models and for not wasting valuable information; the individuals from the US Census were prepared (by some sampling algorithm) to be representative. One critical step in distribution regression is to compute bag-level embedding vector, and its quality depends heavily on both the embedding algorithm and the representativeness of the input data. When the input data is not representative enough, say, non-*i.i.d.* and only convey partial information of the bag, there is no hope the embedding vector will be accurate. Qi et al. (2017a) also provided important stability analysis for PointNet, intrinsically a distribution regression model for point cloud classification. We can borrow their results for our sensitivity analysis. The architecture of PointNet *max* pools individual embeddings to obtain a point cloud embedding. The authors proved that for every input set S, given max pooling u and a composite function $f = \gamma \circ u$ where γ is a continuous function (treated as a final stage regressor or classifier), there exists a lower bound set C_S and an upper bound set N_S such that $f(T) = f(S)$ if $C_S \subseteq T \subseteq N_S$. In other words, $f(S)$ is unchanged up to input corruption if all points in C_S are preserved; also unchanged with extra noise points up to N_S. The two boundary sets C_S and N_S can be derived given S, and C_S acts like a sparse set of the input S. Similarly, a general distribution regression model with max-pooling is insensitive to spillovers as long as they stay within the boundaries. For some applications, mean-pooling or sum-pooling may work better than max-pooling, but we cannot derive analogous boundary sets C_S, N_S as in the max-pooling scenario. The sensitivity analysis for these cases are still lacking.

LLPs can be treated as a special case of distribution regression with sum pooling, so the sensitivity to spillover is lacking. It also requires the input data to be representative of the underlying distribution.

In the end, we provide future directions for each work.

For CGMs, we identified two future directions: designing more efficient inference algorithms and applying CGMs to real-world applications. For the first direction, Vilnis et al. (2015) discovered the approximate MAP inference in CGMs can be solved by proximal gradient descent based algorithms. We experimentally compared our message passing algorithm to several of them in synthetic experiments, see Section 2.6. The results showed those algorithms converged faster than message passing, but on some parameter settings they converged to poorer values. It is worth understanding the limits and use cases of different algorithms, their convergence properties and assumptions.

The second direction is to apply CGMs to real-world problems. Firstly, we have obtained some preliminary results on constructing bird migration routes for several species. But there are practical challenges we have not completely overcome: for most species the data (bird counts) is sparse and noisy, and the bird totals vary a lot across time while CGMs assume the population size is fixed. To overcome these limitations, we applied Gaussian Process Regression (GPR) as a preprocessing step. GPR helps to interpolate missing values, smooth noisy observations, and maintain the population size constant. But it needs to learn some kernel parameters at first, and the learning is computationally intensive. We need to train GPR for each species and when a new year's worth of data for a species is available we have to retrain the GPR. Moreover, the higher the spatiotemporal resolution we use to build the migration routes (e.g., from weekly to daily), the more expensive the learning. Future work may target building more efficient GP training using sparse or incremental GP, or train the data preprocessing and CGMs in an end-to-end fashion.

The second application domain connects to differential privacy. Bernstein et al. (2017) provided a new approach for differentially private learning of undirected graphical models using CGMs. Their approach consists of a release mechanism and a learning algorithm to collectively ensure privacy. The release mechanism adds Laplace noise – the most common noise mechanism in differential privacy – to the *sufficient statistics* of the *private* individuals. The learning algorithm uses CGMs to work on noisy observations of the sufficient statistics in a principled way. They took initial step towards applying CGMs to a real-world human mobility data, and experimentally demonstrated the advantage of CGMs over alternative private learing algorithms. There could be more work done in this privcy domain, and extend to more practical and large-scale problems.

For LLPs, the challenge to our *exact* inference algorithm is that for large bag-size problems, it is computationally expensive for multiclass problems and numerically unstable when computing messages using FFT due to numerical underflow. Future direction could be to design more efficient, scalable, and stable approximate inference algorithms. One straightforward change to the current EM algorithm could be to use SGD to process one bag at a time and update model parameters immediately after. A second future direction is to design more general instance-level models. They can be designed as a direct generalization of LLPs and inspired by distribution regression. Distribution regression solves $f(X) = \rho(\texttt{pool}_i \phi(x_i))$ where ρ and ϕ are two transformation functions, $\phi(x_i)$ is the local embedding of instance x_i and $\texttt{pool}_i \phi(x_i)$ is the pooled global embedding of the bag. LLPs can be treated as a special case of distribution regression where ϕ infers individual labels, `pool` is the sum pooling, and ρ simply copies the pooled result. A more general instance-level model could be designed to solve $f(X) = \rho(\texttt{concat}(\phi(x_i), \texttt{global}))$ that concatenates both instance-level embedding and global embedding before feeding to ρ. The global embedding could be simply $\texttt{pool}_i \phi(x_i)$, or provided information of the bag, or both. If ρ and

ϕ are two neural networks, this model can be trained end-to-end as performed in DeepSets and PointNet. PointNet has be applied for part segmentation, that requires assigning a label to each point in a point cloud.

For distribution regression, future directions could be to investigate theoretically and experimentally how the number of bags, bag sizes and embedding dimension affect the choice of distribution regression method, and how sensitive the choice of method is to the violation of the *i.i.d.* assumption.

1.2 Published and Non-published Work

My published work

- Approximate inference in Collective Graphical Models.
  [ICML 2013] Daniel Sheldon, Tao Sun, Akshat Kumar, and Thomas Dietterich.
- Message Passing for Collective Graphical Models.
  [ICML 2015] Tao Sun, Dan Sheldon, and Akshat Kumar
- Differentially Private Learning of Undirected Graphical Models Using Collective Graphical Models.
  [ICML 2017] Garrett Bernstein, Ryan McKenna, Tao Sun, Daniel Sheldon, Michael Hay, and Gerome Miklau
- A Probabilistic Approach for Learning with Label Proportions Applied to the US Presidential Election.
  [ICDM 2017] Tao Sun, Dan Sheldon, and Brendan O'Connor

The first three papers are about CGMs. I included the content of the first two in Chapter 2. The last paper is about probabilistic LLPs and is included in Chapter 3.

My un-published work

- A software for real-world bird migration, collaborated with Liping Liu and advised by Dan Sheldon.
- Calibrating moth data: initial experiment design and evaluation.
- Chapter 4: systematic evaluation of several methods for distribution regression on several applications.

1.3 Background for Graphical Models

In this section we will provide general background for graphical models for Chapter 2 and Chapter 3. We will describe the graphical models from the exponential families point of view, and present the fact that the marginal inference can be treated as a variational optimization problem, which is the key for developing the message passing algorithms and the proximal gradient based algorithms for solving the inference problem in Chapter 2.

1.3.1 Exponential Family, Sufficient Statistics, and Marginal Inference

An *exponential family* (Wainwright and Jordan, 2008)is defined as

$$\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{1}{Z_{\boldsymbol{\theta}}} h(\mathbf{x}) \exp(\boldsymbol{\theta}^T \phi(\mathbf{x})) \tag{1.1}$$

where $\boldsymbol{\theta}$ is the vector of *natural parameters*, $\phi(\mathbf{x})$ is the *sufficient statistics*, whose value contains all the information needed to compute any estimator of the parameter vector $\boldsymbol{\theta}$. $h(\mathbf{x})$ is a *base measure* independent of $\boldsymbol{\theta}$, and $Z_{\boldsymbol{\theta}}$ is the *partition function* (normalizer to make $\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{x})$ a distribution). For discrete $\mathbf{x}$

$$Z_{\boldsymbol{\theta}} = \sum_{x \in \mathcal{X}} h(\mathbf{x}) \exp(\boldsymbol{\theta}^T \phi(\mathbf{x})) \tag{1.2}$$

For continuous $\mathbf{x}$, replace the sum with integral. Many popular distributions are in the exponential family, such as the Gaussian, Poisson, Laplace, and Discrete distribution. From now on, we will omit the dependence of $\boldsymbol{\theta}$ for $Z_{\boldsymbol{\theta}}$ and $\mathbb{P}_{\boldsymbol{\theta}}$ to simplify the notation. We will emphasize the dependence of $\boldsymbol{\theta}$ when necessary.

A graphical model can be represented by a graph structure $G = (V, E)$, where V and E are the set of vertices and edges. We also use the index set $\mathcal{I}$ to represent the set of all the (maximal) cliques of the graph.

Marginal inference computes the expected sufficient statistics:

$$\boldsymbol{\mu} := \mathbb{E}_{\mathbb{P}_{\boldsymbol{\theta}}}[\phi(\mathbf{x})] = \nabla_{\boldsymbol{\theta}} \log Z \tag{1.3}$$

When all the sufficient statistics $\{\phi_\alpha, \alpha \in \mathcal{I}\}$ are indicator functions, the marginal of clique α equals to the probability of α: $\mu_\alpha = \mathbb{E}_{\mathbb{P}_{\boldsymbol{\theta}}}[\phi_\alpha(\mathbf{x})] = \mathbb{P}_\alpha$.

The set of marginals $\boldsymbol{\mu}$ is a compact representation of $\mathbb{P}_{\boldsymbol{\theta}}$ and is easier to reason about (Wainwright and Jordan, 2008, Proposition 3.2). Especially, the marginal inference can be cast as a variational optimization problem

$$\boldsymbol{\mu}^\star = \arg\min_{\boldsymbol{\mu} \in \mathcal{M}} -\langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle - \mathbb{H}(\boldsymbol{\mu}) \tag{1.4}$$

where $\mathbb{H}(\boldsymbol{\mu})$ is the entropy of $\boldsymbol{\mu}$ and $\mathcal{M}$ is the standard *marginal polytope*, define as the set of all vectors of expected sufficient statistics $\boldsymbol{\mu}$ that are realizable from some probability distribution

$$\mathcal{M} = \left\{ \boldsymbol{\mu} \in \mathbb{R}^d : \exists p \text{ s.t. } \boldsymbol{\mu} = \mathbb{E}_p[\phi(\mathbf{x})] \text{ for some } p(\mathbf{x}) \geq 0, \sum_{\mathbf{x}} p(\mathbf{x}) = 1 \right\} \tag{1.5}$$

Since $\mathcal{M}$ is exponentially large, it is usually infeasible to optimize over. Instead, we relax $\mathcal{M}$ to the *local polytope* $\mathcal{M}'$ that only satisfies the following *consistency* constraints and $\mathcal{M}'$ is a convex set:

$$\mathcal{M}' = \left\{ \boldsymbol{\mu} \geq 0 : \sum_{x_i} \mu_i(x_i) = 1, \forall i \in V, \sum_{x_j} \mu_{ij}(x_i, x_j) = \mu_i(x_i), \forall (i,j) \in E \right\} \tag{1.6}$$

Define the *Bethe entropy* as

$$\mathbb{H}_{\text{Bethe}}(\mu) = -\sum_{i \in V} (\nu_i - 1) \mathbb{H}_i(\mu_i) + \sum_{(i,j) \in E} \mathbb{H}_{ij}(\mu_{ij}) \tag{1.7}$$

where ν_i is the degree of node i and $\mathbb{H}_i$, and $\mathbb{H}_{ij}$ are the entropies for vertex i and edge ij respectively. The marginal inference given the *Bethe free energy* is

$$\boldsymbol{\mu} = \arg\min_{\mu \in \mathcal{M}'} -\langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle - \mathbb{H}_{\text{Bethe}}(\boldsymbol{\mu}) \tag{1.8}$$

For trees, the Bethe entropy is concave (Heskes, 2006) and $\mathcal{M}' = \mathcal{M}$, so $\boldsymbol{\mu}$ is exact. For cyclic graphs, Bethe entropy is not concave and $\mathcal{M}'$ is approximate, so $\boldsymbol{\mu}$ is an approximate solution.

CHAPTER 2

COLLECTIVE GRAPHICAL MODELS

This chapter focuses on Collective Graphical Models (CGMs), introduced to model the problems of learning and inference about a population of *i.i.d.* individuals when only noisy aggregate observations are available.

We will review the background in Section 2.1, and analyze the complexity of the inference problem in CGMs in Section 2.2: unlike inference in convertional graphical models, exact inference in CGMs is NP-hard even for tree-structured models. We will then develop a tractable convex approximation to the NP-hard MAP inference problem in CGMs in Section 2.3, and show how to use the MAP inference to approximate the *marginal inference* within the EM framwork. We demonstrate empirically that, by using off-the-shelf solvers, these approximations can reduce the computational cost of both the inference and the learning by orders of magnitude, and at the same time provide solutions of equal or better quality than the only existing Gibbs sampling alternative.

The approximate inference using off-the-shelf solvers does not scale to realistic-sized problems. In Section 2.4 we highlight a close connection between the approximate MAP inference in CGMs and the *marginal inference* in standard graphical models. The connection leads to a novel Belief Propagation (BP) style algorithm for CGMs. This BP algorithm exploits the graph structure, that the off-the-shelf solvers ignores, and is orders of magnitude faster. In Section 2.5, we will also discuss approximate MAP inference based on proxmial gradient descent. It is a whole slew of optimization algorithms that is guaranteed to converge and enjoys fast convergence. We will evaluate

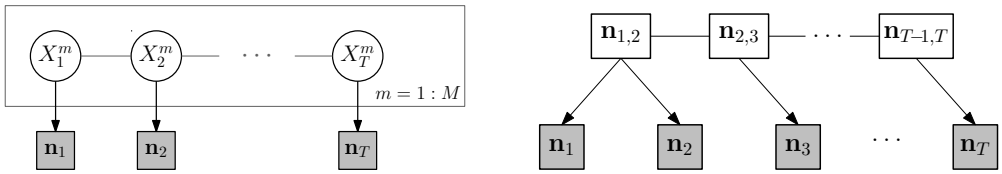


Figure 2.1: CGM example: (a) Individual are explicitly modeled. (b) After marginalization, the latent variables are the sufficient statistics of the individual model.

the aforementioned algorithms in a synthetic bird migration setting. We will also apply these algorithms to a new application: human mobility under the differential privacy setting.

2.1 Introduction to CGMs

2.1.1 Motivation

Sheldon and Dietterich (2011) introduced *Collective Graphical Models* (CGMs) to model the problem of learning and inference with noisy aggregate data. CGMs are motivated by the growing number of applications where data about individuals are not available, but aggregate population-level data in the form of counts or contingency tables are available. For example, the US Census Bureau cannot release individual records for survey effort and privacy reasons, so they commonly release low-dimensional contingency tables that classify individuals according to a few demographic attributes. In ecology, survey data provide counts of animals trapped, seen, heard, or otherwise detected in different locations, but they cannot identify individuals.

CGMs are generative models that serve as a link between individual behavior and aggregate data. As a concrete example, consider the model illustrated in Figure 2.1(a) for modeling bird migration from observational data collected by citizen scientists through the eBird project (Sheldon et al., 2007; Sheldon, 2009; Sullivan et al., 2009). Inside the plate, an independent Markov chain describes the migration of each bird

among a discrete set of locations: X_t^m represents the location of the mth bird at time t. Outside the plate, *aggregate* observations are made about the spatial distribution of the population: the variable $\mathbf{n}_t$ is a vector whose ith entry counts the number of birds in location i at time t. By observing temporal changes in the vectors $\{\mathbf{n}_t\}$, one is possible to infer migratory routes without tracking individual birds.

A key to efficient inference in CGMs is the fact that, when *only* aggregate data are being modeled, the same data-generating mechanism can be described much more compactly by analytically marginalizing away the individual variables to obtain a direct probabilistic model for the sufficient statistics (Sundberg, 1975; Sheldon and Dietterich, 2011). Figure 2.1(b) illustrates the resulting model for the bird migration example. The new latent variables $\mathbf{n}_{t,t+1}$ are tables of sufficient statistics: the entry $n_{t,t+1}(i,j)$ is the number of birds that fly from location i to location j at time t. For large populations, the resulting model is much more amenable to inference, because it has many fewer variables and it retains a graphical structure analogous to that of the individual model. However, the reduction in the number of variables comes at a cost, the new variables are tables of integer counts, which can take on many more values than the original discrete variables in the individual model, and this adversely affects the running time of inference algorithms.

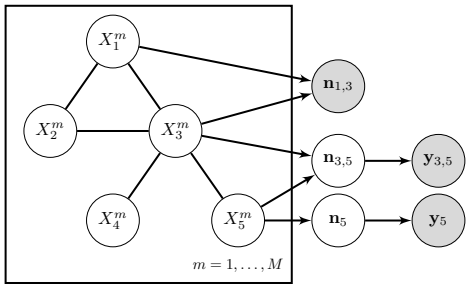


Figure 2.2: General observational model: observe contingency tables on small sets of variables (or plus noise).

In general CGMs, any discrete graphical model can appear inside the plate to model individuals in a population (for example Figure 2.2), and observations are made in the form of (noisy) low-dimensional contingency tables (Sheldon and Dietterich, 2011). A key problem we would like to solve is learning the model parameters (of the individual model) from the aggregate observations, for which inference is the key subroutine. Unfortunately, standard inference techniques applied to CGMs quickly become computationally intractable as the population size increases, due to the large number of latent individual-level variables that are all connected by the aggregate counts.

2.1.2 Problem Statement

In this subsection, we describe the generative model in more details, introduce the CGM distribution, and state the marginal and MAP inference problems for CGMs.

2.1.2.1 Generative Model for Aggregate Data

Let $G = (V, E)$ be an undirected graph with N vertices, and consider the following *pairwise* graphical model over the discrete random vector $\boldsymbol{X} = (X_1, \ldots, X_N)$. Let $\mathbf{x} = (x_1, \ldots, x_N)$ be a particular assignment (for simplicity, assume each x_i takes values in the same finite set $[L] = \{1, \ldots, L\}$). The *individual probability model* is

$$\begin{aligned} p(\mathbf{x}; \boldsymbol{\theta}) &= \frac{1}{Z(\boldsymbol{\theta})} \prod_{(i,j) \in E} \prod_{x_i', x_j'} \phi_{ij}(x_i', x_j'; \boldsymbol{\theta})^{\mathbb{I}[x_i' = x_i, x_j' = x_j]} \\ &= \frac{1}{Z(\boldsymbol{\theta})} \prod_{(i,j) \in E} \phi_{ij}(x_i, x_j; \boldsymbol{\theta}). \end{aligned} \tag{2.1}$$

Here $\mathbb{I}(\cdot)$ is the indicator function and $\phi_{ij} \colon [L]^2 \mapsto \mathbb{R}^+$ are edge potentials defined on edge $(X_i, X_j) \in E$. These local potentials are controlled by a parameter vector $\boldsymbol{\theta}$, and $Z(\boldsymbol{\theta})$ is the partition function. We assume G is a tree. For graphical models that are not trees or have higher-order potentials, our results can be generalized to junction

trees, with the usual blowup in space and running time depending on the clique-width of the junction tree.

To generate the aggregate data, first assume M vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}$ are drawn *i.i.d.* from the individual probability model to represent the individuals in a population. Aggregate observations are then made in the form of contingency tables on small sets of variables, which count the number of times that each possible combination of those variables appears in the populations. Specifically, we define the contingency *node table* $\mathbf{n}_i = (n_i(x_i) : x_i \in [L])$ over nodes of the model and *edge table* $\mathbf{n}_{ij} = (n_{ij}(x_i, x_j)) : x_i, x_j \in [L]$ over edges of the model

$$n_i(x_i) = \sum_{m=1}^{M} \mathbb{I}[X_i^{(m)} = x_i],$$
$$n_{ij}(x_i, x_j) = \sum_{m=1}^{M} \mathbb{I}[X_i^{(m)} = x_i, X_j^{(m)} = x_j].$$

Define $\mathbf{n}$ to be the concatenation of all edge tables $\mathbf{n}_{ij}$ and all node tables $\mathbf{n}_i$. This is a random vector that depends on the entire population.

The joint probability of a population of M individuals is

$$\begin{aligned}
g(\mathbf{n}, \boldsymbol{\theta}) := p(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}; \boldsymbol{\theta}) &= \prod_{m=1}^{M} p(\mathbf{x}^{(m)}; \boldsymbol{\theta}) \\
&= \frac{1}{Z(\boldsymbol{\theta})^M} \prod_{(i,j) \in E} \prod_{x_i', x_j'} \phi_{ij}(x_i', x_j'; \boldsymbol{\theta})^{\sum_{m=1}^{M} \mathbb{I}[x_i' = x_i^{(m)}, x_j' = x_j^{(m)}]} \\
&= \frac{1}{Z(\boldsymbol{\theta})^M} \prod_{(i,j) \in E} \prod_{x_i, x_j} \phi_{ij}(x_i, x_j; \boldsymbol{\theta})^{n_{ij}(x_i, x_j)} \\
&= \exp\{ \sum_{(i,j) \in E} \sum_{x_i, x_j} n_{ij}(x_i, x_j) \log \phi_{ij}(x_i, x_j; \boldsymbol{\theta}) - M \log Z(\boldsymbol{\theta}) \} \\
&= \exp\{ \boldsymbol{\eta}^T \mathbf{n}_{ij} - M \log Z(\boldsymbol{\theta}) \}, \qquad (2.2)
\end{aligned}$$

where $\boldsymbol{\eta} = \{\log \phi_{ij}(x_i, x_j; \boldsymbol{\theta})\}$ is the vector of natural parameters, the edge tables $\mathbf{n}_{ij}$ are the *sufficient statistics* of the individual model and will play a prominent role we will see later on.

In CGMs, one obtains noisy observations $\mathbf{y}$ of some subset of the statistics $\mathbf{n}$ and then seeks to answer queries about the sufficient statistics given $\mathbf{y}$ (e.g., for the purpose of learning the parameters $\boldsymbol{\theta}$) through the conditional (posterior) distribution

$$p(\mathbf{n} \mid \mathbf{y}; \boldsymbol{\theta}) \propto p(\mathbf{n}; \boldsymbol{\theta}) p(\mathbf{y} \mid \mathbf{n}). \tag{2.3}$$

The first term $p(\mathbf{n}; \boldsymbol{\theta})$ is the *CGM distribution*, we will describe how it is derived in section 2.1.2.3. We refer to the second term, $p(\mathbf{y} \mid \mathbf{n})$, as the *noise model* or the *CGM evidence term*. It is often assumed that $p(\mathbf{y} \mid \mathbf{n})$ is log-concave in $\mathbf{n}$, which makes the negative log-likelihood convex in $\mathbf{n}$, though most of results later on do not rely on that assumption.

For node-based noise model, each entry $y_i(x_i)$ has one of the following distributions:

Exact observation: $y_i(x_i) = n_i(x_i)$

Noisy observation: $y_i(x_i) \mid n_i(x_i) \sim \text{Pois}(\alpha \cdot n_i(x_i))$

The Poisson model is motivated by the bird migration problem for count observations, and models birds being counted at a rate proportional to their true density. While it is helpful to focus on these two observation models, considerable variations are possible without significantly changing the results:

1. Observations of the different types can be mixed.

2. Higher-order contingency tables, such as the edge tables $\mathbf{n}_{ij}$, may by observed, either exactly or noisily.

3. Some table entries may be unobserved while others are observed, or, in the noisy model, they may have multiple independent observations.

4. The Poisson model can be replaced by any other noise model, for example, we will use the Laplace noise model for the application of collective human mobility, which is the common noise mechanism for differential privacy.

In Section 2.2, the exact observation model is used to prove the hardness results, while the Poisson model and Laplace model are used in the experiments. Since the exact model can be obtained as the limiting case of a log-concave noise model (e.g., $y \mid n \sim \mathcal{N}(n, \sigma^2)$ as $\sigma^2 \to 0$), we do not expect that noisy observations lead to a more tractable problem.

2.1.2.1.1 Example. For modeling bird migration, assume $X = (X_1, \ldots, X_T)$ is the sequence of discrete locations (e.g., map grid cells) visited by an individual bird, and the graphical model $p(\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{t=1}^{T} \phi_t(x_t, x_{t+1}; \boldsymbol{\theta})$ is a chain model governing the migration of an individual, where the parameter vector $\boldsymbol{\theta}$ controls how different relevant factors (distance, direction, time of year, etc.) influence the affinity $\phi_t(x_t, x_{t+1}; \boldsymbol{\theta})$ between locations x_t and x_{t+1}. In the CGM, M birds of a given species independently migrate from location to location according to the chain model. The node table entries $n_t(x_t)$ indicate how many birds are in location x_t at t. The edge table entries $n_{t,t+1}(x_t, x_{t+1})$ count how many birds move from location x_t to location x_{t+1} from time t to time $t+1$. A reasonable model for eBird data is that the number of birds of the target species counted by a birdwatcher is a Poisson random variable with mean proportional to the true number of birds $n_t(x_t)$, or $y_t(x_t) \mid n_t(x_t) \sim \text{Pois}(\alpha n_t(x_t))$, where α is the detection rate. Given only the noisy eBird counts and the prior specification of the Markov chain, the goal is to answer queries about the distribution $p(\mathbf{n} \mid \mathbf{y}; \boldsymbol{\theta})$ to inform us about migratory transitions made by the population. Because the vector $\mathbf{n}$ consists of sufficient statistics, these queries also provide all the relevant information for learning the parameter $\boldsymbol{\theta}$ from this data.

2.1.2.2 Inference Problems

We wish to learn the parameters $\boldsymbol{\theta}$ of the *individual model* from the aggregate observations $\mathbf{y}$. To do this, we need to know the values of the sufficient statistics of the individual model – namely, $\mathbf{n}_{ij}$ for all edges $(i, j) \in E$. Our observation models do not directly observe these. Fortunately, we can consider $\mathbf{n}$ as a latent variable and apply the standard EM algorithm, in which case we need to know the expected values of the sufficient statistics given the observations: $\mathbb{E}[\mathbf{n}_{ij} \mid \mathbf{y}]$, or more loosely $\mathbb{E}[\mathbf{n} \mid \mathbf{y}]$ (will show in Section 2.3).

This leads us to define two inference problems: marginal inference and MAP inference. The *aggregate marginal inference* problem is to compute the expected (conditional) sufficient statistics $\mathbb{E}[\mathbf{n} \mid \mathbf{y}]$ (see Eq. (1.3)), which are exactly the quantities needed for the E step of the EM algorithm[1].

The *aggregate MAP inference* problem is to find the tables $\mathbf{n}$ that jointly maximize $p(\mathbf{n} \mid \mathbf{y})$. A primary focus of CGMs is approximate algorithms for approximate MAP inference. One reason for conducting MAP inference is the usual one: to reconstruct the most likely values of $\mathbf{n}$ given the evidence as a way of "reconstruction" (e.g., for bird migration, reconstruct the migration routes). However, a second and important motivation is the fact that the *posterior mode* of $p(\mathbf{n} \mid \mathbf{y})$ is an *excellent approximation for the posterior mean* $\mathbb{E}[\mathbf{n} \mid \mathbf{y}]$ in this model (as we will show in Section 2.3), so approximate MAP inference also gives an approximate algorithm for the important marginal inference problem needed for the EM algorithm.

2.1.2.3 CGM Distribution

In the setting we are considering, our observations and queries only concern aggregate quantities. The observations are (noisy) counts and the queries are MAP or

[1]Sheldon and Dietterich (2011) also showed how to generate samples from $p(\mathbf{n}_{ij} \mid \mathbf{y})$, which is an alternative way to query the posterior distributions without storing a huge tabular potential.

marginal probabilities over sufficient statistics (which are also counts). In this setting, we don't care about the values of the individual variables $\left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}\right\}$, so we can marginalize them away. This marginalization can be performed analytically to obtain a probability model with many fewer variables. It results in a model whose random variables are the vector $\mathbf{n}$ of sufficient statistics[2] and the vector $\mathbf{y}$ of observations.

Sundberg (1975) originally described the form of the CGM distribution $p(\mathbf{n}; \boldsymbol{\theta})$ for a graphical model that is decomposible (*i.e.,* its cliques are the nodes of some junction tree), in which case its probabilities can be written in closed form in terms of the marginal probabilities of the original model. For pairwise models, given the marginal probabilities $\mu_i(x_i) = \Pr(X_i = x_i; \boldsymbol{\theta})$ and $\mu_{ij}(x_i, x_j) = \Pr(X_i = x_i, X_j = x_j; \boldsymbol{\theta})$[3]:

$$p(\mathbf{n}; \boldsymbol{\theta}) = M! \prod_{i \in V} \prod_{x_i} \left(\frac{n_i(x_i)!}{\mu_i(x_i)^{n_i(x_i)}} \right)^{\nu_i - 1} \prod_{(i,j) \in E} \prod_{x_i, x_j} \frac{\mu_{ij}(x_i, x_j)^{n_{ij}(x_i, x_j)}}{n_{ij}(x_i, x_j)!} \cdot \mathbb{I}[\mathbf{n} \in \mathbb{L}_M^{\mathbb{Z}}]. \tag{2.4}$$

Here, ν_i is the degree of node i, and the final term $\mathbb{I}[\cdot]$ is a hard constraint that restricts the support of the distribution to vectors $\mathbf{n}$ that are valid sufficient statistics of some ordered sample. Sheldon and Dietterich (2011) showed that, for trees or junction trees, this requirement is satisfied if and only if $\mathbf{n}$ belongs to the *integer-valued scaled local polytope* $\mathbb{L}_M^{\mathbb{Z}}$ defined by the following constraints:

$$\mathbb{L}_M^{\mathbb{Z}} = \Big\{ \mathbf{n} \in \mathbb{Z}_+^{|\mathbf{n}|} \;\Big|\; M = \sum_{x_i} n_i(x_i) \;\forall i \in V, \tag{2.5}$$
$$n_i(x_i) = \sum_{x_j} n_{ij}(x_i, x_j) \;\forall i \in V, x_i \in [L], j \in N(i) \Big\},$$

[2]Note $\mathbf{n}$ is the concatenation of $\mathbf{n}_i$ and $\mathbf{n}_{ij}$ and only $\mathbf{n}_{ij}$ is the sufficient statistics. Since $\mathbf{n}_i$ can be easily derived from $\mathbf{n}_{ij}$, sometimes we loosely claim $\mathbf{n}$ as the sufficient statistics.

[3]If marginal probabilities are not given, they can be computed by performing inference in the individual model.

where $N(i)$ is the set of neighbors of i. The reader will recognize that $\mathbb{L}_M^{\mathbb{Z}}$ is equivalent to the standard local polytope (Eq. (1.6)) of a graphical model except for two differences:

1. The marginals, which in our case represent counts instead of probabilities, are scaled to sum to the population size M instead of summing to one.

2. These counts are constrained to be integers.

The set $\mathbb{L}_M^{\mathbb{Z}}$ is the true support of the CGM distribution. Let $\mathbb{L}_M$ be the relaxation of $\mathbb{L}_M^{\mathbb{Z}}$ obtained by removing the integrality constraint, *i.e.,* the set of real-valued vectors with non-negative entries that satisfy the same constraints.

Liu et al. (2014) refined the result of $p(\mathbf{n};\boldsymbol{\theta})$ above to be written in terms of the original potentials instead of marginal probabilities. Applied to our tree-structured models, this gives the following distribution:

$$p(\mathbf{n};\boldsymbol{\theta}) = M!\frac{\prod_{i\in V}\prod_{x_i}(n_i(x_i)!)^{\nu_i-1}}{\prod_{(i,j)\in E}\prod_{x_i,x_j} n_{ij}(x_i,x_j)!}\cdot g(\mathbf{n},\boldsymbol{\theta})\cdot \mathbb{I}[\mathbf{n}\in\mathbb{L}_M^{\mathbb{Z}}]. \tag{2.6}$$

The first term is a base measure (it does not depend on the parameters) that counts the number of different ordered samples that give rise to the sufficient statistics $\mathbf{n}$. The second term, $g(\mathbf{n},\boldsymbol{\theta})$, is the joint probabilities of *any* ordered sample with sufficient statistics $\mathbf{n}$ as defined in Eq. (2.2).

The distribution $p(\mathbf{n};\boldsymbol{\theta})$ is the *CGM distribution*, which is defined over the random variables $\mathbf{n}_{ij}$ and $\mathbf{n}_i$. The CGM distribution satisfies a *hyper Markov property*: it has conditional independence properties that follow the same essential structure as the original graphical model (Dawid and Lauritzen, 1993). (To see this, note that Eq. (2.4) or Eq. (2.6) factor into separate terms for each node and edge contingency table; when the hard constraints of $\mathbb{L}_M^{\mathbb{Z}}$ are also included as factors, this has the effect of connecting the tables for edges incident on the same node, as illustrated in Figure 2.1.)

We can combine the CGM distribution $p(\mathbf{n};\boldsymbol{\theta})$ (as prior) with the evidence term $p(\mathbf{y} \mid \mathbf{n})$ (as likelihood) to derive an explicit expression for the (unnormalized) posterior $p(\mathbf{n} \mid \mathbf{y};\boldsymbol{\theta}) \propto p(\mathbf{n};\boldsymbol{\theta})p(\mathbf{y} \mid \mathbf{n})$.

2.1.2.4 Generalization

For general graph structures, Sheldon and Dietterich (2011) gave a probability model analogous to Eq. (2.4) defined over a junction tree for the graphical model. In that case, the vector $\mathbf{n}$ includes contingency table $\mathbf{n}_C$ for each clique C of the junction tree. Different junction trees may be chosen for a particular model, which will lead to different definitions of the latent variables $\mathbf{n}$ and thus slightly different inference problems, but always the same marginal distribution $p(\mathbf{y})$ over observed variables. Higher-order contingency tables may be observed as long as each observed table $\mathbf{n}_A$ satisfies $A \subseteq C$ for *some* clique C, so it can be expressed using marginalization constraints such as $\mathbb{L}_M^{\mathbb{Z}}$. The approximate inference algorithms in Section 2.3 extend to these more general models in a straightforward way by making the same two approximations presented in that section for the expression $\log p(\mathbf{n} \mid \mathbf{y};\boldsymbol{\theta})$ to derive a convex optimization problem.

2.2 Computational Complexity

There are a number of natural parameters quantifying the difficulty of inference in CGMs: the population size M; the number of variables N (e.g., $N = T$ for T time steps in the bird migration model); the variable cardinality L (e.g., $L = \ell^2$ for a grid of $\ell \times \ell$ cells for bird migration); and the clique-width K (largest clique size) of the junction tree used to perform inference, which is bounded below by the tree-width of G plus one. The inputs are: the vector $\mathbf{y}$, the integer M, and the CGM distribution. The vector $\mathbf{y}$ has at most NL entries of magnitude $O(M)$, so each can be represented

in $\log M$ bits. The CGM is fully specified by the potential functions in Eq. (2.1), which has size $O(NL^2)$. Thus the input size is $\text{poly}(N, L, \log M)$.

We first describe the best known running time for exact inference in trees ($K = 2$), which are the focus of this paper, and then generalize to running time on junction trees for general graphical models. The proof sketch can be found in Appendix A.1.

Theorem 1. *When G is a tree, message passing in the CGM solves the aggregate MAP or marginal inference problems in time $O(N \cdot \min(M^{L^2-1}, L^{2M}))$.*

Theorem 2. *Message Passing on a junction tree with maximum clique size K and maximum variable cardinality L takes time $O(N \cdot \min(M^{L^K-1}, L^{KM}))$.*

Thus, if either L or M is fixed, message passing runs in time polynomial in the other parameter. When M is constant, then the running time $O(NL^{KM})$ is exponential in the clique-width K, which captures the familiar case of discrete graphical models. When L is constant, however, the running time $O(NM^{L^K-1})$ is not only exponential in L but *doubly-exponential* in the clique-width. Thus, despite being polynomial in one of the parameters, message passing is unlikely to give satisfactory performance on real problems. Finally, the next result tells us that we should not expect to find an algorithm that is polynomial in both parameters.

Theorem 3. *Unless $P = NP$, there is no algorithm for MAP or marginal inference in a CGM that is polynomial in both M and L. This remains true when G is a tree and $N = 4$.*

The proof is by reducing the exact 3-dimensional matching to CGM where both M and L grow with the input size. The detail can be found in Appendix A.1.

2.3 Approximate MAP Inference

In this section, we address the problem of MAP inference in CGMs under the noisy observation model from subsection 2.1.2.1. That is, the node tables for an observed

set are corrupted independently by noise, our goal is to find $\mathbf{n} \in \mathbb{L}_M^{\mathbb{Z}}$ to maximize the following (log-posterior) objective:

$$\log p(\mathbf{n} \mid \mathbf{y}; \boldsymbol{\theta}) = \log p(n; \boldsymbol{\theta}) + \log p(\mathbf{y} \mid \mathbf{n}) + \text{constant}$$

Henceforth, we will suppress the dependence on $\boldsymbol{\theta}$ to simplify notation when discussing inference with respect to fixed parameters $\boldsymbol{\theta}$.

2.3.1 Approximate Objective Function

As highlighted in Section 2.2, it is computationally intractable to directly optimize $\log p(\mathbf{n} \mid \mathbf{y})$. Therefore, we introduce two approximations. First, we relax the feasible set from $\mathbb{L}_M^{\mathbb{Z}}$ to $\mathbb{L}_M$ (*i.e.,* removing the integrality requirement). For large population size M, the effect of allowing fractional values is minimal. Second, as it is hard to incorporate factorial terms $\log n!$ directly into an optimization framework, we employ Stirling's approximation: $\log n! \approx n \log n - n$.

Given the CGM distribution $\log p(\mathbf{n})$, and using these two approximations:

$$\begin{aligned}
\log p(\mathbf{n} \mid \mathbf{y}) &= \sum_i (\nu_i - 1) \sum_{x_i} \log n_i(x_i)! - \sum_{(i,j)} \sum_{x_i, x_j} \log n_{ij}(x_i, x_j)! \\
&\quad + \sum_{(i,j)} \sum_{x_i, x_j} n_{ij}(x_i, x_j) \log \phi_{ij}(x_i, x_j) - M \log Z + \log p(\mathbf{y} \mid \mathbf{n}) + c \\
&\approx \sum_i (\nu_i - 1) \sum_{x_i} n_i(x_i) \log n_i(x_i) - \sum_i (\nu_i - 1) \sum_{x_i} n_i(x_i) \\
&\quad - \sum_{(i,j)} \sum_{x_i, x_j} n_{ij}(x_i, x_j) \log n_{ij}(x_i, x_j) + \sum_{(i,j)} \sum_{x_i, x_j} n_{ij}(x_i, x_j) \\
&\quad + \sum_{(i,j)} \sum_{x_i, x_j} n_{ij}(x_i, x_j) \log \phi_{ij}(x_i, x_j) - M \log Z + \log p(\mathbf{y} \mid \mathbf{n}) + c \\
&= \sum_i (\nu_i - 1) \sum_{x_i} n_i(x_i) \log n_i(x_i) - \sum_{(i,j)} \sum_{x_i, x_j} n_{ij}(x_i, x_j) \log n_{ij}(x_i, x_j) \\
&\quad + \sum_{(i,j)} \sum_{x_i, x_j} n_{ij}(x_i, x_j) \log \phi_{ij}(x_i, x_j) + \log p(\mathbf{y} \mid \mathbf{n}) + c.
\end{aligned}$$

We have absorbed all constants in c. Since our goal is now to find $\mathbf{n} \in \mathbb{L}_M$ to maximize the log-posterior $\log p(\mathbf{n} \mid \mathbf{y}; \boldsymbol{\theta})$, we arrive at the following optimization problem:

$$\min_{\mathbf{z} \in \mathbb{L}_M} F_{\mathrm{CGM}}(\mathbf{z}) := E_{\mathrm{CGM}}(\mathbf{z}) - H_B(\mathbf{z}). \tag{2.7}$$

$$E_{\mathrm{CGM}}(\mathbf{z}) = -\sum_{(i,j) \in E} \sum_{x_i, x_j} z_{ij}(x_i, x_j) \log \phi_{ij}(x_i, x_j) - \log p(\mathbf{y} \mid \mathbf{z}),$$

$$H_B(\mathbf{z}) = -\sum_{(i,j) \in E} \sum_{x_i, x_j} z_{ij}(x_i, x_j) \log z_{ij}(x_i, x_j) + \sum_{i \in V} (\nu_i - 1) \sum_{x_i} z_i(x_i) \log z_i(x_i).$$

We write $\mathbf{z}$ instead of $\mathbf{n}$ to emphasize that the contingency tables are now real-valued.

Theorem 4. *The optimization problem* (2.7) *for approximate MAP inference in tree-structured CGM is convex if and only if* $p(\mathbf{y} \mid \mathbf{z})$ *is log-concave.*

Proof. The quantity $H_B(\mathbf{z})$ is the *Bethe entropy*. It's well known the Bethe entropy is concave over the local polytope of a tree graph (Heskes, 2006). The only difference of H_B here from the conventional Bethe entropy is that the variables $\mathbf{z} \in \mathbb{L}_M$ instead of $\mathbf{z} \in \mathbb{L}_1$, *i.e.,* $\mathbf{z}$ are normalized to sum to M instead of 1, but scaling in this way does not affect concavity. Also the constraints in $\mathbb{L}_M$ are identical to the constraints for pariwise and node marginals used in Bethe entropy (Eq. (1.6)). On the other hand, the *CGM energy function* $E_{\mathrm{CGM}}(\mathbf{z})$ is convex if the noise model $p(\mathbf{y} \mid \mathbf{z})$ is log-concave in $\mathbf{z}$, and therefore F_{CGM} is convex. □

Given this theorem, we can solve our approximate MAP inference using off-the-shelf solvers

2.3.2 MAP Inference for EM

We now describe how the approximate MAP inference problem for CGMs can be used to significantly accelerate the E-step of the EM algorithm for learning the parameters $\boldsymbol{\theta}$ of the *individual model*. Let $\mathbf{x} = (\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(M)})$ be the latent variables for all individuals in the population, and $\mathbf{y}$ be the (noisy) aggregate observations. The

EM algorithm iteratively finds parameters $\boldsymbol{\theta}$ that maximize the following expected log-likelihood:

$$Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{t-1}) = \mathbb{E}_{\mathbf{x}|\mathbf{y};\boldsymbol{\theta}^{t-1}}[\log p(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta})]$$

where $\boldsymbol{\theta}^{t-1}$ denotes the parameters from the previous iteration. When the joint distribution $p(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta})$ is from an exponential family, as in our case, then the problem simplifies to maximizing $\log p(\bar{\mathbf{n}}, \mathbf{y}; \boldsymbol{\theta})$, where $\bar{\mathbf{n}} = \mathbb{E}[\mathbf{n} \mid \mathbf{y}]$ is the expected value of the sufficient statistics $\mathbf{n} = \mathbf{n}(\mathbf{x}, \mathbf{y})$ used to define the model; these are exactly the latent variables in the CGM. In general, this expectation is difficult to compute and requires specialized sampling approaches as in Sheldon and Dietterich (2011).

Instead, we will show that the approximate MAP solution, which approximates the mode of the distribution $p(\mathbf{n} \mid \mathbf{y}; \boldsymbol{\theta})$, is also an excellent approximation for its mean $\mathbb{E}[\mathbf{n} \mid \mathbf{y}]$. While this may seem surprising, recall that the random variables in question take values that are relatively large non-negative integers. A good analogy is the Binomial distribution (a CGM with only one variable), for which the mode is very close to the mean, and the mode of the continuous extension of the Binomial pmf is even closer to the mean. These characteristics make our approach qualitatively very different from a typical "hard EM" algorithm. Our experiments show that the approximate mode $\arg\min_{\mathbf{z} \in \mathbb{L}_M} F_{\mathrm{CGM}}(\mathbf{z})$ can be computed extremely quickly and is an excellent substitute for the mean. It is typically a much *better* approximation of the mean than the one found by Gibbs sampling for reasonable time budgets, and this makes the overall EM procedure many times faster.

2.4 Message Passing for CGMs

We can solve the approximate MAP inference using generic optimization solvers, as described in Section 2.3. However, the generic solvers ignore the structure of a

graphical model, and does not scale to realistic-sized problems. In this section, We will develop a message passing algorithm for approximate MAP inference, that would overcome this limitation. We will start by introducing some related work.

2.4.1 Related Work

Yedidia et al. (2000) published an influential paper showing that the loopy Belief Propagation (BP) algorithm for marginal inference in graphical models can be understood as a fixed-point iteration that attempts to satisfy the first-order optimality conditions of the Bethe free energy, which approximate the true variational free energy. The result shed considerable light on the convergence properties of BP and led to many new ideas for approximate variational inference.

In this section, we highlight a connection between the Bethe free energy and the objective function for approximate MAP inference in CGMs (Section 2.3). We then follow reasoning similar to that of Yedidia et al. (2000) to derive a novel message passing algorithm for CGMs. The algorithm, *Non-Linear energy Belief Propagation* (NLBP), has the interesting property that message passing updates are identical to BP, *with the exception that edge potentials change in each step* based on the gradient of the nonlinear "evidence terms" $\log p(\mathbf{y} \mid \mathbf{n})$ that are present in the CGM objective but not in the Bethe free energy. NLBP is a strict generalization of BP to deal with the presence of these additional nonlinear terms.

2.4.2 Message Passing Algorithm

We will derive an efficient message passing algorithm to solve the approximate MAP optimization problem. We start by comparing the MAP objective for CGMs to the *Bethe free energy* for standard graphical models as follows:

$$F_B(\mathbf{z}) = E_B(\mathbf{z}) - H_B(\mathbf{z}),$$
$$E_B(\mathbf{z}) = -\sum_{(i,j)\in E}\sum_{x_i,x_j} z_{ij}(x_i,x_j)\log\phi_{ij}(x_i,x_j)$$

where $\mathbf{z} \in \mathbb{L}_1$, the standard local polytope, is defined in Eq. (1.6)

The function $F_{\text{CGM}}(\mathbf{z})$ (defined in Eq. (2.7)) and $F_B(\mathbf{z})$ differ only in the energy term: while the standard energy $E_B(\mathbf{z})$ is linear in $\mathbf{z}$, the CGM energy $E_{\text{CGM}}(\mathbf{z})$ is non-linear (but typically convex). In what follows, we will generalize the analysis by Yedidia et al. (2000) of Pearl's classical belief propagation (BP) algorithm (1988) to derive a BP algorithm for arbitrary non-linear energies $E(\mathbf{z})$ such as the one in the CGM MAP objective.

Classical BP maintains a set of messages $\{m_{ij}(x_j)\}$ from nodes to their neighbors, which are updated according to the rule:

$$m_{ij}(x_j) \propto \sum_{x_i} \phi_{ij}(x_i,x_j) \prod_{k\in N(i)\setminus j} m_{ki}(x_i)$$

Upon convergence, the node marginals are $z_i(x_i) \propto \prod_{k\in N(i)} m_{ki}(x_i)$ and the edge marginals are $z_{ij}(x_i,x_j) \propto \phi_{ij}(x_i,x_j)\prod_{k\in N(i)\setminus j} m_{ki}(x_i)\prod_{l\in N(j)\setminus i} m_{lj}(x_j)$, normalized to sum to one. Yedidia et al. (2000) showed that if BP converges, it reaches a zero-gradient point of the Lagrangian of the Bethe free energy with respect to the constraint $\mathbf{z} \in \mathbb{L}_1$, which is the standard local polytope. In practice, if BP converges on a loopy graph, it usually converges to a minimum of the Bethe free energy (Heskes, 2003). For trees, BP always converges to the global minimum and the Bethe free energy is equal to the true variational free energy, so BP is an exact method for marginal inference. For graph with cycles, the Bethe free energy is non-convex and both the Bethe free energy and the constraint set $\mathbb{L}_1$ are approximations of their counterparts in the exact variational inference problem (Wainwright and Jordan, 2008), so loopy BP is an approximate marginal inference method. A key contribution of Yedidia et al.

Algorithm 1: Non-Linear Belief Propagation

Input: Graph $G = (V, E)$, (non-linear) energy function $E(\mathbf{z})$, population size M
Output: Solution to $\min_{\mathbf{z} \in \mathbb{L}_M} E(\mathbf{z}) - H_B(\mathbf{z})$ (upon convergence)

Initialization : $m_{ij}(x_j) = 1$, $\widehat{\phi}_{ij}(x_i, x_j) = \phi_{ij}(x_i, x_j)$,
$z_{ij}(x_i, x_j) \propto \phi_{ij}(x_i, x_j), \ \ \forall (i,j) \in E, x_i, x_j.$

while $\neg$ *converged* **do**

Execute the following updates in any order:

$$\widehat{\phi}_{ij}(x_i, x_j) = \exp\Big\{ - \frac{\partial E(\mathbf{z})}{\partial z_{ij}(x_i, x_j)} \Big\} \tag{2.8}$$

$$m_{ij}(x_j) \propto \sum_{x_i} \widehat{\phi}_{ij}(x_i, x_j) \prod_{k \in N(i) \setminus j} m_{ki}(x_i) \tag{2.9}$$

$$z_{ij}(x_i, x_j) \propto \widehat{\phi}_{ij}(x_i, x_j) \prod_{k \in N(i) \setminus j} m_{ki}(x_i) \prod_{l \in N(j) \setminus i} m_{lj}(x_j) \tag{2.10}$$

end

Extract node marginals: $z_i(x_i) \propto \prod_{k \in N(i)} m_{ki}(x_i)$

(2000) was to reveal the nature of this approximation by its connection to the Bethe free energy.

2.4.2.1 Non-Linear Energy Belief Propagation

We now present a generalized belief propagation algorithm to solve problems in the form of Eq. (2.7):

$$\min_{\mathbf{z} \in \mathbb{L}_M} F(\mathbf{z}) := E(\mathbf{z}) - H_B(\mathbf{z}) \tag{2.11}$$

where the energy function $E(\mathbf{z})$ need not be linear with respect to node and edge marginals $\mathbf{z}$. As with standard BP, we first present the algorithm and then show the connection to the Lagrangian. Algorithm 1 shows Non-Linear Belief Propagation (NLBP). Note that the *only* difference from standard BP is that we replace the edge potential $\phi_{ij}(x_i, x_j)$ by the exponentiated negative gradient of $E(\mathbf{z})$. For the standard linear energy $E_B(\mathbf{z})$, this is always equal to the original edge potential, and we recover standard BP. For non-linear energies, the gradient is not constant with respect to $\mathbf{z}$, so, unlike in standard BP, we must track the value of the marginals $\mathbf{z}$ (normalized to

sum to M) in each iteration so we can use them to update the current edge potentials. Note that the algorithm stores the current edge potential $\hat{\phi}_{ij}$ as separate variables, which is not necessary but will add useful flexibility in ordering updates.

One subtle aspect of NLBP is that the vector $\mathbf{z}$ is overcomplete since edge marginals determine the node marginals, and therefore the gradient of $E(\mathbf{z})$ may depend on details of how the function is defined. For example, the two CGM noise models

$$y_i(x_i) \mid \mathbf{z} \sim \text{Poisson}(\alpha z_i(x_i)),$$
$$y_i(x_i) \mid \mathbf{z} \sim \text{Poisson}(\alpha \sum_{x_j} z_{ij}(x_i, x_j)),$$

give the same distribution over $\mathbf{y}$ but yield log-likelihood functions $\log p(\mathbf{y} \mid \mathbf{z})$ (and thus energy functions $E_{\text{CGM}}(\mathbf{z})$) that differ in their gradient with respect to $\mathbf{z}$. To resolve this ambiguity, we assume the energy function $E(\mathbf{z})$ (and hence the CGM noise model $p(\mathbf{y} \mid \mathbf{z})$) is always written as a function of *only the edge variables* $\{\mathbf{z}_{ij}\}$. This can be considered a non-linear generalization of the standard practice of absorbing unary potentials into binary edge potentials in a graphical model, and explains why only the gradient with respect to edge variables appears in the updates of the algorithm.

Theorem 5. *Suppose the NLBP message passing updates converge and the resulting vector* $\mathbf{z}$ *has strictly positive energies. Then* $\mathbf{z}$ *is a constrained stationary point of* $F(\mathbf{z})$ *in problem* (2.11) *with respect to the set* $\mathbb{L}_M$*. If* G *is a tree and* $E(\mathbf{z})$ *is convex, then* $\mathbf{z}$ *is a global minimum.*

The derivation is relatively straightforward, interested readers could find in Appendix A.1. Note the proof of the theorem does *not* rely on the convexity of the noise term $\log p(\mathbf{y} \mid \mathbf{z})$ except to guarantee that a global minimum is reached in the case of tree-structured models. Also note that NLBP maintains positive marginals as long as the gradient of $E(\mathbf{z})$ is finite (which is analogous to the assumption of positive potentials in the linear case), so the assumption of positivity is not overly

restrictive. Unlike standard BP, which is guaranteed to converge in one pass for trees, in NLBP the edge potentials change with each iteration so it is an open question whether convergence is guaranteed even for trees. In practice, we find it necessary to damp the updates to messages (Heskes, 2003) and marginals $\mathbf{z}$, and that sufficient damping always leads to convergence in our experiments. See Algorithm 2 for details of damping.

2.4.2.2 Edge Evidence vs. Node Evidence

In our applications we consider two primary types of CGM observations, one where noisy edge counts are observed and one where noisy node counts are observed. In both cases, we assume the table entries are corrupted independently by a univariate noise model $p(y \mid z)$:

$$p_{\text{edge}}(\mathbf{y} \mid \mathbf{z}) = \prod_{(i,j) \in E} \prod_{x_i, x_j} p(y_{ij}(x_i, x_j) \mid z_{ij}(x_i, x_j))$$

$$p_{\text{node}}(\mathbf{y} \mid \mathbf{z}) = \prod_{i \in V} \prod_{x_i} p(y_i(x_i) \mid z_i(x_i))$$

The first model occurs in our human mobility application: a data provider wishes to release *sufficient statistics* (edge tables) but must add noise to those statistics to maintain differential privacy. The second model occurs in bird migration: birdwatchers submit counts that provide evidence about the locations of birds at a particular time, and not about the migratory transitions they make.

With noisy edge counts, it is clear how to update the edge potentials within NLBP. Let $\ell(z \mid y) = -\log p(y \mid z)$. Eq. (2.8) becomes

$$\widehat{\phi}_{ij}(x_i, x_j) = \phi_{ij}(x_i, x_j) \exp\left\{\ell'(z_{ij}(x_i, x_j) \mid y_{ij}(x_i, x_j))\right\},$$

where ℓ' is the partial derivative with respect to the marginal. With noisy node counts, we rewrite $p(\mathbf{y} \mid \mathbf{z})$ using only the edge tables. We choose to write $z_i(x_i) =$

Algorithm 2: Feasibility Preserving NLBP

Input same as Algorithm 1, damping parameter $\alpha \geq 0$

Initialization : $\mathbf{z} \leftarrow \text{STANDARD-BP}\big(\{\boldsymbol{\phi}_{ij}\}\big)$

while $\neg$ *converged* **do**

$\quad \widehat{\phi}_{ij}(x_i, x_j) \leftarrow \exp\Big\{ -\dfrac{\partial E(\mathbf{z})}{\partial z_{ij}(x_i, x_j)} \Big\}, \forall (i,j) \in E$

$\quad \mathbf{z}^{\text{new}} \leftarrow \text{STANDARD-BP}\big(\{\widehat{\boldsymbol{\phi}}_{ij}\}\big)$

$\quad \mathbf{z} \leftarrow (1-\alpha)\mathbf{z} + \alpha \mathbf{z}^{\text{new}}$; // damped updates

end

$\frac{1}{\nu_i} \sum_{j \in N(i)} \sum_{x_j} z_{ij}(x_i, x_j)$ as the average of the marginal counts obtained from all incident edge tables. This leads to symmetric updates in Eq. (2.8):

$$\widehat{\phi}_{ij}(x_i, x_j) = \phi_{ij}(x_i, x_j) \exp\left\{ \frac{1}{\nu_i} \ell'(y_i(x_i) \mid z_i(x_i)) + \frac{1}{\nu_j} \ell'(y_j(x_j) \mid z_j(x_j)) \right\}$$

where $z_i(x_i)$ and $z_j(x_j)$ are marginal counts of $\mathbf{z}_{ij}$.

2.4.2.3 Update Schedules and Feasibility Preservation

The NLBP algorithm is a fixed-point iteration that allows updating of edge potentials, messages, and the marginals in any order. We first considered a *naive schedule*, where message updates are sequenced as in standard BP (for trees, in a pass from leaves to root and then back). When message m_{ij} is scheduled for update, the operations are performed in the order listed in Algorithm 1: first the edge potential is updated, then the message is updated, and then all marginals that depend on m_{ij} are updated. Unlike BP, this algorithm does not converge in one round, so the entire process is repeated until convergence. In our initial experiments, we discovered that the naive schedule can take many iterations to achieve a solution that satisfies the consistency constraints among marginals (Eq. (2.5)).

We devised a second *feasibility-preserving schedule* (Algorithm 2) that always maintains feasibility and has the appealing property that it can be implemented as a simple wrapper around the standard BP. This algorithm specializes NLBP to alternate

between two phases. In the first phase, edge potentials are frozen while message and marginals are updated in a full pass through the tree. This is equivalent to one call to the standard BP algorithm, which, for trees, is guaranteed to converge in one pass and return feasible marginals. In the second phase, only edge potentials are updated. Algorithm 2 maintains the property that its current iterate $\mathbf{z}$ is always a convex combination of feasible marginals returned by standard BP, so $\mathbf{z}$ is also feasible.

2.5 Projected Proximal Gradient Descent for CGMs

We will complete our discussion of approximate MAP inference for CGMs following Vilnis et al. (2015). These new algorithms are based on proximal gradient descent for composite optimization, and have theoretical guaranteed convergence. By exploring these algorithms, we wish to inspire new research ideas and directions.

2.5.1 Background on Composite Optimization and Proximal Algorithms

Composite optimization minimizes $h = f + R$, where we have an oracle for minimizing R in closed form (R usually takes the form of a regularizer). The vanilla approach of minimizing h is by *projected proximal gradient*:

$$x_{t+1} = \arg\min_{x \in \mathcal{X}} \langle \nabla f(x_t), x \rangle + \frac{1}{2\eta_t} \|x - x_t\|_2^2 + R(x),$$

given some decreasing sequence of learning rate η_t, and $\mathcal{X}$ is the feasible set.

It is straightforward to generalize to a broader family of optimization by replacing the Euclidean distance $\|\cdot\|_2$ with *any* Bregman divergence (Bregman, 1967), defined as

$$B_\varphi(x, x_0) = \varphi(x) - \varphi(x_0) - \langle \nabla \varphi(x_0), x - x_0 \rangle, \tag{2.12}$$

where φ is the *distance-generating function* and must be strongly-convex.

This family of proximal algorithms is very rich, we will illustrate with *regularized dual averaging* (RDA) (Nesterov, 2009) and *mirror descent* (MD) (Beck and Teboulle, 2003).

The update for RDA is:

$$x_{t+1} = \arg\min_{x \in \mathcal{X}} \langle \bar{g}_t, x \rangle + \frac{\beta_t}{t} \varphi(x) + R(x),$$

where $\bar{g}_t = \frac{1}{t} \sum_{k=1}^{t} \nabla f(x_k)$ and β_t is the learning rate. When R is strongly convex, we can take $\beta_t = 0$ and RDA can be interpreted as using Bregman divergence generated by the strongly convex function $\varphi + R$.

The update for MD is:

$$x_{t+1} = \arg\min_{x \in \mathcal{X}} \langle \nabla f(x_t), x \rangle + \frac{1}{\eta_t} B_\varphi(x, x_t) + R(x),$$

2.5.2 Proximal Algorithms for CGMs

Vilnis et al. (2015) proposed to use composite optimization for solving the augmented marginal inference problem:

$$\boldsymbol{\mu}^* = \arg\min_{\boldsymbol{\mu} \in \mathcal{M}} -H(\boldsymbol{\mu}) - \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle + L_{\boldsymbol{\psi}}(\boldsymbol{\mu}) \tag{2.13}$$

where $\mathcal{M}$ is the standard marginal polytope and $L_{\boldsymbol{\psi}}(\boldsymbol{\mu})$ is some arbitrary function of the marginal vector $\boldsymbol{\mu}$, parameterized by $\boldsymbol{\psi}$. $L_{\boldsymbol{\psi}}(\boldsymbol{\mu})$ could be nonlinear and enforce many types of non-local properties such as each sentence should have at least one verb in an NLP application. $L_{\boldsymbol{\psi}}(\boldsymbol{\mu})$ can be either convex or non-convex.

We can reformulate our approximate MAP inference Eq. (2.7) more compactly as

$$\mathbf{z}^* = \arg\min_{\mathbf{z} \in \mathbb{L}_M} -H_B(\mathbf{z}) - \langle \boldsymbol{\theta}, \mathbf{z} \rangle - \log p(\mathbf{y} \mid \mathbf{z}), \tag{2.14}$$

Algorithm 3: Bethe-RDA

Input: Graph $G = (V, E)$, non-linear term $L(\mathbf{z})$, log-potential $\boldsymbol{\theta}$

Init : $\mathbf{z}_0 \leftarrow \text{STANDARD-BP}(\boldsymbol{\theta})$, $\boldsymbol{\theta}_0 = \boldsymbol{\theta}$

while $\neg$ *converged* **do**

$\beta_t \geq 0$

$\bar{g}_t = \frac{t-1}{t}\bar{g}_{t-1} + \frac{1}{t}\nabla L(\mathbf{z}_t)$

$\boldsymbol{\theta}_t = \boldsymbol{\theta} - \frac{t}{\beta_t + t}\bar{g}_t$

$\mathbf{z}_t \leftarrow \text{STANDARD-BP}(\boldsymbol{\theta}_t)$

end

where $\boldsymbol{\theta} = \{\log \phi_{ij}(x_i, x_j)\}$[4]. It's obvious that Eq. (2.14) has exactly the same format as Eq. (2.13) with $L_{\boldsymbol{\psi}}(\mathbf{z}) = -\log p(\mathbf{y} \mid \mathbf{z})$.

For composite optimization $h = f + R$, define $f(\mathbf{z}) = L_{\boldsymbol{\psi}}(\mathbf{z}) = -\log p(\mathbf{y} \mid \mathbf{z})$ and $R(\mathbf{z}) = -\langle \boldsymbol{\theta}, \mathbf{z} \rangle - H_B(\mathbf{z})$. For trees with n nodes, since $-H_B(\mathbf{z})$ is $\frac{1}{2}(2n-1)^{-2}$-strongly convex with respect to the 2-norm over the interior of the marginal polytope $\mathcal{M} = \mathbb{L}_M$ (Fu et al., 2013), and there exists tractable oracle for minimizing $R(\mathbf{z})$, which is the standard belief propagation (BP) for marginal inference, we can apply RDA with $R(\mathbf{z}) = \varphi(\mathbf{z})$ as follows:

$$\begin{aligned}
\mathbf{z}_{t+1} &= \arg\min_{\mathbf{z} \in \mathbb{L}_M} \langle \bar{g}_t, \mathbf{z} \rangle - \frac{\beta_t}{t}\left(H_B(\mathbf{z}) + \langle \boldsymbol{\theta}, \mathbf{z} \rangle\right) - H_B(\mathbf{z}) - \langle \boldsymbol{\theta}, \mathbf{z} \rangle \\
&= \arg\min_{\mathbf{z} \in \mathbb{L}_M} \langle \bar{g}_t - \frac{\beta_t + t}{t}\boldsymbol{\theta}, \mathbf{z} \rangle - \frac{\beta_t + t}{t} H_B(\mathbf{z}). \\
&= \arg\min_{\mathbf{z} \in \mathbb{L}_M} -\left\langle \boldsymbol{\theta} - \frac{t}{\beta_t + t}\bar{g}_t, \mathbf{z} \right\rangle - H_B(\mathbf{z}) \qquad (2.15)
\end{aligned}$$

where $\bar{g}_t = \frac{1}{t}\sum_{k=1}^{t} \nabla L_{\boldsymbol{\psi}}(\mathbf{z}_k)$. The update is equivalent to marginal inference with new parameters $\boldsymbol{\theta}_t = \boldsymbol{\theta} - \frac{t}{\beta_t + t}\bar{g}_t$ and could be solved by standard BP oracle. Algorithm 3 presents the Bethe-RDA (Vilnis et al., 2015) algorithm. Since $R(\mathbf{z})$ is strongly convex, the convergence rate is $O(\ln(t)/t)$ (Xiao, 2010).

[4]With slight abuse of notation, we use $\boldsymbol{\theta}$ for all log potentials. The potentials can be parametric in themselves as well, *i.e.,* $\phi_{ij}(x_i, x_j) = \phi_{ij}(x_i, x_j; \boldsymbol{\theta}')$.

We leave in Appendix A.2 the derivation of Mirror Descent and Accelerated RDA of , the former converges in $O(\ln(t)/t)$ and the latter converges in $O(1/t^2)$.

2.5.2.1 Discussion

The work by Vilnis et al. (2015) assumes $\nabla L(\mathbf{z})$ exists, but it could be the cases that $L(\mathbf{z})$ is non-smooth and/or non-differentiable. For example, the application of human mobility corrupts the sufficient statistics with Laplace noise, and therefore $L(\mathbf{z}) = -\log p(\mathbf{y} \mid \mathbf{z})$ is non-differentiable around 0. Alternative algorithms should be developed for these cases, such as following the work by Nesterov (2005) that addresses smooth minimization of non-smooth functions, or simply we use sub-gradients instead.

2.6 Evaluation

In this section, we first illustrate on a benchmark bird migration problem (1) the accuracy and speed advantage of approximate MAP inference for CGMs compared to the exact inference, and (2) to what extent NLBP and proximal algorithms could accelerate the CGM inference and learning. Then, we demonstrate the benefits of NLBP by evaluating on a new application: collective human mobility. We simulate a task where a data provider wishes to release data about human mobility, and anonymize the data with Laplace noise to preserve individual privacy.

2.6.1 Bird Migration

In this subsection, we run a sequence of evaluations to demonstrate

1. exact MAP inference is intractable even for small problem instances (illustrated with chain graph and very small population size), while approximate MAP inference using generic convex optimization solvers approximates the exact solutions very closely and is orders of magnitude faster;

2. the generic convex optimization solver for approximate MAP inference is also orders of magnitude faster than the previous only alternative, the Gibbs sampler for the *marginal inference*;

3. the extent to which NLBP and Proximal methods accelerate CGM inference and learning.

For all experiments, we generated data from a chain-structured CGM to simulate wind-dependent migration of a population of M birds from the bottom-left to the top-right corner of an $\ell \times \ell$ grid to mimic the seasonal movement of a migratory songbird from a known winter range to a known breeding range. Thus, the variables X_t of the individual model are the grid locations of the individual birds at times $t = 1, \cdots, T$, and have cardinality $L = \ell^2$. The transition probabilities between grid cells were determined by a log-linear model with four parameters that control the effect of features: the distance from x_t to x_{t+1}, the consistency of the transition direction with wind direction, the consistency of the transition direction with the intended destination, and the preference to move. The parameters $\boldsymbol{\theta}_{\text{true}}$ were selected manually to generate realistic migration trajectories. After generating data for a population of M birds, we computed node contingency tables and generated observations from the Poisson noise model $y \sim \text{Poisson}(\alpha n)$ ($\alpha = 1$) for every node. For the experiments that do not involve learning, we perform inference in the same model used to generate the data – that is, the marginal probabilities $\mu_{ij}(\cdot, \cdot)$ and $\mu_i(\cdot)$ in the CGM are those determined by $\boldsymbol{\theta}_{\text{true}}$.

2.6.1.1 Accuracy of Approximate MAP Solutions

We solved the approximate MAP convex optimization problem using MATLAB's interior point solver.

To evaluate the impact of the two approximations in our approximate MAP algorithm, we first compare its solution $\mathbf{n}^*_{\text{approx}}$ to the exact solution $\mathbf{n}^*_{\text{exact}}$ obtained

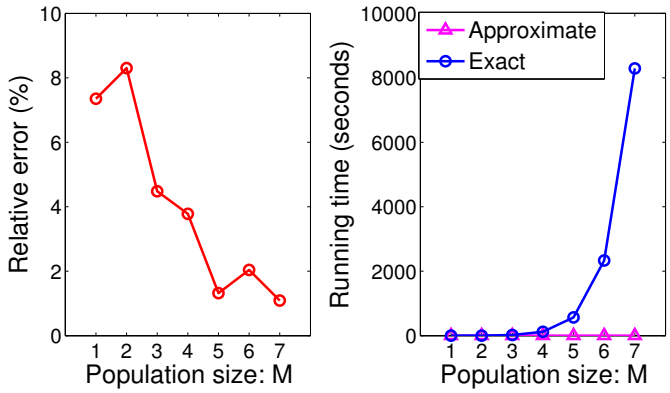


Figure 2.3: The effect of population size M on accuracy and running time of approximate MAP inference ($L = 4, T = 6$). Left: relative error vs. M. Right: running time vs. M.

by message passing for small models ($L = 4, T = 6$). We expect the fractional relaxation and Stirling's approximation to be more accurate as M increases. By Theorem 1, the running time of message passing in this model is $O(M^{15})$, we are limited to tiny populations. Nevertheless, Figure 2.3 shows that the relative error $\|\mathbf{n}^*_{\text{approx}} - \mathbf{n}^*_{\text{exact}}\|_1 / \|\mathbf{n}^*_{\text{exact}}\|_1$ is already less than 1% for $M = 7$. For all M, approximate MAP takes less than 0.2 seconds, while the running time of exact MAP scales very poorly.

2.6.1.2 Marginal Inference

We next evaluated the approximate MAP inference algorithm to show that it can solve the EM marginal inference problem more quickly and accurately than Gibbs sampling. For these experiments, we fixed $M = 1000$, $T = 20$ and increased the grid size L. The largest models ($L = 19^2$) result in a latent vector with $(T-1)L^2 \approx 2.5$ million entries. The goal is to approximate $\mathbb{E}[\mathbf{n} \mid \mathbf{y}]$. Since we cannot compute the exact answer for non-trivial problems, we run ten very long runs of Gibbs sampling (10 million iterations), and then compare each Gibbs run, as well as the approximate MAP, to the reference solution obtained by averaging the nine remaining Gibbs runs; this yields ten evaluations for each method.

Table 2.1: Comparison of Gibbs vs. MAP: seconds to achieve the same relative error compared to the reference solution.

L	9	16	25	36	49
MAP time	0.9	1.9	3.4	9.7	17.2
Gibbs time	161.8	251.6	354.0	768.1	1115.5

To speedup Gibbs sampling, we developed an optimized C implementation of the algorithm of (Sheldon and Dietterich, 2011) and developed an adaptive rejection sampler for discrete distributions to perform the log-concave sampling by that algorithm (Gilks and Wild, 1992; Sheldon, 2013).

The optimization solver quickly finds optimal solution to the approximate MAP problem. Table 2.1 shows the time the solver takes, with varying problem size L, compared to the reference solution: Gibbs consistently takes 50 to 100 times longer to find a solution as close to the reference solution as the one found by MAP.

We conjecture that the approximate MAP solution may be *extremely close to the ground truth*. Taking $L = 49$ for example, each Gibbs solution has a relative difference of about 0.09 from the reference solution computed using the other nine Gibbs runs, suggesting there is still substantial error in the Gibbs runs after 10 million iterations. Furthermore, each time we increase the number of Gibbs iterations used to compute a reference solution, the MAP relative error decreased, suggesting the reference solution was getting closer to the MAP solution.

Based on these observations, from now on, we will take the solution of the generic convex optimization solver as the baseline for EM marginal inference and evaluate new approximate MAP inference algorithms against it.

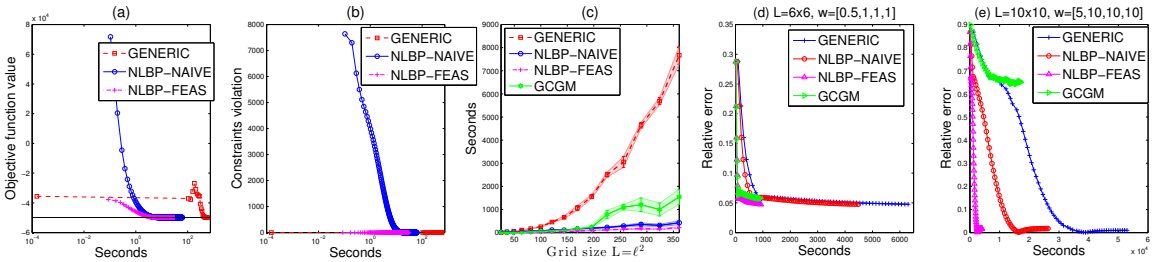


Figure 2.4: **Inference**: Comparison of approximate MAP inference algorithms on 15×15 grid: (a) convergence of objective function, (b) convergence of constraint violation, (c) running time vs. number of grid cells (shaded error bars are 95% confidence intervals computed from 15 repeated trials). **Learning**: (d–e) relative error vs. runtime (seconds) for 60 EM iterations and 3 instances; (d) grid size 6×6 and $\boldsymbol{\theta}_{true} = [0.5, 1, 1, 1]$, (e) grid size 10×10 and $\boldsymbol{\theta}_{true} = [5, 10, 10, 10]$.

2.6.1.3 Approximate MAP inference using NLBP

We compared the speed and accuracy of NLBP and proximal algorithms both as standalone inference methods and as subroutines for learning vs. the generic solver baseline and inference in the Gaussian approximation of CGMs (Liu et al., 2014). We report results for $\boldsymbol{\theta}_{\text{true}} = (5, 10, 10, 10)$. The results for other parameter settings (e.g., those from Liu et al. (2014)) were very similar.

We compare MATLAB's interior point solver (GENERIC), NLBP with the naive message passing schedule (NLBP-NAIVE), feasibility-preserving NLBP (NLBP-FEAS), and the Gaussian approximation (GCGM) for performing inference in CGMs.

Figure 2.4(a–b) show the convergence behavior of the first three algorithms, which solve the same approximate MAP problem, in terms of both objective function and constraint violation for $L = 15^2$. The objective values of the two NLBP algorithms converge to the optimum an order of magnitude more quickly than the generic solver. Both GENERIC and NLBP-FEAS maintain feasibility, but NLBP-NAIVE takes a long time to achieve feasibility – much longer than it does to converge to the optimal objective value. Since GCGM takes a different approach to inference, we do not evaluate it directly in terms of the objective and constraints of the approximate MAP problem. However, we note that when either the grid size or parameter values are large, GCGM

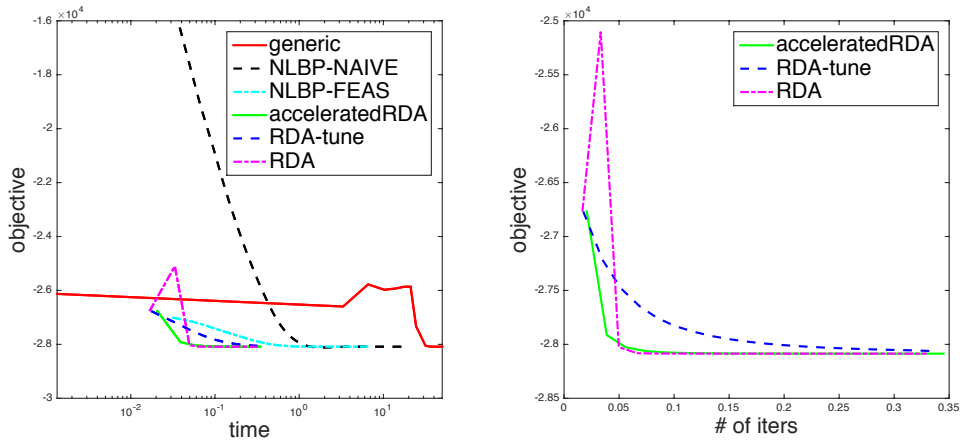


Figure 2.5: **Inference**: Compare approximate MAP inference algorithms on 10×10 grid: (left) convergence of objective function, (right) details for proximal algorithms.

produces marginals that violate the consistency constraints, which may explain why it has difficulty in parameter learning in these cases (see below).

Figure 2.4(c) shows the total time to convergence as a function of problem size for all four algorithms. Both NLBP variants are very efficient and their running time scale much better than that of GENERIC. NLBP-FEAS is approximately twice as fast as NLBP-NAIVE, and is approximately four times faster than GCGM.

2.6.1.4 Approximate MAP inference using Proximal algorithms

On the same benchmark, we evaluated the performance of several proximal algorithms for approximate MAP inference.

We chosen to use three variants of regularized dual averaging (RDA). Since the distance-generating function φ is strongly convex, we can set $\beta_t = 0$ (RDA) or not (RDA-TUNE).

Figures 2.5 (left) shows the convergence behavior of all algorithms for a $L = 10 \times 10$ grid. NLBP converges to the optimal solution an order of magnitude faster than the generic solver, and RDA and its variants converge an order of magnitude faster than NLBP.

Figure 2.5(right) shows the details of the three RDA algorithms, note the learning rate parameter β_t may lead to slower convergence. For accelerated RDA, since the algorithm "damps" the marginals and emphasizes more recent gradients, it does not overshoot and converges the fastest.

2.6.1.5 Learning

Finally, we evaluated approximate MAP as a substitute for marginal inference within the full EM learning procedure. We initialized the parameter vector $\boldsymbol{\theta}$ randomly and then ran the EM algorithm using different inference routines in the E step. In the M step, we applied a gradient-based solver to update the parameters $\boldsymbol{\theta}$ of the log-linear model for transition probabilities. For each algorithm we measured the relative error of the learned parameter vectors from $\boldsymbol{\theta}_{\text{true}}$, defined as $\|\boldsymbol{\theta}^{(t)} - \boldsymbol{\theta}_{\text{true}}\|_1 / \|\boldsymbol{\theta}_{\text{true}}\|_1$, where $\boldsymbol{\theta}^{(t)}$ are the parameters from the t-th EM iteration.

Sheldon et al. (2013) demonstrated that approximate MAP using off-the-shelf solver is an excellent substitute for marginal inference with the EM algorithm, both in terms of accuracy and running time.

We generated data by fixing parameters $\boldsymbol{\theta}_{\text{true}}$ and generating three independent realizations of the entire bird migration process ($T = 20$) to simulate observing the seasonal migration of the same species across three different years. Each realization had different wind covariates, and was treated within EM as an independent data instance. For EM details, see Sheldon et al. (2013); Liu et al. (2014)

Figure 2.4(d–e) show the reduction in error over 60 EM iterations for each algorithm on 6×6 and 10×10 grids. The results confirm the speed advantages of NLBP over the generic solver. All algorithms converge to a similar level of error, except for GCGM in the largest grid size and parameter setting, which is consistent with the results for inference. Both NLBP variants converge much more quickly than GENERIC. The speed

advantage of NLBP-FEAS over NLBP-NAIVE is even greater within the EM procedure. GCGM is only competitive for the setting with small grid size and parameter values.

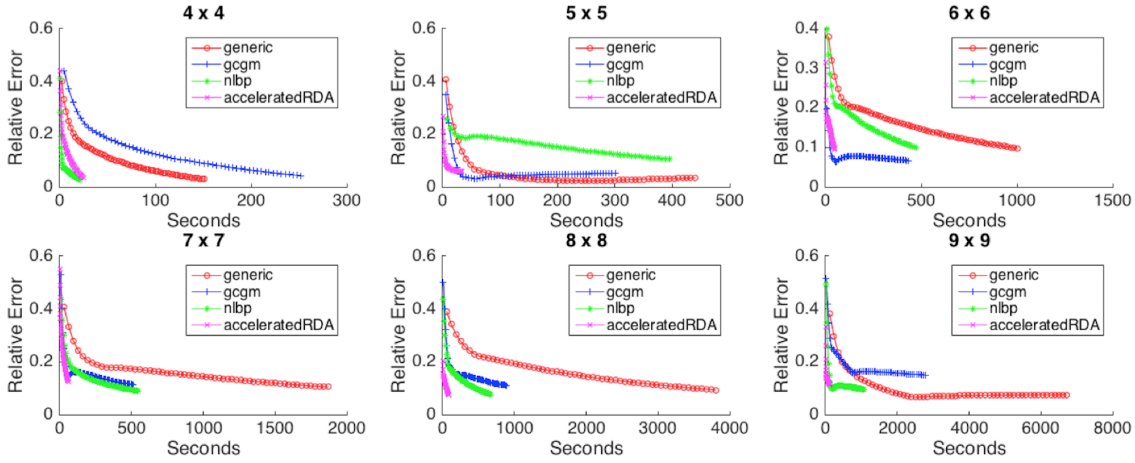


Figure 2.6: **Learning**: Relative error vs. runtime. $\boldsymbol{\theta}_{\text{true}} = [0.5, 1, 1, 1]$.

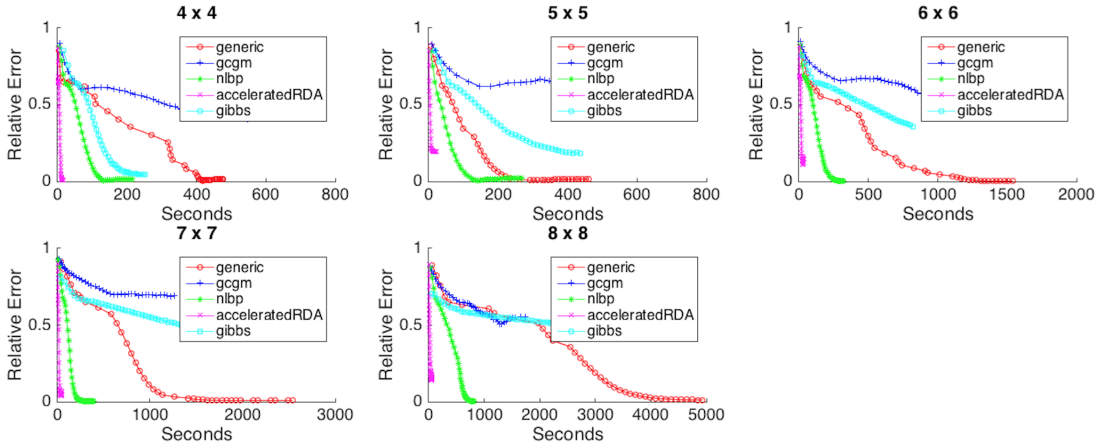


Figure 2.7: **Learning**: Relative error vs. runtime. $\boldsymbol{\theta}_{\text{true}} = [5, 10, 10, 10]$.

Figures 2.6 and 2.7 show the reduction in relative error in 40 EM iterations on a 4×4 grid and a 9×9 grid. Here we encompass more algorithms for comparison. We only showed one proximal algorithm, the accelerated RDA, which performs the best in the inference experiments. The NLBP in the figure is the feasibility-preserving version of NLBP. Both NLBP and accelerated RDA converge much faster than the

generic solver. As before we considered two parameter settings. In the large parameter setting, NLBP converges to lower level of error than the accelerated RDA, which is not fully understood yet.

In summary, feasibility-preserving NLBP is the best performer so far. We will use it as a strong baseline in new applications.

2.6.2 Human Mobility

We now turn to a novel application of CGMs. We address the problem of learning the parameters of a chain-structured graphical model for human mobility, where, unlike the bird migration model, we have access to *transition counts* (edge counts) instead of node counts. Transition counts are sufficient statistics for the model, so learning with *exact* transition counts would be straightforward. However, we assume the available data are corrupted by noise to maintain privacy of individuals. The problem becomes one of learning with noisy sufficient statistics.

In particular, our application simulates the following situation: a mobile phone carrier uses phone records to collect information about the transitions of people among a discrete set of regions, for example, the areas closest to each mobile tower, which form a Voronoi tesselation of space (Song et al., 2010; de Montjoye et al., 2013). Data is aggregated into discrete time steps to provide hourly counts of the number of people that move between each pair of regions. The provider wishes to release this aggregate data to inform public policy and scientific research about human mobility. However, to maintain the privacy of their customers, they choose to release data in a way that maintains the privacy guarantees of *differential privacy* (Dwork and Roth, 2013; Mir et al., 2013). In particular, they follow the Laplace mechanism and add independent Laplace noise to each aggregate count (Dwork and Roth, 2013).

2.6.2.1 Ground-Truth Model

We are interested in fitting models of daily commuting patterns from aggregate data of this form. We formulate a synthetic version of this problem where people migrate among the grid cells of a 15×14 rectangular map. We simulate movement from home destinations to work destinations across a period of $T = 10$ time steps (e.g., half-hour periods covering the period from 6:00 a.m. to 11:00 a.m.). We parameterize the joint probability of the movement sequence for each individual as:

$$p(\mathbf{x}_{1:10}) = \frac{1}{Z} \cdot \phi_1(x_1) \cdot \left(\prod_{t=1}^{9} \psi(x_t, x_{t+1}) \right) \cdot \phi_{10}(x_{10}).$$

The potentials ϕ_1 and ϕ_{10} represent preferences for home and work locations, respectively, while ψ is a pairwise potential that scores transitions as more or less preferred. For the ground truth model, we use compact parameterizations for each potential: ϕ_1 and ϕ_{10} are discretized Gaussian potentials (that is, $\phi(x_t)$ is the value of a Gaussian density over the map measured at the center of grid cell x_t) centered around a "residential area" (top right of the map) and "commercial area" (bottom left). For the transition potential, we set $\phi(x_t, x_{t+1})$ proportional to $\exp\big(-||v_t - v_{t+1}||^2/(2\sigma^2)\big)$, where v_t and v_{t+1} are the centers of grid cells x_t and x_{t+1}, to prefer short transitions over long ones.

2.6.2.2 Data Generation

To generate data, we simulated $M = 1$ million trajectories from the ground truth model, computed the true transition counts, and then added independent Laplace noise to each true count n to generate the noisy count y. The Laplace noise is controlled by a scale parameter b:

$$p(y \mid n) = \text{Laplace}(b; n) = \frac{1}{2b} \exp\left\{ -\frac{|y - n|}{b} \right\}.$$

To explore the relative power of edge counts versus node counts for model fitting, we also performed a version of the experiments where we marginalized the noisy transition counts to give only noisy node counts $y_t(x_t) = \sum_{x_{t+1}} y_{t,t+1}(x_t, x_{t+1})$ as evidence.

2.6.2.3 Parameters and Evaluation

We wish to compare the abilities of CGM-based algorithms and a baseline algorithm to recover the true mobility model. When *fitting* models, it would be a severe oversimplification to assume the simple parametric form used to generate data. Instead, we use a fully parameterized model with parameters $\boldsymbol{\theta} = (\log \phi_1, \log \phi_{10}, \log \psi)$. Here $\log \phi_1$ and $\log \phi_{10}$ are arbitrary $L \times 1$ vectors, and $\log \psi$ is an arbitrary $L \times L$ table. Note that this parameterization is over-complete, and hence not identifiable. To evaluate fitted models, we will compare their *pairwise marginal distributions* to those of the ground truth model: unlike the potentials, the pairwise marginals uniquely identify the joint distribution. The pairwise MAE is defined as the mean absolute error among all $L^2 \times (T-1)$ entries of the pairwise marginals. We also considered node MAE, which is the mean error among the $L \times T$ entries of the node marginals. Note that these *do not* uniquely identify the distribution, but node MAE is an interesting metric for comparing the ability to learn with node evidence vs. edge evidence.

2.6.2.4 Algorithms

Is it possible to estimate parameters of a graphical model given only noisy sufficient statistics? An "obvious" approach is to ignore the noise and perform maximum-likelihood estimation using the noisy sufficient statistics $\mathbf{y}$ in place of the true ones $\mathbf{n}$. To the best of our knowledge, this is the only previously available approach, and we use it as a baseline. The approach has been criticized in the context of general multidimensional contingency tables (Yang et al., 2012). To maximize the likelihood with respect to our parameters, we use a gradient-based optimizer with message passing

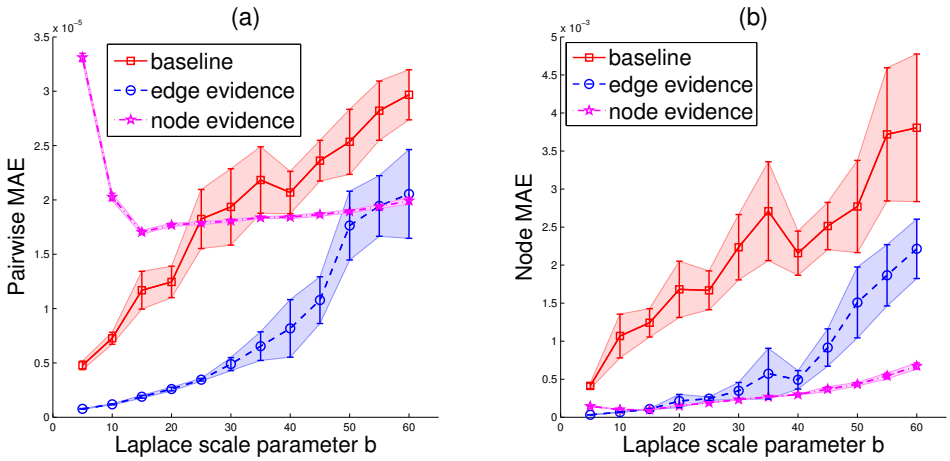


Figure 2.8: Pairwise / Node MAE vs Laplace scale parameter b after 250 EM iterations. Shaded regions shows 95% confidence intervals computed from 10 trials for each setting.

as a subroutine to compute the likelihood and its gradient (Koller and Friedman, 2009).

For the CGM-based approach, we treat the true sufficient statistics as hidden variables and use EM to maximize the likelihood. The overall EM approach is the same as in the bird migration model. When the evidence is noisy edge counts, we first run the baseline algorithm and use those parameters to initialize EM. When the evidence is noisy node counts, the baseline algorithm does not apply and we initialize the parameters randomly.

2.6.2.5 Results

Figure 2.8(a) shows the quality of the fitted models (measured by pairwise MAE) vs. the scale of the Laplace noise. For the CGM-based algorithms, we ran 250 EM iterations, which was enough for convergence in almost all cases. Initializing EM with the baseline parameters helped achieve faster convergence (not shown). The results demonstrate that the CGM algorithm with edge evidence improves significantly over the baseline for all values of b. As expected, the node evidence version of the CGM algorithm performs worse, since it has access to less information. However, it is

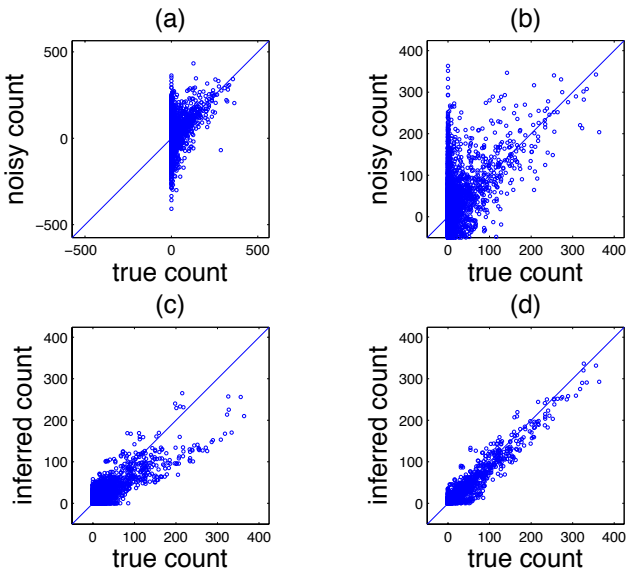


Figure 2.9: Scatter plots of approximate vs. true edge counts for a small problem ($L = 4 \times 7, T = 5, M = 10000, b = 50$): (a) original noisy edge counts, (b) shown only in the same range as (c-d) for better comparison, (c) reconstructed counts after 1 EM iteration, (d) reconstructed counts after EM convergence.

interesting that the CGM with only node evidence outperforms the baseline (which has access to more information) for larger values of b.

Figure 2.8(b) shows *node* MAE vs b for the same fitted models. In other words, it measures the ability of the methods to find models that match the ground truth on single time-step marginals. We see that both CGM algorithms are substantially better than the baseline, and the CGM algorithm with *less information* (node counts only) performs slightly better. We interpret this as follows: node evidence alone provides enough information to match the ground truth model on node marginals; the additional information of the noisy edge counts helps narrow the model choices to one that also matches the ground truth edge marginals. However, this does not explain why the node evidence performs *better* than edge evidence for node MAE. We leave a deeper investigation of this for future work—it may be a form of implicit regularization.

Figure 2.9 provides some insight into the EM algorithm and it's ability to reconstruct edge counts. The original, noisy counts have considerable noise and sometimes take negative values (panels (a) and (b)). After one EM iteration (panel (c)), the reconstructed counts are now feasible, so they can no longer be negative, and they are closer to the original counts. After EM converges, the reconstructed counts are much more accurate (panel (d)).

2.7 Conclusion

In this section, we presented the hardness results and approximate algorithms for the problems of MAP inference and marginal inference in collective graphical models (CGMs). We showed that exact inference by message passing runs in time that is polynomial either in the population size or the cardinality of the variables, but there is no algorithm that is polynomial in both of these parameters unless P=NP. We then showed that the MAP problem can be formulated approximately as a non-linear convex optimization problem. Next we highlighted a close connection between the approximate MAP inference in CGMs and marginal inference in standard graphical models. Inspired by this connection, we derived the non-linear belief propagation (NLBP) algorithm and presented a feasibility-preserving version of NLBP that can be implemented as a simple wrapper around standard BP. In the end, we discussed the proximal gradient descent-based algorithms for approximate MAP inference, inspired by (Vilnis et al., 2015). These algorithms are also simple wrappers around standard BP but enjoy faster convergence, they provide opportunities for designing faster and convergent inference algorithms for CGMs.

Empirically, we first demonstrated that the approximate MAP inference is very accurate even for modest population sizes and that approximate MAP inference is an excellent substitute for marginal inference for computing the E step of the EM algorithm. Our approximate MAP inference algorithm leads to a learning procedure

that is much more accurate and runs in a fraction of the time of the previous only known alternative, the Gibbs sampling algorithm.

By applying NLBP and proximal algorithms to a synthetic benchmark problem for bird migration modeling, we showed that NLBP runs significantly faster than a generic convex optimization solver and is significantly more accurate than inference in the Gaussian approximation of CGMs when the grid size or parameter values are large. The feasibility-preserving version of NLBP is twice as fast as the naive NLBP. We also demonstrated the proximal algorithms run an order of magnitude faster than NLBP, but converge to an inferior level of relative errors than NLBP in the learning problem with large parameter setting, which calls for future examination. In the end, we demonstrated the utility of the NLBP algorithm by contributing a novel application of CGMs for modeling human mobility. In this application, CGMs provide a way to fit graphical models when the available sufficient statistics have been corrupted by noise to maintain the privacy of individuals.

CHAPTER 3

LEARNING WITH LABEL PROPORTIONS (LLPS)

Chapter 3 and 4 study a new class of Learning with Aggregate Data problems, which has the basic setting that we have many "bags" of instances, as well as bag-level aggregate supervisions. It is different from standard supervised learning problems that every instance has its own supervision. Based on specific applications, we may want to develop either bag-level models or instance-level models. The former has been developed as distribution regression, multi-instance learning, learning from sets, ecological inference, etc. For the latter, which is the focus of this chapter, we will study Learning with Label Proportions and apply it to the election problem.

Ecological inference (EI) is a classical problem from political science to model voting behavior of individuals given only aggregate election results. Flaxman et al. (2015) recently solved EI using distribution regression, and applied it to analyze US presidential elections. However, their distribution regression unnecessarily aggregates individual-level covariates available from census microdata (Flaxman et al., 2016), and ignores known structure of the aggregation mechanism. We instead formulate the problem as *learning with label proportions* (LLP), and develop a new, probabilistic, LLP method to solve it. Our model treats individual votes as latent variables and uses *cardinality potentials* to efficiently perform exact inference over latent variables during learning, We also introduce a novel message-passing algorithm to extend cardinality potentials to multivariate probability models for use within multiclass LLP problems. We show experimentally that LLP outperforms distribution regression for predicting

individual-level attributes, and that our method is as good as or better than existing state-of-the-art LLP methods.

3.1 Introduction

Ecological inference (EI) is the problem of making inferences about individuals from aggregate data (King, 2013). EI originates in political science, where its history is closely intertwined with the specific application of inferring voting behavior of individuals or demographic groups from vote totals for different regions. EI maps onto a growing number of problem settings within machine learning—including distribution regression (Flaxman et al., 2015; Szabó et al., 2016), optimal transport (Muzellec et al., 2017), learning with label proportions (Kück and de Freitas, 2005; Yu et al., 2013; Patrini et al., 2014), multiple-instance learning (Dietterich et al., 1997; Hajimirsadeghi and Mori, 2016), and collective graphical models (Sheldon et al., 2013; Sheldon and Dietterich, 2011; Sun et al., 2015) – where one wishes to perform supervised learning, but supervision is only available at an aggregate level, e.g., as summary statistics for "bags" of instances.

We consider the classical EI problem of analyzing voting behavior, motivated in particular by US presidential elections. Although there has been vigorous historical debate about the inherent limitations of EI (King, 1999, 2013; Freedman et al., 1998; Schuessler, 1999), work in machine learning makes it clear that, if one is careful to state assumptions and goals clearly, it is indeed possible to learn individual-level models from aggregate data (Szabó et al., 2016; Bernstein and Sheldon, 2016; Patrini et al., 2014), at least asymptotically. One must still be careful to map these results back to the application at hand; for example, a typical result may be that it is possible to consistently estimate the parameters of an individual-level model given enough group-level observations. This would not imply the ability to infer the outcome or behavior of a single individual.

We will focus on the EI voting problem formulated in (Flaxman et al., 2015, 2016), where a collection of individual-level demographic covariates is available for each geographical region in addition to the region-level voting totals. In the US, individual-level demographic data for different regions is readily available from the US Census Bureau (ACS). In (Flaxman et al., 2015, 2016), the EI problem is then formulated as *distribution regression* (aka *learning from distributions*), where a function is learned to map directly from the distribution of covariates within each region to voting proportions. This is accomplished by creating kernel mean embedding vectors for each region, and learning a standard regression model to map mean embedding vectors to voting proprortions. Theoretical results about distribution regression support the ability of such an approach to correctly learn a model to make region-level predictions for new regions, assuming they are distributed in the same way as the regions used to train the regression model (Szabó et al., 2016).

We argue that EI is more naturally modeled as a learning with label proportions (LLP) problem. Distribution regression treats voting proportions as generic labels associated with the covariate distributions for each region, which ignores a great deal of information about the problem. First, we know the precise aggregation mechanism: there is one vote per individual and these votes are added to get the totals. Second, in solving the problem, distribution regression unnecessarily aggregates the information we *do* know about individuals (the covariates) by constructing a mean embedding. In contrast, LLP acknowledges that the voting proportions come from counting the number of times each label appears in a bag of instances. It is able to use individual covariates and reason relative to the actual aggregation mechanism.

We posit a simple and natural probabilistic latent variable model for EI that places it within an LLP framework. In our model, each individual possesses an observed covariate vector $\mathbf{x}_i$ and an unobserved label y_i (the candidate they voted for), and, within each region, the total number of votes for each candidate is obtained by aggregating the

y_i values. We then learn a logistic regression model mapping directly from covariates to individual-level labels. Because the individual labels are unobserved at training time, we use expectation maximization (EM) for learning. The key computational challenge is therefore to perform inference over the y_i values given the totals. We show that techniques for graphical models with *counting potentials* (Gupta et al., 2007; Tarlow et al., 2012) solve this problem exactly and efficiently. Furthermore, we develop a novel message-passing algorithm to extend counting potentials to multivariate probability models, and thus multiclass classifiction. The result is the the first direct maximum-likelihood approach to LLP based on the "obvious" latent variable model.

To evaluate different EI methods, we design a realistic testbed for designing synthetic problems that mimic the voting prediction problem. We use geographic boundaries and individual-level covariates that match those used in analysis of the US presidential elections. We then design a variety of synthetic tasks where we withhold one covariate and treat this as the variable to be predicted. At training time, only aggregated per-region counts are provided for the withheld variable. Within this framework we control factors such as the number of individuals per region and the number of classes to obtain a variety of realistic EI tasks. For the task of learning models to make individual-level predictions, we show that LLP methods significantly outperform distribution regression, and that our fully probabilistic approach to LLP outperforms other existing state-of-the-art methods. We also assess the ability of different LLP methods as well as distribution regression to predict the voting behavior of different demographic groups in the 2016 US Presidential Election by making predictions using EI and then comparing the results with exit poll data. We find that EI methods do better on qualitative tasks, such as ordering subgroups by their level of support for Hillary Clinton, than they do in predicting precise voting percentages.

3.2 Related Work

LLP has been applied to many practical applications such as object recognition (Kück and de Freitas, 2005), ice-water classification (Li and Taylor, 2015), fraud detection (Rueping, 2010), and embryo selection (Hernández-González et al., 2013).

Early approaches to LLP do not atempt to infer individual labels: the Mean Map approach of Quadrianto et al. (2009) directly estimates the sufficient statistics of each bag by solving a linear system of equations. The sufficient statistics summarize the information from each bag that is relevant for estimating model parameters. The Inverse Calibration method of Rueping (2010) treats the mean of each bag as a "super-instance" (similar to the kernel *mean embedding* used in the distribution regression approach to EI (Flaxman et al., 2015)) and treats label proportions for each bag as target variables within a variant of Support Vector Regression. In contrast, our work explicitly models individual labels and the structural dependency between individual labels and their aggregate class counts.

Several recent LLP approaches reason explicitly about individual labels, but not in a fully probabilistic manner. Stolpe and Morik (2011) first cluster training data given label proportions, and classify new instance using either the closest cluster label, or a new classifier trained from cluster-predicted training data labels. Alter-$\propto$SVM (Yu et al., 2013) poses a joint large-margin optimization problem over individual labels and model parameters, and solves it using alternating optimization. One step in the alternating optimization imputes individual labels.

Alternating Mean Map (AMM) (Patrini et al., 2014) and Alter-CNN (Li and Taylor, 2015) alternate between inferring individual lables and updating model parameters. However, all of these approaches infer "hard" labels for each instance (either 0 or 1). Alter-CNN is formulated probabilistically, but uses "Hard"-EM for learning, which is a heuristic approximate version of EM. In contrast, our method is conventional

maximum-likelihood estimation in the straightforward probability model, and we conduct marginal inference instead of MAP inference over missing individual labels.

Several other papers formulate probabilistic models for LLP, but, unlike our method, resort to some form of approximate inference, such as "hard"-EM (Li and Taylor, 2015) or MCMC (Kück and de Freitas, 2005; Hernández-González et al., 2013). Hernández-González et al. (2013) also propose an EM approach with exact probability calculations in the E step, but using brute-force algorithms that do not scale beyond very small problems; for larger problems they instead resort to approximate inference. In contrast to all previous work, we apply exact and efficient inference algorithms using counting potentials (Gupta et al., 2007; Tarlow et al., 2012) to directly maximize the likelihood in the natural probability model.

One of the latest and closest work to ours is by (Rosenman and Viswanathan, 2018). This work also studies individual voting preference, given both aggregate count data and individual-level characteristics. The major difference from ours is that they model the count data by Poisson Binomial (PB), which is the sum of independent and *not* identically distributed Bernoulli random variables. By applying the Central Limit Theorem (CLT) to PB, they can write the objective and gradient explicitly and run gradient descent. On the contrary, we don't have the PB assumption and don't have closed form log-likelihood of the aggregate counts, and we resort to EM.

3.3 Background and Problem Statement

We now formally introduce the ecological inference problem and describe how it fits within an LLP context. Recall that we assume the availability of individual-level covariates and region-level voting totals for each voting region. In this section, we restrict our attention to binary prediction problems, i.e., we assume there are only two voting options (e.g., candidates from the two major US political parties). We will generalize to multiclass problems in Section 3.4.3.

Within a single voting region, let $\mathbf{x}_i$ denote a vector of demographic covariates for the ith individual, and let $y_i \in \{0, 1\}$ denote that individual's vote, e.g., $y_i = 1$ if the individual votes for the Purple party. Note that we never observe y_i, and we obtain a sample of $\mathbf{x}_i$ values from the Public Use Microdata Sample (ACS). We also observe z, the total number of votes for the Purple party, and n, the total number of voters in the region.

Assume there are B regions in total, and, following LLP terminology, refer to regions as "bags" of instances. The underlying data for bag b is $\{(\mathbf{x}_i, y_i)\}_{i \in \mathcal{I}_b}$ where $\mathcal{I}_b$ is the index set for bag b. In the training data, instead of observing individual y_i values, we observe $z_b = \sum_{i \in \mathcal{I}_b} y_i$, the number of positive instances in the bag. The overall training data is

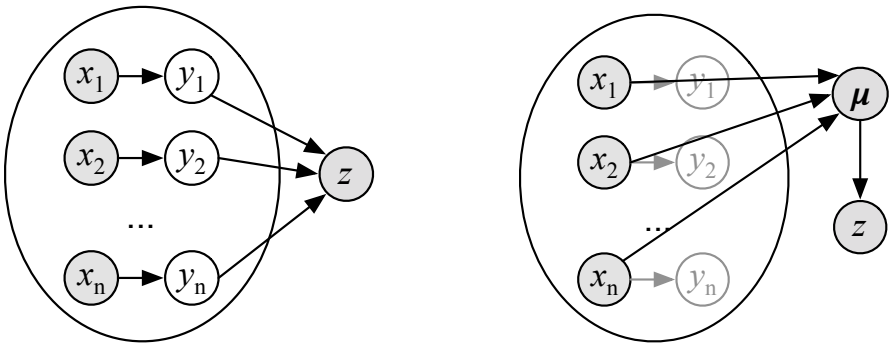
$$\big(\{\mathbf{x}_i\}_{i \in \mathcal{I}_1}, z_1\big), \ldots, \big(\{\mathbf{x}_i\}_{i \in \mathcal{I}_B}, z_B\big),$$

where each example consists of a bag of feature vectors and total number of positive votes for the bag. The goal is to learn a model to do one of two tasks. The first task is individual-level prediction: predict y_i given a new $\mathbf{x}_i$. The second task is bag-level prediction: predict z_b given a new bag $\{\mathbf{x}_i\}_{i \in \mathcal{I}_b}$ without any labels.

3.3.1 Comparison Between LLP and Distribution Regression

The generative model for LLP is illustrated in Figure 3.1a. The figure shows a single bag, for which the feature vectors $\mathbf{x}_i$ and vote total z are observed, but the individual labels y_i are unobserved. The conditional distribution of z is specified by the deterministic relationship $z = \sum_i y_i$. In a probabilistic LLP model, $p(y_i \mid x_i)$ is Bernoulli, and the modeler may choose any regression model for the success probability, such as logistic regression, as we do below, or a CNN (Li and Taylor, 2015).

For comparison, Figure 3.1b illustrates the mean embedding approach to distribution regression (Flaxman et al., 2015; Szabó et al., 2016). Here, the y_i variables



(a) LLP (b) Distribution Regression

Figure 3.1: LLP and distribution regression models for EI. (a) In LLP, there is a latent variable y_i for each individual, and $z = \sum_i y_i$ is the number of positive instances. (b) In distribution regression, y_i is ignored; $\boldsymbol{\mu}$ is an aggregated summary of the $\mathbf{x}_i$'s, and a regression model is learned to map directly from $\boldsymbol{\mu}$ to z.

are ignored, as is the known relationship between the y_i variables and z. Instead, the (empirical) distribution of the $\mathbf{x}_i$ values in the bag is summarized by a mean embedding into a reproducing kernel Hilbert space. In practice, this means computing the average $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n}\phi(\mathbf{x}_i)$ of expanded features vectors $\phi(\mathbf{x}_i)$ for each individual. Then, a standard regression model is learned to predict z directy from $\boldsymbol{\mu}$. Distribution regression introduces a tradeoff between the ability to preserve information about the individual-level covariates and complexity of the model. A feature expansion corresponding to a characteristic kernel preserves information about the distribution of the covariates, but is necessarily infinite and must be approximated; a very high-dimensional approximation will likely lead to increased variance in the following regression problem. If a simple feature expansion, such as a linear one, is used, it is clear the approach discards significant information about the individual-level covariates by simply computing their mean. LLP avoids this tradeoff by leveraging known structure about the problem.

3.4 Our Approach

We now present our approach, which is based on the generative model in Figure 3.1a and the EM algorithm. We adopt the logistic regression model $p(y_i = 1 \mid \mathbf{x}_i; \boldsymbol{\theta}) = \sigma(\mathbf{x}_i^T \boldsymbol{\theta})$ where $\sigma(u) = 1/(1 + e^{-u})$. It is straightforward to consider more complex regression models for $p(y_i = 1 \mid \mathbf{x}_i; \boldsymbol{\theta})$ without changing the overall approach. This completes the specification of the probability model

We now turn to learning. A standard approach is to find $\boldsymbol{\theta}$ to maximize the conditional likelihood $p(\mathbf{z} \mid \mathbf{x}; \boldsymbol{\theta}) = \sum_{\mathbf{y}} p(\mathbf{y}, \mathbf{z} \mid \mathbf{x}; \boldsymbol{\theta})$ where

$$p(\mathbf{y}, \mathbf{z} \mid \mathbf{x}; \boldsymbol{\theta}) = \prod_{b=1}^{B} \left(\prod_{i=1}^{n_b} p(y_i \mid \mathbf{x}_i; \boldsymbol{\theta}) \right) p(z_b \mid \mathbf{y}_b).$$

In this equation, let $\mathbf{y}_b = \{y_i\}_{i=1}^{n_b}$ denote the set of labels in bag b, let $\mathbf{x}_b = \{\mathbf{x}_i\}_{i=1}^{n_b}$ denote the set of feature vectors, and let $\mathbf{y}$, $\mathbf{z}$, and $\mathbf{x}$ denote the concatenation of the $\mathbf{y}_b$, z_b, and $\mathbf{x}_b$ variables from all bags. Also recall that $p(z_b \mid \mathbf{y}_b) = \mathbb{I}[z_b = \sum_{i=1}^{n_b} y_i]$. The obvious challenge to this learning problem is that the $\mathbf{y}$ variables are unobserved.

3.4.1 EM

EM is the standard tool for learning with latent variables (Dempster et al., 1977). At first, we will derive the EM algorithm for our model.

The model is:

$$p(\mathbf{y}, \mathbf{z} \mid \mathbf{x}; \boldsymbol{\theta}) = \prod_{b=1}^{B} \left(p(z_b \mid \mathbf{y}_b) \prod_{i=1}^{n_b} p(y_i \mid \mathbf{x}_i; \boldsymbol{\theta}) \right)$$

where $p(z_b \mid \mathbf{y}_b) = \mathbb{I}[z_b = \sum_{i=1}^{n_b} y_i]$.

We wish to maximize the marginal log-likelihood of the observed data $\mathbf{z}$ conditioned on $\mathbf{x}$, i.e., maximize

$$\ell(\boldsymbol{\theta}) = \log p(\mathbf{z} \mid \mathbf{x}; \boldsymbol{\theta}) = \log \prod_b p(z_b \mid \mathbf{x}_b; \boldsymbol{\theta})$$
$$= \sum_b \log \sum_{\mathbf{y}_b} p(z_b, \mathbf{y}_b \mid \mathbf{x}_b; \boldsymbol{\theta})$$

We may introduce a variational distribution $\mu_b(\mathbf{y}_b)$ and apply Jensen's inequality to obtain a lower bound $Q(\boldsymbol{\theta})$ for $\ell(\boldsymbol{\theta})$:

$$\begin{aligned}
\ell(\boldsymbol{\theta}) &= \sum_b \log \sum_{\mathbf{y}_b} \mu_b(\mathbf{y}_b) \frac{p(z_b, \mathbf{y}_b \mid \mathbf{x}_b; \boldsymbol{\theta})}{\mu_b(\mathbf{y}_b)} \\
&\geq \sum_b \sum_{\mathbf{y}_b} \mu_b(\mathbf{y}_b) \log \frac{p(z_b, \mathbf{y}_b \mid \mathbf{x}_b; \boldsymbol{\theta})}{\mu_b(\mathbf{y}_b)} \\
&= \sum_b \sum_{\mathbf{y}_b} \mu_b(\mathbf{y}_b) \log \frac{p(\mathbf{y}_b \mid \mathbf{x}_b; \boldsymbol{\theta}) p(z_b \mid \mathbf{y}_b)}{\mu_b(\mathbf{y}_b)} \\
&= -\sum_b KL(\mu_b \| p(\cdot \mid \mathbf{x}_b; \boldsymbol{\theta})) - \sum_b KL(\mu_b \| p(z_b \mid \cdot)) \\
&:= Q(\boldsymbol{\theta})
\end{aligned}$$

Given the value $\boldsymbol{\theta}_t$ of the current parameters, this lower bound is maximized when the KL divergence reaches 0, which happens when $\mu_b(\mathbf{y}_b) = p(\mathbf{y}_b \mid z_b, \mathbf{x}_b; \boldsymbol{\theta}_t)$ for all bags bs. After making this substitution and dropping the constant $\sum_b \sum_{\mathbf{y}_b} -\mu_b(\mathbf{y}_b) \log \mu_b(\mathbf{y}_b)$, and following the definition of the Q function in standard EM, we rewrite $Q(\boldsymbol{\theta})$ as:

$$\begin{aligned}
Q(\boldsymbol{\theta}) &= \mathbb{E}_{\mathbf{y}|\mathbf{z},\mathbf{x}}[\log p(\mathbf{y}, \mathbf{z} \mid \mathbf{x}; \boldsymbol{\theta})] \\
&= \sum_b \mathbb{E}_{\mathbf{y}_b|z_b,\mathbf{x}_b;\boldsymbol{\theta}_t} \left(\log p(z_b \mid \mathbf{y}_b) + \sum_i \log p(y_i \mid \mathbf{x}_i; \boldsymbol{\theta}) \right) \\
&= \sum_b \sum_i \mathbb{E}_{y_i|z_b,\mathbf{x}_b;\boldsymbol{\theta}_t} \log p(y_i \mid \mathbf{x}_i; \boldsymbol{\theta}) + \text{const}
\end{aligned}$$

In the tth iteration, we will select $\boldsymbol{\theta}$ to maximize $Q(\boldsymbol{\theta})$. The term $\log p(z_b \mid \mathbf{y}_b)$ is constant with respect to $\boldsymbol{\theta}$ and is ignored during the optimization.

In the M step, we maximize

$$Q(\boldsymbol{\theta}) = \sum_b \sum_i \mathbb{E}_{y_i|z_b,\mathbf{x}_b;\boldsymbol{\theta}_t} \log p(y_i \mid \mathbf{x}_i; \boldsymbol{\theta}) \tag{3.1}$$

Specializing to our logistic regression model, the lower bound simplifies to:

$$Q(\boldsymbol{\theta}) = \sum_b \sum_i q_i \log \sigma(\mathbf{x}_i^T \boldsymbol{\theta}) + (1-q_i) \log(1-\sigma(x_i^T \boldsymbol{\theta})) \tag{3.2}$$

where $q_i := p(y_i = 1|z_b, \mathbf{x}_b; \boldsymbol{\theta}_t)$. We dropped an dditional constant term from Eq. (3.1).

The M step, which requires maximizing $Q(\boldsymbol{\theta})$ given the q_i values, is straightforward. Eq. (3.2) is the standard logistic cross-entropy loss, but uses "soft" labels q_i in instead of the 0-1 labels. It can be optimized with standard solvers.

The E step, however, is challenging. It requires computing the posterior distribution $p(y_i \mid \mathbf{x}_b, z_b; \boldsymbol{\theta}_t)$ over a single y_i value given the observed data $\mathbf{x}_b$ and z_b for bag b and the current parameters $\boldsymbol{\theta}_t$. This corresponds exactly to inference in the graphical model shown in Figure 3.1a. Note that all variables are coupled by the hard constraint $\mathbb{I}[z_b = \sum_{i=1}^{n_b} y_i]$. It is not clear based on standard graphical model principles that efficient inference is possible.

3.4.2 Efficient Exact Inference with Cardinality Potentials

Tarlow et al. (2012) showed how to perform efficient marginal inference for a set of n binary random variables $y_1, \ldots, y_n$ described by a probability model of the form:

$$q(y_1, \ldots, y_n) \propto \phi\big(\sum_i y_i\big) \prod_i \psi_i(y_i), \tag{3.3}$$

where each variable has a unary potential $\psi_i(y_i)$, and there is a single *cardinality potential* $\phi(\sum_i y_i)$ that couples all variables y_i via their sum. Our model fits in this form. Consider the model for a single bag, and dispense with the bag index, so that the

variables are $\mathbf{x} = \{\mathbf{x}_i\}$, $\mathbf{y} = \{y_i\}$ and z. Our model for the bag has unary potentials $\psi_i(y_i) = p(y_i \mid \mathbf{x}_i; \boldsymbol{\theta})$ and a counting potential $\phi\big(\sum_i y_i\big) = \mathbb{I}[z = \sum_i z y_i]$. The method of (Tarlow et al., 2012) computes the marginal probability $q(y_i)$ for all i in $O(n \log^2 n)$ time. In our model $q(y_i) = p(y_i \mid \mathbf{x}, z; \boldsymbol{\theta})$ is exactly what we wish to compute during the E step, so this yields an E step that runs efficiently, in $O(n \log^2 n)$ time.

We now give details of the inference approach, but present them in the context of a novel generalization to the case when $y_1, \ldots, y_n \in \{0, 1\}^k$ are binary *vectors* of length k. Such a generalization is necessary for the most direct extension of our LLP approach to multiclass classification, in which $p(y_i \mid \mathbf{x}_i; \boldsymbol{\theta})$ is a categorical distribution over three or more alternatives. In Section 3.4.3, we describe this approach in more detail as well as a different and faster "one-vs-rest" approach to multiclass LLP.

Henceforth, assume that $y_1, \ldots, y_n$ are binary vectors that follow a joint distribution in the same form as Equation (3.3). To preview the meaning of the multivariate model, the binary vector y_i will be the "one-hot" encoding of the class for individual i, the unary potential is $\psi_i(y_i) = p(y_i \mid \mathbf{x}_i; \boldsymbol{\theta})$,[1] and the counting potential $\phi\big(\sum_i y_i\big) = \mathbb{I}[\sum_i y_i = \mathbf{z}]$ will encode the constraint that the total number of instances in each class matches the observed total, where $\mathbf{z}$ is now a vector of counts for each class. The description of the multivariate model is symbolically nearly identical to the scalar case.

The key observation of (Tarlow et al., 2012) is that it is possible to introduce auxiliary variables that are sums of hierarchically nested subsets of the y_i variables, and arrange the auxiliary variables in a binary tree with $z = \sum_i y_i$ at the root. Then, inference is performed by message-passing in this binary tree.

Figure 3.2 illustrates the construction as a factor graph for an example with $n = 4$. The nodes are arranged in a binary tree. Each internal node z_p is connected to two

[1]Note: this factor has 2^k entries indexed by the binary values $y_{i1}, \ldots, y_{ik}$. In this particular model, the binary vector is a one-hot vector, so $\psi_i(y_{i1}, \ldots, y_{ik})$ is nonzero if and only if there is a single nonzero y_{ij}. The inference technique also applies to arbitrary distributions over binary vectors, for which potentials would not have this structure.

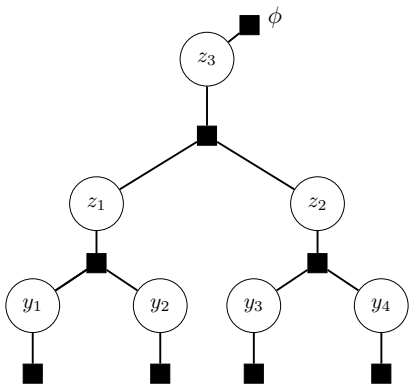


Figure 3.2: Illustration auxiliary variables arranged in a binary tree factor graph for inference with a cardinality potential. Each non-leaf node is a deterministic sum of its children. The root node z is equal to $\sum_i y_i$.

children, which we will denote z_l and z_r (for left and right), by a factor which encodes that z_p is deterministically equal to the sum of z_l and z_p, i.e., $\psi(z_p, z_l, z_r) = \mathbb{I}[z_p = z_l + z_r]$. The unary factors at each leaf are the original unary potentials $\psi_i(y_i)$. The auxiliary nodes and factors enforce that the root node z satisfies $z = \sum_i y_i$. Then, the factor attached to the root node is the cardinality potential $\phi(z) = \mathbb{I}[z = z_{\text{obs}}]$, where z_{obs} is the observed total.

This model is a tree-structured factor graph. Hence, exact inference can be performed by message passing using a leaf-to-root pass followed by a root-to-leaf pass. Although there are only $O(n)$ messages, the support of the variables grows with height in the tree. A node z_l at height i is a sum over 2^i of the y_i values, so it is a vector with entries in $\{0, 1, \ldots, 2^i\}$. We will write this as $z_l \in [m]^k$ where $m = 2^i$ and $[m] = \{0, 1, \ldots, m\}$. Note that m is never more than n.

The message passing scheme is illustrated in Figure 3.3. For any internal node z_u, let $\alpha_u(z_u)$ denote the incoming factor-to-variable message from the factor immediately below it. Similarly, let $\beta_u(z_u)$ be the incoming factor-to-variable message from the factor immediately above it. Because each internal variable is connected to exactly

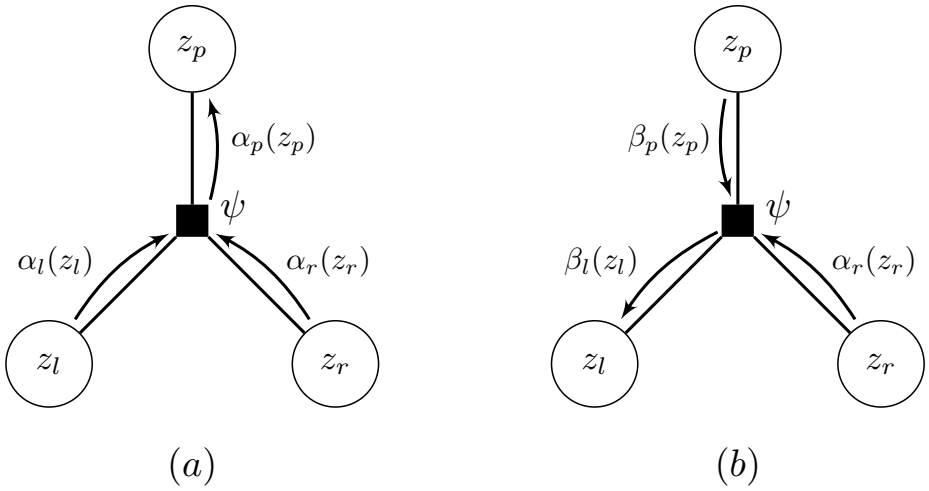


Figure 3.3: Illustration of messages from the factor $\psi = \mathbb{I}[z_p = z_l + z_r]$. (a) The upward message $\alpha_p(z_p)$ is computed from $\alpha_l(z_l)$ and $\alpha_r(z_r)$; (b) The downward message $\beta_l(z_l)$ is computed from $\beta_p(z_p)$ and $\alpha_r(z_r)$, similarly for $\beta_r(z_r)$. See text for details.

two factors, the variables will simply "forward" their incoming messages as outgoing messages, and we do not need separate notation for variable-to-factor messages.

The message operation for the upward pass is illustrated in Figure 3.3(a). The factor ψ connects children z_l and z_r to parent z_p. We assume that $z_l, z_r \in [m]^k$ for some m, and therefore $z_p \in [2m]^k$. The upward message from ψ to z_p is

$$\begin{aligned}\alpha_p(z_p) &= \sum_{z_l \in [m]^k} \sum_{z_r \in [m]^k} \alpha_l(z_l)\alpha_r(z_r)\mathbb{I}[z_p = z_l + z_r] \\ &= \sum_{z_l \in [m]^k} \alpha_l(z_l)\alpha_r(z_p - z_l). \end{aligned} \tag{3.4}$$

Similarly, the downward message to z_l, illustrated in Figure 3.3(b), has the form

$$\begin{aligned}\beta_l(z_l) &= \sum_{z_p \in [2m]^k} \sum_{z_r \in [m]^k} \beta_p(z_p)\alpha_r(z_r)\mathbb{I}[z_p = z_l + z_r] \\ &= \sum_{z_p \in [2m]^k} \beta_p(z_p)\alpha_r(z_p - z_l). \end{aligned} \tag{3.5}$$

Upward message computation:

$$\alpha_p(z_p) = \sum_{z_l \in [m]^k} \alpha_l(z_l)\alpha_r(z_p - z_l)$$

Downward message computation:

$$\beta_l(z_l) = \sum_{z_p \in [2m]^k} \beta_p(z_p)\alpha_r(z_p - z_l)$$

$$\beta_l(z_r) = \sum_{z_p \in [2m]^k} \beta_p(z_p)\alpha_l(z_p - z_r)$$

Figure 3.4: Summary of message passing for cardinality potentials. Each message operation is a convolution; the entire message can be computed in $O(m^k \log m)$ time by the multi-dim FFT. The overall running time to compute all messages is $O(n^k \log^2 n)$.

Note that the upward and downward message operations in Equations (3.4) and (3.5) both have the form of a convolution. Specifically, if we let $*$ denote the convolution operation, then $\alpha_p = \alpha_l * \alpha_r$, and $\beta_l = \beta_p * \hat{\alpha}_r$, where $\hat{\alpha}_r(z_{r1}, \ldots, z_{rk}) = \alpha_r(m - z_{r1}, \ldots, m - z_{rk})$ is the factor with the ordering of entries in every dimension reversed. While a direct iterative convolution implementation can compute each message in $O(m^{2k})$ time, a more efficient convolution using multidimensional fast Fourier transform (FFT) takes only $O(m^k \log m)$ time.

The max computation time for a single message is $O(n^k \log n)$, for the messages to and from the root. It can be shown that the total amount of time to compute all messages for each level of the tree is $O(n^k \log n)$, so that the overall running time is $O(n^k \log^2 n)$. The upward and downward message-passing operations are summarized in Figure 3.4. Note when we consider binary classification, we can simplify $y_i \in \{0, 1\}$ indicating voting for the positive class or not. The overall running time is $O(n \log^2 n)$.

3.4.3 Multiclass Classification

We explore two different methods to extend our LLP approach to multi-class classification: softmax or multinomial regression and one-vs-rest logistic regression. Consider the case when there are $C \geq 3$ classes. It is convenient to assume y_i is encoded using the "one-hot" encoding, i.e., as a binary vector of length C with $y_{ic} = 1$ if and only if the label is c. For each bag, we now observe the vector $z = \sum_i y_i$; the entry z_c is the total number of instances of class c in the bag.

3.4.3.1 Softmax regression

The obvious generalization of our logistic regression model to multiclass classification is multinomial or softmax regression. In this case, $p(y_i \mid \mathbf{x}_i; \boldsymbol{\theta})$ is a categorical distribution with probability vector $\mu_i = \mathbb{E}[y_i]$ obtained through a regression model. The entry μ_{ic} is the probability that y_i encodes class c, and is given by:

$$\mu_{ic} = \frac{\exp(\boldsymbol{\theta}_c^T \mathbf{x}_i)}{\sum_{c'} \exp(\boldsymbol{\theta}_{c'}^T \mathbf{x}_i)}.$$

The parameters $\boldsymbol{\theta}$ of the model now include a separate parameter vector $\boldsymbol{\theta}_c$ for each class c.

Our EM approach generalizes easily to this model. The M step remains a standard softmax regression problem. The E step requires computing the posterior probability vector $q_i = \mathbb{E}[y_i \mid \mathbf{x}, z; \boldsymbol{\theta}]$ for every instance in the bag. This is exactly the problem we solved in the previous section for cardinality potentials over binary vectors. Since each y_i, μ_i, and q_i vector sums to one, we may drop one entry prior to performing inference, and complete the E step for a bag with n instances in $O(n^{C-1} \log^2 n)$ time.

This approach is appealing because it is follows a widely used multiclass classification model, which is the natural generalization of logistic regression. However, a drawback is that the running time of the E step grows exponentially with the number

of classes, which may be too slow in practice when the numbers of instances or classes grows large.

3.4.3.2 One-vs-Rest Classification

An obvious alternative to softmax regression is one-vs-rest logistic regression, in which a separate logistic regression model is learned for each class c by training on the binary task of whether or not an instance belongs to class c. At test time, the class that predicts the highest probability is selected. This model has been observed to work well in many settings (Rifkin and Klautau, 2004). For our LLP model, it has a significant running-time advantage: each classifier can be trained in the binary prediction setting, so the E step will always run in $O(n \log^2 n)$ time, regardless of the number of classes.

3.5 Evaluation

3.5.1 Overview

Our approach is designed for the setting where the learner has access to individual-level covariates, but the outcome variable has been aggregated at the bag level. Given voting results and Census demographic data, we would like to, for example, infer what proportion of a particular minority group voted for a particular candidate. In this setting,

- The aggregate supervision is at the bag-level: voting is anonymous, and proportions of how people voted are known only at coarse aggregations by geographic region, from officially released precinct-level vote totals.

- Demographics are individual-level: the U.S. Census releases anonymized "micro-data," which consists of covariate vectors of demographic attributes of individuals. It is not known how individuals voted, of course.

Flaxman et al. (2015, 2016) apply distribution regression for this problem, performing aggregation on microdata demographics as a preprocessing step. In order to test our hypothesis that individual-level modeling can improve these inferences, we conduct a series of experiments:

- (§3.5.2): Synthetic experiments. We follow (Yu et al., 2014) and hide a known attribute from the individual-level data, and at training time our model accesses the attribute only as aggregated proportions per region. We evaluate models in their ability to accurately predict the hidden attribute for individuals.

- (§3.5.3): 2016 presidential elections. We look at the same task as in (Flaxman et al., 2016): trying to infer the individual-level conditional probability $p(\text{vote Dem} \mid f)$ for any arbitrary feature $f(x)$ of an individual's covariates x (e.g., "person is Hispanic/Latino and lives in Illinois"). We train the model with official per-geography voting tallies for the outcome, and a large set of Census demographic covariates for individuals, and aggregate the model's prediction for individuals to analyze $f(x)$ selections. Quantitative performance is evaluated by comparing to separate survey evidence (exit polls) for the same election.

Individual-level census data (x) is obtained from American Community Survey's Public Use Microdata Sample (PUMS), which covers all of the United States (including D.C. and Puerto Rico).[2] Its millions of records each represents a single person or demographically typical person, along with a survey weight representing how many people are represented by that record, based on the Census' statistical inferences to correct for oversampling, non-response, and to help preserve privacy. We use Flaxman et al.'s open-source preprocessor (used for (Flaxman et al., 2016))[3] to select and

[2] `https://www.census.gov/programs-surveys/acs/data/pums.html`

[3] `https://github.com/dougalsutherland/pummeler`

process the Census data. It merges PUMS data from 2012–2015, resulting in 9,222,638 records, with an average survey weight of 24.2.

PUMS is coded and partitioned geographically by several hundred Public Use Microdata Areas (PUMAs), which are contiguous geographic units with at least 100,000 people each. Since PUMAs do not exactly coincide with counties for which official electoral statistics are reported, the processing scripts merge them with overlapping counties (taking connected components of the bipartite PUMA-county overlap graph), resulting in 979 geographical regions. On average each region contains 9,420 PUMS records, representing on average 228,342 (stdev 357,204) individuals per region, when accounting for survey weights.

Each raw individual record x is comprised of 23 continuous covariates such as income and age, and 97 categorical covariates such as race, education, and spoken language. Flaxman et al. (2015) used FastFood (Le et al., 2013) to approximate a kernel map $\phi(x)$, then averaged $\phi(x)$ vectors to produce per-region mean embeddings for distribution regression.

Materials posted for later work (Flaxman et al., 2016) suggest that linear embeddings may perform similarly as nonlinear, random Fourier features-based embeddings. To simplify our investigation, we only consider linear embeddings in this work.

We use the same preprocessing routine to further process the covariates; it standardizes continuous covariates to z-scores, binarizes categorical covariates as features, and adds regions' geographical centroids, resulting in 3,881 dimensions for x in the model.

For reference, descriptions of several covariates are shown in Table 3.1, including ones inferred for the synthetic prediction experiments as well as exit poll analysis.[4] In some cases, the number of categories results from coarsening the original Census

[4]Details of all variables are available at: `https://www2.census.gov/programs-surveys/acs/tech_docs/pums/data_dict/PUMSDataDict15.txt`

Table 3.1: Covariates

Covariate	Num. Classes	Description
SEX	2	Gender
DIS	2	With or without disability
WKL	3	When last worked
SCHL	4	Educational attainment (high school or less, some college/assoc degree, college graduate, postgraduate study)
RAC1P	5	Race (White, Black, Asian, Hispanic/Latino, Other)

categories (e.g. SCHL has 25 categories in the original data, but is coarsened to 4 for prediction purposes), using the same preprocessing rules as in previous work. (The 3,881 dimensions for modeling use the original non-coarsened covariates.)

Finally, for computational convenience, we perform two final reductions on the x data before model fitting. First, for most experiments we subsample a fixed number of individuals per region, sampling them with replacement according to the PUMS survey weights. Second, we use PCA to reduce the covariates to 50 dimensions, which preserves approximately 80% of the variance in x.

3.5.2 Synthetic experiments

3.5.2.1 Partial synthetic data

We create partial synthetic data following the same procedure as (Yu et al., 2014): we hide a known discrete attribute from x and try to predict it from the other attributes. At training time, we supply it as supervision only as an aggregated count by region ($z_b = \sum_{i \in \mathcal{I}_b} y_i$). We evaluate models in their ability to predict the hidden attribute for individuals in held-out data.

The training data is prepared by sub-sampling either 10 or 100 individual records per region (as described in §3.5.1); we test both settings since prior literature has occasionally examined performance as a function of bag size.

The test set is constructed to include 10,000 records sampled from all regions (by survey weight), from records that were never selected for the training set.

For certain hidden response variables, some covariates are duplicates or near-duplicates, which makes the prediction problem too easy. We make the problem harder by removing attributes in x that have at least one value with Pearson correlation higher than 0.7 with the response (*i.e.,* hidden attribute). For example, if NATIVITY (native or foreign born) was the response variable, it has high absolute correlations to two different binarized values of CIT (citizenship status): CIT_4 (US citizen by naturalization) and CIT_1 (born in the US) have correlations (with NATIVITY) of 1 and 0.85, respectively. Furthermore, DECADE_nan (Decade of entry is N/A, meaning born in the US), and WAOB_1 (world area of birth is US state) also have high absolute correlations (both 0.85). Thus *all* CIT, DECADE, and WAOB attributes are removed. Depending on which hidden attribute is used as response and how many covariates are highly correlated, the number of covariates (out of 3,881) we removed ranges from 0 for HICOV (health insurance available) to 965 for WKL (when last worked).

3.5.2.2 Models

We test a series of logistic regression models, all of which can make predictions for individuals.

- `individual`: an oracle directly trained on the individual labels. This is expected to outperform all other methods, which can only access aggregated counts.
- `mean-emb`: logistic regression trained with mean embedding vectors and label proportions for each region. (Since the sampling already accounts for survey weights, the mean embedding vector for one region is the simple average $\hat{\boldsymbol{\mu}}_b = \frac{1}{n}\sum_{i\in\mathcal{I}_b}\mathbf{x}_i$.)
- `AMM`: logistic regression trained on bags of instances and label proportions. For multiclass problems, we use a one-vs-rest (ovr) approach (Patrini et al., 2014).

- `cardinality`: our method, trained on bags of instances and label proportions, for binary labels (§3.4.1,3.4.2).
- `card-exact`: our method for multiclass problems, with exact inference (§3.4.3).
- `card-ovr`: our method for multiclass problems, with an alternative one-vs-rest formulation, using binary cardinality inference.

Following (Patrini et al., 2014), we initialize all LLP methods (`AMM`, `cardinality`, `card-exact`, `card-ovr`) from `mean-emb`'s learned parameters. We expect LLP methods would improve `mean-emb` by a large margin.

3.5.2.3 Results

Figure 3.5 shows predictive accuracies of all trained models on the test set, for several hidden attributes (with 2, 3, and 4 categories each), with the mean and standard deviation of performance across 10 different trials. Each trial consists of one sampled training set, shared between all models (all trials use the same test set). Results are broadly similar for other hidden attributes (omitted for space).

The results show:

1. LLP outperforms mean embeddings: AMM and cardinality models substantially improve on `mean-emb`, presumably because they can exploit individual-level covariate information and the structure of the aggregation mechanism.
2. Our cardinality method is the most accurate LLP method: it outperforms AMM, the previous state-of-the-art with statistically significant differences for most sample sizes of regions and individuals. The cardinality method is a little slower than AMM, though the difference is minimal in the larger bag setting. Asymptotically, the running time for the "E"-steps of `AMM` and `card-ovr` are nearly the same: $O(kn \log n)$ and $O(kn \log^2 n)$, respectively.

3. For multiclass, `card-ovr` performs similarly as `card-exact`, and is computationally advantageous: `card-ovr` takes $O(kn \log^2 n)$ in runtime and only requires a 1D FFT, versus `card-exact`'s $O(n^{k-1} \log^2 n)$ runtime and additional memory consumption for multidimensional FFT. The exact method also has some precision issues (numerical underflow of downward messages) when running message passing in larger binary trees. Future work could pursue improvements to exact multiclass cardinality inference; in the meantime, we recommend one-vs-rest as an effective practical solution.

4. General observations: as expected, the oracle individual model outperforms all others. Also note that the larger per-bag samples (100) result in a harder problem than smaller (10) per-bag samples, since the training-time aggregation hides more information when bags are larger.

3.5.3 2016 US Presidential Election Analysis

For real-world experiments, our goal is to infer an individual's or a demographic group's voting preference via the individual-level conditional probability $p(\text{vote Dem} \mid f)$ for any arbitrary feature $f(x)$ of an individual's covariates x. We will compare our predictions for such subgroups to an alternative, well-established source of information, the exit polls at state and national levels.

3.5.3.1 Experiment

This experiment requires exit polls for validation, and voting data for model training. Exit polls are surveys conducted on a sample of people at polling stations, right after they finish voting; we use the widely reported 2016 exit poll data collected by Edison Research.[5]

[5]This questionnaire was completed by 24,537 voters at 350 voting places throughout the US on Election Day, from 28 states intended to be representative of the U.S. We use data scraped from the CNN website, available at: `https://github.com/Prooffreader/election_2016_data`.

Voting data (z) is based on county-level official results,[6] aggregated to the 979 regions. This results in a tuple of vote totals $(v_D, v_R, v_{\text{oth}})$ for each region: how many people voted for Clinton (D), Trump (R), or another candidate. Since the PUMS data includes information on all people—including nonvoters—we add in nonvoters to the third category, resulting in the following count vector for each region:

$$z = (v_D, v_R, S - v_D - v_R)$$

where S is the PUMS estimate of the number of persons in the region (sum of survey weights). This is a somewhat challenging setting for the model, requiring it to learn what type of people vote, as well as their voting preference.

We test the mean embedding model, AMM, and the one-vs-rest cardinality model, using all 3,881 covariates. Unlike the previous section, we give the mean embedding model additional access to *all* instances in the data (mean embeddings are constructed from a weighted average of all PUMS records per region), following previous work. By contrast, for the LLP models we sample 1000 individuals per region. PCA is again applied for all models in the same manner as before, and the LLP models are again initialized with `mean-emb`'s learned parameters.

For evaluation, we prepare a held-out dataset with a 1% subsample from all regions in the 28 exit poll states. After training each model, we make predictions on held-out records and aggregate them to state-level and nation-level statistics, so they can be compared against exit polls. We specifically infer fraction of the two-party vote

$$p(\text{vote D} \mid \text{vote D or R}, f(x)) = \frac{n_{D,f}}{n_{D,f} + n_{R,f}}$$

[6]We use Flaxman et al. (2016)'s version of the data, scraped from NBC.com the day after the election: `https://github.com/flaxter/us2016`

where $n_{D,f}$ and $n_{R,f}$ are counts of the model's (hard) predictions for individuals in the test set with property $f(x)$: for example, $n_{D,f}$ (and $n_{R,f}$) might be the number of Clinton (and Trump) voters among Hispanics in Illinois. These quantities are calculated from exit polls as well for comparison.

3.5.3.2 Results

The scatter plots in Figure 3.6 show predictions made by different methods vs. the exit poll results. The columns correspond to methods, and the rows correspond to the feature used to define subgroups. Each data point represents the subgroup for one feature value (e.g., males) in one state. There are up to 28 points per feature value; there may be fewer due to missing statistics in some state-level exit polls. For example, only 1% of respondents in Iowa exit polls were Asian, and the Iowa exit polls do not report the voting breakdown for Asians.

The scatter plots show that EI methods are indeed able to make correct inferences, but also make certain mistakes. For most methods and feature values (e.g., `mean-emb`, males), the state-level predictions are strongly linearly correlated with the exit polls—that is, the method correctly orders the states by how much individuals from different groups in the state supported Clinton. However, subgroups are often clustered at different locations away from the 1:1 line, indicating systematic error for that group—this is especially prominent for SCHL, where all methods tend to overestimate support for Clinton among college graduates, and underestimate support among individuals with high school or less education or with some college. In other examples, such as `mean-emb` for ETHNICITY=white, the overall positioning of the subgroup is correct and the state-level predictions are well *correlated* with the exit polls, but the slope is wrong. This suggests that the model has not correctly learned the magnitude of other features that vary by state. Overall, the LLP methods appear qualitatively better

(predictions more clustered around the 1:1 line) for SEX and ETHNICITY, while there is no clear "winner" for SCHL.

It is also interesting to aggregate the state-level predictions to national-level predictions. Table 3.2 shows national-level predictions as well exit polls for subgroups defined by gender (SEX), race (RAC1P), and educational attainment (SCHL). We see here that the models make mostly correct qualitative comparisons, such as: Which subgroup has a higher-level of support for Clinton? Does the majority of a subgroup support Clinton or Trump? However, the models make notable errors predicting the majority among men and women. Moreover, the models have a difficult time predicting the exact percentages even when the qualitative comparisons are correct.

To quantify these issues further and to gain a better comparison between the methods, Table 3.3 evaluates the methods based on national-level predictions according to three different metrics for each feature:

1. *Binary prediction* is the number of subgroups for which the method correctly predicts which candidate receives the majority (e.g., "a majority of males supported Trump", "a majority of women supported Clinton").

2. *AUC* measures the ability of the methods to order subgroups by their level of support for Clinton (e.g., "females showed higher support for Clinton than males"). It is measured by ordering the groups by *predicted* support for Clinton, and then measuring the fraction of pairs of groups that are in the correct order relative to the exit polls; this is related to the area under the ROC curve (Fawcett, 2006).

3. *Weighted RMSE* measures the numerical accuracy of the predictions. It is the square root of the weighted mean-squared error between the predicted and exit poll percentages, with weights given by the size of each subgroup.

The results show that the models are indeed generally good at the comparison tasks, as shown by the binary prediction and AUC metrics. However, they have considerable error (RMSE more than 5% in all cases) predicting percentages. There is no clear winner among the methods across all metrics. The mean embedding model has the lowest AUC for two out of three variables, and is tied on the third variable.

Table 3.2: National-level voting predictions for Clinton per demographic group

Covariate		**mean-emb**	**AMM**	**card**	**exit**
SEX	M	0.57	0.51	0.41	0.44
	F	0.44	0.47	0.45	0.57
RAC1P	Latino/Hispanic	0.63	0.69	0.77	0.70
	White	0.42	0.38	0.31	0.39
	Black	0.74	0.93	0.99	0.92
	Asian	0.58	0.91	1.00	0.71
	Others	0.77	0.83	0.94	0.61
SCHL	High school or less	0.44	0.32	0.18	0.47
	Some college	0.34	0.34	0.26	0.46
	College graduate	0.71	0.73	0.71	0.53
	Postgraduate	0.60	0.68	0.68	0.61

Table 3.3: Demographic-level model accuracy in predicting voting proportions, compared to exit polls.

Covariate		**Binary prediction**	**AUC**	**Weighted RMSE**
SEX	embed	0/2	0	0.13
	AMM	0/2	0	0.09
	card	1/2	1	0.09
RAC1P	embed	5/5	0.7	0.08
	AMM	5/5	0.9	0.06
	card	5/5	0.8	0.11
SCHL	embed	4/4	0.83	0.12
	AMM	4/4	0.83	0.15
	card	4/4	0.83	0.20

3.6 Conclusion

To analyze the US presidential elections, we formulated the ecological inference problem in the framework of learning with label proportions (LLPs). Compared with previous approaches, we exploit the known aggregation mechanism of the problem, and preserve information in individual-level covariates available to us from Census microdata. We contributed a novel, fully probabilistic, LLP method that outperforms distribution regression and a state-of-the-art LLP method on a range of synthetic tasks. Our probabilistic approach is enabled by adapting message-passing inference algorithms for counting potentials to a natural latent-variable model for LLP. We also applied several methods to analyze the 2016 US presidential election, and found that models frequently make correct comparisons among different choices or groups, but may not predict percentages accurately. Here, no method was a clear winner, but state-level results suggest that LLP methods are in closer agreement with exit polls.

A direction for further exploration is the potential of non-linear methods to improve performance. Previous work used non-linear feature embeddings, and, to a lesser extent, non-linear kernels for classification (Flaxman et al., 2015). In this work we have focused on linear embeddings and linear classifiers. However, our method can support arbitrary non-linear regression models, e.g., neural networks, for the individual-level model to predict y_i from $\mathbf{x}_i$. Exploration of such models is a worthwhile avenue for future research.

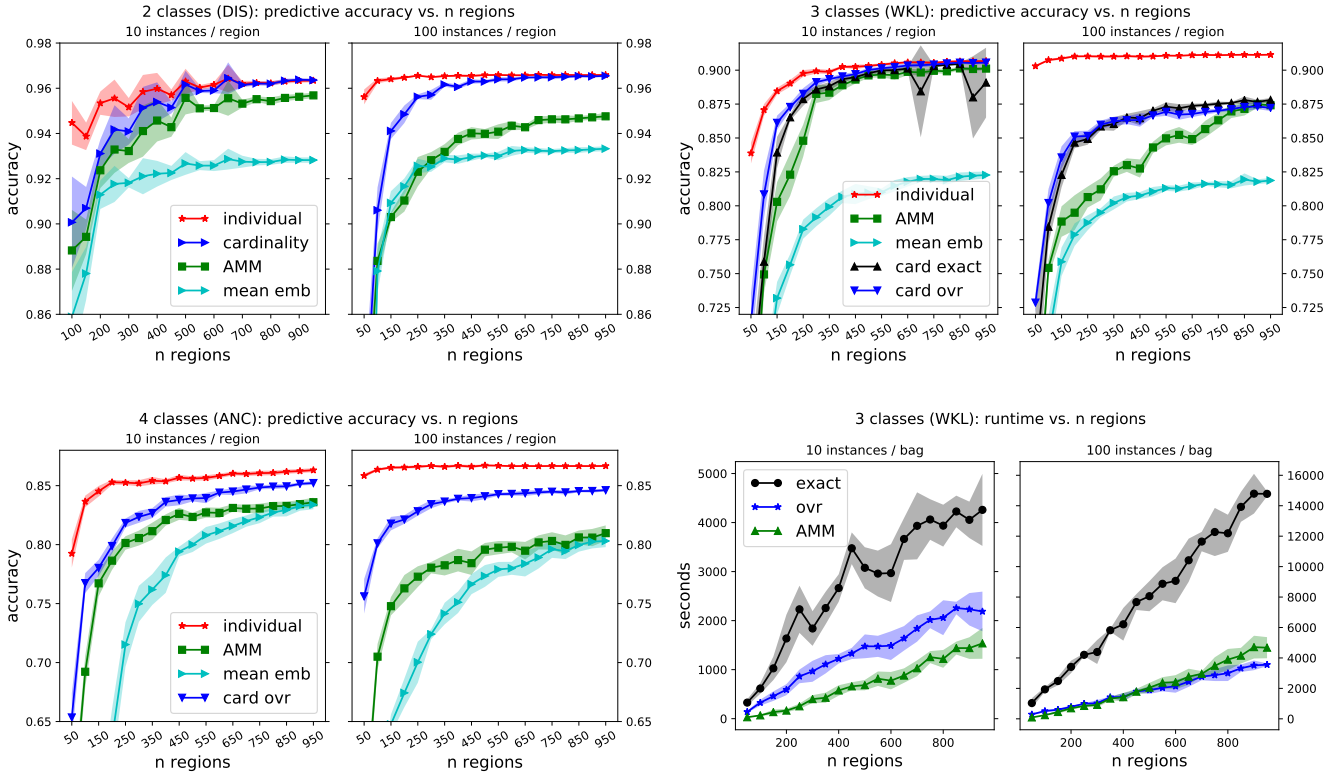


Figure 3.5: Predictive accuracies of trained models for 2, 3, or 4 labels classification tasks. The hidden attributes we chosen are Disability (DIS), When last worked (WKL), and Ancestry recode (ANC). We consider a small bag and a larger bag (10 or 100 instances per bag)) for each hidden attribute. Shaded error bars are 95% confidence intervals computed from 10 repeated trials.

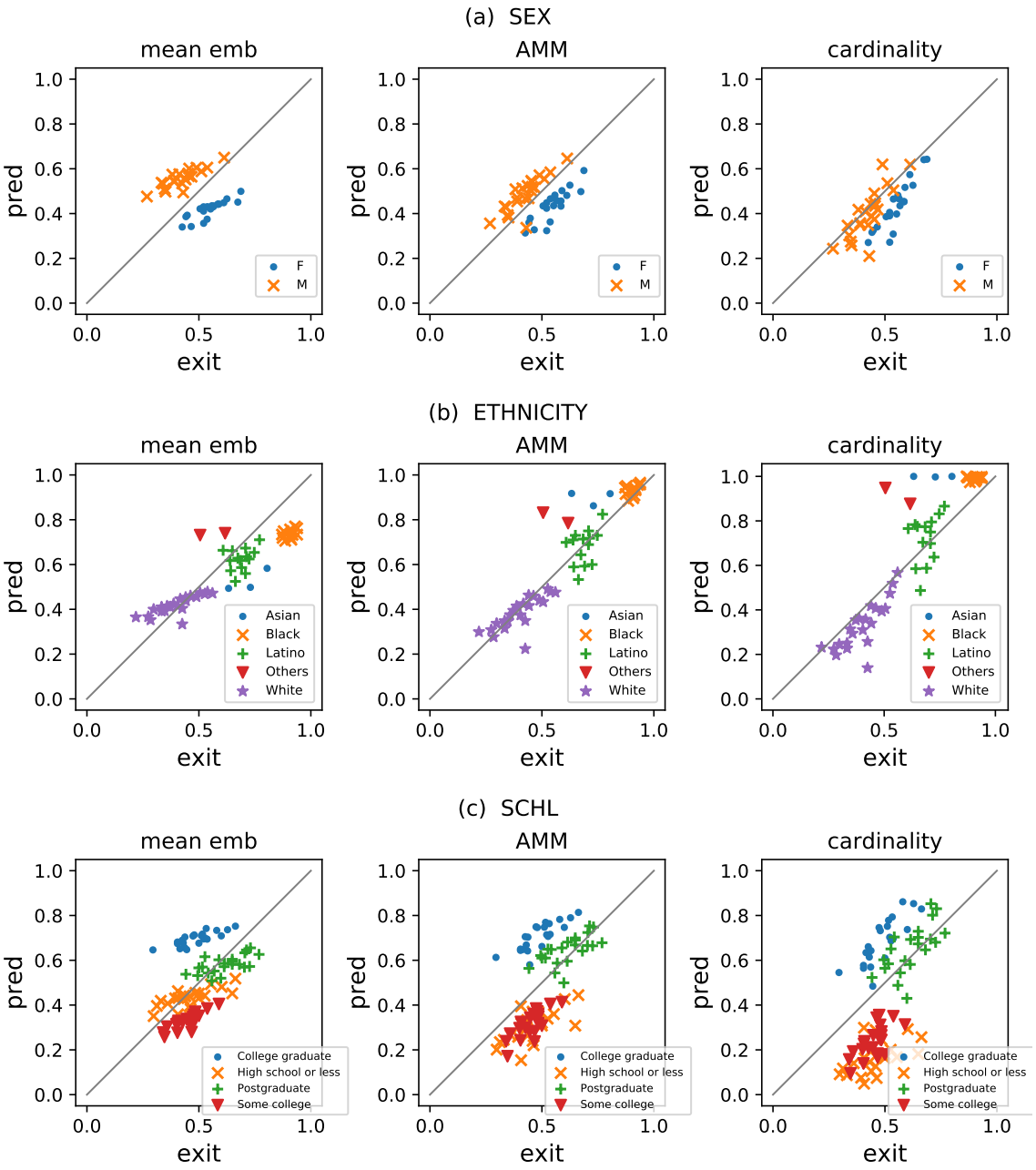


Figure 3.6: Model predictions versus exit polls, by demographic group and state. Each color (demographic category) has up to 28 points for its subpopulation per state (for subgroups large enough such that the exit poll results show voting numbers).

CHAPTER 4

DISTRIBUTION REGRESSION

Standard supervised learning learns a function $f\colon \mathbf{x}_i \mapsto y_i$ that maps from one training instance x_i to its supervision y_i. Distribution regression (also called learning from distributions) tries to learn a function $f\colon \mathbb{P}_i \mapsto y_i$ that maps from one training distribution $\mathbb{P}_i$ to its supervision y_i. In practice, however, these distributions $\mathbb{P}_i$ (for all i) are generally not available. Instead we use samples $\{\mathbf{x}_{ij}\} \sim \mathbb{P}_i$ from $\mathbb{P}_i$ to learn $f\colon \{\mathbf{x}_{ij}\} \mapsto y_i$ that maps from samples to the supervision. We consider each distribution as a bag and we are trying to learn model f given bags of instances.

In this chapter, we will build bag-level models using distribution regression, given bags of instances and bag-level aggregate supervisions. Many related problems, such as ecological inference and multi-instance learning, can be cast as special cases of distribution regression. The learning with label proportions (LLPs) from Chapter 3 can be treated as a special case of distribution regression, with an intermediate step that infers and then aggregates individual supervisions. Distribution regression is a more general learning framework, and has the potential to work with problems with large bag sizes. Distribution regression is widely adopted for many applications such as point estimation (estimating hyperparameters or population statistics), point cloud classification (Qi et al., 2017a,b), estimating mass of galaxy cluster from velocities (Ntampaka et al., 2015, 2016), estimating red shift of galaxy clusters (Connolly et al., 1995; Rozo and Rykoff, 2014), inferring demographic voting behavior (Flaxman et al., 2015, 2016), and modeling radioactive isotope behavior (Jin, 2014).

In this chapter, we will categorize distribution regression to two embedding strategies: the "fixed-embedding" strategy will first construct an embedding vector for each bag independently of the regression target, before feeding the embedding to a (non)linear regressor; the "learned-embedding" strategy will learn the bags' embeddings and the regressor end-to-end. We will start by providing background for the fixed-embedding strategy, introducing the widely studied Kernel Mean Embedding (KME), as well as two quantization-based approaches that are largely ignored by the research community. We will continue by introducing the latest deep learning approaches, which jointly learn the bag-level embeddings and the regressor.

We identified three key factors that directly contribute to the performance of distribution regresssion: bag sizes, number of training bags, and embedding dimension. In order to illustrate their impact, we constructed three relevant hypotheses, and applied to three tasks: estimating population statistics, point cloud classification, and predicting voting preferences. Each task will empirically validate one or two of our hypotheses.

The results from the three tasks show that, given "large enough" training bags and bag sizes, the deep learning approaches would be the best since the joint learning would improve embeddings; given limited number of training bags, the simpler fixed embedding approaches are more efficient and generalize better than the deep learning approaches.

My main contributions to this work:

1. Empirically evaluated different algorithms for distribution regression on three tasks.

2. Identified key elements of distribution regression and showed how a number of different methods can be applied to it.

3. Highlighted quantization-based approaches as competitive baselines.

This chapter is organized as follows. First we will introduce the background of distribution regression, its categorization and major techniques. Next we will identify three major factors that probably affect model performance. We formulate hypotheses for each of them. In the end, we empirically evaluate these hypotheses on three tasks.

4.1 Background

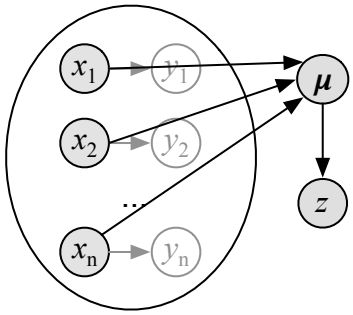


Figure 4.1: Distribution regression. The oval represents ONE bag and $\boldsymbol{\mu}$ is the embedding vector for this bag.

We can describe distribution regression as shown in Figure 4.1. The oval represents one bag and we drop subscripts for indexing bags to simplify the notation. Given bags of instances $\{\mathbf{x}_i\}$ and bag-level aggregate supervision z, we want to build a bag-level model $g\colon \boldsymbol{\mu} \mapsto z$ that maps from the embedding vector $\boldsymbol{\mu}$ for a bag to its supervision z. Our goal is to allow the embedding vector $\boldsymbol{\mu}$ to summarize all the information of the bag's underlying distribution $\mathbb{P}$. Distinct from learning with label proportions (LLPs in Chapter 3), distribution regression skips an intermediate step that explicitly infers individual supervisions y_i and aggregates all individual y_i to obtain z. Therefore, distribution regression ignores the generative process, but enjoys the benefit that there is no expensive inference of individual supervisions and it is easy to work with problems with large bag sizes. The second difference from LLPs is that the aggregate supervision z in distribution regression can be class counts, class

labels, scalars, vectors, etc. While the aggregate supervision z in our probabilistic LLPs in Chapter 3 can only take count statistics.

So the core question is: how to compute the embedding vector $\boldsymbol{\mu}$?

Distribution regression can be described as a two-step procedure: $\{\mathbf{x}_i\} \xrightarrow{f} \boldsymbol{\mu} \xrightarrow{g} z$. First an *embedding function* $f\colon \{\mathbf{x}_i\} \mapsto \boldsymbol{\mu}$ maps samples of a bag to the embedding vector $\boldsymbol{\mu}$ for the bag. Second a *regressor* $g\colon \boldsymbol{\mu} \mapsto z$ maps the embedding vector $\boldsymbol{\mu}$ to its supervision z. If we can obtain the embedding vector $\boldsymbol{\mu}_b$ for each bag b, then $\boldsymbol{\mu}_b \mapsto z_b$ becomes a standard regression problem.

Depending on how to construct the embedding vector $\boldsymbol{\mu}$, we can categorize distribution regression to a "fixed-embedding" strategy or a "learned-embedding" strategy. The former builds $\boldsymbol{\mu}$ via the embedding function f independently of the supervision z and the latter learns both f and g end-to-end. In this section, we will review two fixed-embedding approaches, the Kernel Mean Embedding (KME) and the quantization-based embeddings. We will also review the latest deep learning architectures (Zaheer et al., 2017; Qi et al., 2017a) for jointly learning both f and g.

4.1.1 Fixed-embedding: Kernel Mean Embedding

Most materials in this subsection are abstracted from the monograph by Muandet et al. (2017). We provide minimum descriptions that are necessary for our discussion.

The first fixed-embedding technique is Kernel Mean Embedding (KME). KME is the vast majority of research for distribution regression. KME represents each distribution $\mathbb{P}$ as a mean embedding function

$$\mu_{\mathbb{P}} := \mathbb{E}_{\mathbf{x}\sim\mathbb{P}}[k(\mathbf{x}, \cdot)] = \int_{\mathcal{X}} k(\mathbf{x}, \cdot)\, \mathrm{d}\mathbb{P}(\mathbf{x}) \in \mathcal{H} \tag{4.1}$$

where k is a kernel function and $\mu_{\mathbb{P}}$ transforms $\mathbb{P}$ to an element in the reproducing kernel Hilbert space (RKHS) $\mathcal{H}$. For *characteristic kernels*, KME captures all information about $\mathbb{P}$. As a result,

1. We can use KME to define a metric over the space of probability distributions;

2. KME is suitable for applications that requires a unique characterization of distributions such as two-sample homogeneity tests and distribution regression;

3. Learning algorithms can be extended to the space of probability distributions with minimal assumptions of the underlying data generating process.

Therefore, we can use KME to bypass estimating a density, which is very difficult especially in high dimensional space.

Given a bag of instances, the standard KME estimator is the empirical average

$$\hat{\mu}_{\mathbb{P}} := \frac{1}{n} \sum_{i=1}^{n} k(\mathbf{x}_i, \cdot). \tag{4.2}$$

But one major problem is that for several characteristic kernels we are interested in, $\hat{\mu}_{\mathbb{P}}$ lives in infinite dimensional space. To address the problem, in practice, we use finite dimensional approximation of $\mu_{\mathbb{P}}$ by considering *random feature map* $\hat{\phi}$ (Rahimi and Recht, 2008) in finite dimensional space, and obtains the KME estimator by

$$\tilde{\mu}_{\mathbb{P}} = \frac{1}{n} \sum_{i=1}^{n} \hat{\phi}(\mathbf{x}_i). \tag{4.3}$$

The approximate estimator $\tilde{\mu}_{\mathbb{P}}$ can be easily derived as long as we know how to compute $\hat{\phi}$. Many work follows this direction, such as Random Fourier Feature (RFF) (Rahimi and Recht, 2008), Orthogonal Random Features (ORF) (Felix et al., 2016), and Gaussian quadrature (Dao et al., 2017).

After obtaining $\tilde{\mu}_{\mathbb{P}_i}$ for each bag i, distribution regression is straightforward as standard supervised learning.

Theoretically, recent work by Szabó et al. (2015) analyzed distribution regression as a "two-stage sampling" procedure: we are able to *learn* a distribution regression

model if instances $\{\mathbf{x}_i\}$ within each bag are sampled *i.i.d.* from the bag distribution $\mathbb{P}$ and all bag distributions $\mathbb{P}_i$ are sampled *i.i.d.* from a meta-distribution.

In summary, KME has some underlying limitations that we want to overcome. First, we need to assume a kernel function *a priori* to compute the KME estimator. This requirement is very restrictive for many real-world applications. For example, for the task of predicting voting preferences, it does not make sense to assume a shift-invariant kernel – the most widely used one – for a socioeconomical feature vector $\mathbf{x}_i$. Second, the two-stage sampling assumption is restrictive: bags may not be *i.i.d.* . For example, adjacent voting districts are not independent. Third, KME is computed independently of the supervision of the task, we may wish to build an embedding that is both data- and task-dependent to further improve model performance. We will see how deep learning approaches can help to overcome these limitations.

4.1.2 Fixed-embedding: Quantization-based Approaches

The second fixed-embedding technique uses quantization-based approaches. We consider two of the most commonly used approaches: bag-of-words (BOW) and Fisher vector (FV) (Perronnin and Dance, 2007; Perronnin et al., 2010). They would capture the underlying distribution of each bag to different levels of granularity.

The BOW approach encodes each bag by count statistics. It runs k-means first on all samples from all bags to obtain cluster centers, and represent each bag by a (normalized) histogram of count statistics $\boldsymbol{n}$. The entry n_i is defined to be the total number of instances in the bag belonging to cluster i.

The FV approach encodes each bag with richer information. It runs Gaussian Mixture Models first to obtain model parameters $\boldsymbol{\theta} = \{w_k, \mu_k, \sigma_k, k = 1, \cdots, K\}$, where w_k, μ_k, and σ_k are respectively the mixture weight, mean vector, and standard deviation vector of a diagonal Gaussian. FV will then assign each data instance soft assignments $\gamma(k)$ to all K Gaussians, and represents each bag by a Fisher

vector, defined to be the gradients of the log likelihood of the bag with respect to all Gaussian parameters $\boldsymbol{\theta}$. More specifically, we convert a data instance $x \in \mathbb{R}^D$ to a vector of gradients $\phi(x) = [\varphi_1(x), \cdots, \varphi_K(x)] \in \mathbb{R}^{2KD}$, each φ_i is defined as $\varphi_k(x) = [\frac{\gamma(k)}{\sqrt{w_k}}(\frac{x-\mu_k}{\sigma_k}), \frac{\gamma(k)}{\sqrt{2w_k}}(\frac{(x-\mu_k)^2}{\sigma_k^2} - 1)]$, and then taking the average of all vectors for the bag's Fisher vector.

BOW only accounts for count statistics while FV accounts for both the count statistics and the geometry of each bag. FV was the most successful feature representation for images before the surge of deep learning (Chatfield et al., 2011; Sánchez et al., 2013). For distribution regression, we consider quantization-based approaches as strong baselines, although for some unknown reasons they are not widely adopted in distribution regression research. We will evaluate their strengths and weaknesses in the evaluation section.

4.1.3 Learned-embedding: Deep Learning-based Approaches

Latest deep learning-based approaches for distribution regression assume all bags are sets, *i.e.,* elements in each bag are permutation-equivariant. In what follows, I will present several recent work in this direction.

Zaheer et al. (2017) developed DeepSets, specialized for sets as input. They proved that a system is permutation invariant to the elements of the input set X if the system encodes a score function

$$S(X) = g(\sum_{x \in X} \phi(x)) \tag{4.4}$$

where g and ϕ are two transformation structures and ϕ is shared among all instances $x \in X$ in the input set. They proved this design is a universal approximator to *any* continuous set function $f(X)$. The sum pooling in the score function can be replaced with max pooling. In practice, both g and ϕ are neural networks. We can consider ϕ as the feature extractor and g the regressor and $\text{pool}_i \phi(x_i)$ the embedding function

in the two-step procedure for distribution regression. Both ϕ and g will be learned end-to-end.

They also proposed a permutation-equivariant (PE) layer, which is a general module that can be applied in any set based task and be stacked multiple times – just like stacking convolution layers for images – to capture increasingly complex geometric structure.

They applied DeepSets to both supervised and unsupervised learning tasks. The supervised tasks, that include estimating red-shift of galaxy clusters and point cloud classification, are exactly applications of distribution regression.

Qi et al. (2017a) developed PointNet, a DNN for 3D classification and segmentation given point clouds. Each point cloud is a set and the goal of its architecture design is to encode permutation-equivariance of cloud points, represented by xyz coordinates. The main idea of PointNet is very similar to DeepSets, both applying max pooling over local embeddings of instances processed by a multi-layer perceptron (MLP) $\phi(x)$.

4.2 Key Factors and Hypotheses

In this section, we will identify three key factors for distribution regression and formulate our hypotheses. We will evaluate the hypotheses on three tasks in the next section.

4.2.1 Bag Size

The first key factor for distribution regression is the bag size. To simplify notation, we consider bags with equal sizes, *i.e.,* $N = N_b, \forall b \in B$.

Table 4.1 discusses the embedding function $f\colon \{\mathbf{x}_i\} \mapsto \boldsymbol{\mu}$ in the first stage of the two-step procedure of distribution regression.

For KME, $\boldsymbol{\mu}$ is constructed independently of the distribution of both the input and the target. For example, Random Fourier Features (RFF) (Rahimi and Recht,

Table 4.1: Embedding function f in $\boldsymbol{\mu} = f(\{\mathbf{x}_i\})$

Embedding	Embedding function f
KME	f fixed
Quantization-based	f depends on $p(\mathbf{x})$
DeepSets	f, g trained jointly, f depends on $p(\mathbf{x}, z)$

2008) is the most classical approach to computing the approximate feature map $\phi(\mathbf{x}_i)$ in Eq. 4.3, and it only depends on an kernel choice and the number of Monte-Carlo samples used to construct $\phi(\mathbf{x}_i)$. The bag size N does not directly play an role.

For quantization-based approaches, we construct for each bag either a histogram of count statistics or a Fisher vector, both depending on the underlying bag's distribution $p(\mathbf{x})$. We hypothesize that a large bag size N is required to capture the complex geometrical structure of $p(\mathbf{x})$.

For DeepSets, since the embedding function f and the regressor g are trained end-to-end, f depends on the joint distribution of $p(\mathbf{x}, z)$. We hypothesize that a large bag size N is required to fine-tune the large amount of model parameters and learn the embedding $\boldsymbol{\mu}$ well.

In summary, our hypothesis H1 is:

- Quantization-based approaches need large bag size N to capture $p(\mathbf{x})$.
- DeepSets needs large bag size N to learn $\boldsymbol{\mu}$ well since there are too many model parameters.

4.2.2 Number of Bags

The second key factor is the number of bags, denoted by l. Consider the aa regressor g in $z = g(\boldsymbol{\mu})$, the number l of bags is equivalent to the number of training examples in a standard regression problem. Based on the VC theory and the theory of

bias-and-variance decomposition (Abu-Mostafa et al., 2012), we would expect a large l for small penalty for model complexity and small variance for out-of-sample error.

Our hypothesis H2 is:

- DeepSets can perform the best for large number of bags because it learns the highly nonlinear regressor with large amount of parameters of g.
- Other methods may perform better for small number of bags.

4.2.3 Embedding Dimension

Suppose the input dimension is D and the embedding dimension is d given $\{\mathbf{x}_i \in \mathbb{R}^D\} \xrightarrow{f} \boldsymbol{\mu} \in \mathbb{R}^d \xrightarrow{g} z$. Table 4.2 lists the embedding dimension d for all methods.

Table 4.2: Embedding dimension d

Embedding	Embedding dimension d
KME, DeepSets	d selected independently of the input dim D
k-means	$d = K$, the number of clusters
Fisher vector (FV)	$d = 2KD$

For KME, the embedding dimension d is selected independently of the input dimension D. For example, RFFs construct $\boldsymbol{\mu}$ of length $2X$ where X is the number of sampled frequencies. For DeepSets, d is manually specified in advance. For k-means, d is the length of the histogram, equivalent to the specified number of clusters. For Fisher vector, $d = 2KD$ since we have gradients with respect to all means and all variances from all diagonal Gaussians found by GMM. In practice, we pick large K in hopes of capturing complex geometrical structure of the data. But when the input dimension D is also large, the Fisher vector could be extremely long and leads to the $d \gg n$ problem.

Our hypothesis H3 is: Fisher vector will suffer from large input dimension D, since the classical $d \gg n$ problem will lead to high variance and overfitting.

4.3 Evaluation

We will evaluate the three hypotheses on distribution regression using both "fixed-embedding" and "learned-embedding" strategies on three applications:

1. Estimating population statistics. The synthetic experiments follow (Zaheer et al., 2017). We will try to estimate entropy or mutual information of new bags on four tasks. The synthetic data is generated as follows: first a meta-distribution $\mathcal{M}$ is a Gaussian distribution controlled by parameter(s) $\boldsymbol{\theta}$. We sample from $\mathcal{M}$ by varying $\boldsymbol{\theta}$ to obtain distributions $\mathbb{P}_i \sim \mathcal{M}$ for bags. Therefore every distribution $\mathbb{P}_i$ is also Gaussian. Next for each bag we generate samples $\{\mathbf{x}_j^{(i)}\} \sim \mathbb{P}_i$ from its distribution $\mathbb{P}_i$. We will try to learn from these bags of samples a regressor that could predict a new bag's entropy or mutual information.

2. Point cloud classification. We use the ModelNet10 and ModelNet40 datasets (Wu et al., 2015), and perform the 10-class and 40-class classification tasks. Each class has many training clouds and each cloud consists of points in the form of xyz coordinates.

3. Predicting US presidential election. From 840 electoral regions for the 2012 election and 979 regions for the 2016 election, which cover the contiguous United States, we collect (1) the voting fractions for Democrats and Republican, (2) characteristic individuals' socioeconomical covariates from the US Census. We consider each region as a training bag, the voting preference for Democrats as bag-level supervision, and individuals are sampled from the bag's underlying (unknown) distribution. The task is, for any demographic subgroup, such as the white people live in LA with college degree, ages 30-44, we want to predict the fraction of people that votes for the Democrats.

For each task, we will apply the following methods:

- Kernel mean embedding by Orthogonal Fourier Features (but denoted RFFs as follows). Fixed embedding.
- Bag-of-Words (BOW). Quantization-based approach and fixed embedding.
- Fisher Vector (FV). Quantization-based approach and fixed embedding.
- DeepSets (Zaheer et al., 2017). Learning the embedding and the regressor end-to-end.

The three fixed embeddings will be fed to both linear and kernel ridge regressions.

Table 4.3: Characteristic information of datasets

Task	Number of bags l	Bag size N_i	Instance dimension D
Estimate Pop. Statistics	$128 \sim 65536$	512	≤ 32
Point Cloud Classification	9843	$\{100, 500, 1000\}$	3
Predict Voting 2012	840	mean $\pm$ std: 12864 $\pm$ 21955, median: 7185	4674
Predict Voting 2016	979	mean $\pm$ std: 9421 $\pm$ 19940, median: 5633	3880

Table 4.3 summarizes key characteristic information of the datasets, including the number l of bags, the bag size N_i (for bag i), and the input dimension D. For the first two tasks, since we have large number of training bags, small bag sizes and small feature dimensions, we would expect DeepSets to perform the best. For the voting task, since we have only a small number of training bags, large bag sizes and very high input dimensions, we will expect simpler fixed embedding approaches perform better.

4.3.1 Estimating population statistics

This synthetic experiment follows (Zaheer et al., 2017). The goal is to predict the entropy or mutual information of new bags, given bags of samples for training and prediction. The setups for the four tasks are:

- Rotation: Randomly choose a 2×2 covariance matrix Σ, and then generate $l = 2^7 \sim 2^{16}$ sample sets from $\mathcal{N}(0, R(\alpha)\Sigma R(\alpha)^T)$ of size $N = 512$ for l random

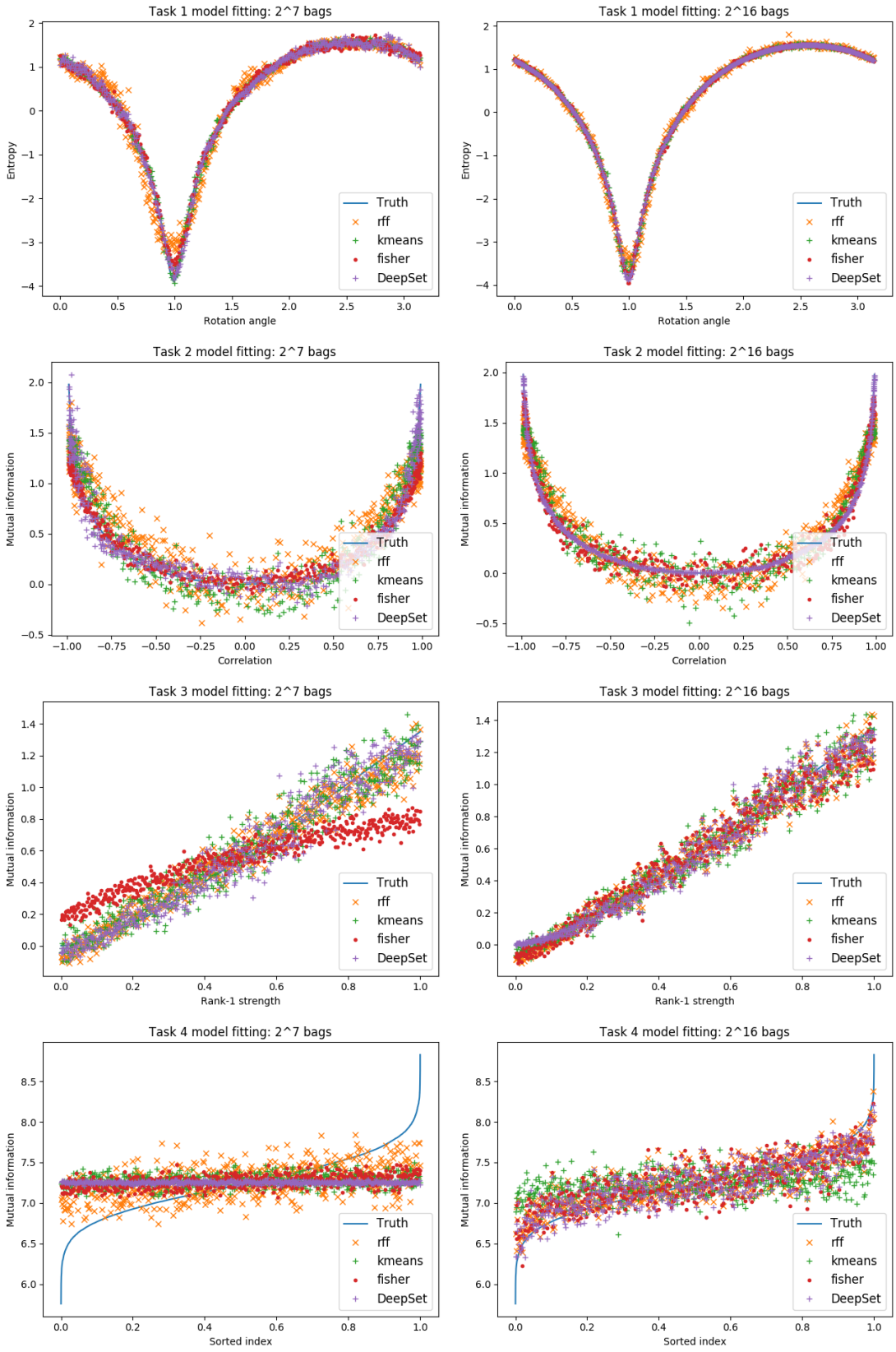


Figure 4.2: Fitting results on a test set for each task using different trained models. Left: all models trained on 2^7 training bags. Right: trained on 2^{16} training bags.

values of $\alpha \in [0, \pi]$, which is the only parameter of the meta distribution. The goal is to learn the entropy of the marginal distribution of first dimension. $R(\alpha)$ is the rotation matrix.

- Correlation: Randomly choose a $d \times d$ covariance matrix Σ for $d = 16$, then generate $l = 2^7 \sim 2^{16}$ sample sets from $\mathcal{N}(0, [\Sigma, \alpha\Sigma; \alpha\Sigma, \Sigma])$ of size $N = 512$ for l random values of $\alpha \in (-1, 1)$. The goal is to learn the mutual information of among the first d and the last d dimensions.

- Rank 1: Randomly choose $v \in \mathbb{R}^{32}$ and then generate $l = 2^7 \sim 2^{16}$ sample sets from $\mathcal{N}(0, I + \lambda vv^T)$ of size $N = 512$ for l random values of $\lambda \in (0, 1)$. The goal is to learn the mutual information.

- Random: Randomly choose $d \times d$ covariance matrices Σ for $d = 32$, and then generate $l = 2^7 \sim 2^{16}$ sample sets from $\mathcal{N}(0, \Sigma)$ of size $N = 512$. The goal is to learn the mutual information.

For BOW and Fisher vector (FV), we specified 200 clusters for running k-means and GMM. We used the default setting from the source code of DeepSets[1]: the feature extractor component used 3 linear layers and Exponential Linear Unit (ELU) as activations except for the last layer; the pooling layer used average pooling; the regressor consists of 4 linear layers and ELU except for the last layer.

First, we will show the predictions on a test set of 512 test bags. We trained our models on $2^7 = 128$ bags and $2^{16} = 65536$ bags respectively to compare the impact of the number of training bags on the performances of the learned models. The results in Figure 4.2 show:

1. Given only 2^7 training bags, the first two tasks are relatively easy since the predictions from all models are close to their ground truths, which are the blue

[1] `https://github.com/manzilzaheer/DeepSets`

lines buried underneath the predictions. We observed that only DeepSets can capture the high-curvature regions, like the bottom in task 1 and the two ends in task 2. We found DeepSets > Fisher Vector > BOW (k-means) > RFF. We use ">" to indicate the l.h.s. predicts better on a holdout testset than the r.h.s. does. The last two tasks are relatively more difficult, since all methods perform poorly in prediction. The predictions from FV bias away from the ground truth in task 3 and none of the methods capture the high-curvature regions in task 4.

2. Given 2^{16} training bags, the predictions from all methods get much more accurate: the predictions from DeepSets become more concentrated on the ground truths and capture more high-curvature regions, and the predictions from FV on task 3 are calibrated back towards the ground truth line.

Figure 4.3 shows the MSEs between the predictions and the ground truth of the test set. We observe (1) more training bags helps to reduce MSE for all methods; (2) DeepSets outperforms other models on the first three tasks and on the last task given more than 2^{11} training bags; (3) RFF perform the worst on "easy" tasks but perform better than quantization-based methods on "more difficult" tasks, and is almost the best on task 4.

We also compared linear ridge regressor to kernel ridge regressor (KRR) on the easy task 1 and the more difficult task 4. The results in Figure 4.2 and 4.3 are based on linear ridge regression. We use grid search to find the best parameters. The results in Figure 4.4 show that KRR is highly unstable for these tasks. When we computed the closed solution for KRR, we got warning messages "Ill-conditioned matrix detected. Results is not guaranteed to be accurate" and "Singular matrix in solving dual problem. Using least-square solution instead". Based purely on these MSE results, linear ridge regression performs better than KRR on these tasks.

The above results support that joint learning by DeepSets could improve the embedding function f and the generalization, therefore the overall performance over

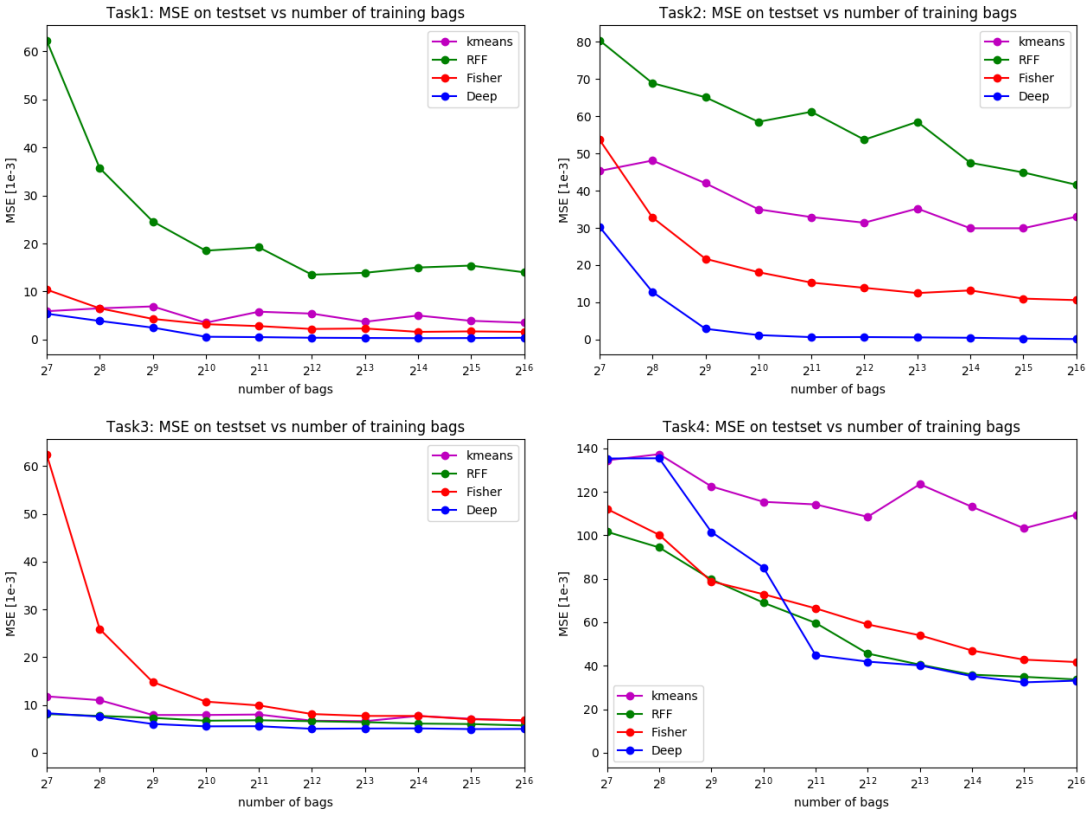


Figure 4.3: MSE between prediction and ground truth on test bags.

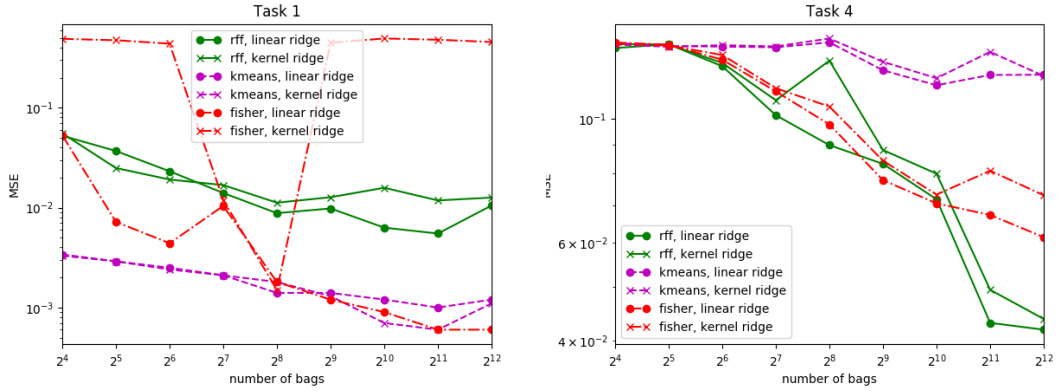


Figure 4.4: Compare linear ridge regression and kernel ridge regression.

fixed-embedding approaches. They also support hypothesis H2: DeepSets outperforms others given large number of training bags, fixed-embedding methods are doing well given small number of training bags.

4.3.2 Point cloud classification

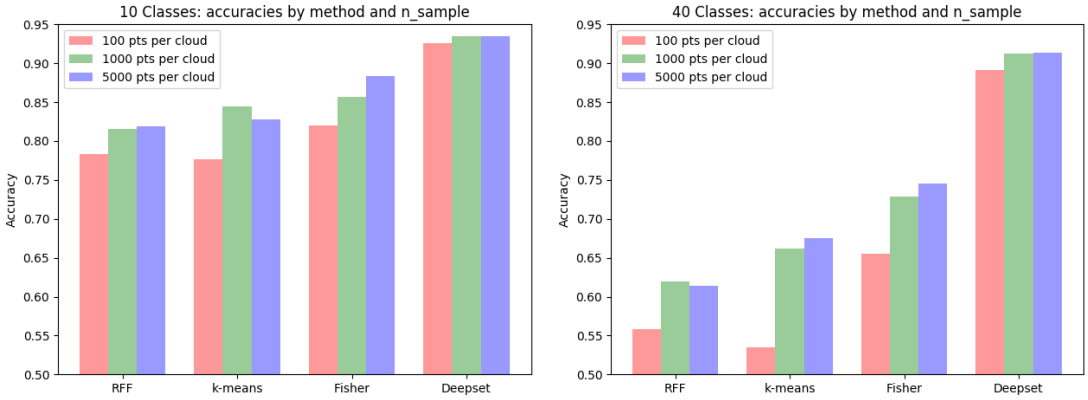


Figure 4.5: Point Cloud: prediction accuracy on test set for different methods and varying bag sizes.

We evaluate model performance on 10-class and 40-class classification tasks. The ModelNet10 and ModelNet40 datasets (Wu et al., 2015) are widely used benchmarks for evaluating computer vision models and algorithms. We have 9843 training clouds, and each cloud contains 10000 cloud points in the form of xyz coordinates. We will extract $N_i \in \{100, 1000, 5000\}$ points uniformly at random for each bag to build our models. We will evaluate hypothesis H1 to find out how bag sizes impact model performance.

We use the default DeepSets architecture for this task. The feature extractor component consists of 3 permutation-equivariant layers followed by tanh as activations. The pooling component takes max pooling. The regressor (here is actually a classifier) includes 2 linear layers with $p = 0.5$ dropout followed by tahh. We tried other architecture settings with different depths and dimensions, and found the results were very close. We speculate that more careful fine-tuning could improve the results a little, but will not overturn the conclusion we drew.

Figure 4.5 shows the prediction accuracy on a test set of 2648 point clouds. It shows (1) DeepSets is the best model. The advantage over all alternatives is more evident on

the 40-class classification task, where DeepSets holds accuracies above 0.9 while all others below 0.75; (2) generally larger bag size helps to improve performance for all methods. There are only two exceptions when we increase bag size from 1000 to 5000: the k-means on the 10-class task and RFF on the 40-class task; (3) quantization-based methods are strong competitors to kernel mean embeddings (in the figure we denote RFF but actually we used orthogonal Fourier features – better than RFF in practice – for embedding).

The results support hypothesis H1: larger bag size (generally) improves performances. For quantization-based approaches, since the embedding function f depends on $p(\mathbf{x})$, large bag size means we have more samples per bag to construct a richer histogram of count statistics or Fisher vector. For DeepSets, large bag sizes help to train parameters of the embedding components of the architecture. Notably, Szabó et al. (2016) suggested optimal bag sizes in theory, which depends on the effective input dimension, the number of training bags, and the smoothness of the regressor.

4.3.3 US presidential election

We collected data for both the 2012 and 2016 US presidential election. The data includes district-level voting results for both Democrats and Republicans, in terms of voting fractions, as well as characteristic individuals from the US Census. The setting assumes the individuals within each district are sampled from the distribution of the district, and distributions of all districts are sampled from an unknown meta-distribution (i.e., the two-level sampling assumption).

The size of the two datasets are (more details in Chapter 3, consider a district as a bag):

- 2012: $l = 840$ bags, bag size N_i=(min: 3091, max: 438037, mean±std:12864±21955, median: 7185), input dimension D =18 real-valued variables + 78 categorical variables (or 4653 after one-hot expansion) + 3 centroids (=4674).

- 2016: $l = 979$ bags, bag size N_i=(min: 2214, max: 256352, mean±std:9421±19940, median: 5633), input dimension D =19 real-valued variables + 101 categorical variables (or 3858 after one-hot expansion) + 3 centroids (=3880).

Here the 3-dimensional centroids are from the Cartesian coordinate system.

Compared to the datasets for estimating population statistics and classifying point clouds, the two datasets have two major differences: the number of training bags l is very small, and the input dimension D is very large. This poses two challenges: (1) DeepSets could fail since there is not enough data to fully train a neural network with a huge number of parameters. (2) Fisher vector could fail since the Fisher vector dimension, which is $2KD$ given K Gaussians and D input dimensions, could be far larger than the number of training bags.

The goal for this task is to build a predictive model so that we can predict the voting preference of *any* demographical group. The exit poll data contains voting fractions for some pre-defined demographical groups, and is the only source of data available for validating our model's predictions.

For DeepSets, we tested several architecture variants: different permutation-equivariant layers; concatenating instance-level "local" embeddings and pooled "global" embedding to enrich feature representation; reducing input by SVD to simplify neural network architecture. In our experiments, more complex architectures did not show to improve performance for this task. Compared to the first two tasks – estimating population statistics and point cloud classification – this task has larger bag sizes and smaller number of bags for training, and may not be enough to train highly complex architectures with too many parameters.

For Fisher vector, since the dimension $d = 2KD$ for K Gaussians and D input dimensions, we encountered the classical problem of *feature dimension far larger than the number of instances* ($d = 1.6M \gg l = 840$ bags for 2012 or $d = 1.9M \gg l = 979$ for 2016). We could reduce the number of Gaussians K for training GMM, but it

would limit the richness of the hypothesis space and introduce high estimation error. In our experiments, we applied SVD to reduce the input dimension to $D = 100$. It explained about 87% of the variance, yet $d = 40K \gg 1K > l$.

For BOW and kernel mean embedding like RFF, the dimension d is fixed (for RFF it is twice of the chosen number of frequencies, $d = 2048$ in our case; for BOW it is the number of clusters, $d = 200$ in our case). These ds are much smaller than d in Fisher vector and DeepSets. The bag-level embeddings are then fed to a (non)linear regressor.

In the experiments, we trained a model on 70% random bags and evaluated on the remaining 30% bags. We repeated 10 times and reported average MSE. We also compared linear and nonlinear regressor with hyperparameters chosen by cross validation.

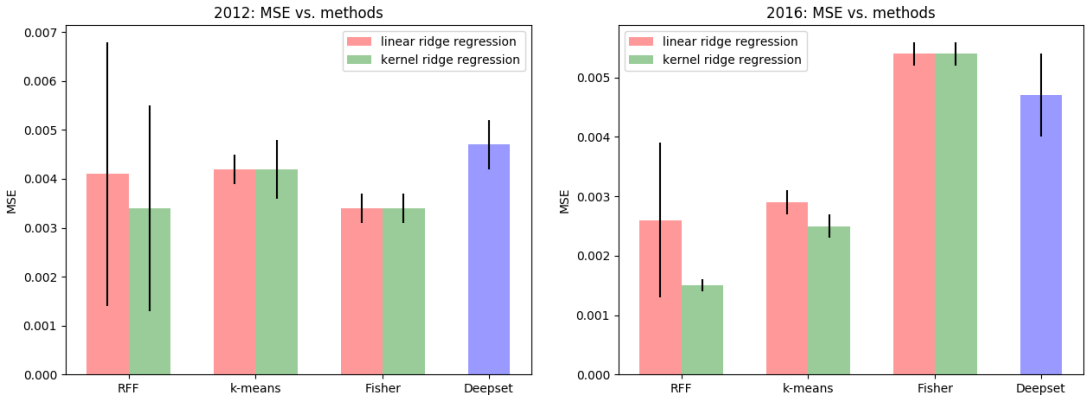


Figure 4.6: Average of MSE on 30% bags over 10 trials.

Figure 4.6 shows the average MSE. It's evident that the fixed-embedding strategy is doing better than DeepSets in average, and nonlinear regressor is doing slightly better than linear regressor. The results support hypothesis H2: that fixed-embeddings are doing just fine given small number of training bags. We expected the results also support hypothesis H3 that large embedding dimension will negatively affects Fisher

vector: for 2016, Fisher vector indeed performs the worst, but for 2012, it's the best performer. It's still unclear why and we don't have enough evidence to support H3.

It's also worth mentioning that the high variance for RFF came from ill-conditioned matrix when computing $(XX^T + \alpha I)^{-1}y$. The solver detected this, used least-square solution instead of solving the dual problem directly, and warned us the result was not guaranteed to be accurate.

4.4 Summary

The choice of method for distribution regression depends on the number of bags, bag sizes, and possibly embedding dimension. We evaluated three hypotheses relevant to these factors on three tasks. Given our experiments, DeepSets is a good choice given large training data, otherwise, fixed-embedding could be better. The results also show that quantization-based embeddings, which is largely ignored by the research community of distribution regression, could be a better choice than kernel mean embedding.

APPENDIX

APPENDIX FOR CGMS

A.1 Computational Complexity

We present the main theorems for the computational complexity of CGMs and the proof sketch, which were developed by Dan Sheldon Sheldon et al. (2013).

Theorem 1 When G is a tree, message passing in the CGM solves the aggregate MAP or marginal inference problems in time $O(N \cdot \min(M^{L^2-1}, L^{2M}))$.

Proof sketch. Because of the hyper-Markov property, the CGM also has the form of a tree-structured graphical model. Message passing gives an exact solution to the MAP or margina inference problem in two passes through the tree, which takes $O(N)$ messages (Koller and Friedman, 2009). In a standard implementation of message passing, the time per message is bounded by the maximum over all factors of the product of the cardinalities of the variables in that factor. However, due to the nature of the hard constraints in the CGM, it is possible to bound the time per message by a smaller number, which is the number of values for the random variable $\mathbf{n}_{ij}$ (details omitted). The number of contingency tables with c entries that sum to M is $\binom{M+c-1}{c-1}$, which is the number of ways to placing M identical balls in c distinct bins. This number is bounded above by M^{c-1} and by $(c-1)^M$, which can be derived as follows:

$$\binom{a+b}{b} = \frac{(a+b)!}{a!b!} = \begin{cases} \frac{a+b}{b} \cdot \frac{a+b-1}{b-1} \cdots \frac{a+2}{2} \cdot \frac{a+1}{1} \leq a^b \\ \frac{b+a}{a} \cdot \frac{b+a-1}{a-1} \cdots \frac{b+2}{2} \cdot \frac{b+1}{1} \leq b^a \end{cases}$$

so $\binom{M+c-1}{c-1} = \binom{M+c-1}{M} = O(\min((c-1)^M, M^{c-1}))$. In a CGM, the table $\mathbf{n}_{ij}$ has L^2 entries, so the number of values for $\mathbf{n}_{ij}$ is $O(\min(M^{L^2-1}, L^{2M}))$, which gives the desired upper bound. □

It is worth noting that the two different upper bounds correspond to the dense and sparse regimes for the contingency tables involved in the computations. For example, when $L^2 \gg M$, the table $\mathbf{n}_{ij}$ is sparse, so it can be stored compactly by recording the values $(x_i^{(m)}, x_j^{(m)})$ for every member of the population. Since there are L^2 possibilities for this pair, and M individuals, the total number of tables is at most L^{2M}. When $M \gg L^2$, then the vast majority of tables will be dense, and the upper bound of M^{L^2-1} is the tighter bound. This number corresponds to a storage scheme that keeps a value from $\{0, \ldots, M\}$ for all but last of the L^2 table entries, which is determined by the others.

For general graphical models, message passing on junction trees can be implemented in a similar fashion. For a clique of size K, the contingency table will have L^K entries, so there are $O(\min(M^{L^K-1}, L^{KM}))$ possible values of the contingency table. This gives us the following result.

Theorem 2 Unless $P = NP$, there is no algorithm for MAP or marginal inference in a CGM that is polynomial in both M and L. This remains true when G is a tree and $N = 4$.

Proof. This proof is by reduction from exact 3-dimensional (3D) matching to a CGM where both M and L grow with the input size. An instance of exact 3D matching consists of finite A_1, A_2 and A_3, each of size M, and a set of hyperedges $T \subseteq A_1 \times A_2 \times A_3$. A hyperedge $e = (a_1, a_2, a_3) \in T$ is said to *cover* a_1, a_2, a_3. The problem of determining whether there is a subset $S \subseteq T$ of size M that covers each element is NP-complete (Karp, 1972).

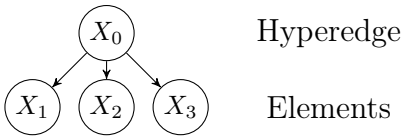


Figure A.1: Reduction from 3-dimensional matching.

To reduce exact 3D matching to inference in CGMs, define a graphical model with random variables X_0, X_1, X_2, X_3 such that $X_0 \in \{1, \cdots, |T|\}$ is a hyperedge chosen uniformly at random, and X_1, X_2, and X_3 are the elements covered by X_0 (see Figure A.1). Define the observed counts to be $n_i(a) = 1$ for all $a \in A_i, i = 1, 2, 3$, which specify that each element is covered exactly once by one of M randomly selected hyperedges. These counts have nonzero probabilities if and only if there is an exact 3D matching. Thus, MAP or marginal inference can be used to decide exact 3D matching. For MAP, there exist tables $\mathbf{n}_{0i}$ such that $p(\{\mathbf{n}_{0i}\}, \{\mathbf{n}_i\}) > 0$ if and only if there is a 3D matching. For marginal inference $p(\{\mathbf{n}_i\}) > 0$ if and only if there is a 3D matching.

Because the model used in the reduction is a tree with only four variables, the hardness result clearly holds under that restricted case. □

Theorem 5 Suppose the NLBP message passing updates converge and the resulting vector $\mathbf{z}$ has strictly positive energies. Then $\mathbf{z}$ is a constrained stationary point of $F(\mathbf{z})$ in problem (2.11) with respect to the set $\mathbb{L}_M$. If G is a tree and $E(\mathbf{z})$ is convex, then $\mathbf{z}$ is a global minimum.

Proof. The proof follows Yedidia et al. (2000, 2005). We will write the Lagrangian of (2.11) and set its gradients to zero to derive the first-order optimality conditions, and then show that these are satisfied by a certain set of Lagrangian multipliers if NLBP converges. The Lagrangian is

$$L(\mathbf{z}, \boldsymbol{\lambda}) = E(\mathbf{z}) - H_B(\mathbf{z}) + \sum_i \lambda_i (\sum_{x_i} z_i(x_i) - M)$$
$$+ \sum_i \sum_{j \in N(i)} \sum_{x_i} \lambda_{ji}(x_i) \left(z_i(x_i) - \sum_{x_j} z_{ij}(x_i, x_j) \right).$$

Since we only consider vectors $\mathbf{z}$ that are strictly positive, we can drop the inequality constraints $\mathbf{z} \geq 0$ when writing the Lagrangian. The partial derivative with respect to the primal variables are:

$$\frac{\partial L(\mathbf{z}, \boldsymbol{\lambda})}{\partial z_{ij}(x_i, x_j)} = \frac{\partial E(\mathbf{z})}{\partial z_{ij}(x_i, x_j)} + \log z_{ij}(x_i, x_j) + 1 - \lambda_{ji}(x_i) - \lambda_{ij}(x_j),$$
$$\frac{\partial L(\mathbf{z}, \boldsymbol{\lambda})}{\partial z_i(x_i)} = (1 - \nu_i)(\log z_i(x_i) + 1) + \lambda_i + \sum_{j \in N(i)} \lambda_{ji}(x_i).$$

Here we have used the assumption that $E(\mathbf{z})$ depends only on edge variables, so $\frac{\partial E(\mathbf{z})}{\partial z_i(x_i)} = 0$. By setting these expressions to zero and factoring out terms that are constant with respect to the individual edge and node marginal tables, we obtain the following first-order conditions:

$$z_{ij}(x_i, x_j) \propto \exp\left\{ \lambda_{ji}(x_i) + \lambda_{ij}(x_j) - \frac{\partial E(\mathbf{z})}{\partial z_{ij}(x_i, x_j)} \right\},$$
$$z_i(x_i) \propto \exp\left\{ \frac{1}{\nu_i - 1} \sum_{j \in N(i)} \lambda_{ji}(x_i). \right\} \tag{A.1}$$

Assume NLBP has converged to a particular set of messages $\{m_{ji}(x_i)\}$ and marginals $\mathbf{z}$ that satisfy Eq. (2.8), (2.9) and (2.10). Construct Lagrangian multipliers as $\lambda_{ji}(x_i) = \log \prod_{k \in N(i) \setminus j} m_{ki}(x_i)$. By substituting these values into Eq. (A.1) and simplifying the node marginal expression, we obtain the fixed point equations for the marginals from the NLBP algorithm, which are assumed to be satisfied. Therefore, for this set of Lagrangian multipliers, the gradient with respect to the primary variables $\mathbf{z}$ is zero. Finally, it is a standard exercise to check that the normalization and consistency

constraints of $\mathbf{z}$ are satisfied when message passing converges, so that the gradient of $L(\mathbf{z}, \boldsymbol{\lambda})$ with respect to $\boldsymbol{\lambda}$ is zero.

This establishes that all partial derivatives of $L(\mathbf{z}, \boldsymbol{\lambda})$ are zero, *i.e.,* $\mathbf{z}$ is an (interior) constrained stationary point. If G is a tree and $E\mathbf{z})$ is convex, then the problem is convex and therefore $\mathbf{z}$ must be a global minimum. □

A.2 Other Proximal Algorithms for CGMs

The Mirror Descent update is:

$$\begin{aligned}
\mathbf{z}_{t+1} &= \arg\min_{\mathbf{z}\in\mathbb{L}_M} \langle g_t, \mathbf{z}\rangle + \frac{1}{\eta_t} B_\varphi(\mathbf{z}, \mathbf{z}_t) + R(\mathbf{z}) \\
&= \arg\min_{\mathbf{z}} \eta_t \langle g_t, \mathbf{z}\rangle + \varphi(\mathbf{z}) - \varphi(\mathbf{z}_t) - \langle \nabla\varphi(\mathbf{z}_t), \mathbf{z} - \mathbf{z}_t\rangle + \eta_t R(\mathbf{z}) \\
&= \arg\min_{\mathbf{z}} \langle \eta_t g_t - \nabla\varphi(\mathbf{z}_t), \mathbf{z}\rangle + \varphi(\mathbf{z}) + \eta_t R(\mathbf{z}) \qquad (\text{let } \varphi = R = -H_B - \langle \boldsymbol{\theta}, \mathbf{z}\rangle) \\
&= \arg\min_{\mathbf{z}} \langle \eta_t g_t + \nabla H_B(\mathbf{z}_t) + \boldsymbol{\theta}, \mathbf{z}\rangle - H_B(\mathbf{z}) - \langle \boldsymbol{\theta}, \mathbf{z}\rangle - \eta_t (H_B(\mathbf{z}) + \langle \boldsymbol{\theta}, \mathbf{z}\rangle) \\
&= \arg\min_{\mathbf{z}} \langle \eta_t g_t + \nabla H_B(\mathbf{z}_t) - \eta_t \boldsymbol{\theta}, \mathbf{z}\rangle - (1 + \eta_t) H_B(\mathbf{z}) \\
&= \arg\min_{\mathbf{z}} - \left\langle \frac{\eta_t}{1 + \eta_t} \left(\boldsymbol{\theta} - g_t - \frac{1}{\eta_t} \nabla H_B(\mathbf{z}_t) \right), \mathbf{z} \right\rangle - H_B(\mathbf{z}), \qquad (A.2)
\end{aligned}$$

and the convergence rate is $O(\ln(t)/t)$ as well (Duchi et al., 2010).

In the end, we will present the accelerated RDA (Nesterov, 2005), which was initially used to solve smooth convex optimization where the uniform average of all past gradients is replaced by an weighted average that emphasizes more recent gradients. Based on the slight variant of Xiao (2010):

$$\begin{aligned}
\mathbf{z}_{t+1} &= \arg\min_{\mathbf{z}\in\mathbb{L}_M} \langle \tilde{g}_t, \mathbf{z}\rangle - \frac{\beta_t + L'}{A_t} (H_B(\mathbf{z}) + \langle \boldsymbol{\theta}, \mathbf{z}\rangle) - H_B(\mathbf{z}) - \langle \boldsymbol{\theta}, \mathbf{z}\rangle \\
&= \arg\min_{\mathbf{z}} \left\langle \tilde{g}_t - \frac{A_t + \beta_t + L'}{A_t} \boldsymbol{\theta}, \mathbf{z} \right\rangle - \frac{A_t + \beta_t + L'}{A_t} H_B(\mathbf{z}). \\
&= \arg\min_{\mathbf{z}} - \left\langle \boldsymbol{\theta} - \frac{A_t}{A_t + \beta_t + L'} \tilde{g}_t, \mathbf{z} \right\rangle - H_B(\mathbf{z}) \qquad (A.3)
\end{aligned}$$

where L' is the Lipschitz parameter for $L(\mathbf{z})$, and A_t, b_t are defined to obtain the optimal convergence rate $O(4LD^2/t^2)$ where D is the diameter of the marginal polytope $\mathbb{L}_M$ that is measured by the strongly convex auxiliary function $\varphi = R = -H_B - \langle \boldsymbol{\theta}, \mathbf{z} \rangle$ and satisfies $-H_B(\mathbf{z}^*) \leq D^2$. According to Theorem 6 in (Xiao, 2010), $A_t = t(t+1)/4$, we get

$$\mathbf{z}_{t+1} = \arg\min_{\mathbf{z} \in \mathbb{L}_M} - \left\langle \boldsymbol{\theta} - \frac{t(t+1)}{t(t+1) + 4L' + 4\beta_t} \tilde{g}_t, \mathbf{z} \right\rangle - H_B(\mathbf{z}), \tag{A.4}$$

thus we update $\boldsymbol{\theta}_t = \boldsymbol{\theta} - \frac{t(t+1)}{t(t+1)+4L'+4\beta_t} \tilde{g}_t$ and then run standard BP oracle given $\boldsymbol{\theta}_t$ to update $\mathbf{z}_{t+1}$. Note we could set $\beta_t = 0$ since $\varphi = R$ is strongly convex.

BIBLIOGRAPHY

Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Lin Hsuan-Tien. *Learning From Data*, volume 4. AMLBook, 2012. 94

U.S. Census Bureau; American Community Survey (ACS). Public Use Microdata Sample (PUMS), 2010-2014 ACS 5-year PUMS and 2015 ACS 1-year PUMS, 2010-2014. Available from https://www.census.gov/programs-surveys/acs/data/pums.html. 57, 61

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003. 38

Garrett Bernstein and Daniel Sheldon. Consistently estimating markov chains with noisy aggregate data. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 1142–1150, 2016. 56

Garrett Bernstein, Ryan McKenna, Tao Sun, Daniel Sheldon, Michael Hay, and Gerome Miklau. Differentially private learning of undirected graphical models using collective graphical models. In *International Conference on Machine Learning*, pages 478–487, 2017. 11

Maria Blettner, Willi Sauerbrei, Brigitte Schlehofer, Thomas Scheuchenpflug, and Christine Friedenreich. Traditional reviews, meta-analyses and pooled analyses in epidemiology. *International journal of epidemiology*, 28(1):1–9, 1999. 7

Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967. 37

Ken Chatfield, Victor S Lempitsky, Andrea Vedaldi, and Andrew Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2011. 91

A JêŐň Connolly, I Csabai, AS Szalay, DC Koo, RG Kron, and JA Munn. Slicing through multicolor space: Galaxy redshifts from broadband photometry. *arXiv preprint astro-ph/9508100*, 1995. 2, 85

Tri Dao, Christopher M De Sa, and Christopher Ré. Gaussian quadrature for kernel features. In *Advances in Neural Information Processing Systems*, pages 6109–6119, 2017. 89

A. P. Dawid and S. L. Lauritzen. Hyper Markov laws in the statistical analysis of decomposable graphical models. *The Annals of Statistics*, 21(3):1272–1317, 1993. 25

Yves-Alexandre de Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3, 2013. 1, 48

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. ISSN 00359246. 63

Thomas G. Dietterich, Richard H. Lathrop, and TomÃąs Lozano-PÃĺrez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1):31 – 71, 1997. ISSN 0004-3702. 56

John C Duchi, Shai Shalev-Shwartz, Yoram Singer, and Ambuj Tewari. Composite objective mirror descent. In *COLT*, pages 14–26, 2010. 110

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Theoretical Computer Science*, 9(3-4):211–407, 2013. 48

Tom Fawcett. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874, 2006. 80

X Yu Felix, Ananda Theertha Suresh, Krzysztof M Choromanski, Daniel N Holtmann-Rice, and Sanjiv Kumar. Orthogonal random features. In *Advances in Neural Information Processing Systems*, pages 1975–1983, 2016. 89

Seth Flaxman, Dougal Sutherland, Yu-Xiang Wang, and Yee Whye Teh. Understanding the 2016 US presidential election using ecological inference and distribution regression with census microdata. *stat.AP arXiv:1611.03787*, 2016. 55, 57, 72, 73, 78, 85

Seth R Flaxman, Yu-Xiang Wang, and Alexander J Smola. Who supported Obama in 2012?: Ecological inference through distribution regression. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 289–298. ACM, 2015. 2, 4, 55, 56, 57, 59, 61, 72, 73, 82, 85

Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *USENIX Security Symposium*, pages 17–32, 2014. 1

David A. Freedman, Stephen P. Klein, Michael Ostland, and Michael Roberts. On "solutions" to the ecological inference problem. *Journal of the American Statistical Association*, 93 (444):1518–22, 1998. 56

Qiang Fu, Huahua Wang, and Arindam Banerjee. Bethe-admm for tree decomposition based parallel map inference. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pages 222–231. AUAI Press, 2013. 39

Dario Garcıa-Garcıa and Robert C Williamson. Degrees of supervision. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems Workshops (NIPS)*, pages 897–904, 2011. 6

W.R. Gilks and P. Wild. Adaptive Rejection sampling for Gibbs Sampling. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 41(2):337–348, 1992. 43

Rahul Gupta, Ajit A Diwan, and Sunita Sarawagi. Efficient inference with cardinality-based clique potentials. In *Proceedings of the 24th international conference on Machine learning*, pages 329–336. ACM, 2007. 58, 60

Hossein Hajimirsadeghi and Greg Mori. Multi-instance classification by max-margin training of cardinality-based markov networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016. 56

Trevor J Hefley and Mevin B Hooten. Hierarchical species distribution models. *Current Landscape Ecology Reports*, 1(2):87–97, 2016. 1

Jerónimo Hernández-González, Iñaki Inza, and Jose A Lozano. Learning bayesian network classifiers from label proportions. *Pattern Recognition*, 46(12):3425–3440, 2013. 59, 60

Jerónimo Hernández-González, Iñaki Inza, and Jose A Lozano. Weak supervision and other non-standard classification problems: a taxonomy. *Pattern Recognition Letters*, 69:49–55, 2016. 6

T. Heskes. Convexity arguments for efficient minimization of the Bethe and Kikuchi free energies. *Journal of Artificial Intelligence Research*, 26(1):153–190, 2006. 15, 29

Tom Heskes. Stable fixed points of loopy belief propagation are minima of the Bethe free energy. *Advances in Neural Information Processing Systems (NIPS)*, 15:359–366, 2003. 32, 35

Jay Jin. *Detection of Sources of Harmful Radiation using Portable Sensors*. PhD thesis, Lawrence Livermore National Laboratory, 2014. 85

RM Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, 1972. 107

Gary King. The future of ecological inference research: A reply to freedman et al. *Journal of the American Statistical Association*, 94:352–355, March 1999. 56

Gary King. *A solution to the ecological inference problem: Reconstructing individual behavior from aggregate data*. Princeton University Press, 2013. 56

D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009. 51, 106

Hendrik Kück and Nando de Freitas. Learning about individuals from group statistics. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 332–339. AUAI Press, 2005. 56, 59, 60

Sanjiv Kumar and Henry A Rowley. Classification of weakly-labeled data with partial equivalence relations. 2007. 5

Ludmila I Kuncheva. Full-class set classification using the hungarian algorithm. *International Journal of Machine Learning and Cybernetics*, 1(1-4):53–61, 2010. 5

Quoc Le, Tamás Sarlós, and Alex Smola. Fastfood-approximating kernel expansions in loglinear time. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013. 73

Fan Li and Graham Taylor. Alter-CNN: An approach to learning from label proportions with application to ice-water classification. In *NIPS 2015 Workshop on Learning and privacy with incomplete data and weak supervision*, 2015. 59, 60, 61

Li-Ping Liu, Daniel Sheldon, and Thomas G. Dietterich. Gaussian approximation of collective graphical models. In *International Conference on Machine Learning (ICML)*, volume 32, pages 1602–1610, 2014. 25, 44, 46

Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pages 94–103. IEEE, 2007. 1

Darakhshan J Mir, Sibren Isaacman, Ramón Cáceres, Margaret Martonosi, and Rebecca N Wright. DP-WHERE: Differentially private modeling of human mobility. In *IEEE International Conference on Big Data*, pages 580–588, 2013. 48

Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, Bernhard Schölkopf, et al. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, 10(1-2):1–141, 2017. 88

Boris Muzellec, Richard Nock, Giorgio Patrini, and Frank Nielsen. Tsallis regularized optimal transport and ecological inference. In *AAAI*, pages 2387–2393, 2017. 56

Yu Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005. 40, 110

Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009. 38

Michelle Ntampaka, Hy Trac, Dougal J Sutherland, Nicholas Battaglia, Barnabás Póczos, and Jeff Schneider. A machine learning approach for dynamical mass measurements of galaxy clusters. *The Astrophysical Journal*, 803(2):50, 2015. 85

Michelle Ntampaka, Hy Trac, Dougal J Sutherland, Sebastian Fromenteau, Barnabás Póczos, and Jeff Schneider. Dynamical mass measurements of contaminated galaxy clusters using machine learning. *The Astrophysical Journal*, 831(2):135, 2016. 85

Giorgio Patrini, Richard Nock, Tiberio Caetano, and Paul Rivera. (Almost) no label no cry. In *Advances in Neural Information Processing Systems*, pages 190–198, 2014. 5, 56, 59, 75, 76

Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988. 32

Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007. 90

Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. *Computer Vision–ECCV 2010*, pages 143–156, 2010. 90

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017a. 2, 9, 85, 88, 92

Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017b. 85

Novi Quadrianto, Alex J Smola, Tiberio S Caetano, and Quoc V Le. Estimating labels from label proportions. *Journal of Machine Learning Research*, 10(Oct):2349–2374, 2009. 59

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008. 89, 92

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of machine learning research*, 5(Jan):101–141, 2004. 71

Evan Rosenman and Nitin Viswanathan. Using poisson binomial glms to reveal voter preferences. *arXiv preprint arXiv:1802.01053*, 2018. 60

Eduardo Rozo and Eli S Rykoff. redmapper ii: X-ray and sz performance benchmarks for the sdss catalog. *The Astrophysical Journal*, 783(2):80, 2014. 85

Stefan Rueping. SVM classifier estimation from group probabilities. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 911–918, 2010. 59

Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *International journal of computer vision*, 105 (3):222–245, 2013. 91

Alexander A. Schuessler. Ecological inference. *Proceedings of the National Academy of Sciences*, 96(19):10578–10581, 1999. doi: 10.1073/pnas.96.19.10578. 56

George Arthur Frederick Seber et al. Estimation of animal abundance and related parameters. 1973. 1

Daniel Sheldon. *Manipulation of PageRank and Collective Hidden Markov Models*. PhD thesis, Cornell University, 2009. 17

Daniel Sheldon. Discrete adaptive rejection sampling. Technical Report UM-CS-2013-012, School of Computer Science, University of Massachusetts, Amherst, Massachusetts, May 2013. 43

Daniel Sheldon and Thomas Dietterich. Collective graphical models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1161–1169, 2011. 1, 2, 17, 18, 19, 23, 24, 26, 30, 43, 56

Daniel Sheldon, M. A. S. Elmohamed, and Dexter Kozen. Collective inference on Markov models for modeling bird migration. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1321–1328, 2007. 1, 17

Daniel Sheldon, Tao Sun, Akshat Kumar, and Thomas G. Dietterich. Approximate inference in collective graphical models. In *International Conference on Machine Learning (ICML)*, volume 28, pages 1004–1012, 2013. 46, 56, 106

Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010. 48

Marco Stolpe and Katharina Morik. Learning from label proportions by optimizing cluster model selection. *Machine Learning and Knowledge Discovery in Databases*, pages 349–364, 2011. 59

B. L. Sullivan, C. L. Wood, M. J. Iliff, R. E. Bonney, D. Fink, and S. Kelling. eBird: A citizen-based bird observation network in the biological sciences. *Biological Conservation*, 142(10):2282 – 2292, 2009. 17

Tao Sun, Dan Sheldon, and Akshat Kumar. Message passing for collective graphical models. In *International Conference on Machine Learning*, pages 853–861, 2015. 56

Tao Sun, Dan Sheldon, and Brendan O'Connor. A probabilistic approach for learning with label proportions applied to the US presidential election. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 445–454. IEEE, 2017. 2, 4, 5

R. Sundberg. Some results about decomposable (or Markov-type) models for multidimensional contingency tables: distribution of marginals and partitioning of tests. *Scandinavian Journal of Statistics*, 2(2):71–79, 1975. 18, 24

Zoltán Szabó, Arthur Gretton, Barnabás Póczos, and Bharath Sriperumbudur. Two-stage sampled learning theory on distributions. In *Artificial Intelligence and Statistics*, pages 948–957, 2015. 4, 89

Zoltán Szabó, Bharath K Sriperumbudur, Barnabás Póczos, and Arthur Gretton. Learning theory for distribution regression. *Journal of Machine Learning Research*, 17(152):1–40, 2016. 56, 57, 61, 102

Daniel Tarlow, Kevin Swersky, Richard S Zemel, Ryan Prescott Adams, and Brendan J Frey. Fast exact inference for recursive cardinality models. In *Proceedings of the Twenty-Eighth Conference Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2012. 58, 60, 65, 66

Luke Vilnis, David Belanger, Daniel Sheldon, and Andrew McCallum. Bethe projections for non-local inference. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pages 892–901. AUAI Press, 2015. 10, 37, 38, 39, 40, 53

M.J. Wainwright and M.I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008. 13, 14, 32

Kevin Winner, Garrett Bernstein, and Dan Sheldon. Inference in a partially observed queuing model with applications in ecology. In *International Conference on Machine Learning*, pages 2512–2520, 2015. 1

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 95, 101

Lin Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11(Oct):2543–2596, 2010. 39, 110, 111

Xiaolin Yang, Stephen E Fienberg, and Alessandro Rinaldo. Differential privacy for protecting multi-dimensional contingency table data: Extensions and applications. *Journal of Privacy and Confidentiality*, 4(1):5, 2012. 50

Jonathan S Yedidia, William T Freeman, and Yair Weiss. Generalized belief propagation. In *Advances in Neural Information Processing Systems (NIPS)*, volume 13, pages 689–695, 2000. 31, 32, 108

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. 108

Felix Yu, Dong Liu, Sanjiv Kumar, Jebara Tony, and Shih-Fu Chang. $\propto$SVM for learning with label proportions. In *Proceedings of The 30th International Conference on Machine Learning*, pages 504–512, 2013. 56, 59

Felix X Yu, Krzysztof Choromanski, Sanjiv Kumar, Tony Jebara, and Shih-Fu Chang. On learning from label proportions. *arXiv preprint arXiv:1402.5902*, 2014. 72, 74

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep sets. *arXiv preprint arXiv:1703.06114*, 2017. 2, 88, 91, 95, 96

Cor Zonneveld. Estimating death rates from transect counts. *Ecological Entomology*, 16(1): 115–121, 1991. 1