Department of Computing

On Conditional Random Fields: Applications, Feature Selection, Parameter Estimation and Hierarchical Modelling

Tran The Truyen

This thesis is presented for the Degree of
Doctor of Philosophy
of Curtin University of Technology

February 2008

Declaration

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgment has been made.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Signature:_____

Date:_____

Contents

List of Figures

List of Tables

Abstract

There has been a growing interest in stochastic modelling and learning with complex data, whose elements are structured and interdependent. One of the most successful methods to model data dependencies is *graphical models*, which is a combination of graph theory and probability theory. This thesis focuses on a special type of graphical models known as Conditional Random Fields (CRFs) (Lafferty *et al.*, 2001), in which the output state spaces, when conditioned on some observational input data, are represented by *undirected* graphical models. The contributions of thesis involve both (a) broadening the current applicability of CRFs in the real world and (b) deepening the understanding of theoretical aspects of CRFs.

On the application side, we empirically investigate the applications of CRFs in two real world settings. The first application is on a novel domain of Vietnamese *accent restoration*, in which we need to restore accents of an accent-less Vietnamese sentence. Experiments on half a million sentences of news articles show that the CRF-based approach is highly accurate. In the second application, we develop a new CRF-based *movie recommendation* system called *Preference Network* (PN). The PN jointly integrates various sources of domain knowledge into a large and densely connected Markov network. We obtained competitive results against well-established methods in the recommendation field.

On the theory side, the thesis addresses three important theoretical issues of CRFs: *feature selection*, *parameter estimation* and *modelling recursive sequential data*. These issues are all addressed under a general setting of *partial supervision* in that training labels are not fully available.

For feature selection, we introduce a novel learning algorithm called *AdaBoost.CRF* that incrementally selects features out of a large feature pool as learning proceeds. AdaBoost.CRF is an extension of the standard boosting methodology to structured and partially observed data. We demonstrate that the AdaBoost.CRF is able to eliminate irrelevant features and as a result, returns a very compact feature set without significant loss of accuracy.

Parameter estimation of CRFs is generally intractable in arbitrary network structures. This thesis contributes to this area by proposing a learning method called *AdaBoost.MRF* (which

stands for AdaBoosted Markov Random Forests). As learning proceeds AdaBoost.MRF incrementally builds a tree ensemble (a forest) that cover the original network by selecting the best spanning tree at a time. As a result, we can approximately learn many rich classes of CRFs in linear time.

The third theoretical work is on modelling *recursive, sequential* data in that each level of resolution is a Markov sequence, where each state in the sequence is also a Markov sequence at the finer grain. One of the key contributions of this thesis is *Hierarchical Conditional Random Fields* (HCRF), which is an extension to the currently popular sequential CRF and the recent semi-Markov CRF (Sarawagi and Cohen, 2004). Unlike previous CRF work, the HCRF does not assume any fixed graphical structures. Rather, it treats structure as an uncertain aspect and it can estimate the structure automatically from the data. The HCRF is motivated by Hierarchical Hidden Markov Model (HHMM) (Fine *et al.*, 1998). Importantly, the thesis shows that the HHMM is a special case of HCRF with slight modification, and the semi-Markov CRF is essentially a flat version of the HCRF.

Central to our contribution in HCRF is a polynomial-time algorithm based on the Asymmetric Inside Outside (AIO) family developed in (Bui *et al.*, 2004) for learning and inference. Another important contribution is to extend the AIO family to address learning with missing data and inference under partially observed labels. We also derive methods to deal with practical concerns associated with the AIO family, including numerical overflow and cubic-time complexity. Finally, we demonstrate good performance of HCRF against rivals on two applications: indoor video surveillance and noun-phrase chunking.

Acknowledgments

For the last four years I have been fortunate to join IMPCA and work with many people who have helped to shape my reasoning, research and future. My deepest appreciation goes to my principal supervisor, Prof. Svetha Venkatesh, for giving me the opportunity to pursue research at IMPCA, teaching me a great deal about research and supporting me all along the way. Her energy and close guidance have helped me stay on the right track, and at the same time given me freedom to pursue my own interest.

I would also like to thank my co-supervisor, Dr. Hung Bui, for introducing me to IMPCA and shaping my study direction. His sharp view and critical thinking are influential and valuable for anyone who want to follow the research path.

I would like to express my gratitude to my second co-supervisor, Dr. Dinh Phung, for his help not only in research but also in life. Things would have been much more difficult for me without his support. Together with Dinh, I have been able to advance my knowledge and skills quickly, making research an enjoyable part of life.

My thanks go to IMPCA for financial support and providing an interesting research environment. Together with Ba Tu, Dinh, Dzung, Hai, Kha, Nam and Sonny, I have enjoyed every Vicobec moment and lunch-time discussion. Other IMPCA guys have also been very helpful: Brett, Daniel, Graeme, Mihai, Patrick, Senjian, Thi and Wanquan. My special thank goes to Ms Mary Mulligan for her administrative support and proofreading.

This thesis is dedicated to my parents for their love, support, wisdom and encouragement wherever I am.

Relevant Publications

Part of this thesis has been published in refereed conference papers. The details are as follows:

Chapter 5:

- Truyen, T.T., Phung, D.Q., Venkatesh, V. (2007). Preference Networks: probabilistic models for recommendation systems. In *6th Australasian Data Mining Conference (AusDM)*, Dec, Gold Coast, Australia.

Chapter 6:

- Truyen, T.T., Bui, H.H., Venkatesh, V. (2005). Boosted Markov networks for activity recognition. In *2nd International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, Dec, Melbourne, Australia.

Chapter 7:

- Truyen, T.T., Phung, D.Q., Bui, H.H., Venkatesh, V. (2006). AdaBoost.MRF: boosted Markov random forests and application to multilevel activity recognition. In *Computer Vision and Pattern Recognition (CVPR)*, Jun, New York, USA.

Abbreviations

Abbreviation	Description
AdaBoost.MRF	AdaBoosted Markov Random Forest
AHMM	Abstract Hidden Markov Model
AIO	Asymmetric Inside Outside
BN	Bayesian Network
BP	Belief Propagation
CRF	Conditional Random Field
DBN	Dynamic Bayesian Network
DCRF	Dynamic Conditional Random Field
EM	Expectation-Maximisation
ESS	Expected Sufficient Statistics
GM	Graphical Model
HCRF	Hierarchical Conditional Random Field
HMM	Hidden Markov Model
HHMM	Hierarchical Hidden Markov Model
ICM	Iterated Conditional Mode
KL	Kullback-Leibler
LHMM	Layered Hidden Markov Model
MAE	Mean Absolute Error
MAP	Maximum A Posteriori
MaxEnt	Maximum Entropy
MCMC	Markov Chain Monte Carlo
ML	Maximum Likelihood
MLE	Maximum Likelihood Estimation
MRF	Markov Random Field
NLP	Natural Language Processing
NNMF	Non-Negative Matrix Factorisation
NP	Noun Phrase
PCFG	Probabilistic Context-Free Grammar
PN	Preference Network
PoE	Product of Experts
POS	Part-of-Speech
RB	Rao-Blackwellisation
RBGS	Rao-Blackwellised Gibbs Sampling
RMN	Relational Markov Network
SVM	Support Vector Machine
WJW	Wainwright, Jaakkola and Willsky

Chapter 1

Introduction

1.1 Motivations

There has been a growing interest in stochastic modelling *structural patterns* with wide spread application in many areas including language processing, bioinformatics, computer vision and social networks. For example, in Natural Language Processing (NLP) (Manning and Schütze, 1999) we are often interested in inferring the (partial or full) syntax tree of a sentence, the hierarchical structure of a document, and the sequence of named entities (e.g. person name, location) in a sentence. In image understanding, the underlying scene of an image can be modelled as a 2-D grid in which each node corresponds to the scene of a raw image pixel. In consumer networks, preferences (e.g. like/dislike) expressed by a set of users on a set of common products and services are interdependent through user-product interactions.

These examples share a common setting in that given some observational data z, which can be easily observed or obtained, we are more interested in modelling and inferring about the structural patterns x emerging from the data. In probabilistic modelling, inferring about x involves computing the conditional distribution $\Pr(x|z)$. There are two general approaches to this problem. The first approach assumes that the underlying pattern x *generates* the observational data z in a *generative process* given by $\Pr(z|x)$. To infer about x, we resort to the Bayes rule $\Pr(x|z) \propto \Pr(x)\Pr(z|x)$. Hence, the problem is broken into two sub-problems: modelling the pattern itself in $\Pr(x)$ and modelling the data generation in $\Pr(z|x)$.

The second approach is more direct as we model the required conditional distribution $\Pr(x|z)$ directly without worrying about $\Pr(x)$. This is particularly important when the data generation distribution $\Pr(z|x)$ is complex, whilst $\Pr(x|z)$ can be quite simple. It also eliminates the potential danger of the generative assumption, which we do not really know

in practice. This approach is often referred to as *discriminative modelling*[1].

Given the discriminative setting the next important question is how can the structural patterns be represented? The main requirements are that the presentation should be both expressive to incorporate various data aspects (e.g. visible and hidden variables), to capture the inherent relationship between the data z and the pattern x, and formal enough to characterise the nature of model estimation and inference.

Graphical models (Pearl, 1988; Lauritzen, 1996) are an important class of probabilistic methods that successfully meet these requirements. They combine the probabilistic theory and the graph theory in a seamless manner. Patterns are represented by a network in which each pattern element is encoded by a node or a subset of nodes, and interactions between pattern elements are materialised by edges between nodes. There are two main types of interaction: *causality* is encoded in *directed* models (also known as Bayesian Networks (BNs)), and *correlation* in *undirected* models (also known as Markov Random Fields (MRFs)). Corresponding to the interaction types, the interaction strength is materialised by local conditional distributions in the directed cases and clique potentials in the undirected cases.

This thesis focuses on a recently introduced sub-class of undirected graphical models that support discriminative modelling known as Conditional Random Field (CRF) (Lafferty *et al.*, 2001). More specifically, given each observational data z, in CRF, the pattern x is represented as a standard MRF. Thus, given multiple data instances, we have multiple graphical models and these models generally share the same set of parameters. CRFs are often parameterised as conditional exponential distributions, which are also known as multi-class logistic regression.

Given these properties, the CRF is at the conjunction between two major areas: probabilistic data structure modelling and statistical machine learning. As standard undirected graphical models, inference in sequential CRFs is very efficient. Equipped with recent advances in numerical optimisation, this enables learning large-scale CRFs with millions of parameters and millions of data instances. Its unique position is perhaps the reason beside the success of the CRF in various areas, including bioinformatics, computer vision and computational linguistics.

However, this position also poses many theoretical challenges for the CRF as a structure modelling and learning machinery. In what follows, we limit to those issues that will be addressed in this thesis.

An issue which has received limited attention in the discriminative setting is *missing pat-*

[1]There is a popular technique to discriminatively estimate $\Pr(x|z)$ from the generative modelling using the Bayes rule. (Minka, 2005a) has made clear that this is *not* discriminative modelling but *discriminative training*.

tern variables in the data (e.g. see (Quattoni *et al.*, 2005)). In learning of standard discriminative models, pattern labels are assumed to be fully available. This style of learning is often called *supervised learning* in the statistical machine learning literature. This contrasts with the other extreme known as *unsupervised learning*, where patterns of interest are totally missing. In this thesis, we are interested in the situation where partial pattern labels are available, and thus we term it by *partially supervised learning*. Typically, this issue arises when there are intrinsic latent variables that are not shown in the data, or there are missing or damaged parts of the patterns due to environmental noise or manual processing.

Another important issue is *feature selection*. Features in the CRF framework are some aspects extracted from observational data z, and often each feature is associated with a free parameter. As we have mentioned, many domains involves millions of features (and therefore parameters), and this is very computationally demanding. Besides, since we do not have to model z, it can be tempting to generate as many features as possible where many of them can be irrelevant for the purpose of modelling the pattern x. In those cases, selecting a compact subset of features is very critical to the success of the CRF implementation.

Perhaps one of the biggest obstacles to adopt CRFs is *parameter estimation in networks with arbitrary structures* but there has been limited work in this area (e.g. see (Sutton and McCallum, 2005, 2007b)). It is well-known that inference in graphical models is only efficient when the structures are chains or trees, but it is intractable in general. It is even worse in learning which typically involves many inference steps in an iterative manner. Typically, one resorts to approximate inference methods (Pearl, 1988; Geman and Geman, 1984; Jordan *et al.*, 1999; Wainwright *et al.*, 2003a, 2005b), but these may corrupt the parameter update steps because we generally assume that the inference steps are exact.

In addition, most applications of the CRFs are limited to flat, sequential structures, possibly due to the efficiency reason mentioned above. In many areas, however, flat sequential models are not adequate but rather *hierarchical structures* are required. For example, a syntactic parsing task in NLP known as noun-phrase chunking (Sang and Buchholz, 2000) requires joint modelling of both noun-phrases (NPs) and part-of-speech tags (POS) as two layers of semantics associated with words in the sentence. The joint modelling is important because on the one hand noun-phrases are often informative to infer the POS tags belonging to them, and on the other hand, a sub-sequence of POS tags may help identify the nature of the phrase for that sub-sequence.

1.2 Aims and Scope

This thesis investigates further into CRFs, which are at the conjunction of the discriminative modelling framework and undirected graphical models. Our objectives are:

- To apply CRFs to new domains with different settings.
- To extend the theory of CRFs in three aspects: feature selection, learning under arbitrary structures, and modelling hierarchical data.

In the application line of work we study two specific areas:

- The accent restoration problem, in which given a sequence of accent-less words, we want to restore original accents without external information. We have chosen Vietnamese as the study subject. We approach the problem by using sequential CRFs to model and learn the accent space.
- Movie recommendation systems in which viewers are provided with specific titles that may interest them. We aim at integrating rich domain knowledge together with preferences expressed by viewers into a single CRF.

In the theoretical investigation we focus on the common theme of learning and inference in CRFs with missing variables. More specifically, the following three aspects are studied in detail:

- The feature selection problem in which we need to select the most discriminative subset of features. We extend the boosting framework (Freund and Schapire, 1997; Schapire and Singer, 1999) to embed the feature selection capacity into the learning process.
- Intractability of parameter estimation of CRFs in arbitrary networks. We exploit the fact that a network is a superimposition of trees and each tree is efficient to learn and infer.
- Generalisation of sequential CRFs to support modelling, learning and inference of hierarchical data. We limit to recursive sequential type of data, in that a node in a sequence at the parent level is a sub-sequence by itself at the child level. We approach the problem by extending the existing Hierarchical Hidden Markov Models (Fine *et al.*, 1998).

1.3 Significance and Contribution

There are two central lines of work that constitute the significance of the thesis: (1) broadening the applicability of CRFs in novel application domains, and (2) deepening the understanding of theory of CRFs in three sub-areas: *feature selection, parameter estimation with general network structures*, and *hierarchical data modelling*. In particular, our contributions are:

- A demonstration that sequential CRFs, especially second-order chains, are suitable for restoring lost accents. This is an important problem because accents may be lost due to formatting or they are not supported by standard keyboards and many display applications. On the algorithmic side languages like Vietnamese pose significant challenge because accent-less sentences are highly ambiguous. In our study we are able to reach high level of accuracy that can be suitable for real world deployment.

- Construction of a novel model called *Preference Networks* (PNs), which is a large and densely-connected CRF for relational databases. PNs are a discriminative relational model that support various queries in recommender systems - an important element in current e-commerce sites. Different from most previous studies in the recommendation field, our model is both formal and expressive in that it supports probabilistic inference and incorporates rich domain knowledge, such as user profile, product content, and collaborative user preferences. We evaluate this model on the movie domain and show that it is competitive against well-known techniques.

- A novel algorithm called AdaBoost.CRF that addresses both feature selection and missing training variables. It is well-known that feature selection is required to eliminate irrelevant information, to improve the prediction accuracy, to aid human interpretation of data and to speed up model execution. AdaBoost.CRF is an extension of the celebrated boosting methodology to the area of structured prediction. It is an efficient algorithm in that feature selection is integrated into the learning process providing a good trade-off between speed and prediction performance. We demonstrate that it is able to select small amounts of features out of a large feature pool while maintaining reasonable accuracy.

- A new parameter estimation algorithm called AdaBoost.MRF for CRFs with arbitrary network structures under missing variables. This provides an answer to the intractability problem in maximum likelihood learning. AdaBoost.MRF is efficient in that it requires only inference in trees, therefore achieving linear complexity in network size. Its predictive power is comparable with well-known parameter estimation methods.

- A discriminative framework called Hierarchical Conditional Random Fields (HCRFs) for modelling, inference and learning recursive sequential data. The data at different resolutions can be jointly modelled in a formal fashion. This eliminates some drawbacks in the popular layered approach by preventing errors to propagate from the lower layer to the higher. This also enables recursive data to be represented as graphical models, allowing rich probabilistic inference. The framework is based on, and is an extension of a generative counterpart known as Hierarchical Hidden Markov Models (Fine *et al.*, 1998; Bui *et al.*, 2004). As a result, it includes a version of Asymmetric Inside-Outside (AIO) algorithm (Bui *et al.*, 2004) for learning and a Generalised Viterbi algorithm for inference.

- A set of techniques to deal with practical issues associated with HCRFs. In particular we have (1) developed a scaling algorithm that is effective in reducing numerical overflow; (2) extended the AIO and the Generalised Viterbi algorithm to cope with arbitrary partial labels; (3) derived an efficient approximate inference scheme based on Rao-Blackwellisation (e.g. see (Casella and Robert, 1996)) and Gibbs sampling (Geman and Geman, 1984); (4) proposed an approximate learning algorithm based on pseudo-likelihood (Besag, 1975); (5) represented a special case of HCRFs with exponential duration distribution as a factor-graph (Kschischang *et al.*, 2001); and (6) shown how to convert discriminative HCRFs to the generative counterparts.

- Two applications of HCRFs in human activity recognition and noun-phrase chunking. We demonstrate that HCRFs are competitive against rival methods.

1.4 Thesis Structure

This thesis is organised into 10 chapters and a number of appendices, in which 6 chapters make up the main contribution of the thesis and the rest are supporting materials. The rest of the thesis is arranged in the following order:

- Chapter 2 selectively reviews background materials that are essential for further development of the thesis. These include general statistical machine learning with structured output spaces, the Maximum Entropy principle and graphical models. The formulation of the principle of Maximum Entropy (MaxEnt) (Shannon, 1948; Jaynes, 1957; Cover and Thomas, 1991) supports the log-linear model utilised in the multi-class logistic classifiers and CRFs. Background on graphical modelling in general, and undirected graphical models in particular, are presented to support the understanding of the theory of CRFs. Finally, we provide a closer look at hierarchical modelling of data, the area that covers a major contribution of the thesis.

- Chapter 3 describes in more detail the main subject of this thesis: the Conditional Random Field. We present common aspects such as modelling, feature selection and parameter estimation and review the most important applications of CRFs. More advanced developments and issues are also discussed.

- Chapter 4 presents a novel application of CRFs in Vietnamese accent restoration. We propose to use sequential CRFs to model and learn the output space of the Vietnamese accent sequences. We apply the stochastic gradient ascent for parameter estimation and compare the performance of CRFs with several rival methods on a large Vietnamese newswire dataset.

- Chapter 5 details the construction of Preference Networks (PNs) for recommendation systems. PNs are large-scale and very densely connected networks that require fast local learning algorithms such as *pseudo-likelihood* of Besag (1975). We show that the PNs are capable of representing the whole rating database provided by a set of users on a set of products or services, and of encompassing varieties of domain knowledge to improve system performance. The chapter also evaluates the PNs against several well-known methods in the area.

- Feature selection is covered in Chapter 6. The chapter presents an extension of boosting called AdaBoost.CRF for parameter estimation of structured models with missing training labels. The chapter documents experimental evidence that suggests the proposed algorithm is effective in selecting a small subset of features from a large feature pool.

- Chapter 7 addresses the problem of parameter estimation in CRFs with arbitrary network structures. We introduce a novel algorithm called AdaBoost.MRF, which is efficient and capable of handling missing labels.

- Hierarchical extensions to the modelling theory of CRFs is given in Chapter 8, and is continued through Chapter 9. Chapter 8 introduces Hierarchical CRF (HCRF) for recursive sequential data. Model definition, representation, and an efficient algorithm for learning and inference in HCRFs are included in the chapter. Chapter 9 addresses practical issues associated with the HCRFs. These include numerical overflow, approximate learning and inference. The chapter also describes experimental evaluations on two different problems: human activity recognition and noun-phrase chunking.

- Chapter 10 summarises the main content of the thesis and outlines future work.

Chapter 2

Related Background

In this chapter, we provide the background on which the thesis is built. As the material is somewhat mathematical, we provide a list of notations in Table 2.1.

2.1 Statistical Machine Learning

2.1.1 Common Setting

Statistical Machine Learning (e.g. see (Hastie *et al.*, 2001)), an intersection of Computer Science and Statistics, aims to build systems that 'learn' from training examples to perform tasks on unseen data. When the training example includes the outcome pattern $x \in \mathcal{X}$ of a given input $z \in \mathcal{Z}$, the learning is said to be *supervised*. The goal is to estimate a classifier $h(z)$

$$h(z) : \mathcal{Z} \rightarrow \mathcal{X} \tag{2.1}$$

that outputs the prediction $\hat{x}$ for a future input z, i.e. $\hat{x} = h(z)$. Another learning type is *unsupervised* in that no outcomes are available for a given input. This section is limited to reviewing supervised learning algorithms that are applicable to the thesis's focus.

Assuming that the data is randomly drawn from a fixed but unknown distribution $\Pr(x, z)$. Learning searches for $h(z)$ that minimises the *expected risk*

$$\mathcal{R}(h) = \int L(x, h(z)) \Pr(x, z) dx dz \tag{2.2}$$

where $L(x, h(z))$ is the measure of mismatch between true output x and the prediction $\hat{x}$

Notation	Description
x	(Joint) state variables
$\mathcal{X}$	Space of state variables, or output space
$x \backslash x_c$	State variables other than x_c
e	Ending indicators
z	Observables
$\Pr(x)$	Model probability
Z	The partition function
$\mathcal{F}$	Free energy
$H[P]$	Shannon's entropy of distribution P
$\mathbb{E}_P[F]$	Expectation of a function F with respect to distribution P
i, j	Index of the graph vertex
d	Level, counting from the root as 1 down to bottom
t	Time index
D, T	Model depth and length
c	Index of the cliques in graph
N	Graph size
k	Index of feature and parameter
K	Feature size
l	Index of data instance
n	Data size
(ϑ, h)	Visible and hidden components of the joint state variable x, respectively
$\mathcal{D}$	The data set
$\mathcal{G}$	The graph
$(\mathcal{V}, \mathcal{E})$	Set of vertices and edges of the graph, respectively
$\mathcal{N}(i)$	Neighbourhood of node i
$\mathbf{w}$	Parameter vector
$\mathbf{f}(.)$	Local feature vector
$\mathbf{F}(.)$	Global feature vector (sum of all active $\mathbf{f}(.)$) in the configuration
$\phi(.), \psi(.)$	Potential functions.
$\mu_{j \to i}(x_i)$	Message from node j to node i
$\delta[.]$	Return 1 is the predicate $[.]$ is true, 0 otherwise

Table 2.1: Notations used in this chapter.

returned by $h(z)$. For example, we may be interested in the error measure

$$L(x, h(z)) = \delta[x \neq h(z)] \tag{2.3}$$

where $\delta[x \neq h(z)]$ returns 1 if $x \neq h(z)$ and 0 otherwise.

However, since $\Pr(x, z)$ is unknown, one resorts to minimise the *empirical risk* (or the loss) on the training data $\mathcal{D} = \{(x^{(l)}, z^{(l)})\}_{l=1}^{n}$

$$\hat{\mathcal{R}}(h) = \frac{1}{n} \sum_{l=1}^{n} L(x^{(l)}, h(z^{(l)})) \tag{2.4}$$

Depending on the loss we can roughly classify the statistical machine learning methods into *probabilistic* and *non-probabilistic* methods. Probabilistic methods aim at arriving at the conditional distribution $\Pr(x|z)$ for prediction. Typically, minimising the empirical loss is converted into maximising the conditional likelihood

$$\mathcal{L} = \frac{1}{n}\sum_{l=1}^{n} \log \Pr(x^{(l)}|z^{(l)}) \tag{2.5}$$

Non-probabilistic methods, on the other hand, employ several different loss functions such as quadratic loss, exponential loss (as in boosting (Freund and Schapire, 1996)), or hinge loss (as in Support Vector Machines - SVMs (Burges, 1998)).

In this thesis we are interested in the *parametric* setting[1], in which h is parameterised by some parameter $\mathbf{w}$. In particular, we will study the linear classification problem in that we want to estimate the following functional

$$G(x, z) = \mathbf{w}^\top \mathbf{F}(x, z) \tag{2.6}$$

where $\mathbf{F}$ is the vector of features that encode dependency between input z and output x. The prediction of a new input is given as

$$\hat{x} = h(z) = \arg\max_{x \in \mathcal{X}} G(x, z) \tag{2.7}$$

The type of probabilistic models we are studying in this thesis is the *multi-class logistic*[2], whose distribution is given as

$$\Pr(x|z; \mathbf{w}) = \frac{\exp(\mathbf{w}^\top \mathbf{F}(x, z))}{\sum_{x'} \exp(\mathbf{w}^\top \mathbf{F}(x', z))} \tag{2.8}$$

In Section 2.2 we will present a theoretical justification for choosing this type of model.

2.1.2 Learning in Structured Output Spaces

Until recent years the field of statistical learning had focused only on unstructured output spaces, in that, there are no direct relations between output variables. However, most of the real world domains involve interdependent variables, in that the output spaces of interest are *structured*. Learning and predicting structured patterns pose new challenges and opportunities that have attracted much interest recently, as evidenced in past workshops (Table 2.2). In fact, structured prediction is now considered as one of the top challenges in

[1] This is opposed to the *non-parametric* setting where no underlying models with specific parameterisation are assumed.

[2] This is also known as (conditional) softmax, exponential family, Gibbs distribution and log-linear model.

statistical machine learning (Lafferty and Wasserman, 2006).

Remarkable work in this area includes (McCallum *et al.*, 2000; Lafferty *et al.*, 2001; Collins, 2002; Taskar *et al.*, 2002) (Altun *et al.*, 2003a, 2004; Taskar *et al.*, 2004; Tsochantaridis *et al.*, 2005), and (Richardson and Domingos, 2006). These works exploit the fact that structured prediction is a multi-class problem so standard machine learning algorithms such as logistic regression, boosting, and SVMs can be applied. The major challenge lies in the size of the output space, which is often exponentially large in the number of variables. For example, if we have N discrete variables, each of which takes S possible values, then the total number of classes that these variables can jointly represent is S^N. Therefore, the main problem is how to efficiently represent and perform learning and inference in these spaces.

To date, the most successful modelling tools for structured spaces are *graphical models* (Lauritzen, 1996). This is a unified framework that includes various previous models such as Markov random fields (MRFs) (Lauritzen, 1996), Bayesian networks (BNs) (Pearl, 1988), hidden Markov models (HMMs) (Rabiner, 1989), Kalman filters and several neural network architectures (Saul *et al.*, 1996; Hinton and Salakhutdinov, 2006), and the recent factor graphs (FGs) (Kschischang *et al.*, 2001). Generally, graphical models represent variables as vertices in a network and probabilistic interdependencies between variables as edges. The global property of the whole network is achieved through local interactions. We will provide more details about these models in Section 2.3.

Previous machine learning in structured output spaces has exploited a rich set of graphical models. For example, the Maximum Entropy Markov model (MEMM) introduced in (McCallum *et al.*, 2000) makes use of directed Markov chains for modelling sequential output data in conjunction with local multi-class logistic classifiers. Similarly, the conditional random field (CRF) (Lafferty *et al.*, 2001) represents output spaces using undirected Markov random fields (also known as Markov networks) and utilises multi-class logistic for learning the input-output mapping. When the CRF is applied to relational domains it becomes the relational Markov network (RMN) (Taskar *et al.*, 2002). These methods are generally probabilistic, in that they estimate the conditional distribution of the output pattern given the input $\Pr(x|z)$.

Non-probabilistic methods have also been applied for structured domains. The work in (Collins, 2002) is an extension of the Perceptron algorithm (Rosenblatt, 1958; Freund and Schapire, 1999). Similarly, work in (Altun *et al.*, 2003a) utilises boosting, (Altun *et al.*, 2004) extends Gaussian processes, and (Taskar *et al.*, 2004; Tsochantaridis *et al.*, 2005) generalises SVMs.

Although much progress in learning with structured output spaces has been made with impressive applications, the field is still in an early stage. There are many remaining issues

to be addressed. These issues come from three sources: those associated with the standard machine learning techniques being employed, those with the underlying graphical models, and those with the interaction between learning algorithms and graphical models. From the statistical machine learning point of view, the main concerns are estimation bias, variance, consistency, generalisation errors and speed[3]. On the graphical models side, exact inference in arbitrary networks is unfortunately intractable. The interaction between the two sides makes some of these issues more challenging. For example, errors made during approximate inference on graphical models may corrupt the execution of the learning process and, as a result, hurt the generalisation power of the classifier. Since statistical properties of these errors are hard to characterise, generalisation errors of learning algorithms may not be estimated. As remarked by Lafferty and Wasserman (2006), the SVM techniques when applied to Markov networks are indeed inconsistent. The only known consistent estimation is based on maximum conditional likelihood, like those used in CRFs. Fortunately, inconsistent estimation in graphical models using a certain class of approximate methods may still be very valuable (Wainwright, 2006).

Year	**Workshop**
2004	NIPS Learning With Structured Outputs Workshop
2004	NIPS Graphical Models and Kernels
2005	NIPS Kernel Methods and Structured Domains
2006	ICML Workshop on Learning in Structured Output Spaces
2007	ICML Workshop on Constrained Optimisation and Structured Output Spaces

Table 2.2: Some recent workshop on learning in structured output spaces.

In the next subsection, we provide a justification of using multi-class logistical distributions through the principle of Maximum Entropy in Section 2.2 and a detailed account of graphical models.

2.2 The Maximum Entropy Principle

Maximum Entropy (MaxEnt) (Jaynes, 1957) is a method for density estimation. To be consistent with the statistical machine learning setting we present here the *conditional* MaxEnt instead.

Suppose we are given an observed data distribution $\widetilde{\Pr}(x, z)$ of the random variables x and z, and some measurement of data $\mathbf{F}(x, z) = (F_1(x, z), F_2(x, z), ..., F_K(x, z))^\top$ that we

[3]The efficiency issue is now recognised as one of the main problem in machine learning, as evidenced in one of NIPS 2007 Workshops.

will call 'features'. A distribution $\Pr(x|z)$ is consistent with the data if

$$\widetilde{\mathbb{E}}[\mathbf{F}(x,z)] \;=\; \mathbb{E}[\mathbf{F}(x,z)] \tag{2.9}$$

where

$$\widetilde{\mathbb{E}}[\mathbf{F}(x,z)] = \sum_{x,z} \widetilde{\Pr}(x,z)\mathbf{F}(x,z) \tag{2.10}$$

is the empirical feature expectation and

$$\mathbb{E}[\mathbf{F}(x,z)] = \sum_{z} \widetilde{\Pr}(z) \sum_{x} \Pr(x|z)\mathbf{F}(x,z) \tag{2.11}$$

is the model feature expectation.

The *Maximum Entropy Principle* states that *among all consistent distributions, if nothing else is known about the data, we should choose the density that is the least biased, i.e. the one closest to the uniform distribution*. The distance between $\Pr(x|z)$ and the uniform distribution $U(x|z) = 1/|\mathcal{X}|$ can be measured by the Kullback-Leibler divergence (Cover and Thomas, 1991):

$$\begin{aligned} KL(\Pr||U) &= \sum_{z} \widetilde{\Pr}(z) \sum_{x} \Pr(x|z) \log \frac{\Pr(x|z)}{U(x|z)} \\ &= -H[\Pr] + \log |\mathcal{X}| \end{aligned} \tag{2.12}$$

where

$$H[\Pr] = -\sum_{z} \widetilde{\Pr}(z) \sum_{x} \Pr(x|z) \log \Pr(x|z) \tag{2.13}$$

is Shannon's entropy (Shannon, 1948). The MaxEnt density estimator minimises the Kullback-Leibler divergence, which is equivalent to maximising the entropy, under the constraints of Equation 2.9.

By using Lagrange multipliers and maximising the entropy in Equation 2.13 with respect to the distribution $\Pr(x|z)$, one arrives at the multi-class logistic (or log-linear) distribution in Equation 2.8. Given this log-linear form, maximising the entropy with respect to $\Pr(x|z)$ is equivalent to maximising the likelihood with respect to parameters associated with features

$$\hat{\mathbf{w}} \;=\; \arg\max_{\mathbf{w}} \mathcal{L}(\mathbf{w}); \quad \text{where} \tag{2.14}$$

$$\mathcal{L}(\mathbf{w}) \;=\; \sum_{x,z} \widetilde{\Pr}(x,z) \log \Pr(x|z;\mathbf{w}) \tag{2.15}$$

A nice property of the MaxEnt is that $\mathcal{L}(\mathbf{w})$ is concave and thus $\hat{\mathbf{w}}$ is unique. Maximising

the likelihood is equivalent to solving the equation where the gradient is equated to zero:

$$\begin{aligned}\nabla\mathcal{L}(\mathbf{w}) &= \sum_{x,z}\widetilde{\Pr}(x,z)\mathbf{F}(x,z) - \sum_{z}\widetilde{\Pr}(z)\sum_{x}\Pr(x|z)\mathbf{F}(x,z) \\ &= \widetilde{\mathbb{E}}[\mathbf{F}(x,z)] - \mathbb{E}[\mathbf{F}(x,z)] \\ &= 0 \end{aligned} \tag{2.16}$$

Thus, solving Equation 2.16 is equivalent to finding the distribution $\Pr(x|z)$ that satisfies the constraints in Equation 2.9. In other words, the *MaxEnt provides a theoretical justification for using multi-class logistical distributions whose parameters are estimated by the maximum likelihood method*.

Algorithmic parameter estimation of MaxEnt models has been addressed in statistics and computer science for past decades. Notable algorithms include the Generalised Iterative Scaling (GIS) introduced in (Darroch and Ratcliff, 1972), and the Improved Iterative Scaling (IIS) in (Berger *et al.*, 1996; Pietra *et al.*, 1997). However, recent empirical evidence (Minka, 2001b; Malouf, 2002; Sha and Pereira, 2003) has suggested that these specialised algorithms are generally outperformed by recently advanced numerical optimisation alternatives such as Conjugate Gradients (Hestenes and Stiefel, 1952) and quasi-Newton methods such as L-BFGS (Liu and Nocedal, 1989; Byrd *et al.*, 1994).

The MaxEnt is particularly popular in computer science in recent years (Zhu *et al.*, 1998; Nigam *et al.*, 1999; Zitnick and Kanade, 2004), especially in the field of NLP after the pioneering work of Berger *et al.* (1996) and Ratnaparkhi (1996). It often achieves competitive performances with state-of-the-art rivals in the domains in which it is applied. The strength of this method comes with the ability to incorporate arbitrary and overlapping features.

The work in (Kazama and Tsujii, 2003) relaxes the equality in the original consistency constraints in Equation 2.9 in the way that the difference between the model feature expectation and the empirical expectation is bounded in a given interval

$$L_k \leq \mathbb{E}[F_k(x,z)] - \widetilde{\mathbb{E}}[F_k(x,z)] \leq U_k, \text{ for } k = 1,2,...,K \tag{2.17}$$

where $L_k < 0 < U_k$. A more comprehensive study of the constraints is described in (Dudík *et al.*, 2007).

2.3 Structured Data and Graphical modelling

As noted early in Section 2.1.2, it is often the case that variables in real data are interdependent, and it is hard, if not impossible, to isolate any variables that are truly independent

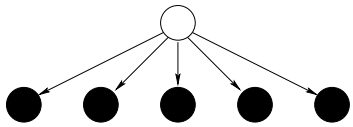


Figure 2.1: Naïve Bayes assumption: words (filled circles) are conditionally independently generated by the topic (empty circle).

of others. However, a holistic analysis of all the variable interactions is very expensive. We have to make some assumption of independence to decompose the complex problem into solvable pieces. One of the most useful assumptions is *conditional independence* in that two sets of variables are independent of each other given some conditions, for example, the connections between these two sets of variables are blocked. For example, words in a document are often assumed to be created with a specific intention, i.e. the topic of the subject being written. The naïve Bayes assumption is that once the topic is chosen, words can be considered as being independently generated (McCallum and Nigam, 1998) (see Figure 2.1). Put it differently, words are conditionally independent given the topic. Of course, such an assumption may be adequate for text classification but it is clearly too simplistic for deeper understanding of text. Words do not just 'happen' to co-occur in texts, but they usually follow certain grammatical structures and conventional usage. Thus, depending on the nature of problem, we may want to vary the level of interdependency, either for ease of analysis or better understanding. More importantly we want a representation scheme that is *expressive* enough to integrate prior knowledge about the domain, and at the same time, provides us with an analytical framework for *efficient* learning, reasoning, interpreting and predicting the data.

2.3.1 Graphical Modelling

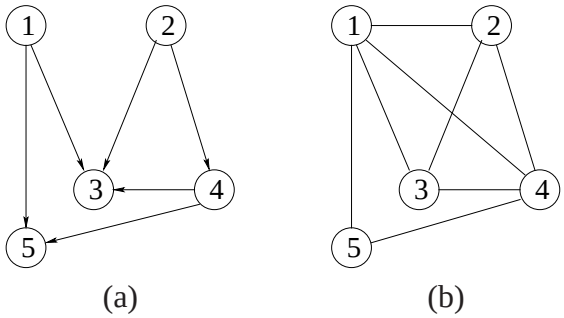


Figure 2.2: Examples of Bayesian Networks (a) and Markov Random Fields (b). (a) is converted to (b) by marrying parents of nodes and dropping arrows.

Graphical Models (GMs) nicely address the above requirements. GMs seamlessly integrate graph theory and probability theory. The formulation is *semi-formal* in the sense that GMs provide a tool for visualisation of interdependency between variables in the data, and at

the same time, obey strict mathematical formulations of conditional dependence and probabilistic consistency. GMs offer a separation between what can be learned from data (in the *learning* phase) and what can be inferred from the model (in the *inference phase*).

Denote by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ a graph that has a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$. Each vertex $i \in \mathcal{V}$ represents a random variable x_i, where $i = 1, 2, ..., N$ and $N = |\mathcal{V}|$. Let $x = (x_i)_{i=1}^N$ denote the joint set of all random variables represented by the graph. In this thesis we are interested in discrete models in which each variable admits values from a finite set of states, i.e. $x_i \in S_i$, where $S_i = \{1, 2, ..., |S_i|\}$. For example, in computer vision, it is common to have the same small set S of possible scenes for all nodes. In language processing, however, the set can be a subset or the whole vocabulary of a language. Continuous variables are also of interest, but their use is beyond the scope of this thesis.

Imposed on the graph $\mathcal{G}$ is a joint distribution of all variables $\Pr(x_1, x_2, ...x_N)$. Reasoning about a particular variable x_i given some evidence x_c can be given as

$$\Pr(x_i|x_c) = \frac{\Pr(x_i, x_c)}{\Pr(x_c)} \tag{2.18}$$

where c is the set of indices. Denote by x_{-c} the set of all variables except the subset c, i.e. $x_{-c} = x \backslash x_c$ and $x = (x_c, x_{-c})$. We have

$$\Pr(x_c) = \sum_{x_{-c}} \Pr(x) = \sum_{x_{-c}} \Pr(x_c, x_{-c}) \tag{2.19}$$

There are two types of GMs: *directed* (Figure 2.2a) and *undirected* (Figure 2.2b). Directed graphical models (also known as Bayesian Networks (Pearl, 1988) and Belief Networks) provide a graphical representation of *causalities* and *influences*. The direction of influence is denoted by an arrow in Figure 2.2a. Undirected graphical models (also known as Markov Random Fields and Markov Networks), on the other hand, encode the *correlations* between variables. Below we describe the directed case and leave the undirected case, which is the focus of this thesis, until Section 2.4.

The main consistency requirement of Bayesian Networks (BNs) is that the graph must be *acyclic* in that there must be no directed cycles in the graph. The direction and the degree of influence are encoded in a conditional distribution $\Pr(x_i|pa(i))$ of the influenced variable x_i given the influencing subset of variables $pa(i)$. In the Bayesian Networks, x_i is often called the child and $pa(i)$ the parents. For instance, in Figure 2.2a, $pa(3) = \{1, 2, 4\}$. The joint distribution $\Pr(x)$ is the product of all local conditional distributions:

$$\Pr(x) = \prod_{i=1}^{N} \Pr(x_i|pa(i)) \tag{2.20}$$

One important property of BNs is that a variable x_i is *conditionally independent* of all other variables given a special surrounding set of variables known as *Markov blanket* which is composed of its parents, its children and its children's parents. For instance, in Figure 2.2a, the Markov blanket of node 5 is $\{1, 4\}$, while it is $\{1, 2, 3, 5\}$ for node 4.

To model data as a BN, we need to determine the graph connectivity structure $\mathcal{E}$ and estimate the conditional distribution $\Pr(x_i|pa(i))$. Determining $\mathcal{E}$ automatically from data is known as *structure learning*. Given the structure, estimating $\Pr(x_i|pa(i))$ is called *parameter learning*.

Structure learning is a hard problem, partly because the structure space of $\mathcal{E}$ is usually explosive in size N and partly because there is no single criterion to define 'goodness' of a structure. More often, we rely on our understanding of the domain to specify $\mathcal{E}$. In many cases the structure of data is obvious, such as the sequence of part-of-speech tags. In other cases there are no magic formulae to design the right model for a given problem. Simple models may be tried and then improved to account for certain aspects of the problem. On the one hand, overly simplistic models can smooth out the real data too much so that only high regularities are kept. On the other hand, there will not be enough regularities to learn the over-complicated models, given limited data. Determining the right complexity for a given data and how much data for a given complexity still remains an art through experiments.

Moreover, the model structure and inference are tightly coupled. Often we want some complex structures to best characterise the problem at hand. However, most of the time, we have to make some trade-offs in favour of simpler structures for inference efficiency.

Parameter learning is often based on maximising the data likelihood $\Pr(x)$. In discrete BNs, learning with fully observed data is quite straightforward: $\Pr(x_i|pa(i))$ is simply the ratio of occurrences of $(x_i, pa(i))$ to the occurrences of $pa(i)$ in the training data. In situations where there are no occurrences of a particular assignment of x_i, *smoothing* is often used to prevent zero probability from propagating to the joint probability in Equation 2.20. For example, in Laplace smoothing, if a particular assignment of x_i does not occur, we assume that it occurs at least once.

However, it is often the case that the data has missing variables. One of the most successful methods in this case is the Expectation-Maximisation (EM) algorithm (Dempster *et al.*, 1977), which we will study in the next subsection.

2.3.2 EM Algorithm

Denote by $x = (\vartheta, h)$, where ϑ is the subset of visible variables, and h the hidden. The EM attempts to maximise the data log-likelihood $\log \Pr(\vartheta|\mathbf{w})$

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} \log \Pr(\vartheta|\mathbf{w}) = \arg\max_{\mathbf{w}} \log \sum_{h} \Pr(\vartheta, h|\mathbf{w}) \tag{2.21}$$

where $\mathbf{w}$ is the model parameters. In Bayesian Networks $\mathbf{w}$ is the set of all local conditional distribution $\{\Pr(x_i|pa(i))\}_{i=1}^{N}$. The summation inside the log function couples the two variables ϑ and h. To decouple them, applying the Jensen's inequality to the concave log function, we have

$$\begin{aligned} \mathcal{L}(\mathbf{w}) = \log \sum_{h} \Pr(\vartheta, h|\mathbf{w}) &\geq \sum_{h} Q(h) \log \frac{\Pr(\vartheta, h|\mathbf{w})}{Q(h)} &(2.22)\\ &= \mathbb{E}_Q[\log \Pr(\vartheta, h|\mathbf{w})] + H[Q] &(2.23) \end{aligned}$$

for any proper distribution $Q(h)$. A nice property of the lower-bound here is that the gap between $\mathcal{L}(\mathbf{w})$ and its lower-bound is closed by setting $Q(h) = \Pr(h|\vartheta; \mathbf{w})$. Since $\log \Pr(\vartheta, h|\mathbf{w})$ is typically decomposable into the sum of simpler components, the lower-bound nicely decouples variables. Let $\mathcal{Q} = \mathbb{E}_Q[\log \Pr(\vartheta, h|\mathbf{w})]$, since $H[Q]$ does not depend on $\mathbf{w}$, maximising the lower-bound with respect to $\mathbf{w}$ is equivalent to maximising $\mathcal{Q}$. This suggests an iterative procedure which loops through two steps until convergence:

- **E-step**: compute $Q(h) = \Pr(h|\vartheta; \mathbf{w}^t)$
- **M-step**: optimise the parameter $\mathbf{w}^{t+1} = \arg\max_{\mathbf{w}} \mathbb{E}_Q[\log \Pr(\vartheta, h|\mathbf{w})]$

Essentially, the **M-step** increases the lower-bound, and the **E-step** closes the gap between the true log-likelihood and the lower-bound. The overall effect is that the log-likelihood monotonically increases until it reaches a local maximum.

2.4 Undirected Graphical Models

This section reviews undirected graphical models, including Markov Random Field (MRF) (e.g. see Lauritzen (1996)) and its generalisation called Factor Graph (Kschischang *et al.*, 2001). Although specific forms of MRFs have been used for a long time, such as the Ising model in physics (e.g. see (Baxter, 1982)), the view of MRFs as a part of graphical models is fairly recent.

2.4.1 Model Representation

As a graphical model, a Markov Random Field specifies a joint distribution $\Pr(x)$ over the undirected graph $\mathcal{G}$. MRFs are essentially more general than Bayesian Networks in the sense that every Bayesian Network can be converted into a MRF by first connecting all the parents of each variable and then dropping the arrows of the edges (Figure 2.2).

Conditional independence is ensured by a property that variables x_i and x_j are conditionally independent of each other if we know values of a subset of variables that block any paths from node i to node j. A useful set of blocking nodes is the Markov blanket, which contains all neighbours of a node. Once the Markov blanket of node i is known, we can be sure that x_i is independent of the rest of the nodes.

This condition is often known as *Markov property*, and is enforced by the Hammersley-Clifford theorem (Lauritzen, 1996), which states that the following factorisation must hold

$$\begin{aligned} \Pr(x) &= \frac{1}{Z}\Phi(x) \\ &= \frac{1}{Z}\prod_c \psi_c(x_c) \end{aligned} \tag{2.24}$$

where x_c is the *maximal clique* defined by the structure of $\mathcal{G}$, and $Z = \sum_x \prod_c \psi(x_c)$ is the normalisation constant (also known as the partition function). A maximal clique is a completely connected subgraph (e.g. the subset $\{1, 2, 3, 4\}$ in Figure 2.2b). The positive clique function $\psi(x_c)$ is often referred to as *potential* or *compatibility function*.

In practice, we may not use this strict factorisation because it may not be natural to visualise, but we further factorise clique potentials into products of smaller sub-potentials. For example, in image modelling we often use the pairwise and singleton potentials, which are defined over edges and nodes, respectively. In this case the distribution is given as[4]

$$\Pr(x) = \frac{1}{Z}\prod_{i\in\mathcal{V}} \phi_i(x_i) \prod_{(i,j)\in\mathcal{E}} \psi_{ij}(x_i, x_j) \tag{2.25}$$

In the context of physical systems, the potential $\psi_c(x_c)$ in Equation 2.24 is often written in terms of *energy* $E_c(x_c)$ as

$$\psi_c(x_c) = \exp(-\frac{1}{\beta}E_c(x_c)) \tag{2.26}$$

where β is a positive quantity commonly referred to as system temperature. The temperature β does not have physical meaning outside physical sciences but it is sometimes used

[4]Traditionally, the computer vision community uses the following form $\Pr(x, z) = \Pr(x)\prod_{i\in\mathcal{V}} \Pr(z_i|x_i)$, where z is generated by x, or z is considered as a noisy version of x.

for algorithm control purposes, especially in simulated annealing (Kirkpatrick *et al.*, 1983; Hofmann and Buhmann, 1997). In this thesis we set $\beta = 1$ for simplicity and this should not change the nature of data modelling.

Let $E(x) = \sum_c E_c(x_c)$ be system energy. A quantity that plays an important role in stochastic evolution of physical systems is the *Helmholtz free-energy* (or just free-energy)

$$\begin{aligned} \mathcal{F} &= \mathbb{E}[E] - H \\ &= \sum_x \Pr(x)E(x) + \sum_x \Pr(x)\log\Pr(x) \qquad (2.27) \\ &= -\log Z \qquad (2.28) \end{aligned}$$

The last equation is obtained by substituting Equation 2.24 and Equation 2.26 into Equation 2.27. There is a common tendency to decrease the free-energy to reach equilibrium of a physical system. Clearly, if we know that the energy of the system has been measured (via $\mathbb{E}[E]$), minimising the free-energy $\mathcal{F}$ is equivalent to maximising the entropy H, and this matches the principle of Maximum Entropy described in Section 2.2. Interestingly, it has been shown in (Yedidia *et al.*, 2005) that minimising an approximation of $\mathcal{F}$ known as Bethe free-energy is equivalent to passing messages in Pearl's famous belief propagation (Pearl, 1988).

2.4.2 Inference

The heart of any graphical models is obviously the inference engine. Let us consider the general case where the joint state variable x has a subset of visible (or observed) variables ϑ, and a subset of hidden (or missing, latent) variables h, i.e. $x = (\vartheta, h)$. In this section we outline most common quantities needed to be computed and leave the algorithmic details for later sections.

2.4.2.1 Inference involved in learning

Let us take a closer look at the computation of data log-likelihood $\log\Pr(\vartheta)$. Under the MRF setting there are two general strategies to maximise it. One is the EM scheme outlined in Section 2.3.2 and the other is the direct optimisation approach.

The EM approach.

Recall that in EM we are concerned with the following quantity

$$\begin{aligned} \mathcal{Q} &= \mathbb{E}_Q[\log \Pr(\vartheta, h)] && (2.29)\\ &= \mathbb{E}_Q[\log \Phi(\vartheta, h)] - \log Z && (2.30)\\ &= \sum_c \mathbb{E}_{Q(h_c|\vartheta)}[\log \psi_c(\vartheta_c, h_c)] - \log Z && (2.31) \end{aligned}$$

In Equation 2.31 we have applied $\Phi(\vartheta, h) = \prod_c \psi_c(\vartheta_c, h_c)$. In the **M-step**, we maximise the function $\mathcal{Q}$, and often need to compute the gradient

$$\nabla \mathcal{Q} = \sum_c \mathbb{E}_{Q(h_c|\vartheta)}[\nabla \log \psi_c(\vartheta_c, h_c)] - \nabla \log Z \tag{2.32}$$

The following proposition shows how $\{\nabla \log Z\}$ is computed.

Proposition 1. *Under the factorisation of Equation 2.24, the following holds:*

$$\nabla \log Z = \sum_c \mathbb{E}_{\Pr(x_c)}[\nabla \log \psi_c(x_c)] \tag{2.33}$$

Proof: Recall that $Z = \sum_x \Phi(x)$, we have

$$\nabla \log Z = \frac{1}{Z} \sum_x \nabla \Phi(x) \tag{2.34}$$

Since $\Phi(x) = \prod_c \psi_c(x_c)$, we have $\log \Phi(x) = \sum_c \log \psi_c(x_c)$, and

$$\begin{aligned} \nabla \log \Phi(x) &= \frac{1}{\Phi(x)} \nabla \Phi(x) = \sum_c \nabla \log \psi_c(x_c), \qquad \text{leading to} && (2.35)\\ \nabla \Phi(x) &= \Phi(x) \sum_c \nabla \log \psi_c(x_c) && (2.36) \end{aligned}$$

Finally, we prove the Proposition 1 using

$$\nabla \log Z = \sum_x \Pr(x) \sum_c \nabla \log \psi_c(x_c) = \sum_c \sum_{x_c} \Pr(x_c) \nabla \log \psi_c(x_c) \tag{2.37}$$

■

The direct optimisation approach.
Unlike the EM, we do not need to compute the auxiliary function $\mathcal{Q}$, but proceed to the log-likelihood directly

$$\begin{aligned} \mathcal{L} &= \log \Pr(\vartheta) = \log \sum_h \Pr(\vartheta, h) = \log \left(\frac{1}{Z} \sum_h \Phi(\vartheta, h) \right) && (2.38)\\ &= \log Z(\vartheta) - \log Z && (2.39) \end{aligned}$$

where $Z(\vartheta) = \sum_h \Phi(\vartheta, h)$. Note that once the observation ϑ is made, the set of free variables of the system is reduced to h, and the distribution becomes:

$$\Pr(h|\vartheta) = \frac{\Pr(\vartheta, h)}{\sum_h \Pr(\vartheta, h)} = \frac{1}{Z(\vartheta)}\Phi(\vartheta, h) \tag{2.40}$$

and thus $Z(\vartheta)$ is a new partition function of the reduced system. We can imagine that there is an evolution from the full system to the reduced system due to the act of observation of ϑ. The change in free-energy during the evolution is then

$$\begin{aligned} \Delta\mathcal{F} &= \mathcal{F}_{reduced} - \mathcal{F}_{full} \\ &= -\log Z(\vartheta) + \log Z \\ &= -\mathcal{L} \end{aligned} \tag{2.41}$$

Thus, *maximum likelihood learning is equivalent to finding the minimum of change in the system free-energy*.

In seeking for the maximiser of the log-likelihood we often compute the gradient

$$\nabla\mathcal{L} = \nabla \log Z(\vartheta) - \nabla \log Z \tag{2.42}$$

Recall that $Z(\vartheta)$ is the partition function of the reduced system with free variables h and distribution $\Pr(h|\vartheta)$, Proposition 1 can be applied as follows

$$\nabla \log Z(\vartheta) = \sum_{c \in C(h)} \mathbb{E}_{\Pr(h_c|\vartheta)}[\nabla \log \psi_c(\vartheta_c, h_c)] \tag{2.43}$$

where $C(h)$ is the set of clique indices in the hidden part of the graph $\mathcal{G}$.

In summary, in EM-based and direct optimisation learning we need to compute the following quantities:

- The 'full' partition function Z, and the 'reduced' partition function $Z(\vartheta)$,
- The local clique marginals $\Pr(x_c)$ and $\Pr(h_c|\vartheta)$.

These inference tasks are only tractable if the structure of the graph $\mathcal{G}$ is a chain or a tree. There exists an efficient message passing over trees known as Pearl's belief propagation (BP) (Pearl, 1988), which require only two passes through all edges (e.g. see (Willsky, 2002; Pearl, 1988)). For general structures, approximations are needed. We will cover exact inference on chains in Sections 2.4.3, 2.4.4 and 2.4.5, and approximate inference on other structures in Section 2.4.6.

2.4.2.2 Inference in pattern prediction

Prediction in MRFs is usually to find the most probable variable assignment

$$\hat{x} = \arg\max_x \Pr(x) \tag{2.44}$$
$$= \arg\max_x \log \Phi(x) \tag{2.45}$$
$$= \arg\min_x \sum_c E_c(x_c) \tag{2.46}$$

This is essentially a combinatorial discrete optimisation problem. Typical computer vision problems such as image restoration (Geman and Geman, 1984) and stereo-matching (Sun *et al.*, 2003), are often recast into the energy-minimisation form of Equation 2.46.

Like partition function estimation, the prediction problem is generally intractable to solve exactly. Efficient approximations to date include the iterated conditional mode (ICM) (Besag, 1986), Pearl's loopy max-product algorithm (Pearl, 1988) and variants (Wainwright *et al.*, 2005a), and the more recent Graph-Cuts (Boykov *et al.*, 2001). Less efficient methods but with theoretical guarantee of convergence can be found in the sampling literature, especially the Simulated Annealing method (Kirkpatrick *et al.*, 1983; Geman and Geman, 1984). The ICM, max-product, and graph-cuts are covered in Section 2.4.7.

2.4.3 First-order Markov Chains

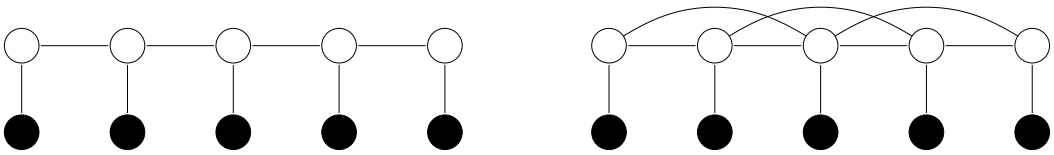


Figure 2.3: Undirected Markov chains: first-order (a) and second-order (b). Filled circles denote observed symbols $\{z_t\}_{t=1}^T$ and empty circles denote state variables $\{x_t\}_{t=1}^T$.

Markov chains, depicted in Figure 2.3, also known as Boltzmann chains (Saul and Jordan, 1995), are the most widely used structure. Our model will involve observables $\{z_t\}_{t=1}^T$ associated with corresponding state variables $\{x_t\}_{t=1}^T$ but we assume that z_t will be absorbed into appropriate local potentials involving x_t.

This subsection presents inference in first-order chains (Figure 2.3a). Extension to second-order chains (Figure 2.3b) and nth-order in general will be covered in the next subsection.

For the chain structure, assuming singleton and pairwise local potentials, the joint potential

is given as

$$\Phi(x) = \Big[\prod_{t\in[1,T]} \phi(x_t) \Big] \Big[\prod_{t\in[2,T]} \psi(x_{t-1}, x_t) \Big] \tag{2.47}$$

where T is the sequence length.

For first-order Markov chains, inference can be made using the *forward-backward* procedure[5].

The forward variable

First, let us define the *forward* variable $\alpha_t(x_t)$ as

$$\alpha_t(x_t) = \sum_{x_{1:t-1}} \prod_{i\in[2,t]} \Big[\phi(x_{i-1})\psi(x_{i-1}, x_i) \Big] \tag{2.48}$$

To derive a recursive relation, let us rewrite Equation 2.48 as

$$\begin{aligned} \alpha_t(x_t) &= \sum_{x_{t-1}} \phi(x_{t-1})\psi(x_{t-1}, x_t) \sum_{x_{1:t-2}} \prod_{i\in[2,t-1]} \Big[\phi(x_{i-1})\psi(x_{i-1}, x_i) \Big] \\ &= \sum_{x_{t-1}} \phi(x_{t-1})\psi(x_{t-1}, x_t)\alpha_{t-1}(x_{t-1}) \end{aligned} \tag{2.49}$$

Let $\alpha_1(x_1) = 1$ for all $x_1 \in S_1$, we can compute all the quantities $\{\alpha_t(x_t)\}_{t=1}^T$ in $\mathcal{O}(T|S|^2)$ time, where $|S| = \max_t |S_t|$. This provides an efficient way to compute the partition function

$$\begin{aligned} Z &= \sum_{x_{1:T}} \Phi(x_{1:T}) \\ &= \sum_{x_T} \alpha_T(x_T)\phi(x_T) \end{aligned} \tag{2.50}$$

Since the recursion often accumulates the numerical scale of the forward variables, it may happen that for large T, we will face either the *under-flow* or the *over-flow* problem. The first case often occurs in directed graphical models such as HMMs because the potentials are always less than unity. The second case is coupled with the undirected graphical models, because it is hard to upper-bound the potential functions, which are usually learnt from data. To avoid this difficulty let us provide some scaling mechanism. Equation 2.49 can be

[5] See (Rabiner, 1989) for details in HMMs.

rewritten as

$$\alpha_t(x_t) = \kappa_{\alpha,t} \sum_{x_{t-1}} \phi(x_{t-1})\psi(x_{t-1}, x_t)\alpha_{t-1}(x_{t-1}) \tag{2.51}$$

where $\kappa_{\alpha,t} > 0$. Typically we use $\kappa_{\alpha,t}$ as a normalisation constant to prevent the numerical values of $\alpha_t(x_t)$ from becoming too small or too large. It is not difficult to see that, after scaling at each time step t, the partition function must be corrected as follows

$$Z = \Big[\prod_{t\in[1,T]} \kappa_{\alpha,t} \Big] \sum_{x_T} \alpha_T(x_T)\phi(x_T) \tag{2.52}$$

Often, we want to work in the log-space instead

$$\log Z = \sum_{t\in[1,T]} \log \kappa_{\alpha,t} + \log \sum_{x_T} \alpha_T(x_T)\phi(x_T) \tag{2.53}$$

The backward variable

In our undirected Markov chains, it is symmetric to define *backward* variables in a similar manner as Equation 2.48

$$\beta_t(x_t) \;\; = \;\; \sum_{x_{t+1:T}} \prod_{j\in[t+1,T]} \Big[\phi(x_j)\psi(x_{j-1}, x_j) \Big] \tag{2.54}$$

which, with appropriate scaling terms $\kappa_{\beta,t}$, also has the recursive relation

$$\beta_t(x_t) = \kappa_{\beta,t} \sum_{x_{t+1}\in S_{t+1}} \beta_{t+1}(x_{t+1})\phi(x_{t+1})\psi(x_t, x_{t+1}) \tag{2.55}$$

Let $\beta_T(x_T) = 1 \; \forall x_T \in S_T$. The log-partition function can also be computed as

$$\log Z = \sum_{t=\in[1,T]} \log \kappa_{\beta,t} + \log \sum_{x_1} \beta_1(x_1)\phi(x_1) \tag{2.56}$$

Of course, the main point is not just the separate forward and backward variables but the relationship between them and how they are used in other inference tasks. For example,

we are often interested in the local marginals

$$\begin{aligned}
\Pr(x_t) &= \sum_{x \setminus x_t} \Pr(x) \\
&\propto \sum_{x \setminus x_t} \Phi(x) && (2.57) \\
&= \sum_{x_{1:t-1}, x_{t+1:T}} \Phi(x_{1:T}) && (2.58) \\
&= Z(x_t) && (2.59)
\end{aligned}$$

where $Z(x_t) = \sum_{x \setminus x_t} \Phi(x)$. By rearranging the factors in the RHS of Equation 2.47, we have

$$\Phi(x) = \Big[\prod_{i \in [2,t]} \phi(x_{i-1}) \psi(x_{i-1}, x_i) \Big] \phi(x_t) \Big[\prod_{j \in [t+1,T]} \phi(x_j) \psi(x_{j-1}, x_j) \Big]$$

then $Z(x_t)$ can be written as

$$\begin{aligned}
Z(x_t) &= \left(\sum_{x_{1:t-1}} \Big[\prod_{i \in [2,t]} \phi(x_{i-1}) \psi(x_{i-1}, x_i) \Big] \right) \phi(x_t) \times \\
&\qquad \times \left(\sum_{x_{t+1:T}} \Big[\prod_{j \in [t+1,T]} \phi(x_j) \psi(x_{j-1}, x_j) \Big] \right) \\
&\propto \alpha_t(x_t) \beta_t(x_t) \phi(x_t) && (2.60)
\end{aligned}$$

In other words, we have

$$\Pr(x_t) = \kappa_t \alpha_t(x_t) \beta_t(x_t) \phi(x_t) \qquad (2.61)$$

where κ_t are appropriate normalisation constants to ensure $\sum_{x_t} \Pr(x_t) = 1$.

A similar trick can be applied to derive the joint marginals. Given the forward and backward variables, we compute the singleton and pair marginals as follows

$$\Pr(x_t, x_{t+1}) = \kappa_{t,t+1} \alpha_t(x_t) \beta_t(x_{t+1}) \phi(x_t) \phi(x_{t+1}) \psi(x_t, x_{t+1}) \qquad (2.62)$$

where $\kappa_{t,t+1}$ are appropriate normalisation constants to ensure $\sum_{x_t} \sum_{x_{t+1}} \Pr(x_t, x_{t+1}) = 1$.

MAP assignment and Viterbi decoding

Viterbi decoding (Rabiner, 1989) is well-known in the HMM literature, and it is equally applicable for the undirected Markov chain. It is a two-step procedure:

- In the first step, we run a maximisation version of the forward $\alpha_t^{\max}(x_t)$, in that all the summarisations in Equation 2.49 are replaced by corresponding maximisations, keeping the local maximal states in a *bookkeeper* Y

$$\alpha_t^{\max}(x_t) = \max_{x_{t-1}} \left[\phi(x_{t-1})\psi(x_{t-1}, x_t)\alpha_{t-1}^{\max}(x_{t-1}) \right] \tag{2.63}$$

$$Y_t(x_t) = \arg\max_{x_{t-1}} \left[\phi(x_{t-1})\psi(x_{t-1}, x_t)\alpha_{t-1}^{\max}(x_{t-1}) \right] \tag{2.64}$$

- In the second step, we need to *backtrack* to decode the best *state sequence* $(x_t)_{t=1}^T$, not just local maximal state.

$$\hat{x}_T = \arg\max_{x_T} \left[\alpha_T^{\max}(x_T)\phi(x_T) \right] \tag{2.65}$$

$$\hat{x}_t = Y_{t+1}(\hat{x}_{t+1}), \text{for } t = T-1, T-2, ..., 1 \tag{2.66}$$

The algorithm takes $\mathcal{O}(T|S|^2)$ time.

There is also an alternative, known as *max-product algorithm* of Pearl, where we make use of both the forward and backward variables in a maximisation manner (e.g. as in Equation 2.63) Substituting the new forward and backward quantities into Equation 2.61, we obtain $\hat{x}$ by finding the maximiser of the local marginals

$$\hat{x}_t = \arg\max_{x_t} \Pr(x_t) \tag{2.67}$$

2.4.3.1 HMMs as special cases

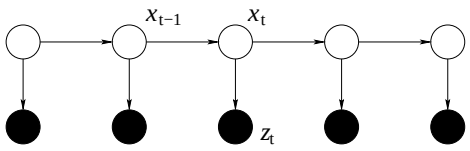


Figure 2.4: Hidden Markov models.

The Hidden Markov Model (Figure 2.4) is a constrained case of the first-order chain, where

we define the potentials as

$$\begin{aligned}
\phi(x_1) &= \pi(x_1)\Pr(z_1|x_1) \text{ where } \sum_{z_1}\Pr(z_1|x_1)=1, \sum_{x_1}\pi(x_1)=1 \\
\phi(x_t) &= \Pr(z_t|x_t), \text{ for } \sum_{z_t}\Pr(z_t|x_t)=1, t\in[2,T] \\
\psi(x_{t-1},x_t) &= \Pr(x_t|x_{t-1}), \text{ where } \sum_{x_t}\Pr(x_t|x_{t-1})=1, t\in[2,T]
\end{aligned}$$

Under these constraints the forward and backward variables have nice probabilistic interpretation. From Equation 2.48, we have

$$\begin{aligned}
\alpha_t(x_t) &= \sum_{x_{1:t-1}}\pi(x_1)\prod_{i\in[2,t]}\Big[\Pr(z_{i-1}|x_{i-1})\Pr(x_i|x_{i-1})\Big] \\
&= \sum_{x_{1:t-1}}\Pr(x_{1:t}|z_{1:t-1}) \\
&= \Pr(x_t, z_{1:t-1}) \qquad (2.68)
\end{aligned}$$

Similarly, from Equation 2.54

$$\begin{aligned}
\beta_t(x_t) &= \sum_{x_{t+1:T}}\prod_{j\in[t+1,T]}\Big[\Pr(z_j|x_j)\Pr(x_j|x_{j-1})\Big] \\
&= \sum_{x_{t+1:T}}\Pr(x_{t+1:T}, z_{t+1:T}|x_t) \qquad (2.69) \\
&= \Pr(z_{t+1:T}|x_t) \qquad (2.70)
\end{aligned}$$

This interpretation is, unfortunately, not present in the undirected counterparts.

The data likelihood arises nicely

$$\begin{aligned}
\Pr(z_{1:T}) &= \sum_{x_t}\Pr(z_{1:T}, x_t) \\
&= \sum_{x_t}\Pr(z_{1:t-1}, x_t)\Pr(z_{t+1:T}|x_t)\Pr(z_t|x_t) \\
&= \sum_{x_t}\alpha_t(x_t)\beta_t(x_t)\Pr(z_t|x_t) \qquad (2.71)
\end{aligned}$$

2.4.4 Second-order Markov Chains

The second-order Markov chains can be converted into the equivalent first-order at the cost of concatenated state space. With a slight abuse of notation, denoted by $\psi(x_{t-2}, x_{t-1}, x_t)$ the second-order potentials. The conversion is carried out by joining two successive nodes x_{t-1} and x_t into a composite-node $y_{t-1} = (x_{t-1}, x_t)$. Let the composite-node poten-

tial be $\phi(y_{t-1}) = \phi(x_{t-1})\psi(x_{t-1}, x_t)$ and the composite-edge potential be $\psi(y_{t-1}, y_t) = \psi(x_{t-1}, x_t, x_{t+1})$. Given these potentials it can be seen that we now have a new first-order Markov chain with the combined state space:

$$y_{t-1} \in S_{t-1} \times S_t \tag{2.72}$$

The naïve implementation of this Markov chain takes $\mathcal{O}((T-1)|S|^4)$ time in this combined state space. However, by paying attention to the fact that the two composite-states $y_{t-1} = (x_{t-1}, x_t)$ and $y_t = (x_t, x_{t+1})$ share x_t, we can implement the forward-backward procedure in $\mathcal{O}((T-1)|S|^3)$ time using

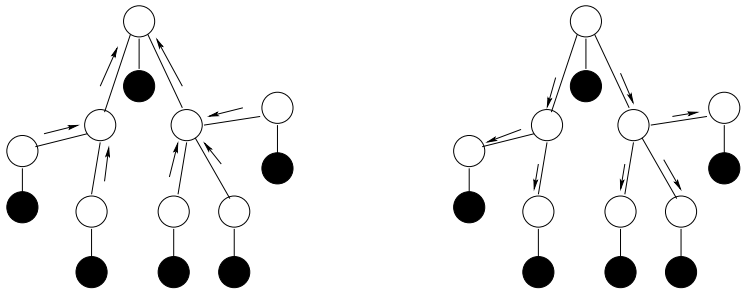
$$\begin{aligned}
\alpha_t(y_t) &= \kappa_{\alpha,t} \sum_{x_{t-1} \in S_{t-1}} \alpha_{t-1}(y_{t-1})\phi(y_{t-1})\psi(y_{t-1}, y_t) \\
\beta_t(y_t) &= \kappa_{\beta,t} \sum_{x_{t+2} \in S_{t+2}} \beta_{t+1}(y_{t+1})\phi(y_{t+1})\psi(y_t, y_{t+1})
\end{aligned}$$

and the joint marginals are computed as

$$\begin{aligned}
\Pr(y_t) &= \kappa_t \alpha_t(y_t)\beta_t(y_t)\phi(y_t) \\
\Pr(y_t, y_{t+1}) &= \kappa_{t,t+1}\alpha_t(y_t)\beta_t(y_{t+1})\phi(y_t)\phi(y_{t+1})\psi(y_t, y_{t+1})
\end{aligned}$$

A similar strategy of state space concatenation can be applied to nth-order Markov chains, i.e. $y_t = (x_t, x_{t+1}, ..., x_{t+n-1})$. In general, the overall complexity will be $\mathcal{O}((T-n+1)|S|^{n+1})$

2.4.5 Tree Models



(a) (b)

Figure 2.5: The two-pass procedure: the upward pass (a) and downward pass (b).

Now we generalise the chains to trees, which are the most complex structures known to be efficient. They have important properties that aid analysis and inference. The joint

distribution can be defined in terms of local marginals as

$$\Pr(x) = \prod_{(i,j)\in\mathcal{E}} \frac{\Pr(x_i, x_j)}{\Pr(x_i)\Pr(x_j)} \prod_{i\in\mathcal{V}} \Pr(x_i) \tag{2.73}$$

The log-partition can be computed as follows

$$\log Z = -\sum_{(i,j)\in\mathcal{E}} \sum_{x_i,x_j} \Pr_{ij}(x_i, x_j) \log \frac{\Pr_{ij}(x_i, x_j)}{\omega_{ij}(x_i, x_j)} + \sum_{i\in\mathcal{V}} (n_i - 1) \sum_{x_i} \Pr_i(x_i) \log \frac{\Pr_i(x_i)}{\phi_i(x_i)} \tag{2.74}$$

where $\omega_{ij}(x_i, x_j) = \phi_i(x_i)\phi_j(x_j)\psi_{ij}(x_i, x_j)$ and n_i is the number of neighbours of node i.

Inference in trees is efficiently carried out by Pearl's *belief propagation* (BP), which is also known as the *sum-product* algorithm. It is a generalisation of the forward-backward procedure described in Section 2.4.3. First we pick one particular node as the root. Since the graph has no loops there is a single path from a node to any other nodes in the graph, and each node, except for the root, has exactly one parent. The forward and backward passes are replaced by the *upward* and *downward* passes:

- In the upward pass, messages are first initiated at the leaves, and are set to 1. Then all messages are sent upward and updated as messages converging at common parents along the paths from leaves to the root. The pass stops when all the messages reach the root.
- In the downward pass, messages are combined and re-distributed downward from the root back to the leaves. The messages are then terminated at the leaves.

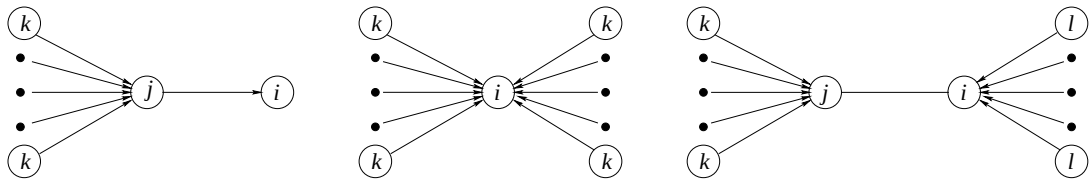


Figure 2.6: Message update (a), node marginal (b) and joint marginal (c).

In general, the message sent from node j to node i in the tree is computed as follows

$$\mu_{j\to i}(x_i) = \kappa_{ji} \sum_{x_j} \phi_j(x_j)\psi_{ij}(x_i, x_j) \prod_{k\in\mathcal{N}(j), k\neq i} \mu_{k\to j}(x_j) \tag{2.75}$$

where $\mathcal{N}(j)$ is the set of neighbours of node j and $\kappa_{ji} > 0$ is some constant. We can compute the log-partition function as soon as the upward pass has reached the root node r:

$$\log Z = \sum_{ji} \log \kappa_{ji} + \log \sum_{x_r} \phi_r(x_r) \prod_{j\in\mathcal{N}(r)} \mu_{j\to r}(x_r) \tag{2.76}$$

Once the messages are terminated, the marginals and joint marginals can be computed as

$$\Pr(x_i) = \kappa_i \phi_i(x_i) \prod_{k \in \mathcal{N}(i)} \mu_{k \to i}(x_i) \tag{2.77}$$

$$\Pr(x_i, x_j) = \kappa_{i,j} \omega_{ij}(x_i, x_j) \left(\prod_{l \in \mathcal{N}(i), l \neq j} \mu_{l \to i}(x_i) \right) \left(\prod_{k \in \mathcal{N}(j), k \neq i} \mu_{k \to j}(x_j) \right) \tag{2.78}$$

The max-product algorithm

Computing the most probable state $\hat{x} = \arg\max_x \Pr(x)$ can be done using the max-product algorithm [6]. This is essentially the sum-product algorithm where all the summations are replaced by maximisations.

Once the messages are computed we can estimate $\hat{x}_i$ by finding the maximiser of the local marginal $\Pr(x_i)$ in Equation 2.77.

2.4.6 Approximate Inference

Popular methods can be broadly classified into two groups: sampling and message passing.

Sampling is a rich literature in physics and statistics, especially under the headline of Markov Chain Monte Carlo (MCMC). The idea is to draw enough samples from the distribution so that the distribution is approximated by the sample frequency. The main problem is that since the state space is often very huge, direct sampling is not computationally applicable. The MCMC methods solve this problem by allowing sampling in a smaller space. See (Metropolis *et al.*, 1953; Hastings, 1970; Kirkpatrick *et al.*, 1983) for early development and application of Metropolis-Hasting method, (Neal, 1993; MacKay, 1996; Andrieu *et al.*, 2003) for MCMC introduction and survey, and (Green, 1995) for a recent important extension. A nice property of sampling is that it can asymptotically converge to the true distribution. In practice, however, it is known to be very slow for many problems. For further implementation issues, see (MacKay, 2003).

Message passing is another important class of approximation methods because of its lightweight and distributed fashion (e.g. see (Minka, 2005b) for a unified view). Loopy belief propagation (McEliece and Cheng, 1998; Murphy *et al.*, 1999; Yedidia *et al.*, 2005) is a particularly important practical method that deserves a separate subsection below. An interesting variant is based on minimising the upper-bound of the log-partition function (Wainwright *et al.*, 2005b).

[6] The max-product is often known as belief propagation in the computer vision community.

Another subset of message passing methods, which attracts much recent attention, falls under the root of *variational methods* (e.g. see (Jordan *et al.*, 1999; Wainwright and Jordan, 2003)). The main idea is to approximate a complex network by some simpler networks (e.g. by removing some edges) and to optimise the difference between the true and the approximation. Depending on the divergence measure, we may obtain the mean-field type (e.g see (Saul *et al.*, 1996; Wiegerinck, 2000; Kappen and Wiegerinck, 2001)) or the expectation propagation type (Minka, 2001a).

2.4.6.1 Gibbs Sampling

Of MCMC methods, Gibbs sampling is popular in the MRF context, especially after the seminal work of Geman and Geman (1984). The idea is, instead of sampling the joint distribution $\Pr(x)$, we cyclically sample the local conditional distribution. Specifically, for networks with pairwise clique potentials as in Equation 2.25, we draw the local values as follows

$$\begin{aligned} \hat{x}_i &\sim \Pr(x_i|\mathcal{N}(i)) \\ &\propto \psi_i(x_i) \prod_{j\in\mathcal{N}(i)} \psi_{ij}(x_i, x_j) \end{aligned} \tag{2.79}$$

where $\mathcal{N}(i)$ is the set of neighbours of node i. This local sampling is easy to perform since it involves only $x_i \in S_i$ and a fixed set of neighbourhood assignment $\mathcal{N}(i)$. After a value is sampled, x_i is assigned to that value and another node is sampled.

2.4.6.2 Loopy Belief Propagation

Loopy BP is the standard BP applied to networks with cycles. Interestingly, physicists have faced similar problems in analysing physical systems such as Ising models. They have proposed the use of Bethe free-energy as an approximation to the true Helmholtz free-energy (see Equation 2.27). Bethe free-energy, like standard BP, is only applicable to systems with singly connected networks. An important recent discovery by Yedidia *et al.* (2005) is that minimising the approximate Bethe free-energy with respect to local marginals is equivalent to seeking stationary points of the loopy BP. Recall from Equation 2.28 that $\mathcal{F} = -\log Z$, minimising the free-energy is equivalent to maximising the log-partition function, which is given in Equation 2.74.

The message passing scheme in loopy BP is similar to that described in Section 2.4.5. However, as the network is loopy, there are no roots and no predefined sending directions. Rather, messages are initiated from all nodes, and are sent in all directions. Because of the cycles present in the network, messages may come back to their sources and create a

non-converging loop. Indeed, there is no guarantee of convergence or quality of approximation. Since the method is applied with the assumption that the network is cycle-free, we can only hope that it will work in networks with large cycles because large cycles will damp the messages enough so that they 'forget' about the origins. On the other hand, densely connected networks will have very small loops, and thus we may not expect a good performance. In addition, loopy BP is known to fail when the interaction (or edge) potentials $\psi_{ij}(x_i, x_j)$ are significantly stronger than the data (or node) potential $\phi_i(x_i)$. The commonly known phenomenon is the oscillation of the messages or other qualities such as the Bethe free energy.

In practice, the BP is often declared converged if the relative change in messages or related quantities like Bethe free energy is small enough, say $10^{-3} - 10^{-5}$ for example. There are two main mechanisms to control the convergence: the message update schedule and damping. Update schedules can be either synchronous or asynchronous. In the synchronous schedule, messages are updated at the same time, and in the asynchronous schedule, messages are updated one by one. Typically, the asynchronous update converges (if it does) much faster than the synchronous counterpart since information between nodes is propagated quicker. There are also several specific schedule schemes that claim to improve the convergence rate (Wainwright *et al.*, 2003a; Elidan *et al.*, 2006; Sutton and McCallum, 2007a; Casado *et al.*, 2007).

Damping is used to reduce the update step size in messages (or sometimes, beliefs). Additive damping has the following form

$$\mu_{j\to i}^{t+1}(x_i) \leftarrow (1-d)\mu_{j\to i}^{t+1}(x_i) + d\mu_{j\to i}^{t}(x_i) \tag{2.80}$$

where $d \in (0, 1]$ is the damping factor, and the superscript t denotes the iteration. Multiplicative damping is also occasionally used

$$\mu_{j\to i}^{t+1}(x_i) \leftarrow \left(\mu_{j\to i}^{t+1}(x_i)\right)^{1-d} \left(\mu_{j\to i}^{t}(x_i)\right)^{d} \tag{2.81}$$

Typically, setting a large value of d yields better convergence quality, but slower rates.

The message passing scheme requires $\mathcal{O}(2|\mathcal{E}||S|)$ memory to store all the messages, where $|\mathcal{E}|$ is number of edges in the graph $\mathcal{G}$. The memory will be very demanding for large images (such as those with height $H = 1000$ and width $W = 1000$, $|S| = 256$; and $|\mathcal{E}| \approx 2HW$).

Despite the lack of guiding theory, empirical evidence has suggested that loopy BP still works well in a wide range of problems (Murphy *et al.*, 1999; Yedidia *et al.*, 2005). It remains one of the most widely used approximate techniques in graphical model applications. Research in improving BP and characterising its convergence is an active area (McEliece

and Cheng, 1998; Yedidia *et al.*, 2005; Welling and Teh, 2001; Weiss and Freeman, 2001a; Wiegerinck and Heskes, 2003; Yuille, 2002; Wainwright *et al.*, 2003a,b; Wainwright and Jordan, 2003; Dechter and Mateescu, 2003; Heskes, 2004; Mooij and Kappen, 2005a,b; Ihler *et al.*, 2005).

2.4.6.3 Variational Methods

In this subsection we deal with structured variational approximation, in that the whole network is partitioned into a number of independent sub-networks (see Figure 2.7). The partitioning effectively removes edges connecting sub-networks. When each sub-network is a single node, the method reduces to the well-known mean-field.

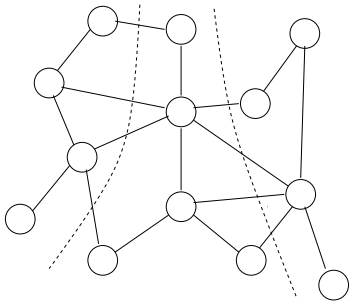


Figure 2.7: Partitioning intractable networks into tractable sub-networks. Dashed lines indicate boundaries between sub-networks.

The main assumption is that the approximate distribution $Q(x)$ of $\Pr(x)$ is factorised as follows

$$Q(x) = \prod_{\alpha} Q_{\alpha}(x_{\alpha}) \tag{2.82}$$

where α is the index of the sub-networks.

Since $Q(x)$ is an approximation to the $\Pr(x)$, the natural goal is to minimise the distance between the two distributions. In the variational approach the Kullback-Leibler divergence is minimised

$$\hat{Q} = \arg\min_{Q} KL(Q||\Pr) \tag{2.83}$$

$$= \arg\min_{Q} \sum_{x} Q(x) \log \frac{Q(x)}{\Pr(x)} \tag{2.84}$$

subject to the constraints in Equation 2.82.

Let $x_{c,\alpha,\beta}$ be the sub-set of clique variables at the boundary between the sub-networks α and β, i.e. $c \in \alpha \cap \beta$. These cliques are split when partitioning, and $x_{c,\alpha}$ belongs to the

sub-network α, $x_{c,\beta}$ belongs to β. Denote the message sent from β to α with respect to the clique c as

$$\mu_{\beta\to\alpha}(x_{c,\alpha}) = \exp\left(\sum_{x_{c,\beta}} Q_{c,\beta}(x_{c,\beta}) \log \psi_{c,\alpha,\beta}(x_{c,\alpha,\beta})\right) \tag{2.85}$$

Then the distribution of the sub-network α is given as

$$Q_\alpha(x_\alpha) \propto \Phi_\alpha(x_\alpha) \prod_{\beta\in\mathcal{N}(\alpha)} \prod_{c\in\alpha\cap\beta} \mu_{\beta\to\alpha}(x_{c,\alpha}) \tag{2.86}$$

where $\mathcal{N}(\alpha)$ is the set of neighbour sub-networks of the sub-network α and $\Phi_\alpha(x_\alpha)$ is the product of local clique potentials belonging the to sub-network α. That is, the distribution of a sub-network in the variational method is proportional to the potential of its variables, and all of the messages coming from its neighbourhood. This is very similar to the case of Belief Propagation (as in Equation 2.77). The only difference is how the messages are computed.

Note that we have assumed each sub-network α to be tractable, in that $Q_{c,\alpha}(x_{c,\alpha}) = \sum_{x_\alpha \setminus x_c} Q_\alpha(x_\alpha)$ can be evaluated efficiently. Thus Equations 2.85 and 2.86 provide a *recursive* relationship between the distributions of sub-networks. originally we do not know any distributions for sure, we need to iteratively run the message updating (Equation 2.85) and distribution revision (Equation 2.86), and hope it will converge.

The derivation details of Equations 2.85 and 2.86 are given in Appendix A.1.

Remark: One of the main problems of variational methods is that it does not handle well the case with zero potentials. Zero potentials mean certain configurations of the local cliques are prohibited, or equivalently, have zero probability. If such cliques are broken due to network partitioning, then the resulting approximation will be inconsistent. This issue does not seem not to have adequate treatment in the literature. Another problem is that if the interaction between nodes at the removed edges is strong, then the resulting approximation will be poor because discriminative information is lost.

2.4.7 Approximate Prediction

2.4.7.1 Iterated Conditional Mode

The ICM (Besag, 1986) is a fast local method that performs greedy search. The idea is quite simple, in that we iteratively find the local optima of the conditional distribution:

$$\hat{x}_i \leftarrow \arg\max_{x_i \in S_i} \Pr(x_i | \mathcal{N}(i)) \tag{2.87}$$

The process is repeated for all nodes in the network until convergence. The main drawback of this method is that it is sensitive to initialisation and may be trapped in poor local optima.

2.4.7.2 Loopy Max-Product Algorithms

The loopy max-product algorithm is the Pearl's max-product algorithm applied for the loopy networks. Messages are sent in all directions along edges and updated at each step as

$$\mu_{j \to i}(x_i) \propto \max_{x_j} \left[\phi(x_j)\psi(x_i, x_j) \prod_{k \in \mathcal{N}(j), k \neq i} \mu_{k \to j}(x_j) \right] \tag{2.88}$$

The maximal beliefs are computed using the same equation as in Equation 2.77. As with message passing algorithms on general graphs, the loopy max-product is not guaranteed to converge, especially in MRFs with strong interaction between nodes, and it requires significant memory to store all messages for large models. Fortunately, the max-product often finds good solutions that are close to the optimum in practice.

There has been a strong interest in loopy max-product algorithms due to its wide applicability in many areas. Beside issues of convergence and quality of solution found by the loopy max-product, we need to take care of large state spaces, especially in computer vision. Work in theoretical characterisation and improvement includes (Weiss and Freeman, 2001b; Yanover and Weiss, 2003; Kolmogorov, 2005; Wainwright *et al.*, 2005a; Meltzer *et al.*, 2005; Kolmogorov and Wainwright, 2005; Felzenszwalb and Huttenlocher, 2006; Kolmogorov, 2006; Coughlan and Shen, 2006; Leordeanu and Hebert, 2006; Ravikumar and Lafferty, 2006; Yanover *et al.*, 2006; Johnson *et al.*, 2007; Sanghavi, 2007; Gupta *et al.*, 2007; Duchi *et al.*, 2007).

2.4.7.3 Graph-Cuts

Graph-Cuts have been shown to be very successful on certain classes of vision problems (Boykov *et al.*, 2001; Szeliski *et al.*, 2006). They are, nevertheless, designed with specific cost functions in mind (i.e. *metric* and *semi-metric*), and therefore inapplicable for generic cost functions such as those resulting from learning.

2.4.8 Factor Graphs

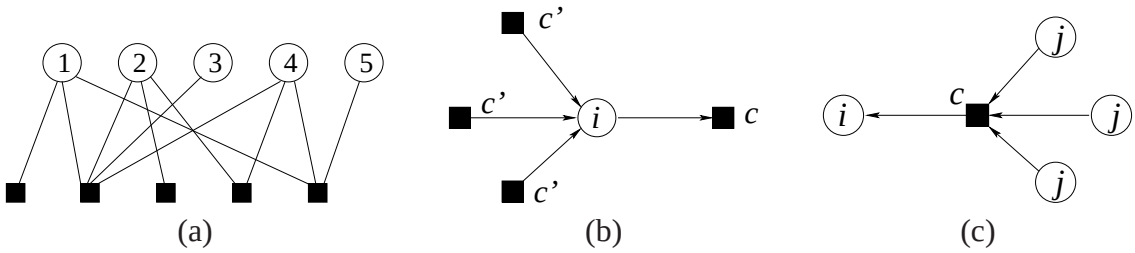


Figure 2.8: Examples of Factor Graph (a), which is a generalisation of the Bayesian Network in Figure 2.2a by grouping local conditional distribution in factor nodes (filled squares). Messages passing from node to factor (b) and factor to node (c).

Factor Graphs (Kschischang *et al.*, 2001; Yedidia *et al.*, 2005) introduce a new way to represent MRFs in that both the variables, the potential functions (called 'factors') and their connections are jointly represented. There are no direct connections between nodes of the same type. Figure 2.8a shows a factor graph, which is converted from the Bayesian Network in Figure 2.2a. Sometimes it is meaningful to have a factor node to encode a particular feature, and thus a variable node can have multiple factor nodes associated with it. As expected, the joint distribution of the variables is defined in the same way as Equation 2.24, but now $\psi_c(x_c)$ is a function associated with the factor c that connects node variables in x_c.

As factor graphs are just an alternative (but more expressive) way of representing MRFs, the Markov property is also preserved. The Markov blanket of a variable node consists of all variable nodes that share some factor nodes with it. Similarly, inference in factor graphs can also be carried out using Pearl's belief propagation. The sum-product algorithm works as follows. Since there are only direct connections between variable nodes and factor nodes, messages are sent from variable nodes to its associated factors, and vice versa. The message sent from a node i to a factor c (Figure 2.8b) is updated as

$$\mu_{i\to c}(x_i) = \prod_{c'\in C(i), c'\neq c} \eta_{c'\to i}(x_i) \tag{2.89}$$

where $C(i)$ is the set of all neighbour factors associated with node i. And the messages

sent from a factor c to a node i (Figure 2.8c) is updated as

$$\eta_{c\to i}(x_i) = \sum_{x_c \setminus x_i} \left(\psi_c(x_c) \prod_{j|x_j \in x_c, j \neq i} \mu_{j\to c}(x_j) \right) \tag{2.90}$$

Finally, the local beliefs (approximate marginals) are computed as

$$b(x_c) \propto \psi_c(x_c) \prod_{j|x_j \in x_c} \mu_{j\to c}(x_j) \tag{2.91}$$

$$b(x_i) = \sum_{x_c \setminus x_i} b(x_c) \tag{2.92}$$

A recent extension of factor graphs is introduced in (Frey, 2003) as a unification of directed and undirected graphical models.

2.5 Probabilistic Hierarchical modelling

Modelling hierarchical aspects in complex dynamic processes is an important research issue in many application domains ranging from computer vision, text information extraction, computational linguistics to biological computation. For example, in a syntactic parsing task known as noun-phrase chunking, noun-phrases (NPs) and part-of-speech tags (POS) are two layers of semantics associated with words in the sentence. Previous methods first tag the POS and then feeds these tags as input to the chunker. The POS tagger takes no information from the NPs. This layered approach, however, may not be optimal, as a noun-phrase is often very informative to infer the POS tags belonging to the phrase. In addition, it suffers from the so-called *cascading error* problem (e.g. see (Finkel *et al.*, 2006)), as the error introduced from the lower layer will propagate to higher tasks. Thus, it is more desirable to *jointly* model and infer both the NPs and the POS tags at the same time (e.g. see (Sutton *et al.*, 2007)).

Many models have been proposed to address this challenge, for which solutions can be largely categorised as either graphical models extending the flat hidden Markov models (HMM) (e.g., the layered HMM (Oliver *et al.*, 2004), the abstract HMM (Bui *et al.*, 2002), hierarchical HMM (HHMM) (Fine *et al.*, 1998; Bui *et al.*, 2004), DBN (Murphy, 2002)) or grammar-based models (e.g., PCFG (Pereira and Schabes, 1992)). These models are all *generative*.

Recent development in discriminative, hierarchical structures include extension of the flat CRFs (e.g. dynamic CRFs (DCRF) (Sutton *et al.*, 2007), hierarchical CRFs (Liao *et al.*, 2007; Kumar and Hebert, 2005)) and conditional learning of the grammars (e.g. see (Miyao

and Tsujii, 2002; Clark and Curran, 2003)). The main problem of the DCRFs is that they are not scalable due to inference intractability. The hierarchical CRFs, on the other hand, are tractable but assume fixed tree structures, and therefore are not flexible to adapt to complex data. For example, in the noun-phrase chunking problem no prior tree structures are known. Rather, if such a structure exists, it can only be discovered after the model has been successfully built and learned.

The conditional probabilistic context-free grammar (C-PCFG) appears to address both tractability and dynamic structure issues. More precisely, in C-PCFGs it takes cubic time in sequence length to parse a sentence. However, the context-free grammar does not limit the depth of semantic hierarchy, thus making it unnecessarily difficult to map many hierarchical problems into its form. Secondly, it lacks a graphical model representation and thus does not enjoy the rich set of approximate inference techniques available in graphical models.

2.5.1 Hierarchical Hidden Markov Models

Hierarchical HMMs are generalisations of HMMs (see Section 2.4.3.1) in the way that a state in an HHMM may not emit a single observation symbol but a sub-sequence of observations, and a state may be a sub-HHMM. On other words, an HHMM is a nested Markov chain. In the model temporal evolution, when a child Markov chain terminates, it returns the control to its parent. Nothing from the terminated child chain is carried forward. Thus, the parent state abstracts out everything belonging to it. Upon receiving the return control the parent then either transits to a new parent, (given that the grand parent has not finished), or terminates.

Figure 2.9 illustrates the state transition diagram of a two-level HHMM. At the top level there are two parent states $\{A, B\}$. The parent A has three children, i.e. $ch(A) = \{1, 2, 3\}$ and B has four, i.e. $ch(B) = \{1, 2, 3, 4\}$. Note that we have assumed that the parents share some common children, i.e. $ch(A) \cap ch(B) = \{1, 2, 3\}$. This structure sharing follows the work of (Bui *et al.*, 2004). At the top level the transitions are between A and B, as in a normal directed Markov chain. Under each parent there are also transitions between child states, which only depend on the direct parent (either A or B). There are special ending states (represented as shaded nodes in Figure 2.9) to signify the termination of the Markov chains. At each time step of the child Markov chain, a child will emit an observational symbol (not shown here).

The temporal evolution of the HHMM can be represented as a dynamic Bayesian network, which was first done in (Murphy and Paskin, 2002). Figure 2.10 depicts a DBN structure of 3 levels. The bottom level is often referred to as *production level*. Associated with each

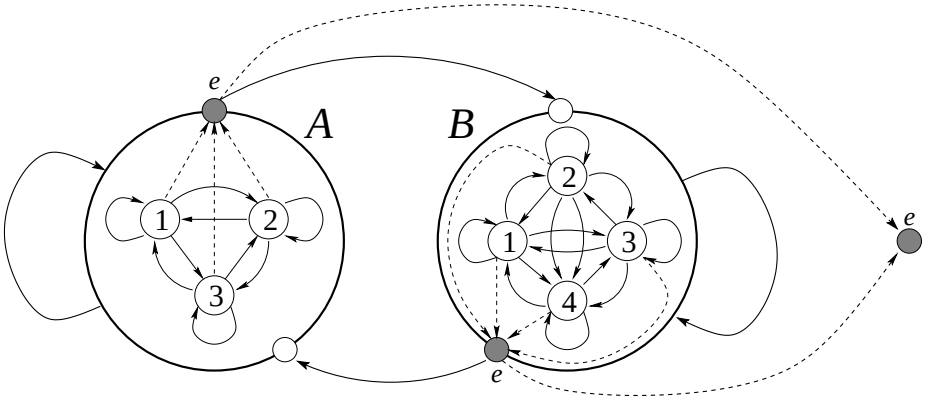


Figure 2.9: The state transition diagram of an HHMM.

state is an ending indicator to signify the termination of the state. Denote by x_t^d and e_t^d the state and ending indicator at level d and time t, respectively. When $e_t^d = 0$, the state x_t^d continues, i.e. $x_t^d = x_{t+1}^d$. And when $e_t^d = 1$, the state x_t^d transits to a new state, or transits to itself. There are hierarchical consistency rules that must be ensured. Whenever a state persists (i.e. $e_t^d = 0$), all of the states above it must also persist (i.e. $e_t^{d'} = 0$ for all $d' < d$). Similarly, whenever a state ends (i.e $e_t^d = 1$), all of the states below it must also end (i.e. $e_t^{d'} = 1$ for all $d' > d$).

Inference and learning in HHMMs follow the Inside-Outside algorithm of the probabilistic context-free grammars. Overall, the algorithm has $\mathcal{O}(|S|^3 DT^3)$ time complexity where $|S|$ is the maximum size of the state space at each level, D is the depth of the model and T is the model length. This is costly for large T.

When representing as a DBN, the whole stack of states $x_t^{1:D}$ can be collapsed into a 'mega-state' of a big HMM, and therefore inference can be carried out in $\mathcal{O}(|S|^{2D} T)$ time. This is efficient for a shallow model (i.e. D is small), but problematic for a deep model (i.e. D is large).

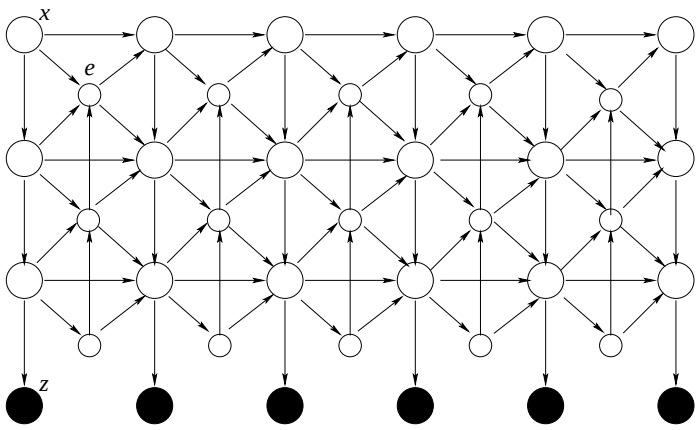


Figure 2.10: Dynamic Bayesian network representation of HHMMs.

2.5.2 Abstract Hidden Markov Models

Like HHMMs, an abstract HMM is a also multi-scale HMM. It is proposed largely for the purpose of *plan recognition* (Kautz and Allen, 1986), a sub-area of Artificial Intelligence where the goal is to recognise the execution plan of an agent acting in an environment. There are four main elements in AHMMs: observation symbol, state, action and abstract policy. States are at the bottom level which emit observation symbols just like an ordinary HMM. The observations typically represent noisy signals the agent perceives from the environment. Right above the state level is the *action* level representing concrete actions of the agent that will alter the agent's states. The actions are assumed to be generated by a stack of *abstract policies*. The policy stack is quite similar to the state stack of HHMMs above the production level. The main difference is that in AHMMs, the policies and their termination depend on the state at production level. In HHMMs, on the contrary, the production states never directly influence the parents.

A DBN representation of the AHMM is given in Figure 2.11. Inference in the AHMM, unfortunately, is generally intractable, except for shallow networks with a small number of abstract policies. Approximate methods, therefore, must be used. In (Bui *et al.*, 2002), the authors employ a sampling based method based on the combination of Rao-Blackwellisation (Casella and Robert, 1996) and Sequential Importance Sampling (e.g. see (Andrieu *et al.*, 2003)).

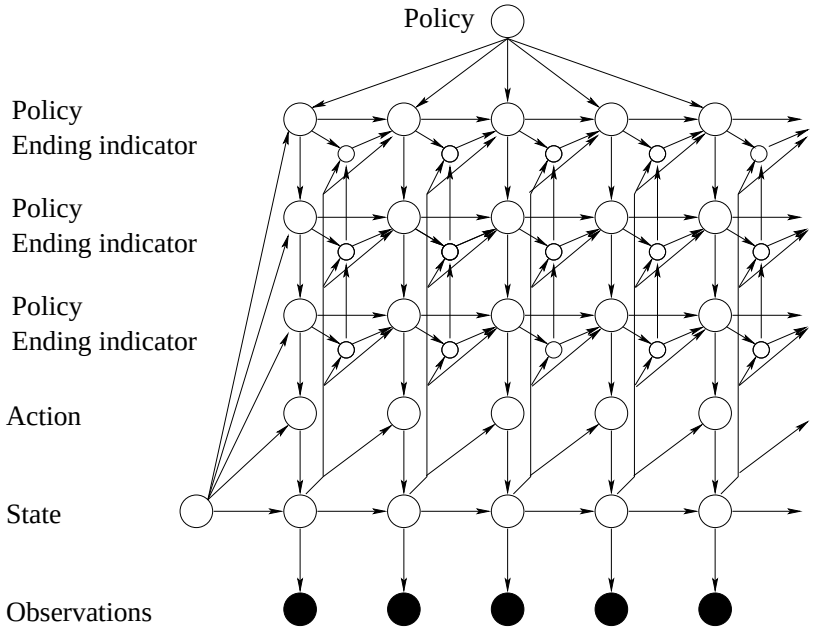


Figure 2.11: Dynamic Bayesian network representation of AHMMs.

2.6 Closing Remarks

This chapter has reviewed background necessary for further development of the thesis. Starting from the general problem of classification with structured output space, we described two important elements of statistical machine learning: multi-class logistic classifiers and an effective data modelling machinery known as graphical models. The former has a strong connection with the exponential family of distributions, and thus with the Maximum Entropy principle. In graphical models, we mainly focus our attention to the undirected setting, which essentially includes the directed counterpart as a special case. Reviewed details include representation, learning and inference under different network structures: the n-order Markov chains, the Markov tree, the general networks, exact and approximate inference, factor graphs, and hierarchical models.

In the next chapter we narrow down the subject to the main focus of this thesis - the Conditional Random Field (Lafferty *et al.*, 2001), which is a combination of the multi-class logistic classifier and undirected graphical models.

Chapter 3

Conditional Random Fields

In this chapter we describe Conditional Random Fields (CRFs) (Lafferty *et al.*, 2001), which are undirected graphical models for structured output. CRFs define distributions over structured output variables conditioned on some input variables. For example, in applications such as Part-of-Speech (POS) tagging, the output variables are a sequence of POS tags that we want to predict from the input sentence. In image scene segmentation the output variables are 2D arrays of scene interpretation of the raw pixels.

3.1 Model Description

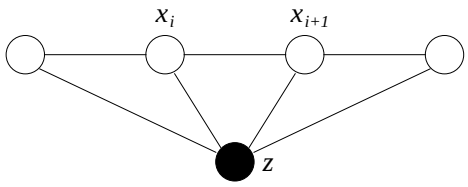


Figure 3.1: A chain-structured CRF. Empty circles denote state variables x and filled circle denotes conditioning variables z.

Denote by z the input variable and $x = (x_1, x_2, ..., x_N)$ the joint output variable. The input variable z represents our knowledge about the domain. The output variable x has some structure that specifies the interactions between its component variables (x_i). For example, in sequential modelling problems $(x_1, x_2, ..., x_T)$ is a chain of length T, and the interactions are between pairs of successive variables (x_i, x_{i+1}) (see Figure 3.1).

We would like to model the mapping from z to x via the conditional distribution $\Pr(x|z)$. Thus we are only interested in the output structure conditioned on the input. The input distribution $\Pr(z)$ is left unspecified. Conditional Random Fields approach the modelling

of $\Pr(x|z)$ by representing x as a Markov random field. More precisely, x is represented using a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$[1] in which each vertex of the graph corresponds to a variable x_i. The joint variable x, when conditioned on z, admits the Markov property in that the conditional distribution of x_i given its neighbours, defined by the graph $\mathcal{G}$, does not depend on other variables outside the neighbourhood. This is formally defined in Definition 1.

Definition 1. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph such that $x = (x_i)_{i \in \mathcal{V}}$ is indexed by the vertices of $\mathcal{G}$. Then (x, z) is a conditional random field such that, when conditioned on z, the random variables x obey the Markov property with respect to the graph: $\Pr(x_i|z, x_j, j \neq i) = \Pr(x_i|z, x_j, j \in \mathcal{N}(i))$, where $\mathcal{N}(i)$ is the neighbourhood of x_i.*

The conditional distribution $\Pr(x|z)$ is therefore given as

$$\Pr(x|z) = \frac{1}{Z(z)}\Phi(x, z) = \frac{1}{Z(z)} \prod_c \psi_c(x_c, z) \tag{3.1}$$

where c is the index of the cliques specified by the structure of the graph $\mathcal{G}$, x_c is the joint variable associated with the clique c, $\psi_c(x_c, z)$ is the non-negative potential function defined over c, and $Z(z) = \sum_x \Phi(x, z) = \sum_x \prod_c \psi_c(x_c, z)$ is the partition function with respect to the input z. The clique potentials specify how local variables interact and how much the interaction contributes to the global distribution. For example, in the chain structured CRF, as illustrated in Figure 3.1, a clique is a segment of the chain that contains two nodes (x_i, x_{i+1}). There is one partition function for each input z to ensure the normalisation of the distribution $\Pr(x|z)$. This is different from standard Markov random fields (Section 2.4) where there is a single partition function for all data cases.

Typically, we parameterise the potential function in an exponential form

$$\psi_c(x_c, z) = \exp(\mathbf{w}^\top \mathbf{f}(x_c, z)) \tag{3.2}$$

where $\mathbf{w} = (w_1, w_2, ..., w_K)^\top \in \mathbb{R}^K$ is the parameter vector, and $\mathbf{f} = (f_1, f_2, ..., f_K)^\top$ is the feature vector. Basically features are functions that encode prior belief about dependency between the conditioning variable z and the output pattern x. Generally the features map the input z and the associated clique variable x_c to some real or binary value. The parameters $\{w_k\}$ are the weights of corresponding feature $\{f_k(.)\}$ and thus specify how features contribute to the global distribution. Note that we have used the same parameter vector across clique potentials. This is known as parameter tying.

[1] It is worth mentioning that the graph $\mathcal{G}$ is not uniquely defined for all data instances. Instead, it depends on the nature of the each data instance. For example, POS tagging, the chain of POS tags varies with sentence length.

Let $\mathbf{F}(x, z) = \sum_c \mathbf{f}(x_c, z)$, then Equation 3.1 becomes

$$\Pr(x|z) = \frac{1}{Z(z)} \exp(\mathbf{w}^\top \mathbf{F}(x, z)) \tag{3.3}$$

which is essentially the conditional Maximum Entropy model (see Section 2.2).

Once the conditional distribution $\Pr(x|z)$ has been estimated, various inference tasks can be performed. The most important task is to predict the output x given the input z using

$$\hat{x} = \arg\max_x \Pr(x|z) = \arg\max_x \mathbf{w}^\top \mathbf{F}(x, z) \tag{3.4}$$

Other common tasks include computing the log-partition function $Z(z)$ and the marginals $\Pr_c(x_c|z)$. In general, inference in a CRF (x, z) (see Definition 1) for each input observation z and state variable x is identical to that in the underlying Markov random field imposed on x. For this reason, we do not describe the details of inference further and readers are referred to the description in Section 2.4.

3.2 Parameter Estimation

In this section we discuss how to estimate $\mathbf{w}$ from training data. First, we describe the case of fully observed data in which all the output patterns are fully specified. We are given a set of n training instances $\mathcal{D} = \{x^{(l)}, z^{(l)}\}_{l=1}^n$. Assume further that these training instances are independently and identically distributed. Note that this assumption does not invalidate the dependencies *within* each output pattern $x^{(l)}$.

The most popular method is based on the maximum likelihood (ML) principle, which selects the parameter that maximises the conditional likelihood.

$$\begin{aligned} \hat{\mathbf{w}} &= \arg\max_{\mathbf{w}} \mathcal{L}(\mathcal{D}; \mathbf{w}); \qquad \text{where} \\ \mathcal{L}(\mathcal{D}; \mathbf{w}) &= \sum_{l=1}^{n} \log \Pr(x^{(l)}|z^{(l)}; \mathbf{w}) \\ &= \sum_{l=1}^{n} \left\{ \mathbf{w}^\top \mathbf{F}(x^{(l)}, z^{(l)}) - \log Z(z^{(l)}) \right\} \end{aligned} \tag{3.5}$$

Typically, we add a quadratic penalty term[2] to the log-likelihood for *regularisation*[3]

$$\mathcal{L}(\mathcal{D};\mathbf{w}) = \sum_{l=1}^{n} \left\{\mathbf{w}^\top \mathbf{F}(x^{(l)}, z^{(l)}) - \log Z(z^{(l)})\right\} - \sum_{k=1}^{K} \frac{w_k^2}{2\sigma_k^2} \tag{3.6}$$

where $\{\sigma_k^2\}_{k=1}^K$ specify how much the penalty is applied. In general, the penalty prevents the absolute values of parameters $\{|w_k|\}$ from becoming too large. This method has a Bayesian interpretation, in that the parameter $\mathbf{w}$ is treated as a random multivariate Gaussian with mean 0 and diagonal covariance matrix. Let $\Pr(\mathbf{w}) \propto \exp(-\sum_{k=1}^K w_k^2/2\sigma_k^2)$ be the prior distribution, the posterior is computed as

$$\begin{aligned} \Pr(\mathbf{w}|\mathcal{D}) &\propto \Pr(\mathbf{w}) \Pr(\mathcal{D}|\mathbf{w}) \\ &= \Pr(\mathbf{w}) \prod_{l=1}^{n} \Pr(x^{(l)}|z^{(l)};\mathbf{w}) \Pr(z^{(l)}) \end{aligned} \tag{3.7}$$

Taking the log of both sides and ignoring the terms associated with $\Pr(z^{(l)})$, which is independent of $\mathbf{w}$, we arrive at the RHS of Equation 3.6. Another less popular choice or prior is the Laplace distribution[4], i.e. $\Pr(\mathbf{w}) \propto \exp(-\sum_k \beta_k |w_k|)$ for $\beta_k > 0$. In general, the Laplace distribution penalises large parameters more severely than the Gaussian, and it often results in many zero parameters.

What remains is to apply optimisation methods to find the maximiser of $\mathcal{L}(\mathcal{D};\mathbf{w})$ in Equation 3.6. An important property of $\mathcal{L}(\mathcal{D};\mathbf{w})$ is that it is concave, and thus there exists a unique global maximiser. A popular method is gradient-based in which we seek to find the solution that sets the gradient of the penalised log-likelihood to zero. The partial derivative is computed as

$$\frac{\partial \mathcal{L}(\mathcal{D};\mathbf{w})}{\partial w_k} = \sum_{l=1}^{n} \left\{ F_k(x^{(l)}, z^{(l)}) - \sum_x \Pr(x|z^{(l)}) F_k(x, z^{(l)}) \right\} - \frac{w_k}{\sigma_k^2} \tag{3.8}$$

$$= n\left\{\mathbb{E}_{\tilde{x}|z}[F_k] - \mathbb{E}_{x|z}[F_k]\right\} - \frac{w_k}{\sigma_k^2} \tag{3.9}$$

where $\mathbb{E}_{\tilde{x}|z}[F_k]$ is the empirical distribution based on training data. Since $\mathbf{F}(x,z) = \sum_c \mathbf{f}(x_c, z)$, we have

$$\frac{\partial \mathcal{L}(\mathcal{D};\mathbf{w})}{\partial w_k} = \sum_{l=1}^{n} \sum_c \left\{ f_k(x_c^{(l)}, z^{(l)}) - \sum_{x_c} \Pr(x_c|z^{(l)}) f_k(x_c, z^{(l)}) \right\} - \frac{w_k}{\sigma_k^2} \tag{3.10}$$

Thus, the computation boils down to computing clique marginals $\Pr(x_c|z)$. This has been

[2]This term is commonly called norm-l_2 regularisation.

[3]Regularisation is necessary for *ill-posed* estimation problem in that a small deviation in the objective function cause large deviation in the solution.

[4]This prior has several other names, for example, norm-l_1 regularisation and Lasso (Tibshirani, 1996) in the context of regression. This method has been widely used recently to achieve sparsity.

described in Section 2.4.

Since setting this gradient to zero does not result in any closed form solution, we typically resort to iterative methods. Since most applications of CRFs are large-scale, pure Newton methods that require computing the second order derivative matrix is impractical. Better choices include the Conjugate Gradients (Hestenes and Stiefel, 1952) and the limited memory quasi-Newton method called L-BFGS (Liu and Nocedal, 1989; Byrd *et al.*, 1994). These two methods are efficient and require only a few gradient evaluations in each round. Indeed, the L-BFGS has been the method of choice since the work of (Sha and Pereira, 2003).

The extension to missing labels is quite straightforward. Let $x^{(l)} = (\vartheta^{(l)}, h^{(l)})$ where $\vartheta^{(l)}$ is the subset of visible patterns and $h^{(l)}$ the hidden. The full likelihood in Equation 3.5 is now replaced by the incomplete likelihood

$$\begin{aligned}
\mathcal{L}_{incom}(\mathcal{D}; \mathbf{w}) &= \sum_{l=1}^{n} \log \Pr(\vartheta^{(l)} | z^{(l)}; \mathbf{w}) \\
&= \sum_{l=1}^{n} \log \sum_{h^{(l)}} \Pr(\vartheta^{(l)}, h^{(l)} | z^{(l)}; \mathbf{w}) \\
&= \sum_{l=1}^{n} \left\{ \log Z(\vartheta^{(l)}, z^{(l)}) - \log Z(z^{(l)}) \right\} \qquad (3.11)
\end{aligned}$$

where $Z(\vartheta^{(l)}, z^{(l)}) = \sum_{h^{(l)}} \Phi(\vartheta^{(l)}, h^{(l)}, z^{(l)})$. The gradient is now

$$\begin{aligned}
\frac{\partial \mathcal{L}_{incom}(\mathcal{D}; \mathbf{w})}{\partial w_k} &= \sum_{l=1}^{n} \left\{ \sum_{h} \Pr(h | \vartheta^{(l)}, z^{(l)}) F_k(\vartheta^{(l)}, h, z^{(l)}) - \sum_{x} \Pr(x | z^{(l)}) F_k(x, z^{(l)}) \right\} \\
&= n \left\{ \mathbb{E}_{h|\vartheta,z}[F_k] - \mathbb{E}_{x|z}[F_k] \right\} \qquad (3.12)
\end{aligned}$$

where

$$\begin{aligned}
\mathbb{E}_{h|\vartheta,z}[F_k] &= \frac{1}{n} \sum_{l=1}^{n} \sum_{h} \Pr(h | \vartheta^{(l)}, z^{(l)}) F_k(\vartheta^{(l)}, h, z^{(l)}) \qquad \text{and} \\
\mathbb{E}_{x|z}[F_k] &= \frac{1}{n} \sum_{l=1}^{n} \sum_{x} \Pr(x | z^{(l)}) F_k(x, z^{(l)})
\end{aligned}$$

respectively.

3.3 Feature Engineering and Selection

There is no doubt that features are very crucial to the success of CRF-based systems because they are meant to capture essential information about the data and relations between the input and output. In some applications, features are the output of the pre-processing step. For example, in image scene segmentation, the input variables $(z_i)_{i=1}^N$ are just raw RGB values of pixels. Using RGB as features is not very informative because scenes like water and sky may locally share very similar sets of colours. For this reason we often apply a set of filters to detect interesting local patterns such as edges and textures, and use these patterns as features.

As CRFs allow arbitrary and complex features to be included, it is easy to arrive at an excessively large feature pool by considering many combinations of basic features. Although coverage is important to ensure that no useful information is missing, there are many problems associated with large feature sets. First, many features are effectively noisy, i.e. they are not indicative of the dependence between the data and the output patterns. For example, in POS tagging, most associations between a particular POS tag and words, which occur in the same sentence but far away from the tag, happens only once in the training data. This rare association cannot be robustly learnt, generally.

Second, since each feature is associated with a parameter, a large feature pool leads to a problem known as *overfitting*. In this case parameters are easily tuned to fit the training data well, but generalise poorly in unseen data.

Third, some practical applications, such as those used in decision making (e.g. in medicine and business), require interpretation of features selected. Complex features may be too difficult to interpret. Finally, large feature pools are costly to process both in terms of storage and run time.

Feature selection has been widely studied in the machine learning literature (Guyon and Elisseeff, 2003) in unstructured output spaces. Broadly speaking there are three approaches to this problem. The *filtering* approach employs some simple and fast heuristics to select the features according to some independent criteria. The *wrapper* approach extensively evaluates the feature combinations according to the final performance measure. And finally, the *embedded* approach incrementally builds the feature set as learning proceeds.

Common filtering methods include the simple cut-off of infrequent features, and correlation or mutual information between the input and the output. The main drawback is that the selection may not correlate well with the final performance. In contrast, the wrapper approach is thorough, but it is computationally expensive because of the combinatorial nature of the problem. It also connects with the learning only through the final evaluation,

and thus information gathered during learning is wasted. The third approach represents a trade-off between these two extremes. For example, in boosting (Schapire and Singer, 1999), features are iteratively added and their weights are adjusted in a greedy manner.

In the case of CRFs, feature selection must be efficient since the CRFs themselves are expensive to evaluate. The filtering-based and embedded approaches are therefore more suitable. The filtering approach, especially the simple frequency cut-off, is popular due to its simplicity. However, the frequency cut-off may select popular but irrelevant features, and as a result, the selected feature set is usually large. For example, in language modelling, common words like 'the' appear almost everywhere and do not often add value to the feature set. On the other hand, it sometimes removes rare but highly relevant features. An alternative to the frequency cut-off is to employ some simplified and efficient version of the CRFs to do the feature selection task. For example, the pseudo-likelihood (Besag, 1975) can be a good simplified version of the true likelihood because of its efficiency and consistency.

There have been some studies following the embedded approach for CRFs. A feature induction method for MRF introduced in (Pietra *et al.*, 1997), incrementally adds features that most reduce the Kullback-Leibler divergence between the model distribution and the empirical observations. Although this method is theoretically interesting, it is iterative and thus requires repeated inference in MRFs, which is intractable in general. For CRFs, a similar approach is used in (McCallum, 2003), in which some simple approximations such as mean fields are employed to improve the feature induction speed. The author of (McCallum, 2003) reports great saving in feature set size in some large-scale NLP applications. Another method that exploits the feature selection property of boosting has been studied in (Altun *et al.*, 2003a; Dietterich *et al.*, 2004).

3.4 Applications

The early motivation of CRFs is from the area of Information Extraction (IE) (Lafferty *et al.*, 2001; Pinto *et al.*, 2003; Peng and McCallum, 2004; Kristjannson *et al.*, 2004), in which given a dataset (mostly texts), we extract relevant information that belongs to some predefined types (such as proper names, locations and time). As text is inherently sequential, imposing a chain structure on the text is both effective in capturing temporal relations, and efficient in inference and learning. As a result, CRFs have been quickly adopted for a wide range of text processing applications, for example, part-of-speech tagging (POS) and chunking (Sha and Pereira, 2003; Sutton *et al.*, 2007) and semantic role labeling (Cohn and Blunsom, 2005). More recently, the application of CRFs has been expanded to word

alignment (Blunsom and Cohn, 2006)[5], question answering (Hickl and Harabagiu, 2006), and document summarisation (Shen *et al.*, 2007).

In fact, CRFs are applicable to any domain that allows supervised learning and such domains were previously dominated by HMMs. These include speech recognition (Gregory and Altun, 2004; Roark *et al.*, 2004; Gunawardana *et al.*, 2005; Liu *et al.*, 2005; Morris and Fosler-Lussier, 2006), word-segmentation (Peng *et al.*, 2004; Zhang *et al.*, 2006a), and activity recognition (Liao *et al.*, 2007; Truyen *et al.*, 2006; Liao *et al.*, 2005; Sminchisescu *et al.*, 2006; Taycher *et al.*, 2006; Quattoni *et al.*, 2007; Vail *et al.*, 2007).

Generally speaking, CRFs are suitable for labeling and segmentation of structured data, given that the labels are available for training. Since a CRF is a conditional MRF, it is not surprising that CRFs have been applied to traditional domains of MRF such as image processing. Specifically, these include image segmentation and labeling, both for static images (Kumar and Hebert, 2004; He *et al.*, 2004; Kumar and Hebert, 2005; Cowans and Szummer, 2005) and video (Winn and Shotton, 2006; Loe *et al.*, 2006). In (Torralba *et al.*, 2005) a random field is used to model the contextual relation between scenes and objects and in (Quattoni *et al.*, 2005) object parts are connected in a hidden tree graph for object classification. CRFs also find application in stereo vision (Scharstein and Pal., 2007).

In recent years there has been much interest in *collective classification* (Jensen *et al.*, 2004; Macskassy and Provost, 2007) in that entities are interconnected so that it is better to classify them collectively rather than individually. For example, Web pages are hyperlinked and those that are linked often belong to the same category (Taskar *et al.*, 2002). In this area, CRFs are often recast as discriminative relational models (Taskar *et al.*, 2002; Sutton and McCallum, 2006; Truyen *et al.*, 2007).

A summary of applications of CRFs is given in Table 3.1. There are a number of available CRF implementations that vary in programming languages and in support of modelling, inference, learning and data pre-processing features. These include the McCallum's MALLET package[6] for general machine learning, Sarawagi's[7] that supports semi-Markov CRFs (Sarawagi and Cohen, 2004), and Murphy's Matlab toolbox[8] for general inference and graphs.

[5]Word alignment is an essential step of statistical machine translation (Brown *et al.*, 1993).
[6]http://mallet.cs.umass.edu
[7]http://crf.sourceforge.net
[8]http://www.cs.ubc.ca/˜ murphyk/Software/CRF/crf.html

Areas	Publications
Speech recognition	(Gregory and Altun, 2004; Roark *et al.*, 2004) (Gunawardana *et al.*, 2005) (Liu *et al.*, 2005; Morris and Fosler-Lussier, 2006)
Word segmentation	(Peng *et al.*, 2004; Zhang *et al.*, 2006a)
POS tagging & phrase chunking	(Sha and Pereira, 2003; Sutton *et al.*, 2007)
Semantic role labeling	(Cohn and Blunsom, 2005)
Information extraction	(Lafferty *et al.*, 2001; Settles, 2004) (Sarawagi and Cohen, 2004) (Peng and McCallum, 2004) (Kristjannson *et al.*, 2004; Zhu *et al.*, 2005)
Image segmentation	(Kumar and Hebert, 2004; He *et al.*, 2004) (Kumar and Hebert, 2005; Lee *et al.*, 2005) (Winn and Shotton, 2006; Loe *et al.*, 2006)
Object recognition/classification	(Torralba *et al.*, 2005; Quattoni *et al.*, 2005)
Stereo vision	(Scharstein and Pal., 2007)
Activity recognition	(Liao *et al.*, 2007; Truyen *et al.*, 2006) (Liao *et al.*, 2005; Sminchisescu *et al.*, 2006) (Quattoni *et al.*, 2007; Vail *et al.*, 2007) (Taycher *et al.*, 2006)
Web page classification	(Taskar *et al.*, 2002)
Word alignment	(Blunsom and Cohn, 2006)
Document summarisation	(Shen *et al.*, 2007)
Question answering	(Hickl and Harabagiu, 2006)
Bioinformatics	(McDonald and Pereira, 2005; Settles, 2004) (Vinson *et al.*, 2007)

Table 3.1: Some selected applications of Conditional Random Fields.

3.5 Discussion and Related Background

3.5.1 Approximate Learning Methods

An implicit assumption made in the discussion of maximum likelihood learning is that we can compute exactly the clique marginals $\Pr(x_c|z)$, and the partition function $Z(z)$. Unfortunately, this only holds for tree-structured CRFs. For general CRFs, approximations must be used. One approach is to utilise stochastic methods by accepting that quantities required for learning can only be approximated. The other approach seeks alternative objective functions other than likelihood that can be computed exactly.

3.5.1.1 Stochastic Gradients

The first approach typically involves *stochastic gradient* methods (Robbins and Monro, 1951; Zhang, 2004; Vishwanathan *et al.*, 2006). Given an approximate gradient $\nabla\mathcal{L}(\mathcal{D};\mathbf{w})$, the parameter is updated as follows

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + \lambda^t \nabla\mathcal{L}(\mathcal{D};\mathbf{w}^t) \tag{3.13}$$

where t is the iteration index, $\lambda^t > 0$ is the learning rate. Under certain conditions of the randomness of $\mathcal{L}(\mathcal{D};\mathbf{w}^t)$ and the learning rate, this type of learning can be very effective.

In what follows we review a number of important techniques that implement stochastic gradients.

Online-learning. This refers to a general learning strategy in which parameters are updated after the trainer observes a training instance. The update rule is the same as Equation 3.13. Even when the true gradients can be evaluated exactly, the rule is still stochastic because we approximate the gradient of all training instances $\mathcal{D}$ by the gradient of just one instance. This learning strategy contrasts with the *batch-learning* strategy, where the parameters are only updated after seeing all the training instances. Typically, online-learning is very greedy, and thus is much faster but can be slightly less accurate than batch-learning. The method is, therefore, of practical significance where speed is more important than accuracy. This includes situations that require immediate corrections such as those in interactive applications. A simple correction is to update parameters after a small block of training instances, and this may stabilise the learning curve and thus improve accuracy.

Perceptron learning (Rosenblatt, 1958; Freund and Schapire, 1999; Collins, 2002). The method updates the parameter based on the mismatch between the prediction $\hat{x}^t$ (see Equation 3.4) and the true pattern x^t, i.e. $\hat{x}^t \neq x^t$:

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + \lambda\{\mathbf{F}(x^t, z^t) - \mathbf{F}(\hat{x}^t, z^t)\} \tag{3.14}$$

It has been proved that if the data is separable, that is, there exists a $\hat{\mathbf{w}}$ that satisfies $\hat{\mathbf{w}}^\top\mathbf{F}(x^{(l)}, z^{(l)}) \geq \hat{\mathbf{w}}^\top\mathbf{F}(x, z^{(l)})\ \forall x \in \mathcal{X}, x \neq x^{(l)}$, then the perceptron will achieve zero training error after finite steps. Although the Perceptron is not designed to maximise the conditional likelihood, it is a good approximation and often works well in practice.

Contrastive Divergence (Hinton, 2002) reduces the number of time-consuming MCMC steps by running only a few samplings from the empirical distribution. This speeds up training significantly because estimating the 'correct' $\Pr(x|z)$ early during the learning process is not necessary. However, this greedy strategy does introduce bias (Carreira-Perpiñán and Hinton, 2005) in that it will not guarantee to converge to the true maximum

likelihood solution. Fortunately, it has been shown empirically that the bias is relatively small for practical purposes (Hinton, 2002; Carreira-Perpiñán and Hinton, 2005).

3.5.1.2 Pseudo-likelihood

Pseudo-likelihood (Besag, 1975) is one of the most popular objective functions as an alternative to the true likelihood. In particular, we employ the following objective function

$$\mathcal{L}(\mathcal{D}, \mathbf{w}) = \sum_{l=1}^{n} \sum_{i \in \mathcal{V}^{(l)}} \log \Pr(x_i^{(l)} | z^{(l)}; \mathbf{w}) \tag{3.15}$$

where $\Pr(x_i|z) = \sum_{x \setminus x_i} \Pr(x|z)$ and $\mathcal{V}^{(l)}$ is the set of vertices of the graph for the l-th data instance. Pseudo-likelihood is attractive because it is efficient, regardless of network structures, and it is theoretically consistent under some regular conditions. In practice, however, the pseudo-likelihood is known to overestimate the interaction between nodes, and it may underperform methods that require approximate inference. Another drawback is that it does not support missing variables.

We can extend the pseudo-likelihood to cover trees instead of nodes (Sutton and McCallum, 2007b). Specifically, we can, therefore, use $\sum_{\tau} \log \Pr(x_\tau | \mathcal{N}(\tau), z)$ as an alternative criterion to maximum likelihood, where x_τ denotes variables associated with the tree τ embedded in the network, and $\mathcal{N}(\tau)$ denotes neighbouring variables of the tree τ.

3.5.1.3 Reranking

Reranking (Collins, 2005) is an interesting strategy to learn CRFs with intractable structures. It is a two-step procedure:

1. In the first step, we learn a base-classifier, which is generally efficient but not as powerful as CRFs. The base classifier can sometimes be an approximation to the CRFs, or those methods that operate only on local variables. For each data instance (in both training and test sets), we obtain from this base-classifier a set of K top predictions. These K outputs are used as the constrained set in the output space in the next step.

2. In the second step, in the training phase, we learn to rerank the K possible outputs using global constraints. Any algorithms that can produce a ranked list of the K outputs can do the job. In the testing phase, the ranker is again used to rerank the K outputs by the base-classifier on the test data.

For the reranking to work, the base-classifier must be strong enough so that its K best outputs generally cover the true output pattern or some of these outputs are at least very close to the true output. Since K is typically much smaller the size of the full output space, re-ranking is very efficient.

3.5.2 Learning Criteria

In parameter estimation, although we have specifically chosen the conditional likelihood as the objective function, it is not the only criteria. Instead, any objective function that satisfies the following properties can be used:

- **Condition 1**: It is asymptotically consistent. e.g. $\lim_{n\to\infty} \hat{\mathbf{w}}_n = \mathbf{w}^*$, where $\hat{\mathbf{w}}_n$ is the optimal parameter when training with n data instances, and $\mathbf{w}^*$ is the true parameter if it exists.
- **Condition 2**: It is efficient to optimise[9]. Generally, this is for the practical purposes. The efficiency comes from two sources. First, the objective function must be easy to compute. Second, the optimisation must converge rapidly.

Condition 1, although a must from a statistician perspective, it is far from being understood. The CRFs can be considered as multi-class classifiers, and the only criterion currently known to be consistent in this multi-class setting is the conditional likelihood (Lafferty and Wasserman, 2006). In addition, the analysis is expected to be much more involved given the fact that the structured output space is typically exponentially large with respect to the number of nodes in the network. The number of nodes, for example, in the case of syntactic analysis of text, is not fixed but dependent on sentence length. The matter is more complicated since often the likelihood of CRFs and its gradient, which are central quantities in maximum likelihood learning, can only be approximately estimated in general. The nature of such approximation (e.g. bias and variance), which depends on the inference methods being used, has not been fully investigated.

Condition 2 is a common sense requirement in practice. For example, applications in language processing may involve training over hundreds of thousands of sentences and millions of features, and typically require to pass through the whole data hundreds of times. The matter is worse if the state space is large because the time complexity is typically quadratic in the number of states. As an example, in speech recognition the number of states (unique words in the vocabulary) is about $10^3 - 10^4$.

[9]This is different from the concept of efficiency in statistics. We are mostly concerned about the speed of inference and numerical optimisation.

To date, there have been no methods that meet both of these conditions. The pseudo-likelihood discussed in Section 3.5.1.2 has proved to be consistent under some conditions. It is a fast method but it also often overestimates the interaction between variables in practice. In fact, while consistency is a good theoretical property, it is too hard to guarantee in practice, so most learning methods are driven by efficiency and practical needs.

For example, in (Sutton and McCallum, 2005) a subset of variables (called pieces) is treated as an independent data instance. This allows efficient learning and at the same time, estimates the local interactions. However, since the joint model is split in pieces, its statistical properties are hard to characterise. Surprisingly, the experiments in (Sutton and McCallum, 2005) reveal that this strategy is comparable with learning the joint model with BP as the approximate underlying inference. There are several possibilities that may justify this result. First, it has been shown in (Sutton and McCallum, 2005) that piece-wise training maximises the lower-bound of the true likelihood. However, the bound can be rather loose, and it can be shown that the piece-wise training attempts to learn a different model with the same parameter set but larger state-space. To see why, let us 'glue' the pieces together by some unity potential functions. Such potential functions do not add anything to the global potential of the joint network, and thus the pieces are still probabilistically independent as before. But now each original node has been split into multiple nodes, each of which belong to a piece, we have more variables, or equivalently, a larger state-space. Learning a larger state-space somehow provides some smoothing, but it can easily over-smooth if there are too many pieces. In fact, the experimental results are compared against the usual global likelihood method with belief propagation as underlying inference. Since BP allows only computation of approximate likelihood and its gradient, the global training is clearly sub-optimal. In terms of final performance, that may explain why piece-wise training is reasonably good.

In Chapter 7, we provide some further treatment that corrects the 'locality' issue of the pseudo-likelihood and piece-wise training but retains the efficiency.

Some other objective functions are motivated by the accuracy metrics being employed for evaluation of the system, for example, the error rate, F-score and BLEU score (Papineni *et al.*, 2001). Error rates and F-scores for CRFs are studied in (Suzuki *et al.*, 2006). Typically, the error rate is computed on a network-wise basis, but in practice, we are often interested in label-wise prediction in that we count an error for any misclassified label. Label-wise accuracy is considered in (Gross *et al.*, 2007), where the objective function is defined as

$$\mathcal{L} = \sum_{l=1}^{n} \sum_{i \in \mathcal{V}^{(l)}} \delta[x_i^{(l)} = \arg\max_{x_i} \Pr(x_i | z^{(l)})] \tag{3.16}$$

Likewise, the label-wise likelihood is advocated in (Kakade *et al.*, 2002)

$$\mathcal{L} = \sum_{l=1}^{n} \sum_{i \in \mathcal{V}^{(l)}} \log \Pr(x_i^{(l)} | z^{(l)}) \tag{3.17}$$

where $\Pr(x_i|z) = \sum_{x \setminus x_i} \Pr(x|z)$. See (Altun *et al.*, 2003b) for some experimental evaluation of network-wise versus label-wise criteria.

3.5.3 Other Topics

There are other aspects of CRFs that we do not address in this thesis:

- Bayesian learning
- Structure learning
- Semi-supervised learning
- Hybrid directed/undirected models
- Hybrid discrete/continuous variables
- Cost-sensitive learning
- Imbalanced data

Bayesian learning has been studied in (Qi *et al.*, 2005), where the prediction on unseen data z is based on

$$\Pr(x|z, \mathcal{D}) = \int_{\mathbf{w}} \Pr(x|z, \mathbf{w}) \Pr(\mathbf{w}|\mathcal{D}) d\mathbf{w} \tag{3.18}$$

Thus, in the Bayesian CRFs, parameters are not estimated but averaged out. Thus this performs model averaging and helps to combat the overfitting problem. $\Pr(\mathbf{w}|\mathcal{D})$ is approximated by a distribution $Q(\mathbf{w})$ using Expectation Propagation (Minka, 2001a).

Structure learning involves discovering the connectivity of the Markov network from data. Research in this area has a quite long history, starting from (Chow and Liu, 1968), but has mostly concerned Bayesian networks. The past few years have witnessed a substantial interest in structure learning of Markov random fields (Parise and Welling, 2007; Meinshausen and Buhlmann, 2006; Banerjee and Natsoulis, 2006; Wainwright *et al.*, 2006; Schmidt *et al.*, 2007). However, this does not automatically translate into CRFs because the Markov networks can vary from one data instance to another. In that situation, estimating a common structure for all data instances does not apply.

Another aspect is *semi-supervision*, or learning with some labeled instances and some unlabeled ones (e.g. see (Chapelle *et al.*, 2006) for a comprehensive account, and (Zhu, 2006) for a constantly updated survey in the field). This is particularly useful in many situations where collecting unlabeled data is fairly easy (e.g. human faces under Webcams or surveillance cameras) but manual labeling is very expensive (e.g. the person's name). Most of the work has only involved unstructured data but the community has recently been paying considerable attention to structured data (Lafferty *et al.*, 2004; Ando and Zhang, 2005; Altun *et al.*, 2006; Jiao *et al.*, 2006; Brefeld and Scheffer, 2006; Mann and McCallum, 2007).

Originally the CRF was defined as an undirected graphical model. However, in some areas it may be beneficial to incorporate directed components into the model. Theoretically, directed parts are just constrained versions of the undirected counterparts in that the local conditional probability $\Pr(x_i|pa(i))$ plays the role of the clique potential $\psi(x_i, pa(i))$ subject to $\sum_{x_i} \psi(x_i, pa(i)) = 1$. Practically, however, representing the component by a directed subgraph is more intuitive, and the constraints may lead to better numerical stability. Early attempts to build a *hybrid representation* include chain-graphs (Buntine, 1995) and factor-graphs (Frey, 2003).

Most of the work involving CRFs so far has assumed discrete state variables. Much of the probabilistic consistency for the discrete cases can be applied for the *continuous variables*. However, here is no straightforward marginalisation over variables even in continuous cases because it now involves integration, which may not have any analytical form. There has long been investigation into Gaussian random fields in general, but investigation into Gaussian CRFs, in particular, is fairly recent (Tappen *et al.*, 2007).

Cost-sensitive learning (Elkan, 2001) addresses the *consequence* of applying the classifiers in the domain rather than just generic criteria such as maximum likelihood or accuracy. This is important in decision making under uncertainty, i.e. when we want to chose an action that maximises an expected utility, or equivalently to minimise an expected cost:

$$\hat{x} = \arg\min_{x} \sum_{x'} \Pr(x'|z) C(x, x') \tag{3.19}$$

where $C(x, x')$ is the cost of choosing x when the true output is x'. There has been considerable research in this area for unstructured output classifiers, but we are only aware of an attempt in (Sen and Getoor, 2006) for CRFs.

Imbalanced data (e.g. see (Japkowicz, 2002)) refers to situations when the class label distribution is far from uniform, i.e. some labels are much more popular than others. For example, in images, it is often the case that the objects of interest are quite small compared to the background. Prior work, which has mostly addressed the problem in classifiers with unstructured output, can be roughly divided into two groups: those with re-sampling

for correcting the class distribution and those with cost-modification for introducing more weight for rare classes. However, these methods cannot be applied directly for structured classifiers such as CRFs because the interaction between labels is not considered. Only very recently there have been some indirect attempts for this class imbalance in structured output spaces. Sen and Getoor (2006) address cost-sensitive learning, which can be considered as one technique to deal with class imbalance problem. The authors propose a method based on weighted features to bias the cost. Another work addressing the problem in a slightly different angle is (Phan *et al.*, 2005), which aims at discovering rare associations. The work in (Lee *et al.*, 2005) proposes a hybrid method called Support vector random field (SVRF) that combine SVMs as a local classifier and CRFs as a global classifier. The authors claim that the SVRF is insensitive to the class imbalance.

3.6 Closing Remarks

In summary, the Conditional Random Field is a recent major advance in statistical machine learning where the combination between graphical models and machine learning is just about 'right'. It is a proven machinery for many real-world tasks in that it often achieves competitive results against state-of-the-art methods. Chapters 4 and 5 demonstrate its applications further in the area of accent restoration and collaborative filtering, respectively.

So what are the drawbacks of CRFs? There are many, some of which we have pointed out in this chapter. The main computational bottleneck is still the intractability of the underlying graphical models for complex structures. We provide some answers to this problem from the learning perspective in Chapter 7.

Second, the development of CRFs so far has only concentrated on hand-specified features. To be truly effective we should incorporate into CRF the feature discovery capacity from unsupervised learning, as well as feature selection. We address the feature selection problem in Chapter 6.

Third, accurate labels are required so that learning can proceed, and this is too expensive in many domains. There should be some mechanism to at least reduce the need for labeling. One approach that we adopt as the common theme of this thesis is *partially supervised learning*, where only part of the labels in the network are needed. For example, in hand labeling of images the areas around segment boundaries are difficult to handle, so they can be left unlabeled.

Certainly, there is much room for exploring the network structures in the CRFs to make the best out of the framework. In Chapters 8 and 9, we generalise the commonly used

chain-CRFs into hierarchical CRFs in a way that each node in a Markov chain is a Markov chain by itself.

Chapter 4

Statistical Vietnamese Accent Restoration

4.1 Introduction

In this chapter we present a novel application of sequential CRFs to the problem of accent restoration, which is a common task for many languages whose 'accents' are not represented by the standard alphabet set in writing. This chapter is limited to the Vietnamese language.

The Vietnamese writing system utilises a set of Latin alphabets, a small set of new alphabets and a set of five tonal marks. A sentence is a sequence of text units known as *syllables* separated by white spaces. One or more consecutive syllables constitute a word, which is the smallest meaningful text unit. Thus, word boundaries are not predefined by white spaces. Vietnamese accent arises when a syllable contains one or more new alphabets, or when it is combined with none or one of the five tonal marks.

Most keyboards today are designed for English, which means without further help, we can only type the Latin alphabet but the accents are lost. For example, a Vietnamese sentence: *bạn hãy thăm Việt Nam ngay hôm nay* ('please visit Vietnam today') will be written as an accent-less sequence as *ban hay tham Viet Nam ngay hom nay*. It is annoying and error-prone for human readers to decode such messages. The accent-less term *ngay* can easily lead to confusion between the original Vietnamese *ngay* ('now' or 'straight') and the plausible alternative *ngày* ('day'). The current solution for this problem is to use typing-aided software to automatically assign the correct Vietnamese characters when users type in a certain pattern. However, some keying methods such as *Telex*[1] require a great deal of

[1]Telex is a technique that encodes accents using extra characters, for example, by typing fast enough, one converts *ngayf* into *ngày* and *hoom* into *hôm*.

practice to master since it is quite unintuitive from English keyboards. Yet another motivating application is with small footprint devices such as Pocket PCs, where there exists some word processing software with handwriting recognition capability so that users need only to write by hand directly on the screen. Again, most of the handwriting recognition software currently works only for English characters.

It is therefore necessary to restore the accents automatically from English-like texts for reducing the typing burden and for backward transliteration (Knight and Graehl, 1998). The difficulty of this problem is due to the high ambiguity of the accent-less text and it cannot be tackled locally because the context of a syllable is also in the accent-less form. Therefore, methods that look for local patterns between syllables like those in contextual spelling correction (Golding and Roth, 1999) may not work properly.

4.2 Background on Accent Restoration

An accent-less sentence $z = (z_1, z_2, ..., z_T)$ can be considered as a result of forward-transliteration of an original Vietnamese sentence $x = (x_1, x_2, ..., x_T)$

$$z_t = L(x_t) \tag{4.1}$$

where $L(x_t)$ is a deterministic function for removing the accent of the Vietnamese syllable x_t, and T is the length of the sentence z. Thus each x_t would yield a unique z_t.

The restoration is defined as finding the Vietnamese sequence $\hat{x}$ given the accent-less sequence z

$$\begin{aligned} \hat{x}|z &= \arg\max_{x \in \mathsf{V}(z)} \Pr(x|z) = \arg\max_{x \in \mathsf{V}(z)} \Pr(x)\Pr(z|x) \\ &= \arg\max_{x \in \mathsf{V}(z)} \Pr(x) \end{aligned} \tag{4.2}$$

where $\mathsf{V}(z)$ is the space of all Vietnamese sentences whose accent-less form is z, i.e. $\mathsf{V}(z) = \{x|z_t = L(x_t) \forall t \in [1, T]\}$, and $\Pr(z|x) = 1$ since the forward transliteration is deterministic.

There are thus two main problems: (1) how to efficiently and effectively estimate the language model $\Pr(x)$, and (2) how to define the search space $\mathsf{V}(z)$. In subsequent subsections we present some solutions for the first problem. The second issue is quite straightforward since for each accent-less syllable z_t we only need to build a *proposal set* of Vietnamese syllables $\mathsf{V}(z_t) = \{x_t\}$. From the corpus, we add a Vietnamese syllable x_t to $\mathsf{V}(z_t)$ if $L(x_t) = z_t$ and x_t is not yet in $\mathsf{V}(z_t)$. The good news is that the number of Vietnamese syllables is quite small (of the order of 10^4), and the number of accent-less syllables is even

smaller (of the order of 10^3), and each accent-less syllable corresponds to 1-24 Vietnamese syllables.

4.2.1 N-gram Models

Clearly, since we do not have enough resources to enumerate all possible sentences in Vietnamese, approximation must be made. Options include the local n-gram models (Manning and Schütze, 1999, ch. 6) and others with global constraints. The simplest method is the unigram model $\Pr_1$ that assumes the complete factorisation:

$$\Pr_1(x) \approx \prod_{t\in[1,T]} \Pr(x_t) \tag{4.3}$$

where each unigram is a syllable. However, syllables by themselves do not generally have meaning. They make sense only if they belong to words. Thus, modelling the relation between unigrams is more important. The bigram model $\Pr_2$ and the trigram model $\Pr_3$ capture this better:

$$\Pr_2(x) \approx \Pr(x_1) \prod_{t\in[2,T]} \Pr(x_t|x_{t-1}) \tag{4.4}$$

$$\Pr_3(x) \approx \Pr(x_1, x_2) \prod_{t\in[3,T]} \Pr(x_t|x_{t-1}, x_{t-2}) \tag{4.5}$$

The bigram model is actually a special case of the first-order HMM (see Section 2.4.3.1 for description and Figure 2.4 for illustration) where the emission probability is one ($\Pr(z_t|x_t) = 1$). Likewise, the trigram model is a second-order HMM.

The n-gram distributions can be estimated by simply counting the number of occurrences as usual. Due to data sparseness we need to smooth over the distribution to assign a non-zero probability to unseen n-grams. In this study, we employ simple Laplace smoothing, which is given as

$$\Pr(x_{t-n+1}, ..., x_t) = \frac{C(x_{t-n+1}, ..., x_t) + 1}{N_n + |V_n|} \tag{4.6}$$

where $C(x_{t-n+1}, ..., x_t)$ is the number of occurrences of the n-grams, N_n is the total occurrences of all n-grams, $|V_n|$ is the estimated vocabulary size of the n-grams of the language. Thus an unseen n-gram will be given the uniform probability of $1/|V_n|$.

Given these three simple models we can proceed to estimate the corresponding conditional probabilities from the data. On the other hand, the bigrams and trigrams model the language better, but reliably estimating the bigrams and trigrams would require a very large

data set. One effective strategy is to join the three schemes together in a Product-of-Expert (PoE) approach (Hinton, 2002):

$$\Pr(x) = \frac{1}{Z}\Pr_1(x)^{w_1}\Pr_2(x)^{w_2}\Pr_3(x)^{w_3} \tag{4.7}$$

where $w_1, w_2, w_3 \geq 0$ are the weights of the component models, and Z is the normalisation constant. The beauty of this approach is that the computational complexity is the same as its components, whilst we can adjust the contribution of the components by tuning the extra parameters w_1, w_2, w_3. The distribution by the PoE is often more peaked than the component parts. For example, if the three component models agree on a particular sentence x, the PoE would yield a high probability. Another important property is that if the trigram model is not discriminative about a particular sentence, while the bigram and the unigram are, the resulting PoE still assigns a reasonable probability to that sentence.

4.2.2 Related Work

Previous research has addressed the accent restoration problem for other languages such as Spanish (Yarowsky, 1994), the Latinised Chinese called Pinyin (Wan and Verspoor, 1998). We can consider this problem as a form of backward transliteration (Knight and Graehl, 1998; Li *et al.*, 2004) which aims to recover the original form of transliterated words. The work in this area is still limited, and seems to focus on a narrow set of proper names and technical terms transliterated between different languages such as Japanese/Chinese and English. An example is to recover the original English words (such as *computer, Washington*) from the Japanese transliteration (Knight and Graehl, 1998). Our work, in contrast, is to recover the whole original Vietnamese sentences from the accent-less forms.

At a larger scale, it is a special case of machine translation (Brown *et al.*, 1993) converting accent-less Vietnamese into correct Vietnamese. Fortunately, the accent restoration problem is expected to be far easier than the translation, because the mapping is word-for-word and no alignments are needed. Moreover, it falls into the lexical ambiguity category, where each accent-less syllable can correspond to many possible Vietnamese alternatives. From this perspective the problem can also be cast as a *word sense disambiguation* problem (Ide and Veronis, 1998), where each alternative roughly plays the role of a 'sense'.

4.3 Modelling using Conditional Random Fields

The n-gram models and their PoE ensemble described in the previous subsection do not exploit the fact that we do not need to model the whole language space of all possible

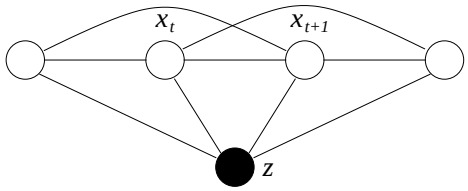


Figure 4.1: A second-order CRF.

sentences (see Equation 4.2). Rather, for each accent-less sentence z, we need only to pay attention to the restricted subspace $\mathsf{V}(z)$. More specifically, the size of the whole language space is the number of combinations of words in the vocabulary, the sentence length, and the permutation of word order in the sentence. On the contrary, the subspace $\mathsf{V}(z)$ is limited to the specific sequence length and word order of z, and only a subset of vocabulary, because the maximum number of accent alternatives for each accent-less unigram is 24. Furthermore, the weight vector in the PoE (Equation 4.7) should be learned automatically from the data.

The conditional nature of the CRFs allows us to directly model the distribution $\Pr(x|z)$, and therefore deals only with the subspace $\{x|x \in \mathsf{V}(z)\}$

$$\Pr(x|z) = \frac{1}{Z(z)} \prod_c \psi_c(x_c, z)$$

where $Z(z) = \sum_{x \in \mathsf{V}(z)} \prod_c \psi_c(x_c, z)$. In this study, we employ CRFs with second-order Markov chains (see Section 2.4.4 for description and Figure 4.1 for illustration). Thus the clique potentials are of the form $\psi_t(x_t, x_{t+1}, x_{t+2}, z)$, where $x_t \in \mathsf{V}(z_t)$, $t \in [1, T-2]$. More specifically, we have

$$\psi_t(x_t, x_{t+1}, x_{t+2}, z) = \exp\{w_{k_1} f_{k_1}(x_t, z) + w_{k_2} f_{k_2}(x_t, x_{t+1}, z) + w_{k_3} f_{k_3}(x_t, x_{t+1}, x_{t+2}, z)\}$$

where $f_{k_1}(x_t, z)$, $f_{k_2}(x_t, x_{t+1}, z)$ and $f_{k_3}(x_t, x_{t+1}, x_{t+2}, z)$ are binary unigram, bigram and trigram features, respectively. The features are given as

$$\begin{aligned}
f_{k_1}(x_t, z) &= \delta[C(x_t) > n_1]\delta[x_t \in \mathsf{V}(z_t)] \\
f_{k_2}(x_t, x_{t+1}, z) &= \delta[C(x_t, x_{t+1}) > n_2]\delta[x_t \in \mathsf{V}(z_t)]\delta[x_{t+1} \in \mathsf{V}(z_{t+1})] \\
f_{k_3}(x_t, x_{t+1}, x_{t+2}, z) &= \delta[C(x_t, x_{t+1}, x_{t+2}) > n_3]\delta[x_t \in \mathsf{V}(z_t)] \times \\
&\quad \times\delta[x_{t+1} \in \mathsf{V}(z_{t+1})]\delta[x_{t+2} \in \mathsf{V}(z_{t+2})]
\end{aligned}$$

where $\{n_1, n_2, n_3\} \geq 0$ are thresholds for the number of occurrences $C(.)$, and $\delta[.]$ is the indicator function. Thus, this feature design implements the simple frequency cut-off feature selection method (see Section 3.3).

The main difference of this CRF compared to the majority of previous CRF applications

is that the label set at each node in the Markov chain is constrained, depending on the accent-less syllable. The problem has been partly addressed in the name of *constrained inference* (Kristjannson *et al.*, 2004) in the context of interactive labeling. In (Kristjannson *et al.*, 2004), at decoding time, when the user of the system gives a label at a given node of the sequence, the system responds by limiting the label set of that node to this given label, and thus improves the performance. The constrained inference is not used at training time. Our work can be considered as an extension to (Kristjannson *et al.*, 2004) by applying constrained inference to an arbitrary subset of labels in both training and decoding. Most other works, especially those in the field of Information Extraction (Cowie and Lehnert, 1996; Lafferty *et al.*, 2001), assume a fixed set of labels is used for all nodes in the sequence.

For parameter estimation we are more interested in the online-setting in which parameters are updated after seeing a training instance (Section 3.5.1.1). This is important because in interactive applications the system may output several possible alternatives and let the user select the one that is the most appropriate to the user's context. Since the style used by each person differs, an accent-less sentence can possibly have many plausible Vietnamese alternatives, so letting the user correct the restoration will allow the system to gradually adjust the parameters to suit the individual styles. In particular, we use both the stochastic gradient and perceptron methods for parameter estimation (detailed in Section 3.5.1.1).

4.4 Experiments

4.4.1 Corpus and Pre-processing

Training size	426×10^3 sentences
Testing size	28×10^3 sentences
Accent-less vocabulary size	1.4×10^3 syllables
Number of accents	24(max), 4(average)
Vietnamese unigram set	7×10^3 unigrams
Vietnamese bigram set	842×10^3 bigrams
Vietnamese trigram set	1264×10^3 trigrams

Table 4.1: Data statistics.

Data is collected from online news articles and split into a training set of 426K sentences and a test set of 28K sentences. The corpus contains a wide range of subjects[2]. The writing styles vary as the material comes from a dozen news sources.

For testing, we first perform the forward-transliteration to obtain the accent-less form and

[2]They are: politics, social issues, IT, family & life-style, education, science, economics, legal issues, health, world, sports, arts & culture, and personal opinions.

then decode back the Vietnamese form. The decoded text is compared against the original. Here we do not distinguish between upper and lower cases. Learning and comparison are done in lower-case.

Since news articles often contain foreign words, acronyms and non-alphabets, we restore only the accents of those in a fixed accent-less vocabulary. To obtain the accent-less vocabulary we remove the accents of the syllables in a Vietnamese dictionary. The accent-less vocabulary has 1.4K syllables, which is much smaller than the typical Vietnamese set of syllables (around 10K). Through the forward-transliteration we obtain the proposal sets, each of which is a set of Vietnamese syllables corresponding to a particular accent-less syllable. The number of Vietnamese syllables that share the same accent-less form ranges from 1 to 24, and is about 4 on average.

The performance is measured within the accent-less vocabulary. The word accuracy is the portion of restored syllables that are correct. A restored sentence is considered correct if all of its restored syllables (within the accent-less vocabulary) are correct.

From the training data we estimate the unigram, bigram and trigram distributions. There are 7K unique unigrams whose accent-less form is in the accent-less dictionary. We count a bigram if it occurs and one of the component unigrams is in the unigram list. We obtain a bigram list of size 842K. If we remove those bigrams that happen only once in the corpus, the list is reduced to 465K. Similarly, we count a trigram if it occurs and one of the component unigrams is in the unigram list. This gives 3137K unique trigrams. Removing the trigrams with a single occurrence, we obtain a trigram list of size 1264K.

We then apply the Laplace smoothing (see Equation 4.6), where vocabulary sizes for unigrams, bigrams and trigrams are estimated to be $10^4, 7 \times 10^8$, and 7×10^{13}, respectively. The unigram vocabulary estimate is from the 7K unigrams we obtain from the corpus. To estimate the bigram vocabulary size, recall that the we count a bigram if one of its component unigram is from the 7K unigram list, and the other unigram can be anything. We estimate that there are about 10^5 unique unigrams outside the list, and this number is multiplied with 7K to yield 7×10^8. Multiplying this result further by 10^5, we obtain the estimate of the trigram vocabulary size. The main statistics of the data is summarised in Table 4.1.

4.4.2 Results

For the Product-of-Experts, we set the weight manually

- first-order PoE (unigram & bigram): $w_1 = 1; w_2 = 1$,
- second-order PoE (unigram & bigram & trigram): $w_1 = 2; w_2 = 2; w_3 = 1$

First, we perform a set of experiments with first-order models, which include the bigrams, the first-order PoE and the first-order CRF. The baseline is the unigram model. One problem with the bigram model is how it handles unseen bigrams. As the majority of Vietnamese words used in writing are bigrams, this means that a bigram is not simply a random combination of two unigrams. Therefore, most random combinations of two unigrams should have an extremely low probability, or at least their probabilities are not equal. The popular Laplace smoothing, on the other hand, tries to assign every unseen bigram an equally small probability under the assumption of prior uniform distribution. This is unrealistic in Vietnamese. To deal with this we assign unseen bigrams with a very low probability, which is practically zero, so that any sequence with unseen bigrams is severely penalised. Although this is not optimal since some plausible bigrams are cut off, it seems to solve the problem. Luckily, the PoE does not have this problem, probably because the unseen bigrams will be compensated by the component unigrams.

In the first-order CRF model, we use only the bigram features. For training, we run the Perceptron for 20 iterations and the Stochastic Gradient for 15 iterations over the whole training data set. This is obviously much slower than the bigram and first-order PoE models since we need to estimate the bigram distribution using only one run through the data. However, such a cost can be well justified by the higher performance of the CRF model compared with the bigram and the PoE as shown in Table 4.2. In this study, we use 465K bigram features for all the first-order models. The simple unigram model works poorly as expected, and its performance is unacceptable for practical use. The PoE, which is just a product of the unigram and the bigram models, works surprisingly well with significant improvement over the bigram model. The CRF model is the winner despite the fact that it uses no more domain information than for the PoE.

Model	**Word accuracy**	**Sentence accuracy**
Baseline	71.83	6.31
Bigram	90.68	30.87
1st-order PoE	92.42	37.54
1st-order CRF (Perceptron)	93.16	38.40
1st-order CRF (Stochastic Gradient)	93.68	41.95
2st-order PoE	93.45	42.72
2st-order CRF (Perceptron)	93.51	41.77
2st-order CRF (Stochastic Gradient)	94.26	44.83

Table 4.2: Word and sentence accuracy (%) of first/second-order models compared with the baseline unigram model.

The second set of experiments is performed with second-order models. For the moment only the second-order PoE is used with 7K unigram features, 465K bigram features and 1264K trigram features. We run the Perceptron for 10 iterations and the Stochastic Gradient for 5 iterations. The last rows in Table 4.2 show the accuracy of the PoE and the CRF. The

trigram model is not used since it performs fairly poorly, possibly due to the limited corpus. Interestingly, the PoE can compensate for the poor estimate of the trigrams by using the unigram and bigram components.

Overall for both experiment sets, the CRF trained by Stochastic Gradient performs best. We observe that the Perceptron minimises the error over *training* data quickly as it is specifically designed for this task. However, this leads to the overfitting problem as the Perceptron does not have any regularisation mechanism. The Stochastic Gradient training, on the other hand, can control the overfitting through the Gaussian penalty term (Equation 3.6).

4.5 Closing Remarks

In this chapter we have applied the CRFs to the Vietnamese accent restoration problem. Experimental results so far indicate that the approach is suitable and achieve good results in the news domain.

In regard to the accent restoration problem, there are several aspects that need further investigation. One aspect of Vietnamese is that the white spaces are not indicators of word boundaries. Most words are composed of two syllables, especially words used in written texts. It is therefore important to incorporate the capability of word segmentation in the language models. From our experience with the popular bigrams, we believe that we need a better smoothing scheme for the Vietnamese language model, which is inherently different from English.

Furthermore, there are different genres and writing styles, and it is likely that a sequence of accent-less syllables can correspond to several plausible Vietnamese sequences, depending on the context of use. A very challenging domain is creative writing, especially poetry, where authors make deliberate use of word reordering and repetition to achieve stylistic and artistic effects. The most challenging form is perhaps spoken language, especially in online environments such as chatting and SMS, where the use of language is largely distorted due to the constraints of writing space and personal interests.

The current study is limited to the online news domain, and clearly the results are biased towards these reporting styles. Thus, one possible direction is to address varying styles by training the algorithms on more data to obtain better coverage. Another option is to detect the style through independent methods or through clustering.

An issue not addressed in this work is the analysis of syntax and semantics. It is likely that the analysis will provide more consistent results. Through the CRF framework, for example, it is possible to incorporate a richer set of features to address the correlation

between sentences in the same paragraph. Also, we can create different models to address different linguistic aspects and then combine them together in the PoE approach.

In the next chapter we will demonstrate another application of CRF in the field of movie recommendation. This utilises a different network structure under a setting known as Relational Markov Networks, where there is only a single (complex and large-scale) Markov network built on top of a relational database.

Chapter 5

Relational Markov Networks for Hybrid Recommendation

5.1 Introduction

In previous chapters we have presented the Conditional Random Field and a real-world application for Vietnamese accent restoration. A common practice when using CRFs is to assume that the structured training instances are independently generated. Typically, we generate a Markov network for each instance and ignore dependencies between instances. However, this practice may not be appropriate for relational domains in which all the entities are related. Such relations often do not allow partitioning the data into independent instances.

To be more concrete, let us study a particular relational domain called automatic recommendation. In this domain, we have a set of users and a set of items. A user is anyone who purchases products (for example, books) or subscribes to services (for example, online movies). An item refers to products or services that users use. Each user typically expresses their preference over a subset of items they have been using. Based on the preferences, the recommender system will predict the next set of preferred items for a given user and ratings on how much the user will like these items.

Recall that a relational domain can be represented by a *schema* that defines entity types, entity attributes and relations between entity types. In recommender systems, there are three entity types: User, Item and Rating (Figure 5.1). The User may have multiple attributes such as *Age* and *Sex*. The Item's attributes may include *Category*, and *Date*. Rating has an ordinal *Score* attribute, which is typically a small integer from the set $\{1, 2, .., |S|\}$. Typically, in each entity of types User and Item, there is also an *ID* attribute for indexing. Rela-

tions between entities are realised by reference attributes pointing to other entities' identity. For example, the attribute Rating.*ByUser* points to the user ID, and Rating.*OnItem* points to the item ID.

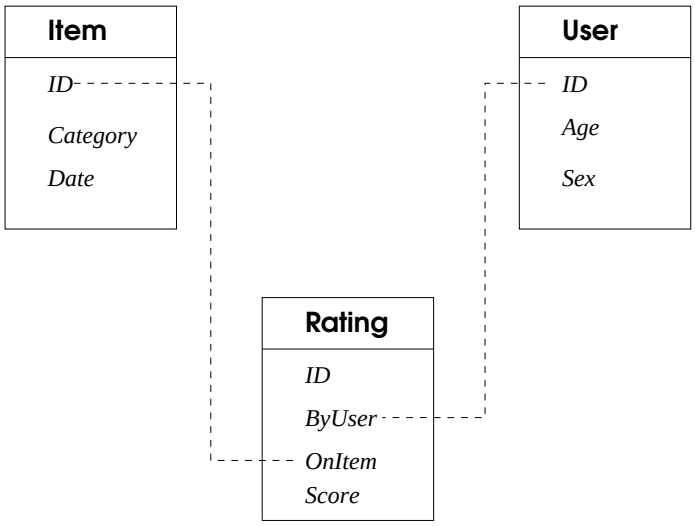


Figure 5.1: Schema for recommender systems.

In the relational database each entity type has multiple instances and their attributes are filled with specific values. These constitute an *instantiation* $\mathcal{I}$ of the schema. For example, in the MovieLens dataset, which we will use for experiments in Section 5.3.6, there are 943 instances of the User entity types, 1682 of Item and 100,000 of Rating. The instantiation of reference attributes defines an instantiation graph that links all the entity instances together.

In our study we are interested in probabilistic modelling and prediction of all attributes Rating.*Score* in the instantiation. That is, we want to define a Markov network over all the ratings. We can manually construct a CRF as in the previous chapter but the model representation can be very expensive and is only applicable for a particular instantiation of the schema. For example, in recommender systems, the size of a rating database can be hundreds of millions of items[1]. In contrast, the schema, as we have seen, can be quite simple and is generic for any instantiation.

In this chapter we will present a method called Relational Markov Network (RMN) (Taskar *et al.*, 2002) that deals with this problem. The RMN exploits the compactness of the relational schema to specify how the network structure of the CRF is constructed. Then in Section 5.3, we propose the use of RMN for preference modelling and prediction in recommender systems, and call the resulting model a *Preference Network* (PN). Differing from previous approaches to recommendation, the PN is a joint model of all preferences and it takes into account a variety of information, including relations between users and between

[1]The Netflix movie rating data has 100 millions entries [http://www.netflixprize.com].

products, user demographics and product attributes. The model then serves as a probabilistic database that supports various queries such as the most probable ratings and top-N new items for a given user. We evaluate the PN on the movie rating database in Section 5.3.6.

5.2 Relational Markov Networks

Relational Markov Networks exploit the relational structure by providing a query language that helps to build the Markov network structure at a *template* level. First, for each schema, the RMNs define a set $\mathbf{C}$ of *relational clique templates*, and each clique template in the set $C \in \mathbf{C}$ includes a subset of entities, a subset of relations between these entities, and a subset of attributes associated with the selected entities. A clique template can be a set of rules (or SQL queries) that tie entities together. This does not depend on specific schema instantiation. For example, in our movie rating example, the clique template can be the same, regardless of the movie data sources being used.

The instantiated clique template $C(\mathcal{I})$ (i.e. C applied to the specific instantiation $\mathcal{I}$) is a set of cliques $\{c \in C(\mathcal{I})\}$. Thus, the set of all cliques of all templates $\{c|c \in C(\mathcal{I}), C \in \mathbf{C}\}$ specifies the graph structure of the Markov network.

What remains is the potential function associated with each clique template C. Let x be the set of attributes in the instantiation that we want to treat as hidden state variables of the resulting Markov network. Let z be the rest of the attributes. The clique potential that realises the template C, and specific clique c defined by the instantiation $C(\mathcal{I})$, therefore, has the form $\psi_C(x_c, z_c)$, where x_c is the subset of hidden state variables in the clique c and z_c is the set of content and reference attributes.

A Relational Markov Network defines the following conditional distribution

$$\Pr(x|z) = \frac{1}{Z(z)} \prod_{C \in \mathbf{C}} \prod_{c \in C(\mathcal{I})} \psi_C(x_c, z_c) \tag{5.1}$$

Using log-linear parameterisation, we write $\psi_C(x_c, z_c) = \exp(\mathbf{w}_C^\top \mathbf{f}_C(x_c, z_c))$.

Thus, given an instantiation $\mathcal{I}$, the RMN produces an *unrolled* CRF. Note that for a particular domain the same set of clique templates can be used for different instantiations. Given a schema and a relational database, the RMN provides a set of queries to construct the CRF. Thus, the RMN is a compact representation of the CRF. The compactness is important for computational reasons. For example, in our study of movie ratings a standard CRF will have a densely connected network of 100,000 nodes. Storing the network structure and all associated potentials will be too expensive. Rather, we should only store the tables of

ratings, users and items and then construct the network fragments and associated potentials on-the-fly based on the clique template specifications.

5.3 Preference Networks for Recommender Systems

5.3.1 Background on Recommender Systems

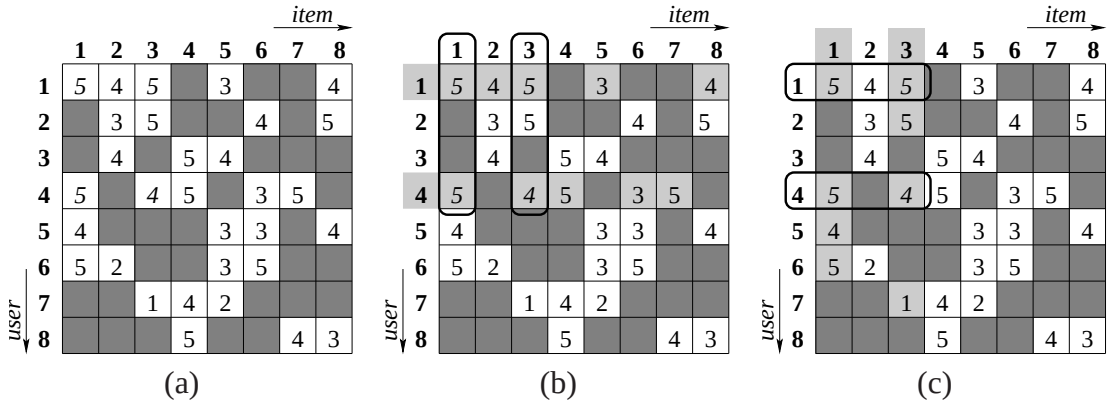


Figure 5.2: (a) Preference matrix; (b) Correlation between user 1 and user 4 are based on common items 1 and 3 co-rated by the two users; and (c) Correlation between item 1 and 3 are based on common users 1 and 4 co-rating the two items.

Recommenders are automated tools to deliver selective information that matches personal preferences. These tools have become increasingly important to help users find what they need from massive amount of media, data and services currently flooding the Internet. Commercial systems currently operating include Amazon[2], Netflix[3] and Google News[4].

Recommender systems make recommendations based on the content of products and services (*content-based*), or based on collective preferences of the crowd (*collaborative filtering*), or both (*hybrid methods*). Typically, content-based methods work by matching product attributes to user-profiles using classification techniques. Collaborative filtering, on the other hand, relies on similarity between users (Resnick *et al.*, 1994) or products (Sarwar *et al.*, 2001) and preferences the user has expressed. Since content and preferences are complementary, hybrid methods often work best when both types of information are available (Balabanović and Shoham, 1997; Basu *et al.*, 1998; Pazzani, 1999; Basilico and Hofmann, 2004).

In general, the recommendation can be stated as follows: given a set of M users, and L items that the users can select from, let $\mathbf{M} = \{x_{ui}\}$ denote the *preference matrix* where $u \in$

[2]http://www.amazon.com
[3]http://www.netfix.com
[4]http://news.google.com

$\{1, 2, ..., M\}$ is the user index, $i \in [1, 2, ..., L]$ is the item index, and x_{ui} is the preference or rating of user u over item i (Figure 5.2a). In many applications a user usually rates only a small number of items and this makes the preference matrix $\mathbf{M}$ extremely sparse. For example, in the MovieLens dataset, only about 6.3% entries in the $\mathbf{M}$ matrix are filled. A common problem in recommender systems is to use all previous preferences and try to estimate the rest of the entries in $\mathbf{M}$ (the *preference prediction problem*). In practice, a preference x_{ui} is expressed either explicitly when a user gives a numerical rating, or implicitly when she chooses to read a particular news article.

Another frequent task is to return a set of N items that the user has not expressed preference for, but may prefer if the items are presented to her. This is known as the *top-N recommendation problem* (Deshpande and Karypis, 2004). Typically the task involves several steps. In the first step a candidate set of promising items is identified. In the second step, these candidates are ranked in decreasing order of relevance and then the top N items are presented to the user. The measure of relevance depends on context, for example, it may be the probability that the user will like the item, or the expected benefit that the user will gain for choosing the item.

Most popular works to date address these two tasks by using some *similarity* measure between related users, or between items. Users are related in the way that they co-rate some common items. One of the most common similarity measure $s(u, v)$ between user u and user v is Pearson's correlation

$$s(u,v) = \frac{\sum_{i \in I(u,v)} (x_{ui} - \bar{x}_u)(x_{vi} - \bar{x}_v)}{\left[\sum_{i \in I(u,v)} (x_{ui} - \bar{x}_u)^2\right]^{\frac{1}{2}} \left[\sum_{j \in I(u,v)} (x_{vj} - \bar{x}_v)^2\right]^{\frac{1}{2}}} \tag{5.2}$$

where $I(u, v)$ is the set of all items co-rated by users u and v, and $\bar{x}_u$ is the average rating by user u. Figure 5.2b illustrates the case where $u = 1, v = 4$ and $I(u, v) = \{1, 3\}$. Prediction of preference of an unseen item for a given user can be computed as (Resnick *et al.*, 1994)

$$x_{ui} = \bar{x}_u + \frac{\sum_{v \in U(i)} s(u,v)(x_{ui} - \bar{x}_v)}{\sum_{v \in U(i)} |s(u,v)|}$$

where $U(i)$ is the set of all users who rate item i. Since in the preference matrix, users and items play equal roles, similarity between items (Sarwar *et al.*, 2001) can also be used for prediction

$$s(i,j) = \frac{\sum_{u \in U(i,j)} (x_{ui} - \bar{x}_u)(x_{uj} - \bar{x}_u)}{\left[\sum_{u \in U(i,j)} (x_{ui} - \bar{x}_u)^2\right]^{\frac{1}{2}} \left[\sum_{v \in U(i,j)} (x_{vj} - \bar{x}_v)^2\right]^{\frac{1}{2}}} \tag{5.3}$$

where $U(i, j)$ is the set of all users who co-rate both items i and j. Figure 5.2c illustrates

the case where $i = 1, j = 3$ and $U(i, j) = \{1, 4\}$. The prediction rule in this case is analogous to Equation 5.3 where the roles of user u and item i are swapped.

Further constraints are often in place, for example, in user-based methods, only K most similar users $v \in U(i)$ with respect to user u are selected. Typically, K ranges from 20 to 100. Another practice is to choose only positive, correlated users in the neighbourhood of user u.

Another approach is based on non-negative matrix factorisation (NNMF) (Lee and Seung, 1999; Rennie and Srebro, 2005; Zhang *et al.*, 2006b). The idea is that the preference prediction problem is to fill the empty entries in the preference matrix $\mathbf{M}$. We approximate $\mathbf{M}$ as follows

$$\mathbf{M} \approx \mathbf{A} = \mathbf{BC} \tag{5.4}$$

where $\mathbf{B} = \{b_{uh}\}$ is an $M \times H$ non-negative matrix, and $\mathbf{C} = \{c_{hi}\}$ is a $H \times L$ non-negative matrix, and H is often much smaller than M and L. In essence, we seek a lower dimension representation of $\mathbf{M}$, in that $\mathbf{B}$ is roughly a low rank basis, and $\mathbf{C}$ is roughly the projection of $\mathbf{M}$ on $\mathbf{B}$. A common method to determine $\mathbf{B}$ and $\mathbf{C}$ is to minimise the following function

$$\sum_{u,i|x_{ui}>0} (x_{ui} - b_{uh}c_{hi})^2 + \mu(\sum_{u,h} b_{uh}^2 + \sum_{h,i} c_{ui}^2) \tag{5.5}$$

where $\mu > 0$ is a regularisation factor. Note that in the first term, we only sum over observed entries in the preference matrix. The optimisation can be done via methods such as gradient descent. Once we find an approximate factorisation the empty entries in $\mathbf{M}$ can be filled by corresponding entries in $\mathbf{A}$.

Probabilistic approaches to the recommendation problem attempt to construct models that explain user ratings (Breese *et al.*, 1998; Heckerman *et al.*, 2001; Hofmann, 2004; Marlin, 2004). Existing work has employed directed graphical models such as Bayesian networks (Breese *et al.*, 1998) and dependency networks (Heckerman *et al.*, 2001), and undirected models such as restricted Boltzmann machines (Salakhutdinov *et al.*, 2007). Many of other probabilistic works perform clustering. This is an important technique for reducing the dimensionality and noise, dealing with data sparsity and more significantly, discovering latent structures. Here, the latent structures are either communities of users with similar tastes or categories of items with similar features. Some representative techniques are mixture models, probabilistic latent semantic analysis (pLSA) (Hofmann, 2004) and latent Dirichlet allocation (LDA) (Marlin, 2004). These methods try to uncover some hidden process which is assumed to generate items, users and ratings. Such a generative process, on one hand, is intuitive and expressive in the way that it expresses prior belief, but on the

other hand may not reliably 'explain' the data well.

Another important class of methods are from machine learning. These methods map the recommendation into a classification problem (Billsus and Pazzani, 1998; Basu *et al.*, 1998; Zhang and Iyengar, 2002; Basilico and Hofmann, 2004; Zitnick and Kanade, 2004). One of the key observations made is that there is some similarity between text classification and rating prediction (Zhang and Iyengar, 2002). There are two ways to convert collaborative filtering into a classification problem (Billsus and Pazzani, 1998). The first is to build a model for each item, and ratings by different users are treated as training instances. The other builds a model for each user, and ratings on different items by this user are considered as training instances (Breese *et al.*, 1998). These treatments, however, are complementary, and there should therefore be a better way to systematically unify them (Basu *et al.*, 1998; Basilico and Hofmann, 2004). That is, the pairs (user,item) are now treated as independent training instances. However, the assumption that training instances are independently generated does not hold in collaborative filtering. Rather all the ratings are interconnected directly or indirectly through common users and items.

5.3.2 Preference Networks

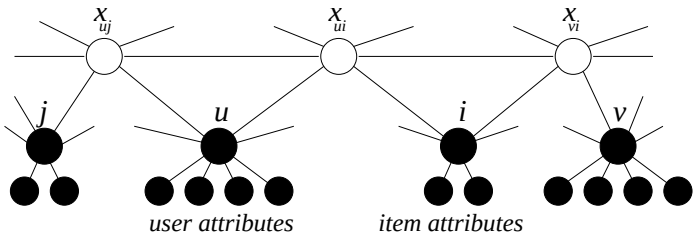


Figure 5.3: A fragment of Preference Networks.

The main goal is to apply the RMN framework for modelling and prediction of ratings. To that end we build a single Markov network for all ratings in the database. Since the ratings reflect user's preferences we call the resulting Markov network a *Preference Network* (PN). We would like the PN to integrate varieties of domain knowledge such as prior rich information of user demographics, item content attributes, correlation information between closed users (Resnick *et al.*, 1994) and between related items (Sarwar *et al.*, 2001). Given the schema (Figure 5.1), we define the following clique templates:

1. *User Identity*: this specifies the association between the two attributes User.*ID* and Rating.*Score*, and captures how likely a user gives a particular scoring.

2. *Item Identity*: this specifies the association between the two attributes Item.*ID* and Rating.*Score*, and captures how likely an item is given a particular scoring.

3. *User Type*: this specifies the association between the item's identity attribute Item.*ID* and the user's attributes such as User.*Age*. It carries the likelihood that an item will be preferred by a particular class of users (e.g. teenagers).

4. *Item Category*: this specifies the association between the user's identity attribute User.*ID* and the item's content attributes such as Item.*Category*. For example, this captures how likely a user would buy a particular type of product (e.g. a VIP customer will be likely to use first-class services). Since a product may fall into multiple categories, this template must be able to take aggregation of categories into account.

5. *User Correlation*: this captures the 'hidden relations' between any two users if they share common interest in a particular item. The common belief is that if two users are both interested in some items, their tastes are similar, and that ratings by one user are indicative of the other's ratings (Resnick *et al.*, 1994). The SQL query is:
   SELECT rating1.*Score*, rating2.*Score*
   FROM Rating rating1, Rating rating2, User user1, User user2
   WHERE rating1.*OnItem* = rating2.*OnItem* and rating1.*ByUser* = user1.*ID* and rating2.*ByUser* = user2.*ID*
   Figure 5.2b depicts two cliques returned by this query for user 1 and user 4.

6. *Item Correlation*: similar to the case of User Correlation, this template captures the 'hidden relations' between any two items if they are co-rated by the same user. The idea is that if two items are co-rated by some users, then qualities of these items are similar and that scores given to one item are informative in predicting the score of the other item. The SQL query is:
   SELECT rating1.*Score*, rating2.*Score*
   FROM Rating rating1, Rating rating2, Item item1, Item item2
   WHERE rating1.*ByUser* = rating2.*ByUser* and rating1.*OnItem* = item1.*ID* and rating2.*OnItem* = item2.*ID*
   Figure 5.2c depicts two cliques returned by this query for item 1 and item 3.

Application of these six clique templates to the rating database results in an unrolled Markov network, or Preference Network. Figure 5.3 depicts a fragment of the PN. Rating.*Score*s are treated as hidden state variables. Denote by x_{ui} the state variable associated with user u and item i. The pair (u, i) is then the index of the network's vertex. There is an edge between any two ratings by the same user u, and an edge between two ratings on the same item i. As a result, a vertex of x_{ui} will be connected with $U(i) + I(u) - 2$ other vertices. Thus, for each user, there is a fully connected sub-network of all ratings that have been made, plus connections to ratings by other users on these items. Likewise, for each item, there is a fully connected sub-network of all ratings by different users on this item, plus connections to ratings on other items by these users. The resulting network $\mathcal{G}$ is typically densely connected because $U(i)$ can be potentially very large (e.g. 10^6).

Two-step modelling

Ideally, we would be interested in modelling all possible ratings, including those which have not yet been in the database. In other words, the ideal model should cover all the empty cells in the preference matrix (Figure 5.2). However, in practice, the matrix size is extremely large (e.g., $10^6 \times 10^6$), making computation intractable. In addition, such modelling is unnecessary because a user is often interested in a moderate number of items. As a result, we adopt a two-step strategy:

- During the learning phase, we limit to model the joint distribution over existing ratings.
- During the prediction/recommendation phase we extend the model to incorporate to-be-predicted entries without changing parameters.

5.3.3 Feature Design and Selection

Corresponding to the six clique templates defined in Section 5.3.2 are feature functions that realise the templates in real data. They are *user-identity, item-identity, user-type, item-category, user-correlation*, and *item-correlation*.

5.3.3.1 Feature design

User identity

Assume that the ratings are integer, ranging from 1 to $|S|$. The average rating $\bar{x}_u$ by user u over items rated roughly indicates the user-specific scale of the rating because the same rating of 4 may mean 'OK' for a regular user, but may mean 'excellent' for a critic. The feature that encodes such belief is given as

$$f_u(x_{ui}, u) = g(|x_{ui} - \bar{x}_u|) \tag{5.6}$$

where $g(y) = 1 - y/(|S| - 1)$ is used to ensure that the feature values is normalised to $[0, 1]$ and $|S|$ is the rating scale.

Item identity

Similarly, we know from the database the average rating $\bar{x}_i$ of item i which roughly indicates the general quality of the item with respect to those who have rated it. We have the following feature

$$f_i(x_{ui}, i) = g(|x_{ui} - \bar{x}_i|), \tag{5.7}$$

User types

Denote by $\mathbf{a}_u$ the vector of attributes for user u. We are interested in seeing the classes of users who like a particular item i through the following mapping

$$\mathbf{f}_i(x_{ui}) = \mathbf{a}_u g(|x_{ui} - \bar{x}_i|) \tag{5.8}$$

Item categories

Denote by $\mathbf{a}_i$ the vector of attributes of item i. Mapping from item attributes to user preference can be carried out through the following feature

$$\mathbf{f}_u(x_{ui}) = \mathbf{a}_i g(|x_{ui} - \bar{x}_u|) \tag{5.9}$$

User correlation

The user correlation features capture the idea that if two users rate the same item then the ratings, after being offset by user's mean rating, should be similar

$$f_{u,v}(x_{ui}, x_{vi}) = g(|(x_{ui} - \bar{x}_u) - (x_{vi} - \bar{x}_v)|) \tag{5.10}$$

Item correlation

The item correlation features capture the fact that if a user rates two items, then after offsetting the goodness of each item, the ratings should be similar;

$$f_{i,j}(x_{ui}, x_{uj}) = g(|(x_{ui} - \bar{x}_i) - (x_{uj} - \bar{x}_j)|) \tag{5.11}$$

5.3.3.2 Feature selection

We employ the filtering approach (Section 3.3) for selecting correlation features. Specifically, we only select those correlation features if the correlations are beyond a certain threshold. Between users, the Pearson's correlation in Equation 5.2 is used. Likewise, the similarity measure in Equation 5.3 is employed as correlation between items. For simplicity, the threshold is set to 0.

It should be noted that correlation features realise correlation clique templates. Thus feature selection is equivalent to clique selection, which in turn defines the connectivity of the Preference Network.

5.3.4 Parameter Estimation

We limit ourselves to supervised learning in that all the ratings $\{x_{ui}\}$ in the training data are known. Since the network structure is dense we resort to the pseudo-likelihood learning method (Section 3.5.1.2). To optimise the parameters we use the stochastic gradient ascent procedure (Section 3.5.1.1). Not only is the stochastic gradient ascent fast, it is also suitable for dealing with dynamic databases in an online setting where the users constantly update the ratings. Typically, 2-3 passes through the entire data are often enough in our experiments.

5.3.5 Prediction

Recall that we employ a two-step modelling. In the learning phase (Section 5.3.4), the model includes all previous ratings. Once the model has been estimated we extend the graph structure to include new ratings that need to be predicted or recommended. Since the number of newly added ratings is typically small compared to the size of existing ratings, it can be assumed that the model parameters do not need to be re-estimated.

5.3.5.1 Preference prediction

The prediction of the rating x_{ui} for user u over item i is given as

$$\hat{x}_{ui} = \arg\max_{x_{ui}} \Pr(x_{ui} \mid \mathcal{N}(u,i), z) \tag{5.12}$$

where $\mathcal{N}(u,i)$ is the neighbourhood of the node x_{ui}. The probability $\Pr(\hat{x}_{ui}|\mathcal{N}(x_{ui}), z)$ is the measure of the *confidence* or the ranking level in making this prediction. This can be useful in practical situations when we need high precision, that is, only ratings with high confidence are presented to the users.

We can jointly infer the ratings x_u of given user u on a subset of items $\mathbf{i} = (i_1, i_2, ..)$ as follows

$$\hat{x}_u = \arg\max_{x_u} \Pr(x_u \mid \mathcal{N}(u), z) \tag{5.13}$$

where $\mathcal{N}(u)$ is the set of all existing ratings that are connected with ratings by user u. In another scenario we may want to recommend a relatively new item i to a set of promising users, we can make joint predictions x_i as follows

$$\hat{x}_i = \arg\max_{x_i} \Pr(x_i \mid \mathcal{N}(i), z) \tag{5.14}$$

where $\mathcal{N}(i)$ is the set of all existing ratings that are connected with ratings on item i. Since the sub-networks in such joint predictions are potentially densely connected, it is only feasible to apply local iterative classification methods such as Iterated Conditional Mode (Section 2.4.7.1), mean fields (Section 2.4.6.3) and relaxation labeling (e.g. see (Pelillo and Refice, 1994)).

5.3.5.2 Top-N recommendation

In order to provide a list of top-N items to a given user, the first step is usually to identify a candidate set of promising items. Then in the second step we rank and choose the best N items from this candidate set according to some measure of relevance.

Identifying the candidate set.
This step should be as efficient as possible and the candidate set should be relatively small compared to the number of items in the database. There are two common techniques used in user and item-based methods. In the user-based technique, for each user we identify the set of K most similar users, and then take the union of all items rated by these K users. Then, we remove from the union those items that the user has previously rated. In the item-based technique (Deshpande and Karypis, 2004), for each item the user has rated we select the K best similar items that the user has not rated. Then, we take the union of all similar items.

Indeed, if $K \rightarrow \infty$, or equivalently, we use all similar users and items in the database, then the item sets returned by the item-based and user-based techniques are *identical*. To see why, we show that every candidate j returned by the item-based technique is also the candidate by the user-based technique, and vice versa. Recall that a pair of items is said to be similar if they are jointly rated by the same user. Let $I(u)$ is the set of items rated by the current user u. So for every item $j \notin I(u)$ similar to item $i \in I(u)$, there must exist a user $v \neq u$ so that $i, j \in I(v)$. Since u and v jointly rate i, they are similar users, which mean that j is also in the candidate set. Analogously, for every candidate j rated by user v, which is similar to u, and $j \notin I(u)$, there must be an item $i \neq j$ jointly rated by both u and v. Thus $i, j \in I(v)$, and therefore they are similar. This means that j must be a candidate for the item-based technique.

One drawback of this neighbourhood-based method is that due to data sparsity the candidate set can be limited, and may not cover what the user is really interested in. For example, if each user rates 5 items, there are, at most, 5 users in her neighbourhood, each of whom rates 4 more items. Thus the candidate set has at most 20 items.

In our Preference Networks, the similarity measure is replaced by the correlation between

users or between items. The correlation is in turn captured by the corresponding correlation parameters. Thus, we can use either the user-user correlation or item-item correlation to identify the candidate set. Furthermore, we can also use both the correlation types and take the union of the two candidate sets.

Ranking the candidate set.
The second step in the top-N recommendation is to rank the candidates according to some scoring methods. Ranking in the user-based methods is often based on the popularity of the item, i.e. the number of users in the neighbourhood who have rated the item. Ranking in the item-based methods (Deshpande and Karypis, 2004) is computed by considering not only the number of raters but the similarity between the item being ranked and the set of items already rated by the user.

Under our Preference Networks formulation, we propose to compute the change in system energy and use it as the ranking measure. Our PN can be thought of as some stochastic physical system whose energy is related to the conditional distribution as follows

$$\Pr(x|z) = \frac{1}{Z(z)} \exp(-E(x,z)) \tag{5.15}$$

where $E(x,z) = -\mathbf{w}^\top \mathbf{F}(x,z)$ is the system energy. Thus the lower the energy the system state x has, the more probable the system is in that state. Denote by $t = (u,i)$ the index of node in the Preference Network. Since the features are function of attributes at node and of pairwise interaction between nodes, the system energy is the sum of node-based energy and interaction energy

$$E(x,z) \;\; = \;\; \sum_{t \in \mathcal{V}} E_t(x_t, z) + \sum_{(t,t') \in \mathcal{E}} E_{t,t'}(x_t, x_{t'} z)$$

Recommending a new item i to a given user u is equivalent to extending the system by adding new rating node $x_t = x_{ui}$. The change in system energy is therefore the sum of node-based energy of the new node, and the interaction energy between the node and its neighbours.

$$\Delta E(x_t, z) = E_t(x_t, z) + \sum_{t' \in \mathcal{N}(t)} E_{t,t'}(x_t, x_{t'}, z)$$

where $\mathcal{N}(t)$ is the neighbourhood of node t. For simplicity, we assume that the state of the existing system does not change after the node addition. Typically, we want the extended system to be in the most probable state, or equivalently the system state with lowest energy. This means that the node that causes the most reduction of system energy will be preferred. Since we do not know the correct state x_t of the new node t, we may guess by predicting

$\hat{x}_t$ (using Equation 5.12). Let us call the energy reduction by this method the *maximal energy change*. Alternatively, we may compute the *expected energy change* to account for the uncertainty in the preference prediction

$$\mathbb{E}[\Delta E(x_t, z)] = \sum_{x_t} P(x_t|\mathcal{N}(t), z)\Delta E(x_t, z) \tag{5.16}$$

5.3.6 Results

5.3.6.1 Data and Experimental setup

We evaluated our method on the MovieLens data[5], collected by the GroupLens Research Project at the University of Minnesota from September 19th, 1997 through April 22nd, 1998. We used the dataset of 100,000 ratings (1-5 scale). This has 943 users and 1682 movies. The data is divided into a training set of 80,000 ratings and the test set of 20,000 ratings. The training data accounts for 852,848 user-based and 411,546 item-based correlation features.

We transform the content attributes into a vector of binary indicators. Some attributes such as sex are categorical and thus are dimensions in the vector. Age requires some segmentation into intervals: under 18, 18-24, 25-34, 35-44, 45-49, 50-55, and 56+. We limit user attributes to age, sex and 20 job categories[6], and item attributes to 19 film genres[7]. Much richer movie content can be obtained from the Internet Movie Database (IMDB)[8]. Then we normalise the binary vectors by dividing it to the number of active vector elements. This makes the content features less sensitive to the amount of available content information.

5.3.6.2 Accuracy of rating prediction

For comparison we implement three methods described in Section 5.3.1: the user-based Pearson's correlation, the item-based correlation method, and the non-negative matrix factorisation. For correlation methods only positive correlations are used for prediction. For matrix factorisation the gradient descent was employed. We set the regularisation parameter as $\mu = 0.01$ and the learning rate of 5×10^{-4}. We experiment with different rank values H and then chose $H = 5$. The gradient descent was stopped after 100 iterations.

[5]http://www.grouplens.org

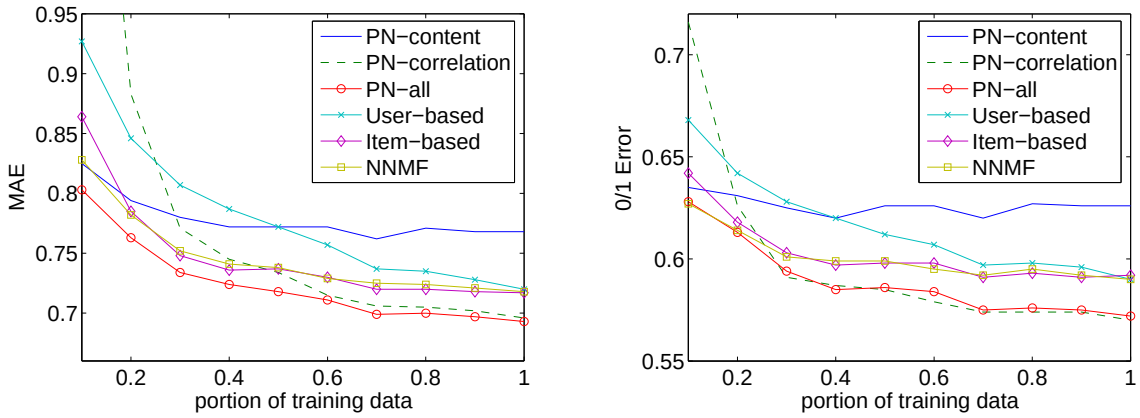
[6]Job list: administrator, artist, doctor, educator, engineer, entertainment, executive, healthcare, homemaker, lawyer, librarian, marketing, none, other, programmer, retired, salesman, scientist, student, technician and writer.

[7]Film genres: unknown, action, adventure, animation, children, comedy, crime, documentary, drama, fantasy, film-noir, horror, musical, mystery, romance, sci-fi, thriller, war and Western.

[8]http://us.imdb.com

Method	MAE	0/1 Error
User-based	0.720	0.590
Item-based	0.717	0.592
NNMF	0.718	0.590
PN	0.693	0.572

Table 5.1: Mean absolute error (MAE) of recommendation methods on MovieLens data. NNMF = Non-negative Matrix Factorisation, PN = Preference Network.



(a) (b)

Figure 5.4: (a) Mean absolute error (MAE) and (b) mean 0/1 error of recommendation methods with respect to training size of the MovieLens data. PN-content: PNs with content-based features only, PN-correlation: PNs with correlation-based features only, PN-all: PNs with all features, and NNMF: Non-negative matrix factorisation.

For the PNs, in the training phrase, we set the learning rate $\lambda = 0.001$ and the regularisation term $\sigma = 1$. Good performance is obtained after 2 iterations.

Two metrics are used: the mean absolute error (MAE)

$$\sum_{(u,i)\in\mathcal{T}'} |\hat{x}_{ui} - x_{ui}|/(|\mathcal{T}'|) \tag{5.17}$$

where $\mathcal{T}'$ is the set of rating indices in the test data, and the mean 0/1 error

$$\sum_{(u,i)\in\mathcal{T}'} \delta(\hat{x}_{ui} \neq x_{ui})/(|\mathcal{T}'|) \tag{5.18}$$

In general, the MAE is more desirable than the 0/1 error because making exact predictions may not be required and making 'close enough' predictions is still helpful. As item-based and user-used algorithms output real ratings, we round the numbers before computing the errors. Results shown in Table 5.1 demonstrate that the PN outperforms all other methods.

Sensitivity to data sparsity

To evaluate methods against data sparsity we randomly subsample the training set, but fix the test set. We report the performance of different methods using the MAE metric in Figure 5.4a and the mean 0/1 errors in Figure 5.4b. As expected, the purely content-based method deals with the sparsity in the user-item rating matrix very well, i.e. when the training data is limited. However, as the content we use here is limited to a basic set of attributes, more data does not help the content-based method further. The correlation-based method (purely collaborative filtering), on the other hand, suffers severely from the sparsity, but outperforms all other methods when the data is sufficient. Finally, the hybrid method, which combines all the content, identity and correlation features, improves the performance of all the component methods, both when data is sparse, and when it is sufficient.

5.3.6.3 Accuracy of top-N list

We produce a ranked list of items for each user in the test set so that these items do not appear in the training set. When a recommended item is in the test set of a user, we call it is a hit. For evaluation, we employ two measures. The first is the *expected utility* of the ranked list (Breese *et al.*, 1998), and the second the MAE computed over the hits. The expected utility takes into account the position j of the hit in the list for each user u

$$R_u = \sum_j \frac{1}{2^{(j-1)/(\alpha-1)}} \tag{5.19}$$

where α is the viewing half-life. Following (Breese *et al.*, 1998), we set $\alpha = 5$. Finally, the expected utility for all users in the test set is given as

$$R = 100 \frac{\sum_u R_u}{\sum_u R_u^{max}} \tag{5.20}$$

where R_u^{max} is computed as

$$R_u^{max} = \sum_{j \in I'(u)} \frac{1}{2^{(j-1)/(\alpha-1)}} \tag{5.21}$$

where $I'(u)$ is the set of items of user u in the test set.

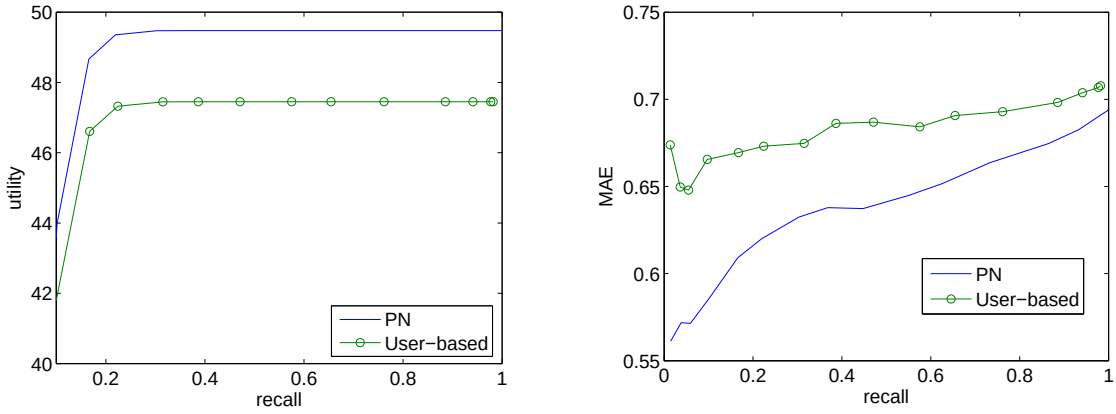
For comparison, we implement a user-based recommendation in that for each user we rank the item based on the number of times it is rated by other (positively) correlated users. Table 5.2 reports results of Preference Network using ranking measure of maximal energy change and expected energy change to produce the top 20 item recommendations.

We vary the rate of recall by varying the value of N, i.e. the recall rate typically improves as N increases. We are interested in how the expected utility and the MAE changes as

Method	MAE	Expected Utility
User-based	0.669	46.61
PN (maximal energy change)	0.603	47.43
PN (expected energy change)	0.607	48.49

Table 5.2: Performance of top-20 recommendation. PN = Preference Network.

a function of recall. The expected energy change is used as the ranking criteria for the Preference Network. Figure 5.5a shows that the utility increases as a function of the recall rate and reaches a saturation level at some point. Figure 5.5b exhibits a similar trend. It supports the argument that when the recall rate is smaller (i.e. N is small), we have more confidence on the recommendation. For both measures, it is evident that the Preference Network has advantages over the user-based method.



(a) (b)

Figure 5.5: Expected utility (a) and Mean absolute error (b) as a function of recall. The lager utility the better. The smaller MAE the better. PN = Preference Network.

5.4 Closing Remarks

This chapter has presented Relational Markov Networks, a compact representation of CRFs in relational domains. We have also applied the RMN to recommendation systems. The whole rating database was modelled by a single Markov network to best exploit the inter-dependency between variables. In terms of feature selection we employed heuristic-based correlation measures. Interestingly, feature selection in this case is not a separate task from network modelling, but rather, it specifies the network structure directly. As the network structure constructed in the case study is large-scale and densely connected, the pseudo-likelihood is used. For optimisation we use stochastic gradient ascent for efficiency and the requirement of dynamic database updating.

In our movie rating application we have treated the attribute Rating.*Score* as a *categorical* variable as in standard CRFs. In this treatment, all values are equally important. However, in fact Rating.*Score* is *ordinal*, in that the difference between the likelihoods of two values that are close (e.g. 1 and 2) should be smaller than that between values that are not close (e.g. 1 and 5). Treatment of ordinal state variables for CRFs is still an open problem and the only work we are aware of is (Mao and Lebanon, 2007).

Looking into wider contexts there are pending problems in CRFs that have not been fully investigated. First, in the heuristics we have used for selecting features, little is known about their effect on learning and final performance evaluation. Feature selection should preferably be embedded in learning so that the progress can be monitored.

Second, as evidenced in the recommendation case study, learning globally with complex network is not possible and we had to resort to the local pseudo-likelihood. This has been known to over-estimate the interaction potentials, and thus, the performance may be sub-optimal. Unfortunately, there have not been any generic global learning algorithms that are both efficient and highly accurate in arbitrary networks.

And finally, the two case studies in the previous and current chapters share a common property in that the data is essentially flat, and there are no hierarchical structures. Further, the model structures are pre-specified and are not inferred directly from the unseen data. In many domains, on the other hand, there is a natural hierarchy where structures are data dependent, and thus cannot be pre-specified. These problems, however, cannot be represented by current modelling in CRFs.

In the rest of this thesis we present investigations into these three issues.

Chapter 6

AdaBoost.CRFs for Feature Selection with Missing Labels

6.1 Introduction

As discussed in Section 3.3, feature selection plays a crucial role in the successful implementation of a CRF-based system. In Chapters 4 and 5 we employed the filtering approach that involves frequency cut-off and correlation measures. These methods help to reduce the number of features significantly, but are not integrated into the learning process and not evaluated against the final prediction performance. Since extensive evaluation of feature combinations in the wrapper approach is extremely expensive, even for small CRF-based systems, it is more reasonable to embed feature selection into learning.

One particular successful learning methodology that exhibits feature selection behaviour is boosting (Freund and Schapire, 1997; Schapire *et al.*, 1998; Schapire and Singer, 1999). In the boosting setting we have access to a pool of 'base learners', and boosting aims to boost the predictive power of these learners by sequentially adding the weighted learners into an ensemble. Base learners can be simple and weak (e.g. decision stumps (Schapire and Singer, 2000)) but they can also be sophisticated (e.g. decision trees (Quinlan, 1993; Dietterich, 2000), neural networks (Bishop, 1995; Drucker *et al.*, 1992), and Hidden Markov Models (Rabiner, 1989; Yin *et al.*, 2004)). In the context of linear classifiers, base learners are features or weighted combination of features. The sequential process of drawing features from the feature pool generally results in a small subset of features.

In this chapter we extend the boosting framework for parameter estimation of CRFs under the condition that some labels may be missing. Thus, learning is *partially supervised*. We adapt a multi-class boosting algorithm known as AdaBoost.MR (Freund and Schapire,

1997; Schapire and Singer, 1999) for partially labeled CRFs. The resulting algorithm is called AdaBoost.CRF. Its effectiveness is demonstrated through experiments on the problem of video-based human activity recognition, in which boosting provides a comparable performance to maximum likelihood estimation (MLE) but with a much smaller subset of features.

6.2 Related Work

Our work is closely related to that in (Dietterich *et al.*, 2004), where boosting is applied to learn parameters of the CRFs using gradient trees (Friedman, 2001). The objective function is the log-likelihood in the standard MLE setting but the training is based on fitting regression trees in a stage-wise fashion. The final decision function is in the form of a linear combination of regression trees. In (Dietterich *et al.*, 2004), functional gradients of the log-loss are used whilst we apply the original gradients of the exponential loss of AdaBoost (Freund and Schapire, 1997; Schapire *et al.*, 1998; Schapire and Singer, 1999). More importantly, the paper in (Dietterich *et al.*, 2004) does not incorporate hidden variables as our work does.

Another work, (Torralba *et al.*, 2005), integrates the message passing algorithm of belief propagation (BP) with a variant of LogitBoost (Friedman *et al.*, 2000). Instead of using the per-network loss as in (Dietterich *et al.*, 2004), the authors of (Torralba *et al.*, 2005) employ the per-label loss (e.g. see (Altun *et al.*, 2003b) for details of the two losses), that is, they use the marginal probabilities. The work in (Torralba *et al.*, 2005) converts the structured learning problem into a more conventional unstructured learning problem. The algorithm thus alternates between a message passing round to update the local per-label log-losses, and a boosting round to update the parameters. However, as the BP is integrated in the algorithm, it is not made clear on how to apply different inference techniques when the BP fails to converge in general networks.

There have been a number of attempts to exploit the learning power of boosting applied to structured models other than CRFs, such as dynamic Bayesian networks (DBNs) (Garg *et al.*, 2003), Bayesian network classifier (Jing *et al.*, 2005), and HMMs (Yin *et al.*, 2004).

6.3 Multi-class Boosting

This section reviews a multi-class boosting algorithm known as AdaBoost.MR (Schapire and Singer, 1999; Collins *et al.*, 2002; Lebanon and Lafferty, 2002; Altun *et al.*, 2003a), based on which our work will be developed. We adopt the functional view of boosting

from (Mason *et al.*, 2000).

Given a pool of features $\{F_m(x,z)\}$, we seek to select a subset $\{F_k(x,z)\}_{k=1}^K$ and corresponding weights $\{w_k\}$. Let $G(x,z) = \sum_{k=1}^K w_k F_k(x,z)$ be a final classifier that outputs the prediction as follows

$$\hat{x} = \arg\max_{x \in \mathcal{X}} G(x,z) \tag{6.1}$$

Given a training set $\mathcal{D} = \{x^{(l)}, z^{(l)}\}_{l=1}^n$, we would expect that $x^{(l)} = \hat{x}^{(l)}$, and thus

$$G(x^{(l)}, z^{(l)}) \geq G(x, z^{(l)}) \tag{6.2}$$

for all $x \in \mathcal{X}$ and $l = 1, 2, ..., n$. Whenever there exists an x that invalidates this assertion, the system suffers a loss. The *rank loss* is defined as

$$\mathcal{L}_{rank} = \frac{1}{n} \sum_{l=1}^{n} \sum_{x} \delta[G(x, z^{(l)}) - G(x^{(l)}, z^{(l)}) > 0] \tag{6.3}$$

where $\delta[.]$ is the indicator function. This rank loss is basically the number of possibilities where the system misclassifies the data. The loss vanishes if the system correctly classifies all the data instances.

However, the rank-loss in Equation 6.3 is difficult to minimise. Therefore we resort to the exponential-loss, which is a smooth, convex upper-bound of the rank-loss:

$$\mathcal{L}_{exp} = \frac{1}{n} \sum_{l=1}^{n} \sum_{x} \exp\{G(x, z^{(l)}) - G(x^{(l)}, z^{(l)})\} \tag{6.4}$$

Term-by-term comparison of Equations 6.3 and Equation 6.4 can easily verify that $\mathcal{L}_{exp}$ is indeed the upper-bound of $\mathcal{L}_{rank}$ up to a constant.

To see the connection between the exponential loss and the log-likelihood, assume a conditional distribution

$$\Pr(x|z) = \frac{1}{Z(z)} \exp(G(x,z)) \tag{6.5}$$

where $Z(z) = \sum_x \exp(G(x,z))$ is the normalisation constant. This assumption makes sense because the prediction rule in Equation 6.1 is identical to the Maximum A Posteriori:

$$\hat{x} = \arg\max_x G(x,z) = \arg\max_x \Pr(x|z) \tag{6.6}$$

Substituting $\Pr(x|z)$ into Equation 6.4 yields

$$\mathcal{L}_{exp} = \frac{1}{n}\sum_{l=1}^{n}\frac{1}{\Pr(x^{(l)}|z^{(l)})} \tag{6.7}$$

This appears similar to the log-loss used in the maximum likelihood estimation

$$\mathcal{L}_{log} = \frac{1}{n}\sum_{l=1}^{n}\log\frac{1}{\Pr(x^{(l)}|z^{(l)})} \tag{6.8}$$

The difference between the exponential loss and the log-loss is about the numerical scale, because of the $\log$ function in the log-loss. However, in (Lebanon and Lafferty, 2002) the authors show that the two loss functions give very close results given enough data. This paper suggests that boosting can be regarded as an (approximate) alternative for the maximum likelihood estimation (MLE). From another related angle, boosting-style MLE algorithms are derived in (Friedman *et al.*, 2000; Collins *et al.*, 2002).

The learning process in boosting is iterative, in that at each step t we greedily seek an update of the functional $G(.)$ that best reduces the loss:

$$G^{t+1} \leftarrow G^t + \alpha^t F_j \qquad \text{where} \tag{6.9}$$

$$(\alpha^t, j) = \arg\min_{\alpha,k}\mathcal{L}_{exp}(G + \alpha F_k) \tag{6.10}$$

The last equation depicts the process of incremental feature selection, i.e. only the best feature is drawn at each step.

6.4 AdaBoost.CRFs

6.4.1 Exponential Loss for Incomplete Data

We view pattern prediction in CRFs as a classification problem. However, in this case the number of distinct classes is exponentially large, i.e. $|S|^{|\mathcal{V}|}$, where $|S|$ is the size of label set, and $|\mathcal{V}|$ is the number of nodes in the Markov network. In our partially supervised setting the label set x has a visible subset ϑ and a hidden subset h, i.e. $x = (\vartheta, h)$. Given n i.i.d observations $\{\vartheta^{(l)}, z^{(l)}\}_{l=1}^{n}$, maximum likelihood learning in CRFs minimises the incomplete log-loss

$$\mathcal{L}_{log} = -\frac{1}{n}\sum_{l=1}^{n}\log\Pr(\vartheta^{(l)}|z^{(l)}) = \frac{1}{n}\sum_{l=1}^{n}\log\frac{1}{\Pr(\vartheta^{(l)}|z^{(l)})} \tag{6.11}$$

Following the development in Section 6.3, we define a new *expected ranking loss* to incorporate hidden variables as follows

$$\mathcal{L}_{rank} = \frac{1}{n}\sum_{l=1}^{n}\sum_{h}\Pr(h|\vartheta^{(l)}, z^{(l)})\sum_{\vartheta\neq\vartheta^{(l)}}\delta[\Delta G(z^{(l)}, \vartheta, h) > 0] \tag{6.12}$$

where $\Delta G(z^{(l)}, \vartheta, h) = G(z^{(l)}, \vartheta, h) - G(z^{(l)}, \vartheta^{(l)}, h)$. This rank loss captures the expected number of times when a classification is wrong. To see why, assume that the classification is right, then $\max_{\vartheta} G(z^{(l)}, \vartheta, h) = G(z^{(l)}, \vartheta^{(l)}, h)$, implying $G(z^{(l)}, \vartheta, h) < G(z^{(l)}, \vartheta^{(l)}, h)$ for all $\vartheta \neq \vartheta^{(l)}$. As for optimisation purposes, we will deal with a smooth, convex upper bound of the rank loss

$$\mathcal{L}_{exp} = \frac{1}{n}\sum_{l=1}^{n}\sum_{h}\Pr(h|\vartheta^{(l)}, z^{(l)})\sum_{\vartheta}\exp(\Delta G(z^{(l)}, \vartheta, h)) \tag{6.13}$$

When $\vartheta = x$ and $h = \emptyset$, i.e. all state variables are observed, this reduces to the rank loss proposed in (Altun *et al.*, 2003a).

A difficulty associated with this formulation is that we do not know the true conditional distribution $\Pr(h|\vartheta^{(l)}, z^{(l)})$. First, we approximate it by the learned distribution at the previous iteration. Thus, the conditional distribution is updated along the way, starting from some guessed distribution, for example, a uniform distribution. Second, we assume the log-linear model as in Equation 6.5, leading to

$$\begin{aligned}\sum_{\vartheta}\exp(\Delta G(z^{(l)}, \vartheta, h)) &= \frac{\sum_{\vartheta}\exp(G(z^{(l)}, \vartheta, h))}{\exp(G(z^{(l)}, \vartheta^{(l)}, h))} \\ &= \frac{1}{\Pr(\vartheta^{(l)}|h, z^{(l)})}\end{aligned}$$

which can be fed into Equation 6.13 to obtain

$$\begin{aligned}\mathcal{L}_{exp} &= \frac{1}{n}\sum_{l=1}^{n}\sum_{h}\frac{\Pr(h|\vartheta^{(l)}, z^{(l)})}{\Pr(\vartheta^{(l)}|h, z^{(l)})} \\ &= \frac{1}{n}\sum_{l=1}^{n}\frac{1}{\Pr(\vartheta^{(l)}|z^{(l)})}\end{aligned} \tag{6.14}$$

We can notice the similarity between the exponential loss in Equation 6.14, and the log-loss in Equation 6.11 as log(.) is a monotonically increasing function. The difference is the exponential scale used in Equation 6.14 with respect to features $\{F_k\}$ as compared to the linear scale in Equation 6.11.

6.4.2 Boosting-based Learning

Applying the greedy update rules in Equations 6.9 and 6.10, we seek the best feature F_j and its coefficient to add to the ensemble $G^{t+1} = G^t + \alpha^t F_j$ so that the loss in Equation 6.13 is minimised.

$$\begin{aligned} (\alpha^t, j) &= \arg\min_{\alpha,k} \mathcal{L}_{exp}(t, \alpha, k), \text{ where} \\ \mathcal{L}_{exp}(t, \alpha, k) &= \frac{1}{n}\sum_{l=1}^{n} \mathbb{E}_{h|\vartheta^{(l)},z^{(l)},t}[\sum_{\vartheta} \exp(\Delta G^{l,t} + \alpha \Delta F_k^{(l)})] \end{aligned} \tag{6.15}$$

and $\mathbb{E}_{h|\vartheta^{(l)},z^{(l)},t}[.]$ is the expectation with respect to the distribution $\Pr(h|\vartheta^{(l)}, z^{(l)}, t)$; and $G^{l,t}$ and $F_k^{(l)}$ are shorthands for $G^t(z^{(l)}, \vartheta, h)$ and $F_k(z^{(l)}, \vartheta, h)$, respectively. Note that this is just an approximation to the loss in Equation 6.13 because we fix the conditional distribution $\Pr(h|\vartheta^{(l)}, z^{(l)}, t)$ obtained from the previous iteration. However, this still makes sense since the learning is incremental, and thus the estimated distribution will get closer to the true distribution along the way. Indeed, this captures the essence of boosting: during each round boosting selects the base learner that best minimises the following loss over the weighted data distribution (Schapire and Singer, 1999)

$$(\alpha^t, j) = \arg\min_{\alpha,k} \frac{1}{n}\sum_{l=1}^{n}\sum_{\vartheta,h} D(l, \vartheta, h, t) \exp(\alpha \Delta F_k^{(l)}) \tag{6.16}$$

where $D(l, \vartheta, h, t)$ is the weighted data distribution

$$D(l, \vartheta, h, t) = \frac{\Pr(h|\vartheta^{(l)}, z^{(l)}, t) \exp(\Delta G^{l,t})}{\sum_{l'} \Pr(h|\vartheta^{(l')}, z^{(l')}, t) \exp(\Delta G^{l',t})} \tag{6.17}$$

Since the data distribution does not contain α, Equation 6.16 is identical to Equation 6.15 up to a constant.

6.4.3 Beam Search

It should be noted that boosting is a very generic framework to boost the performance of the base learner. Thus, we can build more complex and stronger base learners by using some ensemble of features and then later fit them into the boosting framework. However, here we stick to simple base learners, which are features, to make the algorithm compatible with the MLE.

We can select a number of top features and associated coefficients that minimise the loss in Equation 6.16 instead of just one feature. This is essentially a beam search with specified beam size B.

6.4.4 Regularisation

We employ a l_2 regularisation term to make it consistent with the popular Gaussian prior used in conjunction with the MLE of CRFs. It also maintains the convexity of the original loss. The regularised loss becomes

$$\mathcal{L}_{reg} = \mathcal{L}_{non-reg} + \sum_k \frac{w_k^2}{2\sigma_k^2} \tag{6.18}$$

where $\mathcal{L}_{non-reg}$ is either $\mathcal{L}_{log}$ for MLE in Equation 6.11 or $\mathcal{L}_{exp}$ for boosting in Equation 6.13. Note that the regularisation term for boosting does not have the Bayesian interpretation as in the MLE setting but is simply a constraint to prevent the parameters from growing too large, i.e. the model fits the training data too well, which is clearly sub-optimal for noisy and unrepresentative data. The effect of regularisation can be numerically very different for the two losses, so we cannot expect the same σ for both MLE and boosting.

6.5 Efficient Computation

Straightforward implementation of the optimisation in Equation 6.15 or Equation 6.16 by sequentially and iteratively searching for the best features and parameters can be impractical if the number of features is large. This is partly because the objective function, although tractable to compute using dynamic programming in tree-like structures, is still expensive. We propose an efficient approximation which requires only a few vectors and an one-step evaluation. The idea is to exploit the convexity of the loss function $\mathcal{L}_{exp}(t, \alpha, k)$ by approximating it with a convex quadratic function using second-order Taylor's expansion. The change due to the update is approximated as

$$\Delta\mathcal{L}_{exp}(t, \alpha, k) \approx \left.\frac{d\mathcal{L}_{exp}(t, \alpha, k)}{d\alpha}\right|_{\alpha=0} \alpha + \frac{1}{2}\left.\frac{d^2\mathcal{L}_{exp}(t, \alpha, k)}{d\alpha^2}\right|_{\alpha=0} \alpha^2 \tag{6.19}$$

The selection procedure becomes

$$(\alpha^t, j) = \arg\min_{\alpha,k} \mathcal{L}_{exp}(t, \alpha, k) = \arg\min_{\alpha,k} \Delta\mathcal{L}_{exp}(t, \alpha, k)$$

The optimisation over α has an analytical solution

$$\alpha_t = -\frac{\mathcal{L}'_{exp}}{\mathcal{L}''_{exp}} \tag{6.20}$$

Once the feature has been selected the algorithm can proceed by applying an additional line-search step to find the best coefficient as $\alpha^t = \arg\min_\alpha \mathcal{L}_{exp}(t, \alpha, j)$. One way to do

so is to repeatedly apply the update based on Equation 6.20 until convergence.

Up to now we have made an implicit assumption that all computation can be carried out efficiently. However, this is not the case for general CRFs because most quantities of interest involve summation over an exponentially large number of network configurations. Similar to (Altun *et al.*, 2003a), we show that dynamic programming exists for tree-structured networks. However, for general structures approximate inference must be used. This issue will be studied in Chapter 7.

There are three quantities we need to compute: the distribution $\Pr(v^{(l)}|z^{(l)})$ in Equation 6.14, the first and second derivative of $\mathcal{L}_{exp}(t,\alpha,k)$ in Equation 6.19. For the distribution, we have

$$\begin{aligned}\Pr(\vartheta^{(l)}|z^{(l)}) &= \sum_{h}\Pr(\vartheta^{(l)},h|z^{(l)}) \\ &= \frac{Z(\vartheta^{(l)},l)}{Z(l)} \end{aligned} \tag{6.21}$$

where $Z(\vartheta^{(l)},l) = \sum_h \exp(\sum_c G(z^{(l)},\vartheta_c^{(l)},h_c))$ and $Z(l) = \sum_x \exp(\sum_c G(z^{(l)},x_c))$. Both these partition functions are in the form of sum-product, thus, they can be computed efficiently using a single pass through a tree-like structure. The first and second derivatives of $\mathcal{L}_{exp}(t,\alpha,k)$ are then

$$\mathcal{L}'_{exp}|_{\alpha=0} = \frac{1}{n}\sum_{l=1}^{n}\mathbb{E}_{h|\vartheta^{(l)},z^{(l)},t}[\sum_{\vartheta}\exp(\Delta G^{l,t})\Delta F_k^{(l)}] \tag{6.22}$$

$$\mathcal{L}''_{exp}|_{\alpha=0} = \frac{1}{n}\sum_{l=1}^{n}\mathbb{E}_{h|\vartheta^{(l)},z^{(l)},t}[\sum_{\vartheta}\exp(\Delta G^{l,t})(\Delta F_k^{(l)})^2] \tag{6.23}$$

Expanding Equation 6.22 yields

$$\mathcal{L}'_{exp}|_{\alpha=0} = \frac{1}{n}\sum_{l=1}^{n}\frac{1}{\Pr(\vartheta^{(l)}|z^{(l)},t)}\sum_{\vartheta,h}\Pr(\vartheta,h|z^{(l)},t)\Delta F_k^{(l)} \tag{6.24}$$

Recall that $F_k^{(l)}$ in the CRF is decomposed into the sum of clique-based features as $F_k^{(l)}(z^{(l)},x) = \sum_c f_k^{(l)}(z^{(l)},x_c)$. It follows that $\Delta F_k(z^{(l)},x) = \sum_c \Delta f_k(z^{(l)},x_c)$. Thus Equation 6.24 reduces to

$$\mathcal{L}'_{exp}|_{\alpha=0} = \frac{1}{n}\sum_{l=1}^{n}\frac{1}{\Pr(\vartheta^{(l)}|z^{(l)},t)}\sum_{c}\sum_{x_c}\Pr(x_c|z^{(l)},t)\Delta f_k(z^{(l)},x_c) \tag{6.25}$$

which now contains clique marginals and can be estimated efficiently for tree-like structures using a downward and upward sweep. For general structures, loopy belief propagation can provide approximate estimates. Details of the procedure are omitted here due to

space constraints.

However, the computation of Equation 6.23 does not enjoy the same efficiency because the square function is not decomposable. To make it decomposable, we employ Cauchy's inequality to yield the upper bound of the change (Equation 6.19) as

$$\begin{aligned}(\Delta F_k(z^{(l)}, x))^2 &= (\sum_c \Delta f_k(z^{(l)}, x_c))^2 \\ &\leq |C| \sum_c \Delta f_k(z^{(l)}, x_c)^2\end{aligned}$$

where $|C|$ is the number of cliques in the network.

The update using $\alpha = -\mathcal{L}'_{exp}/\tilde{\mathcal{L}}''_{exp}$, where $\tilde{\mathcal{L}}''_{exp}$ is the upper bound of the second derivative $\mathcal{L}''_{exp}$, is rather conservative, so it is clear that a further line search is needed. Moreover, it should be noted that the change in Equation 6.19, due to the Newton update, is

$$\Delta\tilde{\mathcal{L}}_{exp}(\alpha, k) = -0.5\frac{(\mathcal{L}'_{exp})^2}{\tilde{\mathcal{L}}''_{exp}} \tag{6.26}$$

where $\Delta\tilde{\mathcal{L}}_{exp}$ is the upper bound of the change $\Delta\mathcal{L}_{exp}$ due to Cauchy's inequality, so the base learner selection using the optimal change does not depend on the scale of the second derivative bound of $\tilde{\mathcal{L}}''_{exp}$. Thus, the term $|C|$ in Cauchy's inequality above can be replaced by any convenient constant.

The complexity of our boosting algorithm is the same as that in the MLE of the CRFs. This can be verified easily by taking the derivative of the log-loss in Equation 6.11 and comparing it with the quantities required in our algorithm.

6.6 Evaluations

6.6.1 Data and Feature Extraction

We evaluate the proposed AdaBoost.CRF algorithm on the problem of home video surveillance that was previously studied in (Nguyen *et al.*, 2005). The task is to recognise activities performed by a person in a kitchen using two cameras mounted on two opposite ceiling corners. There are three complex activities: SHORT_MEAL, HAVE_SNACK and NORMAL_MEAL. Each of these consists of some of 12 primitive activities (Table 6.1), which are essentially trajectories between landmark points. Specifically, SHORT_MEAL = {1,2,3,4,11}, HAVE_SNACK = {2,5,6,7,8}, and NORMAL_MEAL = {1,2,4,9,10,11,12}.

No.	Activity	No.	Activity
1	Door→Cupboard	7	Fridge→TV chair
2	Cupboard→Fridge	8	TV chair→Door
3	Fridge→Dining chair	9	Fridge→Stove
4	Dining chair→Door	10	Stove→Dining chair
5	Door→TV chair	11	Fridge→Door
6	TV chair→Cupboard	12	Dining chair→Fridge

Table 6.1: Primitive activities, from Nguyen *et al.* (2005).

The raw data consists of 90 video sequences from which noisy coordinates of the person at each time step are extracted using a background subtraction algorithm. The coordinate sequences are then used as the observations since they are deemed relevant for the task of recognising sub-trajectories. Each time step is manually annotated by two labels: the complex and primitive activities. The labels are given at training time to learn the model and used as ground-truth to evaluate the accuracy of the model's prediction. The data is divided into training and testing sub-sets with 45 sequences each.

Although the data is hierarchical, we restrict our attention to modelling and recognising the primitive activities only. The data is divided into three subsets corresponding to the three complex activities. Thus, the problem is inherently sequential for which a chain-structured CRF is appropriate and thus efficient. The state space of each subset is limited to the corresponding primitive activities.

For all the experiments reported here, we train the model using the MLE along with the limited memory quasi-Newton method (L-BFGS) and we use the proposed boosting scheme with the help of a line search, satisfying Amijo's conditions (Nocedal and Wright, 1999). For regularisation, the same σ is used for all features for simplicity and is empirically selected. In the training data, only 50% of labels are randomly given for each data slice in the sequence. For the performance measure, we report the per-label error and the average F_1-score over all distinct labels[1].

From the raw observation of coordinates, we extract five observational features at each time step τ: $g(z,\tau) = \{g_m(z,\tau)\}_{m=1}^{5}$. These include the (X, Y) coordinates, the u_X & u_Y velocities, and the speed $\sqrt{u_X^2 + u_Y^2}$, respectively. These observational features are approximately normalised so that they are of comparable scale.

[1]The F_1-score is computed as $F_1 = 2 \times R \times P/(R + P)$, where R is the recall rate, and P is precision.

6.6.2 Effect of Feature Selection

We design three feature sets. The first set, called *activity-persistence*, captures the fact that activities are in general persistent. The set is divided into data-association features

$$f_{l,m}(z, x_\tau) = \delta[x_\tau = l]g_m(z, \tau) \tag{6.27}$$

where $m = 1, .., 5$, and label-label features

$$f_{l,m}(z, x_{\tau-1}, x_\tau) = \delta[x_{\tau-1} = x_\tau]\delta[x_\tau = l] \tag{6.28}$$

Thus the set has $K = 5|S| + |S|$ features, where $|S|$ is the size of the label set.

The second feature set consists of *transition-features* that are intended to encode the activity transition nature as follows

$$f_{l_1,l_2,m}(z, x_{\tau-1}, x_\tau) = \delta[x_{\tau-1} = l_1]\delta[x_\tau = l_2]g_m(z, \tau) \tag{6.29}$$

Thus the size of the feature set is $K = 5|S|^2$.

The third set, called the *context set*, is a generalisation of the second set. Observation-features now incorporate neighbouring observation points within a sliding window of width W

$$g_m(z, \tau, \epsilon) = g_m(z, \tau + \epsilon) \tag{6.30}$$

where $\epsilon = -W_l, ..0, ..W_u$ with $W_l + W_u + 1 = W$. This is intended to capture the correlation of the current activity with the past and the future, or the temporal *context* of the observations. The second feature set is a special case with $W = 1$. The number of features is a multiple of that in the second set, which is $K = 5W|S|^2$.

The boosting studied here has a beam size $B = 1$, i.e. each round picks only one feature to update its weight. Tables 6.2, 6.3 and 6.4 show the performance of the training algorithms on test data of all three scenarios (SHORT_MEAL, HAVE_SNACK and NORMAL_MEAL) for the three feature sets, respectively. Note that the infinite regularisation factor σ means that there is no regularisation. In general, sequential boosting appears to be slower than the MLE because it updates only one parameter at a time. For the activity persistence features (Table 6.2), the feature set is compact but informative enough so that the MLE attains a reasonably high performance. Due to this compactness, the feature selection capacity is almost eliminated, leading to poorer results as compared with the MLE.

However, the situation changes radically for the activity transition feature set (Table 6.3) and for the context feature set (Table 6.4). When the observation context is small, i.e.

Table 6.2: Performance on three data sets, activity-persistence features. Here, SM = SHORT_MEAL, HS = HAVE_SNACK , NM = NORMAL_MEAL, Agthm = algorithm, itrs = number of iterations, ftrs = number of selected features, % ftrs = portion of selected features.

Data	SM	SM	HS	HS	NM	NM
Agthm	MLE	Boost	MLE	Boost	MLE	Boost
σ	∞	∞	∞	∞	∞	∞
error(%)	10.3	16.6	12.4	14.5	9.7	17.2
F_1(%)	86.0	80.2	84.8	82.1	87.9	77.4
itrs	100	500	100	200	100	200
# ftrs	30	30	30	30	42	35
% ftrs	100	100	100	100	100	83.3

Table 6.3: Performance on activity transition features

Data	SM	SM	HS	HS	NM	NM
Agthm	MLE	Boost	MLE	Boost	MLE	Boost
σ	∞	∞	∞	∞	∞	∞
error(%)	18.6	10.1	13.0	10.8	15.0	16.5
F_1(%)	75.8	89.3	86.8	85.7	81.4	80.9
itrs	59	200	74	100	53	100
# ftrs	125	57	125	44	245	60
% ftrs	100	45.6	100	35.2	100	24.5

$W = 1$, boosting consistently outperforms the MLE whilst maintaining only a partial subset of features ($< 50\%$ of the original feature set). The feature selection capacity is demonstrated more clearly with the context-based feature set (W = 11), where less than 9% of features are selected by boosting for the SHORT_MEAL scenario, and less than 3% for the NORMAL_MEAL scenario. The boosting performance is still reasonable despite the fact that a very compact feature set is used. There is therefore a clear computational advantage when the learned model is used for classification.

6.6.3 Learning the Activity-Transition Model

In this section we demonstrate that the activity transition model can be learned by both the MLE and boosting. The transition feature sets studied previously do not separate the transitions from data, so the transition model may not be correctly learned. We design another feature set, which is the bridge between the activity-persistence and the transition feature set. Similar to the activity persistence set, the new set is divided into data-association

Table 6.4: Performance on context features with window size $W = 11$

Data	SM	SM	HS	HS	NM	NM
Agthm	MLE	Boost	MLE	Boost	MLE	Boost
σ	2	2	∞	∞	∞	∞
error(%)	15.3	9.6	9.4	11.2	9.3	16.6
F_1(%)	81.6	87.7	89.3	86.6	87.7	78.1
itrs	51	200	22	100	21	100
# ftrs	1375	115	1375	84	2695	80
% ftrs	100	8.36	100	6.1	100	3.0

Table 6.5: Activity transition matrix of SHORT_MEAL data set

Activity	1	2	3	4	11
1	1	1	0	0	0
2	0	1	1	0	1
3	0	0	1	1	0
4	0	0	0	1	0
11	0	0	0	0	1

features, as in Equation 6.27, and label-label features

$$f_{l_1,l_2}(x_{\tau-1}, x_\tau) = \delta[x_{\tau-1} = l_1]\delta[x_\tau = l_2] \tag{6.31}$$

Thus the set has $K = 5|S| + |S|^2$ features.

Given the SHORT_MEAL data set, and the activity transition matrix in Table 6.5, the parameters corresponding to the label-label features are given in Tables 6.6 and 6.7, as learned by boosting and MLE, respectively.

At first sight it may be tempting to select non-zero parameters and their associated transition features, and hence the corresponding transition model. However, as transition features are non-negative (indicator functions), the model actually penalises the probabilities

Table 6.6: Parameter matrix of SHORT_MEAL data set learned by boosting

Activity	1	2	3	4	11
1	1.8	0	-5904.9	-5904.9	0
2	-5904.9	3.6	0	-5904.9	0
3	-5904.9	-5904.9	2.425	0	-5904.9
4	-5904.9	-5904.9	-5904.9	2.4	-5904.9
11	-5904.9	-5904.9	-5904.9	-5904.9	2.175

Table 6.7: Parameter matrix of SHORT_MEAL data set learned by MLE

Activity	1	2	3	4	11
1	10.81	4.311	-5.7457	-5.3469	-1.8398
2	-2.2007	15.056	3.6388	-5.6644	0.41921
3	-5.3565	-2.3131	9.3656	1.6575	-2.3736
4	-5.4103	-4.556	-4.1142	7.1332	-5.2976
11	-3.17	-0.09001	-2.9518	-4.8741	8.9128

of any configurations that activate negative parameters exponentially, since $\Pr(x|z) \propto \exp(w_k F_k(x_{\tau-1}, x_\tau))$. Therefore, huge negative parameters practically correspond to improbable configurations. If we replace all non-negative parameters in Table 6.6 and 6.7 by 1, and the rest by 0, we actually obtain the transition matrix in Table 6.5. The difference between boosting and MLE is that boosting penalises the improbable transitions much more severely, thus leading to much sharper decisions with high confidence. Note that for this data set boosting learns a much more correct model than the MLE, with an error rate of 3.8% (F_1 = 93.7%), in constrast to 15.6% (F_1 = 79.5%) by the MLE without regularisation, and 11.8% (F_1 = 85.0%) by the MLE with $\sigma = 5$.

6.6.4 Effect of Beam Size

Recall that the beam search described in Section 6.4.3 allows the base learner to be an ensemble of B features. When $B = K$, all the parameters are updated in parallel, so it is essentially similar to the MLE, and thus no feature selection is performed. We run a few experiments with different beam sizes B, starting from 1, which is the main focus of this study, to the full parameter set K. As B increases, the number of selected features also increases. However, it is inconclusive about the final performance. It seems that when B is large, the update is quite poor, leading to slow convergence. This is probably because the diagonal matrix resulting from the algorithm is not a good approximation to the true Hessian used in Newton updates. It suggests that there exists a good, but rather moderate beam size that performs best in terms of both the convergence rate and the final performance.

An alternative is just to minimise the exponential loss in Equation 6.13 directly by using any generic optimisation method (e.g. see (Altun *et al.*, 2003a,b)). However, this approach, although fast to converge, loses the main idea behind boosting, which is to re-weigh the data distribution on each round to focus more on hard-to-classify examples as in Equation 6.16. These issues are left for future investigation.

6.7 Closing Remarks

We have presented a scheme to exploit the discriminative learning power of the boosting methodology and the semantically rich structured model of CRFs and integrated them into a boosting based CRF framework which can handle missing variables. We have demonstrated the performance of the newly proposed algorithm (AdaBoost.CRF) over the standard maximum-likelihood frameworks on video-based activity recognition tasks. The built-in capacity of feature selection by boosting suggests an interesting application area in small footprint devices with limited processing.

However, in our algorithm, we have assumed that the underlying inference is efficient in computing clique marginals. This assumption, unfortunately, only holds for a restricted class of tree-like Markov network structures. For general networks, approximate inference must be used. The drawback of this approximation is that since the first and second derivatives cannot be computed exactly, it is very hard to analyse the convergence property of optimisation method used in the learning algorithm. For example, the updating rule in Equation 6.20 may be corrupted. One possible approach to handle this problem of stochastic derivatives is to apply stochastic gradient methods as in Chapter 4 and Chapter 5 with the hope that the long term effect of these methods will average out the randomness introduced by inference approximation. An alternative approach is to employ approximate loss functions that support exact inference. This type of loss and convergence properties are easier to characterise. This will be presented in the next chapter.

Chapter 7

AdaBoost.MRF for Learning CRFs with General Structures

7.1 Introduction

In the last chapter we have addressed the problem of feature selection under partially supervised conditions. The underlying inference of the learning process is assumed to be efficient. However, this only holds for tree-like structures, and learning CRFs in general structures is intractable.

There are two general approaches to deal with this problem: stochastic and deterministic. Stochastic methods allow running parameter updates *even* with inexact computation, and they carefully control the learning process in the way that it may converge to the true maximum likelihood solution. Deterministic methods, on the other hand, work only with exact computation, but deterministically approximate the true likelihood by more efficient objective functions.

In the stochastic approach attention is paid to the quality of the stochastic process, e.g. convergence, bias and variance. However, under the general network setting these issues are poorly understood. More specifically, as we have presented in Section 2.4.6, approximate inference can be carried out in different ways, either through sampling or through message passing algorithms. Unfortunately, sampling can be extremely slow to reach good approximation and message passing algorithms are not guaranteed to converge. Under the practical constraints of running time these methods often result in approximate quantities required in parameter estimation, causing the optimisation loop to stop prematurely.

In the deterministic approach, since there is no approximate inference that affects the quality of the parameter updating process we can focus our attention to the parameter esti-

mation. The art is to maintain a good balance between inference efficiency and the quality of approximation of the objective function to the true likelihood. Examples include pseudo-likelihood (see Section 3.5.1.2 for description and Chapter 5 for an application in recommender systems), piece-wise pseudo-likelihood and piece-wise likelihood (see Section 3.5.2).

This chapter addresses the intractability of parameter estimation under general structures by following the deterministic approach. We introduce a novel algorithm called AdaBoost.MRF. The name comes from the fact that it is based on AdaBoost - a boosting algorithm we have studied in the context of feature selection in the previous chapter. The second part of the algorithm stands for Markov Random Forest, or the collection of Markov trees induced by the graph under study. We exploit the fact that a graph is a superimposition of many spanning trees, which are intractable jointly for inference but efficient individually. The main part of the algorithm is a method to effectively distribute the parameter estimation task to individual trees and then combine the results at the end. We show that under mild assumptions the AdaBoost.MRF is guaranteed to reach the unique optimum. Furthermore, since the AdaBoost.MRF considers all the variables in the MRFs, the problem of hidden variable can also be handled effectively.

We demonstrate the effectiveness of the AdaBoost.MRF on the home video surveillance data described in Chapter 6. However, this time we jointly model multiple levels of activities using a grid CRF, known as Factorial CRF (FCRF) (Sutton *et al.*, 2007) instead of the flat CRF as in Chapter 6. Differing from previous applications of the FCRF we tackle the problem of missing labels. We compare our AdaBoost.MRF with the standard maximum likelihood method, which uses Loopy BP (Section 2.4.6) and its variant (Wainwright *et al.*, 2005b) as the underlying inference engines. To evaluate the effectiveness of the discriminative FCRFs against generative methods, we implement a variant of the layered hidden Markov models (LHMMs) (Oliver *et al.*, 2004), that has previously been applied for activity recognition. Differing from the original LHMMs, our variant can handle partially observed state variables to make it compatible with the FCRFs considered in this paper.

7.2 AdaBoost.MRF

In this section we describe AdaBoost.MRF, the boosting algorithm for parameter estimation of general Markov random fields. As in the previous chapter, we consider the general case where the state label x may have a hidden component h and a visible component ϑ, that is $x = (\vartheta, h)$.

7.2.1 Boosted Markov Random Forests

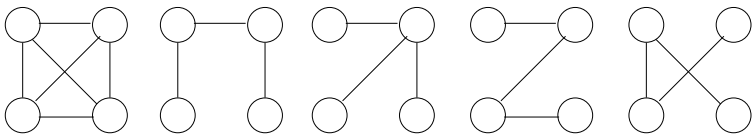


Figure 7.1: An example of Markov network (left-most) and some spanning trees (right).

Recall from Section 6.3 that given a set of weak learners $\{F_k(x,z)\}_{k=1}^K$, boosting seeks a linear combination that makes a strong learner as follows: $G(x,z) = \sum_{k=1}^K w_k F_k(x,z)$, where $\{w_k\}_{k=1}^K$ are corresponding weights.

In the fully supervised setting with n i.i.d observations $\{x^{(l)}, z^{(l)}\}_{l=1}^n$, we want to minimise the exponential loss of Equation 6.4. In partial supervision, on the other hand, we are given only the visible part $\vartheta^{(l)}$ of $x^{(l)}$ for the instance l. We propose to minimise the *incomplete* loss given by

$$\mathcal{L}_{inco} = \sum_l \sum_{\vartheta} \exp(G(\vartheta, z^{(l)}) - G(\vartheta^{(l)}, z^{(l)})) \tag{7.1}$$

In this setting, at step t, the strong learner $G^t(\vartheta, z)$ is updated by adding a weak-learner $F^t(\vartheta, z)$ to the previous $G^{t-1}(\vartheta, z)$ as

$$G^t(\vartheta, z) = G^{t-1}(\vartheta, z) + \alpha^t F^t(\vartheta, z) \tag{7.2}$$

where α^t is the weight of each weak learner in the ensemble. The weak learner and its weight are chosen to minimise the loss in Equation 7.1, i.e.

$$(F^t, \alpha^t) = \arg\min_{F,\alpha} \mathcal{L}_{inco} \tag{7.3}$$

As we are interested in estimating the distribution $\Pr(\vartheta|z)$ we may choose the weak learner as $F(\vartheta, z) = \log \Pr(\vartheta|z)$. However, if we use the distribution defined over the general Markov networks, the computation of the weak learner itself is intractable. To address this issue we propose the use of spanning trees as weak learners. Thus, spanning trees are weak approximations to the whole network. The spanning tree-based learners are 'weak' because they are crude approximations of the true model

$$F(\vartheta, z) = \log \Pr_\tau(\vartheta|z) \tag{7.4}$$

where τ is the index of the spanning trees in the network. This choice also allows incorpo-

ration of the hidden information since

$$F(\vartheta, z) = \log \sum_{h} \Pr_{\tau}(\vartheta, h|z) \tag{7.5}$$

The strong learner G is therefore a collection of trees, and hence we call our boosting method AdaBoost.MRF (AdaBoosted Markov Random Forests). Figure 7.1 shows a simple example of a four-node network and some spanning trees.

7.2.2 Loss Bound using Hölder's Inequality

With the tree selection procedure described in the previous subsection, and given the fact that the strong learner is the weighted sum of the tree log-likelihood, the incomplete exponential loss (Equation 7.1) at the step t becomes

$$\begin{aligned} \mathcal{L}_{inco} &= \sum_{l,\vartheta} \exp\left\{ \sum_{j=1}^{t} \alpha^j \left(\log \Pr_{\tau_j}(\vartheta|z^{(l)}) - \log \Pr_{\tau_j}(\vartheta^{(l)}|z^{(l)}) \right) \right\} \\ &= \sum_{l} \frac{\sum_{\vartheta} \prod_j \Pr_{\tau_j}(\vartheta|z^{(l)})^{\alpha^j}}{\prod_j \Pr_{\tau_j}(\vartheta^{(l)}|z^{(l)})^{\alpha^j}} \end{aligned} \tag{7.6}$$

Although the evaluation of each weak learner is tractable, the sum over all visible variables in the numerator is unfortunately intractable, except for the special case when all selected spanning trees are the same[1].

Fortunately, there exists a technique that helps to remove the summation in the numerator. The idea is to apply the Hölder's inequality (Hardy *et al.*, 1952, Theorem 11) (see Appendix A.2 for details) to the numerator

$$\sum_{\vartheta} \prod_{j} \Pr_{\tau_j}(\vartheta|z^{(l)})^{\alpha^j} \leq \prod_{j} \left(\sum_{\vartheta} \Pr_{\tau_j} \left(\vartheta|z^{(l)}\right)^{\alpha^j r_j}\right)^{1/r_j} \tag{7.7}$$

where $\sum_j 1/r_j = 1$ and $r_j > 0$. If we can ensure that $\alpha^j > 0$ and $\alpha^j r_j = 1$ for all j, or

[1]This case can only happen if the Markov network is originally a tree or during the course of learning, no other structures can compete with one particular tree. The former case is not interesting because any learning method will do and we suspect that the latter case rarely happens unless the tree is a very good approximation to the original network.

$\sum_j \alpha^j = 1$, we obtain

$$\begin{aligned}\mathcal{L}_{inco} &\leq \sum_l \frac{\prod_j \left\{ \sum_\vartheta \Pr_{\tau_j}(\vartheta|z^{(l)}) \right\}^{\alpha^j}}{\prod_j \Pr_{\tau_j}(\vartheta^{(l)}|z^{(l)})^{\alpha^j}} \\ &= \sum_l \frac{1}{\prod_j \Pr_{\tau_j}(\vartheta^{(l)}|z^{(l)})^{\alpha^j}} \\ &= \mathcal{L}_H \end{aligned} \tag{7.8}$$

since $\sum_\vartheta \Pr_{\tau_j}(\vartheta|z^{(l)}) = 1, \forall l, j$.

Using the fact that $\log \Pr_{\tau_j}(\vartheta|z)$ is a weak learner we can rewrite the upper bound loss $\mathcal{L}_H$ as

$$\mathcal{L}_H(G) = \sum_l \exp\left\{ -\sum_j \alpha^j \log \Pr_{\tau_j}(\vartheta^{(l)}|z^{(l)}) \right\} \tag{7.9}$$

$$= \sum_l \exp\left\{ -G^t(\vartheta^{(l)}, z^{(l)}) \right\} \tag{7.10}$$

It can be seen that the new bound is tractable to evaluate, and is also convex so that a global minimum exists. We use the new loss $\mathcal{L}_H$ for learning. The domain of $\mathcal{L}_H$ is therefore a linear space of functions (Mason *et al.*, 2000), which are $\{F^t(\vartheta, z^{(l)}) = \log \Pr_{\tau_t}(\vartheta|z^{(l)})\}$ in our case.

The requirement $\sum_j \alpha^j = 1$ can be met by defining the following ensemble

$$G^t(\vartheta, z) = (1-\alpha^t)G^{t-1}(\vartheta, z) + \alpha^t F^t(\vartheta, z) \tag{7.11}$$

$$= G^{t-1}(\vartheta, z) + \alpha^t s^t(\vartheta, z) \text{ where} \tag{7.12}$$

$$s^t(\vartheta, z) = F^t(\vartheta, z) - G^{t-1}(\vartheta, z) \tag{7.13}$$

From Equation 7.12, it can be seen that $s^t()$ plays the role of the search direction with respect to the functional $G()$. Each previous weak learner's weight is scaled down by a factor of $1-\alpha^t$ as

$$\alpha_*^j \leftarrow \alpha^j(1-\alpha^t) \tag{7.14}$$

for $j = 1, ..., t-1$, so that $\sum_{j=1}^{t-1} \alpha_*^j + \alpha^t = \sum_{j=1}^{t-1} \alpha^j(1-\alpha^t) + \alpha^t = 1$, since $\sum_{j=1}^{t-1} \alpha^j = 1$.

7.2.3 Weak Learners, Convergence and Complexity

7.2.3.1 Selecting the best tree

We now show how to carry out the stepwise optimisation in Equation 7.3 with the incomplete loss replaced by the upper bound $\mathcal{L}_H(G)$ in Equation 7.10.

The loss $\mathcal{L}_H(G)$ as a function of $G(\vartheta, z)$ can be minimised by moving in the opposite direction of gradient

$$\nabla\mathcal{L}_H(\vartheta, z^{(l)}) = \begin{cases} -\exp(-G^{t-1}(\vartheta^{(l)}, z^{(l)})) & \text{if } \vartheta = \vartheta^{(l)} \\ 0 & \text{otherwise} \end{cases} \tag{7.15}$$

However, as the functional gradient $\nabla\mathcal{L}_H(G)$ and and the functional direction s in Equation 7.12 may not belong to the same function space, direct optimisation may not apply. In (Mason *et al.*, 2000) the authors propose to find s^t which points to the decreasing direction of $\mathcal{L}_H$, i.e.

$$\langle\nabla\mathcal{L}_H, s\rangle < 0 \tag{7.16}$$

Thus the best search direction s^t is the solution of

$$s^t = \arg\min_s \langle\nabla\mathcal{L}_H, s\rangle \tag{7.17}$$

The step size α^t is determined using a line search or by setting it to a small constant $\in (0, 1)$.

Let us define the weight of data instance l

$$D^{t-1}(l) = \frac{\exp\{-G^{t-1}(\vartheta^{(l)}, z^{(l)})\}}{\sum_l \exp\{-G^{t-1}(\vartheta^{(l)}, z^{(l)})\}} \tag{7.18}$$

These weights play the role of data distribution which is updated as boosting proceeds. Substituting Equation 7.15 into Equation 7.17, we have

$$s^t \;=\; \arg\min_s \sum_l -D^{t-1}(l)s(\vartheta^{(l)}, z^{(l)}) \tag{7.19}$$

As $s(\vartheta^{(l)}, z^{(l)}) = F(\vartheta^{(l)}, z^{(l)}) - G^{t-1}(\vartheta^{(l)}, z^{(l)})$, minimising with respect to $s(\vartheta^{(l)}, z^{(l)})$ and $F(\vartheta^{(l)}, z^{(l)})$ is equivalent, since $G^{t-1}(\vartheta^{(l)}, z^{(l)})$ is a constant. Recall from Equation 7.4 that $F(\vartheta^{(l)}, z^{(l)}) = \log \Pr_\tau(\vartheta^{(l)}|z^{(l)}; w_\tau)$, this minimisation translates to selecting the best tree

τ_t and its parameters $\mathbf{w}_{\tau_t}$ as follows

$$(\tau_t, \mathbf{w}_{\tau_t}) = \arg\max_{\tau, \mathbf{w}_\tau} \sum_l D^{t-1}(l) \log \Pr_\tau(\vartheta^{(l)} | z^{(l)}; \mathbf{w}_\tau) \tag{7.20}$$

Our final result has a satisfying interpretation: *the functional gradient descent step tries to solve the maximum re-weighted log-likelihood problem (Equation 7.20) for each tree and selects the best tree with the largest re-weighted log-likelihood*. As boosting proceeds, some trees may be more likely to be selected than others, so the accumulated weights of trees may be different.

From Equation 7.18 it can be seen that after adding the learner s^t to the ensemble in Equation 7.12, the data distribution is updated as

$$\begin{aligned} D^t(l) &\propto \exp(-F_t(\vartheta^{(l)}, z^{(l)})) \\ &= \exp(-G^{t-1}(\vartheta^{(l)}, z^{(l)}) - \alpha^t s^t(\vartheta^{(l)}, z^{(l)})) \\ &= D^{t-1}(l) \exp(-\alpha^t s^t(\vartheta^{(l)}, z^{(l)})) \end{aligned} \tag{7.21}$$

This distribution must be re-normalised as

$$D^t(l) \leftarrow \frac{D^t(l)}{\sum_{l=1}^n D^t(l)} \tag{7.22}$$

Since $\alpha^t > 0$, the weight increases if $s^t = F^t - G^{t-1} < 0$. It can be interpreted that *for a given data instance l, if the new weak learner F^t is less likely than the average of previous weak learners G^{t-1}, the AdaBoost.MRF will increase the weight for that data instance*. This is different from the usual boosting behaviour where the data weight increases if the strong learner fails to correctly classify the instance. The AdaBoost.MRF seems to maximise data likelihood rather than minimise training error, and this is particularly desirable for density estimation.

7.2.3.2 Convergence property

We now provide a formal support for the convergence of the tree selection procedure in Equation 7.20.

The search direction s satisfying the condition in Equation 7.16 is called *gradient-related* to G^t (Bertsekas, 1999, p.35). We have the following convergence result (Bertsekas, 1999, Proposition 1.2.3)

Proposition 2. *Given a* Lipschitz continuity *condition on $\nabla\mathcal{L}_H$, i.e. $\|\nabla\mathcal{L}_H(G) - \nabla\mathcal{L}_H(G')\| \leq M\|G - G'\|$, for some $M > 0$, $\forall G, G' \in \mathcal{F}$, where $\mathcal{F}$ is the function space, a gradient-*

related search direction s^t, and a reasonably (positive) small step size α^t that satisfies

$$\epsilon \leq \alpha^t \leq (2-\epsilon)\frac{|\langle \nabla \mathcal{L}_H(G^{t-1}), s^t \rangle|}{M\|s^t\|^2} \tag{7.23}$$

where ϵ is a fixed positive scalar. Then

$$\lim_{t\to\infty} G^t = \arg\min_G \mathcal{L}_H(G) \tag{7.24}$$

The Lipschitz continuity condition can be satisfied in our case because $\mathcal{L}_H$ is twice differentiable, and the Hessian $\nabla^2\mathcal{L}_H$ is bounded (Bertsekas, 1999, p. 48). The constant M is hard to find analytically, so in our implementation we set the step size to a small constant $\alpha^t = 0.05$, and we have found it is sufficient in our experiments. The algorithm terminates when we cannot find any weak learner s that satisfies the condition in Equation 7.16.

7.2.3.3 Complexity

The running time of AdaBoost.MRF scales linearly in number of trees R. Recall from Section 2.4.5 that inference in trees with $|\mathcal{V}|$ nodes, $|S|$ states per node takes $\mathcal{O}(2|\mathcal{V}|S^2)$ time. If we only consider limited spanning trees, just enough to cover the whole network, then R can be quite moderate. For example, for a fully connected network we just need $R = |\mathcal{V}|$, and in a grid-like network (Figure 7.3a), $R = 2$ is enough (Figure 7.4).

7.2.4 Combining the Parameters

Up to this point we have successfully estimated the parameters of individual trees, and thus the strong learner in the boosting sense, which is sufficient for classification purposes. The prediction of output pattern x given the input z is given as

$$\hat{x} = \arg\max_{x\in\mathcal{X}} G(x, z) \tag{7.25}$$

However, our ultimate goal is to (approximately) estimate the parameters of the original network, which is a superimposition of individual trees. This subsection argues for a sensible method for such an approximate estimation.

Recall that $G(x,z) = \sum_t \alpha^t F^t(x,z)$ and $F^t(x,z) = \log \Pr_{\tau_t}(x|z)$, thus

$$G(x,z) = \sum_t \alpha^t \log \Pr_{\tau_t}(x|z) \tag{7.26}$$

Assume that the tree distribution also belongs to the exponential family, that is

$$\Pr_\tau(x|z;\mathbf{w}_\tau) = \frac{1}{Z(z;\mathbf{w}_\tau)}\exp(\mathbf{w}_\tau^\top \mathbf{F}(x,z)) \tag{7.27}$$

where $\mathbf{w}_\tau$ is the tree parameter vector and $Z(z;\mathbf{w}_\tau) = \sum_x \exp(\mathbf{w}_\tau^\top \mathbf{F}(x,z))$. Assume further that the trees share the same feature functions $\mathbf{F}(x,z)$. We require that the parts of the parameters $\mathbf{w}_\tau$, that correspond to cliques outside the trees to be zero. Thus

$$G(x,z) = \sum_t \alpha^t(\mathbf{w}_{\tau_t}^\top \mathbf{F}(x,z)) - \sum_t \alpha^t Z(z;\mathbf{w}_\tau) \tag{7.28}$$

Let

$$\mathbf{w} = \sum_t \alpha^t \mathbf{w}_{\tau_t} \tag{7.29}$$

then Equation 7.28 becomes

$$G(x,z) = \mathbf{w}^\top \mathbf{F}(x,z) - \sum_t \alpha^t Z(z;\mathbf{w}_\tau) \tag{7.30}$$

Combining this with Equation 7.25 leads to

$$\hat{x} = \arg\max_{x\in\mathcal{X}} \mathbf{w}^\top \mathbf{F}(x,z) \tag{7.31}$$

Obviously we want $\hat{x}$ to be the MAP assignment of the the original network, that is $\hat{x} = \arg\max_x \Pr(x|z)$. One reasonable way is to assume that $\Pr(x|z)$ is parameterised by the exponential family with parameter $\mathbf{w}$ and feature set $\mathbf{F}(x,z)$. The network distribution can be written in terms of component tree distributions as

$$\begin{aligned}
\Pr(x|z) &\propto \exp\left\{\langle \sum_t \alpha^t \mathbf{w}_{\tau_t}, \mathbf{F}(x,z)\rangle\right\} && (7.32)\\
&= \prod_t \exp\left\{\alpha^t \langle \mathbf{w}_{\tau_t}, \mathbf{F}(x,z)\rangle\right\} && (7.33)\\
&\propto \prod_t \Pr_{\tau_t}(x|z)^{\alpha^t} && (7.34)
\end{aligned}$$

As $\Pr(x|z)$ is a distribution, we have

$$\Pr(x|z) = \frac{\prod_t \Pr_{\tau_t}(x|z)^{\alpha^t}}{\sum_x \prod_t \Pr_{\tau_t}(x|z)^{\alpha^t}} \tag{7.35}$$

Thus, the combined model is a Logarithmic Opinion Pool (LogOP) (Heskes, 1998; Pennock and Wellman, 1999). Each model $\Pr_{\tau_t}(x|z)$ is an 'expert' providing an estimate of the true distribution $Q(x|z)$. The aggregator $\Pr(x|z)$ is indeed a minimiser of the weighted sum

of Kullback-Leibler divergences between the $Q(x|z)$ and each $\Pr_{\tau_t}(x|z)$ (Heskes, 1998)

$$\Pr(x|z) = \arg\min_{Q(x|z)} \sum_t \alpha^t \sum_x Q(x|z) \log \frac{Q(x|z)}{\Pr_{\tau_t}(x|z)} \tag{7.36}$$

The work of (Heskes, 1998) shows that $\Pr(x|z)$ is closer to the true distribution $Q(x|z)$ than the average of all individual experts $\Pr_{\tau_t}(x|z)$. Our boosting algorithm can be seen as an estimator of the weighting factors $\{\alpha^t\}$.

The AdaBoost.MRF is summarised in Figure 7.2.

Input: $l = 1, 2, ..., n$ data pairs, graphs $\{\mathcal{G}^{(l)} = (\mathcal{V}^{(l)}, \mathcal{E}^{(l)})\}$
Output: parameter vector $\mathbf{w}$
Begin
  Select spanning trees for each data instance
  Initialise $\{D_{l,0} = 1/n\}$, and $\alpha^1 = 1$
  For each boosting round $t = 1, 2, \ldots$
    Train all trees given weighted data $\{D^{t-1}(l)\}$
    */*Select the best tree distribution*/*
    $(\tau_t, \mathbf{w}_{\tau_t}) = \arg\max_{\tau, \mathbf{w}_\tau} \sum_l D^{t-1}(l) \log \Pr_\tau(\vartheta^{(l)}|z^{(l)}; \mathbf{w}_\tau)$
    $F^t = \log \Pr_{\tau_t}(\vartheta^{(l)}|z^{(l)})$
    $s^t = F^t - G^{t-1}$
    If $\sum_l D^{t-1}(l) s^t(\vartheta^{(l)}, z^{(l)}) \leq 0$ **Then** go to Output
    If $t > 1$ **Then** select the step size $0 < \alpha^t < 1$
    */*Update the strong learner*/*
    $G^t = (1 - \alpha^t) G^{t-1} + \alpha^t F^t$
    */*Scale down the previous learner weights*/*
    $\alpha^j \leftarrow \alpha^j (1 - \alpha^t)$, for $j = 1, ..., t-1$
    */*Update the data weight*/*
    $D^t(l) \leftarrow D^{t-1}(l) \exp(-\alpha^t s^t(\vartheta^{(l)}, z^{(l)}))$
    $D^t(l) \leftarrow \frac{D^t(l)}{\sum_{l=1}^n D^t(l)}$
  End
  Output $\mathbf{w} = \sum_t \alpha^t \mathbf{w}_{\tau_t}$
End

Figure 7.2: AdaBoost.MRF - AdaBoosted Markov Random Forests.

Pennock and Wellman (1999) offer an interesting discussion on the relation between Markov networks, the LogOP, and the properties of desirable aggregators which the LogOP satisfies. Our method is based on the idea of superimposition, or *union* of sub-networks, that is, if a node or an edge belongs to the aggregated network it must belong to one of the individual sub-networks. In (Smith *et al.*, 2005) the authors consider the combination of different models but they share the same underlying simple chain structure. Models are trained independently and then combined using the LogOP. The model weights $\{\alpha^t\}$ are then estimated by maximising the likelihood of the combined models. This approach is fine as long as the underlying structure is tractable.

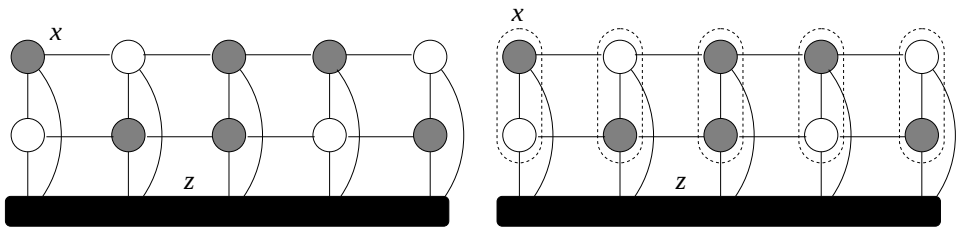
Another related idea is the the product-of-experts (Hinton, 2002), where all weights are unity. In (Hinton, 2002) sampling is used to overcome the intractability, which may not converge within a limited time. By contrast, our method is efficient as it deals directly with trees.

7.2.5 AdaBoost.MRF as Guided Search for MLE

As we rely on the boosting capacity to boost weak learners to a strong one, we do not need to reach the maximum of the weighted log-likelihood in each round. We can simply run a few training iterations and take the partial results as long as the condition in Equation 7.16 is met. To speedup the learning, we can initialise the parameters for each weak learner to the previously learned values.

This procedure has an interesting interpretation for tree-structured networks. As we do not have to select the best spanning trees anymore, the algorithm simply optimises the re-weighted log-likelihood in a stage-wise manner. We argue that this approach can be attractive because more information from the data distribution can be used to guide the MLE, and it can create more diverse weak classifiers.

7.3 Evaluation



(a) FCRF (b) Collapsed FCRF

Figure 7.3: Factorial CRF with missing labels (a), and the collapsed version into a chain (b). Filled circles and bars are data observations, empty circles are hidden labels, shaded labels are the visible.

We evaluate the AdaBoost.MRF on the same home video surveillance dataset described in Chapter 6. Recall that the data is hierarchical, in that the complex human activities are composed of primitive activities. However, this property was not considered in Chapter 6 as we did not have efficient tools for learning more complex structures than chains and trees. In this chapter we model each data sequence as a grid (see Figure 7.3). In other words we build a two level Factorial CRF (FCRF) (Sutton *et al.*, 2007). The bottom level represents all 12 primitive activities and the top level 3 complex activities. Note that the

setting of the bottom level in this chapter is different from that in Chapter 6 in the sense that the state space is the union of sub-state spaces considered in Chapter 6.

Differing from the original setting of the FCRF in (Sutton *et al.*, 2007), we allow some missing labels in training data. Specifically, we randomly provide half the labels for each level. For testing, the MAP assignments resulted from Pearl's loopy max-product algorithm are compared against the ground-truth.

In Figure 7.3 circles represent state variables (corresponding to labels) and the bottom filled bar is the whole observation sequence (the sequence of coordinates in this case). Empty circles represent missing labels.

Since the model hierarchy is not deep, exact estimation of marginals can be carried out by collapsing all the states at the current time into a mega-state (see Figure 7.3b) and performing a *forward-backward* procedure. Approximate inference using the BP (see Section 2.4.5) and a BP-variant by Wainwright, Jaakkola and Willsky (WJW) (Wainwright *et al.*, 2005b) methods has the complexity of $\mathcal{O}(2I|\mathcal{E}||S|^2)$, where I is the number of message passing rounds, $|\mathcal{E}|$ is the number of edges in the network, and $|S|$ is the state size per node. However, the number of rounds I is not known analytically and there has not been any theoretical estimate of it yet.

In our AdaBoost.MRF, inference in the trees takes $\mathcal{O}(2|\mathcal{V}|S^2)$ time, where $|\mathcal{V}|$ is the number of nodes in the network. Thus, for $|\mathcal{D}|$ data instances and R trees, the AdaBoost.MRF costs $\mathcal{O}(4|\mathcal{D}|R|\mathcal{V}|S^2)$ in total time for each gradient evaluation as we need to take both $\Phi(\vartheta, z)$ and $\Phi(z)$ into account. Similarly, the BP and WJW-based ML requires $\mathcal{O}(4|\mathcal{D}|I|\mathcal{E}|S^2)$ time. In fully connected networks, $|\mathcal{E}| = \frac{1}{2}|\mathcal{V}|(|\mathcal{V}|+1)$, and in grid FCRFs, $|\mathcal{E}| \approx 2|\mathcal{V}|$. If we take only $R = |\mathcal{V}|$ trees for the former fully connected networks, and $R = 2$ for the grids, the total complexity per gradient evaluation of the BP and WJW-based maximum likelihood and the AdaBoost.MRF will be similar up to a constant I. We summarise the complexities in Table 7.1.

BP/WJW	AdaBoost.MRF
$\mathcal{O}(4\|\mathcal{D}\|I\|\mathcal{E}\|S^2)$	$\mathcal{O}(4\|\mathcal{D}\|R\|\mathcal{V}\|S^2)$

Table 7.1: Complexity per gradient evaluation.

7.3.1 Feature Extraction

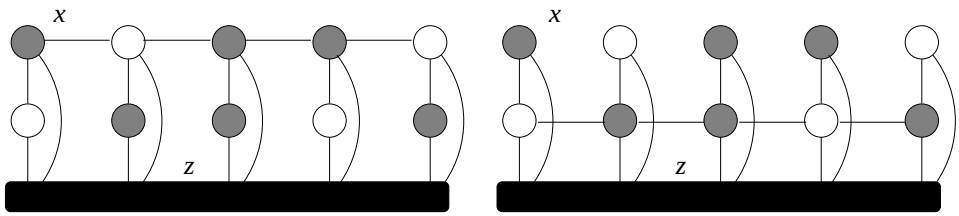
At the bottom level of the FCRFs in our study the observational feature set described in Chapter 6 is reused. At the top level, however, instant information such as velocities offers limited help since the complex activities often span long periods. Instead of using the real

coordinates (X, Y) for data association we quantize them into 24 squares in the room. We also use much larger sliding windows with $s_1 = s_2 = 20$. To avoid computational overhead we take $\epsilon = -s_1, -s_1 + 5, ..., s_2 - 5, s_2$.

There are also state features that capture the state transition between time steps at both levels, and features that encode the state emission from the parent state at the top level to the child at the bottom. For simplicity we use indicator functions for both cases.

7.3.2 Spanning Trees for AdaBoost.MRF

The AdaBoost.MRF algorithm described in Figure 7.2 requires the specification of a set of spanning trees which will be used as weak classifiers. Given the grid structure considered in this experiment there are many spanning trees that can be extracted. However, since the nature of our problem is about temporal regularities where the slice structure is repeated over time, it is natural to decompose the network into trees in a such a way that the structural repetition is maintained. With this hint there are two most noticeable trees that stand out as shown in Figure 7.4, which roughly correspond to the top and bottom chains respectively.



(a) Top process (b) Bottom process

Figure 7.4: (a,b) Two process view of the FCRF in activity modelling: (a) the complex activity, and (b) the primitive.

With the same method the number of trees for dynamic models that respect the Markov assumption is reduced drastically. If we impose further restrictions that each state can only interact with the levels right above and below it, then the number of trees can be manageable (e.g. see Figure 7.5 for another example).

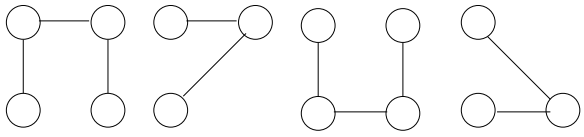


Figure 7.5: 2-slice structures of spanning trees for the FCRFs whose 2-slice structure is given in the left-most graph in Figure 7.1.

7.3.3 Segmentation and Annotation Results

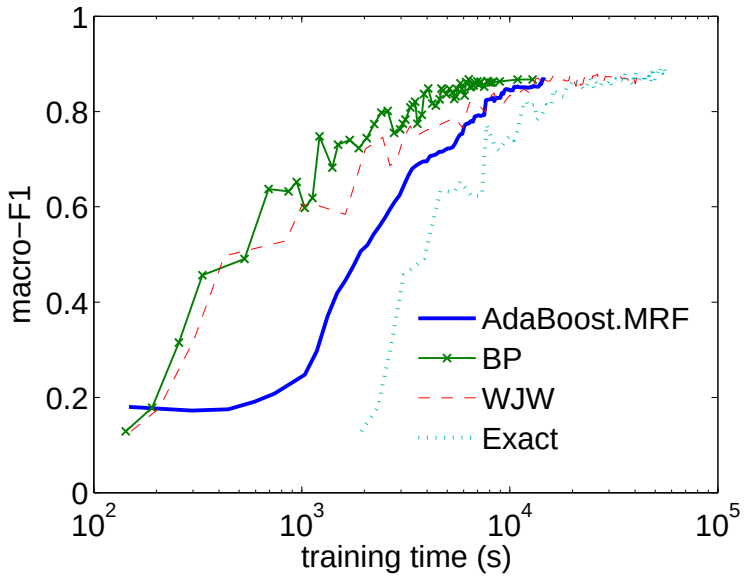


Figure 7.6: Macro-averaged F_1 scores at the bottom layer vs training time.

For segmenting and annotating data we apply Pearl's loopy max-product algorithm.

For comparison we implement ML learning methods based on BP, WJW and exact inference for FCRFs. We also evaluate the effectiveness of the FCRFs against the Layered HMMs (LHMMs) (Oliver *et al.*, 2004), where the output of the bottom HMM is used as the input for the top HMM. Since, it is difficult to encode rich feature information in the LHMMs without producing very large state spaces, we limit the LHMMs features to be the discretised positions, and the differences between current position and the previous and next ones. Our new implementation of LHMMs differs from the original in (Oliver *et al.*, 2004) for each HMM has been extended to handle the partially observed states. All learning algorithms are initialised uniformly. For segmentation purposes we report the macro-averaged F_1 scores on a per-label basis.

For parameter optimisation of the (re-weighted) log-likelihood, initially we used the limited memory quasi-Newton method (L-BFGS) as suggested in the CRF literature, but it seems to be slower and it converges prematurely to poor solutions for the BP and the exact inference. The conjugate-gradient (CG) method works better in our experiments. For the Markov forests we run only two iterations of CG per boosting round with the initial parameters from the previously learned ones as we only need to meet the condition in Equation 7.16. The WJW inference loop is stopped if the messages have converged at the rate of 10^{-4} or after 100 rounds. It appears that the final performance of BP is sensitive to the choice of convergence rates, while it is fairly stable for the WJW. For example, the F_1 scores at the bottom level for BP are $0.84, 0.87$ and 0.82 corresponding to the rates of

Table 7.2: Macro-averaged F_1 scores for top and bottom layers.

Algorithm	Top-layer	Bottom-layer
AdaBoost.MRF	0.98	0.87
BP	0.99	0.87
WJW	0.98	0.87
Exact	0.98	0.88
LHMM	0.88	0.67

$10^{-3}, 10^{-4}$ and 10^{-5}, respectively. Below we report only the case of 10^{-4}, which appears to be the best both in terms of accuracy and speed. Learning algorithms for the FCRFs are stopped after 100 iterations if they have not converged at the rate of 10^{-5}.

The performance of the AdaBoost.MRF and its alternatives is reported in Figure 7.6 and Table 7.2, respectively. Overall, after enough training time, the AdaBoost.MRF performs comparably with the ML methods based on BP and WJW. The exact inference ML method gives slightly better results as expected, but at the cost of much slower training time. However, it should be stressed that inference in our AdaBoost.MRF always converges, while it is not guaranteed in the BP and WJW and it is generally intractable in the exact method. The complexity per evaluation of the log-likelihood gradient is known and fixed for the AdaBoost.MRF, while for the BP and the WJW, it is generally dependent on the convergence criteria and how much the distribution is different from uniform (see Table 7.1).

Figure 7.7 shows the AdaBoost.MRF segmentation details of 22 randomly selected sequences which are concatenated together.

7.4 Closing Remarks

We have presented a novel method for using boosting in parameter estimation of the general CRFs with hidden variables. The algorithm AdaBoost.MRF offers an efficient way to tackle the intractability of the maximum likelihood method by breaking the model into tractable trees and combining them to recover the original networks. We apply the algorithm to learn the FCRF for the problem of multilevel activity recognition and segmentation.

As shown in our experiments, it appears that the AdaBoost.MRF exhibits a structure learning behaviour since it may selectively pick some trees more frequently than others, giving higher weights to those trees. An important issue we have left unanswered is that how to automatically select the optimal tree at each round without knowing the set of trees in advance.

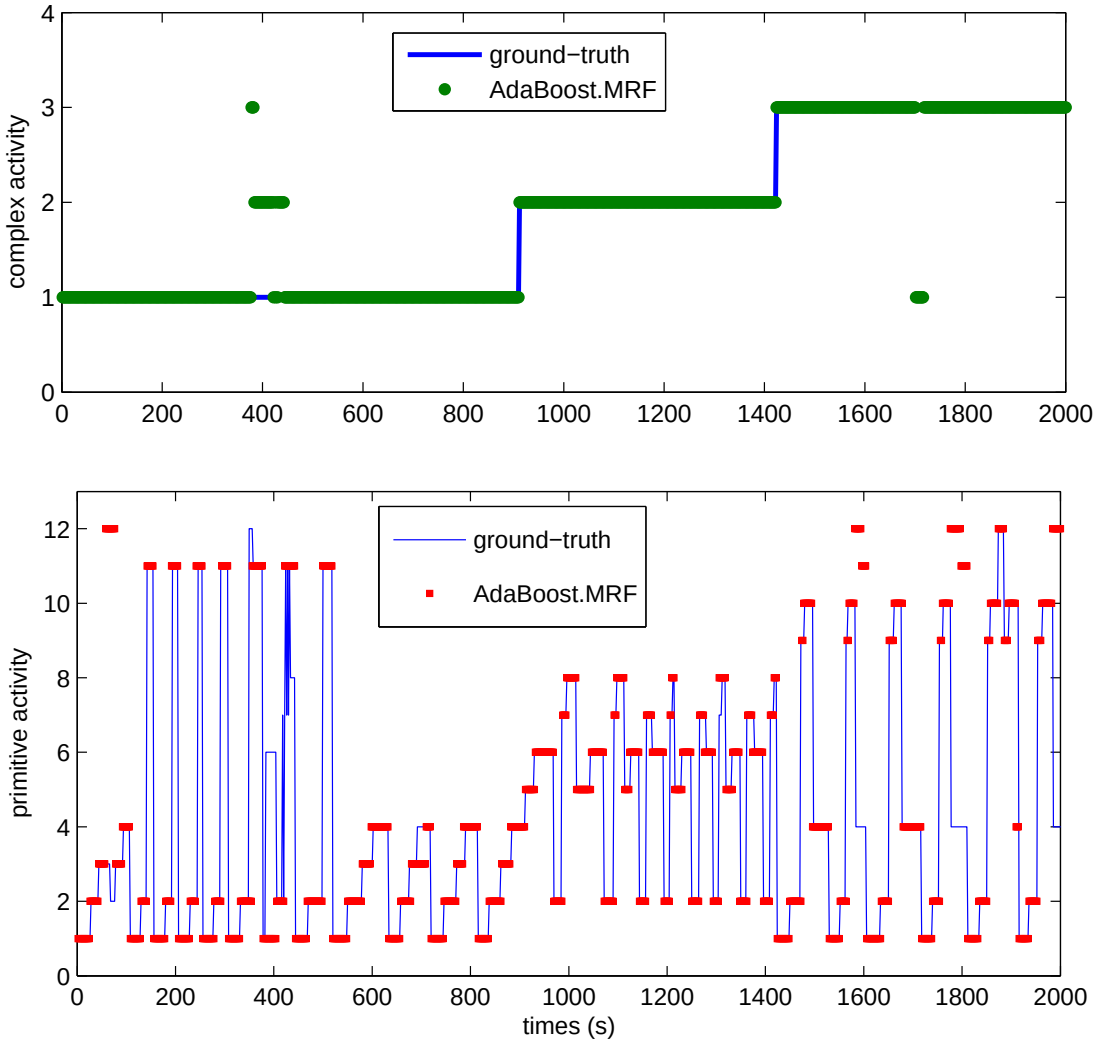


Figure 7.7: The segmentation compared with the ground-truth at top (top graph) and bottom levels (bottom graph).

There have been no exact methods (the AdaBoost.MRF is still an approximate method) that can perform inference and learning on arbitrary multilevel data. On the other hand there are classes of multilevel temporal data that are strictly nested, in the sense that the life span of the higher level semantics exclusively contains the life span of the lower ones. This constraint may give rise to more efficient inference and learning. We will investigate this issue in the next two chapters.

Chapter 8

Hierarchical Conditional Random Fields for Recursive Sequential Data

8.1 Introduction

In the previous two chapters we have investigated two aspects associated with general CRFs: feature selection and efficient learning with general structures. In this chapter, we turn our attention to the third aspect, *hierarchical data modelling*.

Hierarchies are indeed a natural property of many domains in that each level is an *abstraction* of lower level details. We have seen in the previous chapter that high level human activities may include sub-activities at more primitive levels. In vision, objects are composed of parts, which in turn are a combination of visual cues such as edges, dots and textures. Similarly, in natural language processing (NLP) syntax trees are inherently hierarchical. For example, in the partial parsing task known as noun-phrase (NP) chunking (Sang and Buchholz, 2000), there are four levels: the sentence, noun-phrases, part-of-speech (POS) tags and unigrams. In this setting, the sentence is a sequence of NPs and non-NPs, each phrase is a sub-sequence of POS tags, and finally each POS tag possible consists of one unigram (as in English) or more (as in Chinese and Vietnamese).

A popular approach to deal with hierarchical data is to build a cascaded model: each level is modelled separately, and the output of the lower level is used as the input of the level right above it (e.g. see (Oliver *et al.*, 2004)). For instance, in NP chunking this approach first builds a POS tagger and then constructs a chunker that incorporates the output of the tagger. This approach is clearly sub-optimal because the POS tagger takes no information of the NPs and the chunker is not aware of the reasoning of the tagger. In contrast, a noun-phrase is often very informative to infer the POS tags belonging to the phrase. As a result,

this layered approach often suffers from the so-called *cascading error* problem as the error introduced from the lower layer will propagate to higher levels.

A more holistic approach is to build a joint representation of all the levels. However, complex models are likely to suffer inference intractability. There must be appropriate constraints that allow efficient inference. Fortunately there exists a class of hierarchical models that satisfy both the requirements of joint representation and efficiency. More specifically, the models are recursive and sequential, in that each level is a sequence and each node in a sequence can be decomposed further into a sub-sequence of finer grain at the lower level.

There has been substantial investigation of these types of model, especially in the area of probabilistic context-free grammars (e.g. see (Manning and Schütze, 1999, Chapter 11)). However, grammars are often unbounded in depth and thus difficult to represent by graphical models. A more restricted version known as hierarchical hidden Markov model (HHMM) (Fine *et al.*, 1998) offers clearer representation in that the depth is fixed and the semantic levels are well defined. It can also be represented as a Dynamic Bayesian Network (DBN) (Murphy and Paskin, 2002). Essentially, the HHMM is a nested HMM in the sense that each state is a sub HMM by itself.

In this chapter we follow a similar route to generalise chain-structured CRFs to nested CRFs. As a result, we propose a novel model called *Hierarchical Conditional Random Field* (HCRF), which is an undirected conditional graphical model of nested Markov chains. Thus HCRF is the combination of the discriminative nature of CRFs and the nested modelling of the HHMM. To be more concrete let us return to the Noun-Phrase chunking example. The problem can be modelled as a three-level HCRF, where the root represents the sentence, the second level the NP process, and the bottom level the POS process. The root and the two processes are conditioned on the sequence of words in the sentence. Under the discriminative modelling of the HCRF, rich contextual information such as starting and ending of the phrase, the phrase length, and the distribution of words falling inside the phrase can be effectively encoded. On the other hand, such encoding is much more difficult for HHMMs.

For learning and inference we derive an efficient algorithm based on the Asymmetric Inside-Outside (AIO) of (Bui *et al.*, 2004) that exhibits cubic time complexity. We also develop a generalised Viterbi algorithm for decoding the optimal state assignment for a given observational sequence.

Notations and Chapter Organisation

This chapter introduces a number of new mathematical notations which we include in Table 8.1 for reference.

Notation	Description
$x_{i:j}^{d:d'}$	Subset of state variables from level d down to level d' and starting from time i and ending at time j, inclusive.
$e_{i:j}^{d:d'}$	Subset of ending indicators from level d down to level d' and starting from time i and ending at time j, inclusive.
$\zeta_{i:j}^{d,s}$	Set of state variables and ending indicators of a sub model rooted at s^d, level d, spanning a sub-string $[i, j]$
σ	Contextual clique
i, j, t	Time indices
τ^d	Set of all ending time indices, e.g. if $i \in \tau^d$ then $e_i^d = 1$
r, s, u, v, w	State
$R_{i:j}^{d,s,z}$	State-persistence potential of state s, level d, spanning $[i, j]$
$\pi_{u,i}^{d,s}$	Initialisation potential of state s at level d, time i initialising sub-state u
$A_{u,v,i}^{d,s,z}$	Transition at level d, time i from state u to v under the same parent s
$E_{u,i}^{d,s,z}$	Ending potential of state z at level d and time i, and receiving the return control from the child u
$\Phi[\zeta, z]$	The global potential of a particular configuration ζ
S^d	The number of state symbols at level d
$\Delta_{i:j}^{d,s}$	The symmetric inside mass for a state s at level d, spanning a substring $[i, j]$
$\hat{\Delta}_{i:j}^{d,s}$	The full symmetric inside mass for a state s at level d, spanning a substring $[i, j]$
$\Lambda_{i:j}^{d,s}$	The symmetric outside mass for a state s at level d, spanning a substring $[i, j]$
$\hat{\Lambda}_{i:j}^{d,s}$	The full symmetric outside mass for a state s at level d, spanning a substring $[i, j]$
$\alpha_{i:j}^{d,s}(u)$	The asymmetric inside mass for a parent state s at level d, starting at i and having a child-state u which returns control to parent or transits to new child-state at j
$\lambda_{i:j}^{d,s}(u)$	The asymmetric outside mass, as a counterpart of asymmetric inside mass $\alpha_{i:j}^{d,s}(u)$
$\psi(.), \varphi(.)$	Potential functions.

Table 8.1: Notations used in this chapter.

The rest of the chapter continues with the HCRF model definition and parameterisation in Section 8.2. Section 8.3 defines building blocks required for common inference tasks. These blocks are computed in Section 8.3.2 and 8.3.3. Parameter estimation follows in Section 8.4. Section 8.5 presents the generalised Viterbi algorithm. We analyse the complexity of the AIO algorithm in Section 8.6 and conclude the chapter in Section 8.7.

8.2 Model Definition

Consider a hierarchically nested Markov process with D levels. Then as in the HHMMs (Fine *et al.*, 1998), the parent state embeds a child Markov chain whose states may in turn contain child Markov chains. The family relation is defined in the *model topology*, which is a state hierarchy of depth D. The model has a set of states S^d at each level $d \in [1, D]$, i.e. $S^d = \{1...|S^d|\}$, where $|S^d|$ is the number of states at level d. For each state $s^d \in S^d$ where $1 \leq d < D$, the topological structure also defines a set of children $ch(s^d) \subset S^{d+1}$. Conversely, each child s^{d+1} has a set of parents $pa(s^{d+1}) \subset S^d$. Unlike the original HHMMs where the child states belong exclusively to the parent, the HCRFs allow arbitrary sharing of children between parents. For example, in Figure 8.1, $ch(s^1 = 1) = \{1, 2, 3\}$, and $pa(s^3 = 1) = \{1, 2, 4\}$. This helps to avoid an explosive number of sub-states when D is large, leading to fewer parameters and possibly less training data and time. The shared topology has been investigated in the context of HHMMs in (Bui *et al.*, 2004).

The temporal evolution in the nested Markov processes with sequence length of T operates as follows:

- As soon as a state is created at level $d < D$, it *initialises* a child state at level $d + 1$. The initialisation continues downward until reaching the bottom level[1].
- As soon as a child process at level $d + 1$ *ends*, it returns control to its parent at level d, and in the case of $d > 1$, the parent either *transits* to a new parent state or returns to the grand-parent at level $d - 1$.

The main requirement for the hierarchical nesting is that the life span of the child process belongs exclusively to the life span of the parent. For example, consider a parent process at level d starts a new state $s^d_{i:j}$ at time i and persists until time j. At time i the parent initialises a child state s^{d+1}_i which continues until it ends at time $k < j$, at which the child state transits to a new child state s^{d+1}_{k+1}. The child process exits at time j, at which the control from the child level is returned to the parent $s^d_{i:j}$. Upon receiving the control the parent state $s^d_{i:j}$ may transit to a new parent state $s^d_{j+1:l}$, or end at j, returning the control to the grand-parent at level $d - 1$.

We are now in a position to specify the nested Markov processes in a more formal way. Let us introduce a multi-level temporal graphical model of length T with D levels, starting from the top as 1 and the bottom as D (Figure 8.2). At each level $d \in [1, D]$ and time index

[1] In HHMMs, the bottom level is also called *production* level, in which the states emit observational symbols. In HCRFs, this generative process is not assumed.

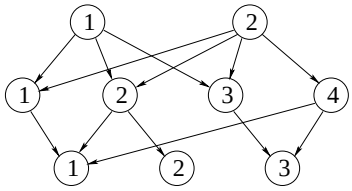


Figure 8.1: The shared topological structure.

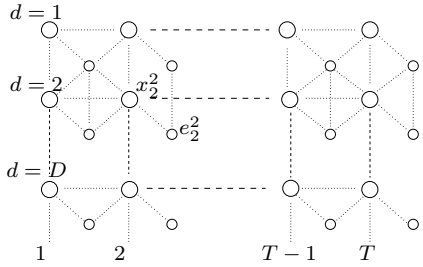


Figure 8.2: The multi-level temporal model.

$i \in [1, T]$, there is a node representing a state variable $x_i^d \in S^d = \{1, 2, ..., |S^d|\}$. Associated with each x_i^d is an ending indicator e_i^d which can be either 1 or 0 to signify whether the state x_i^d ends or persists at i. The nesting nature of the HCRFs is now realised by imposing the specific constraints on the value assignment of ending indicators (Figure 8.3).

- The top state persists during the course of evolution, i.e. $e_{1:T-1}^1 = 0$, $e_T^1 = 1$.
- When a state finishes, all of its descendants must also finish, i.e. $e_i^d = 1$ implies $e_i^{d+1:D} = 1$.
- When a state persists, all of its ancestors must also persist, i.e. $e_i^d = 0$ implies $e_i^{1:d-1} = 0$.
- When a state transits, its parent must remain unchanged, i.e. $e_i^d = 1$, $e_i^{d-1} = 0$.
- The bottom states do not persists, i.e. $e_i^D = 1$ for all $i \in [1, T]$.
- All states end at T, i.e. $e_T^{1:D} = 1$.

Figure 8.3: Hierarchical constraints.

Thus, specific value assignments of ending indicators provide *contexts* that realise the evolution of the model states in both hierarchical (vertical) and temporal (horizontal) directions. Each context at a level and associated state variables form a *contextual clique*, and we identify four contextual clique types:

- *State-persistence* : This corresponds to the life time of a state at a given level (see Figure 8.4). Specifically, given a context $c = (e_{i-1:j}^d = (1, 0, .., 0, 1))$, then $\sigma_{i:j}^{persist,d} = (x_{i:j}^d, c)$, is a contextual clique that specifies the life span $[i, j]$ of any state $s = x_{i:j}^d$.
- *State-transition* : This corresponds to a state at level $d \in [2, D]$ at time i transiting to a new state (see Figure 8.5a). Specifically, given a context $c = (e_i^{d-1} = 0, e_i^d = 1)$

then $\sigma_i^{transit,d} = (x_{i+1}^{d-1}, x_{i:i+1}^{d}, c)$ is a contextual clique that specifies the transition of x_i^d to x_{i+1}^d at time i under the same parent x_{i+1}^{d-1}.

- *State-initialisation* : This corresponds to a state at level $d \in [1, D-1]$ initialising a new child state at level $d+1$ at time i (see Figure 8.5b). Specifically, given a context $c = (e_{i-1}^d = 1)$, then $\sigma_i^{init,d} = (x_i^d, x_i^{d+1}, c)$ is a contextual clique that specifies the initialisation at time i from the parent x_i^d to the child x_i^{d+1}.

- *State-ending* : This corresponds to a state at level $d \in [1, D-1]$ to end at time i (see Figure 8.5c). Specifically, given a context $c = (e_i^d = 1)$, then $\sigma_i^{end,d} = (x_i^d, x_i^{d+1}, c)$ is a contextual clique that specifies the ending of x_i^d at time i with the last child x_i^{d+1}.

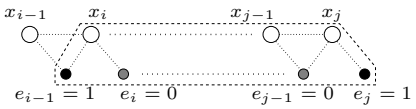


Figure 8.4: An example of a state-persistence sub-graph.

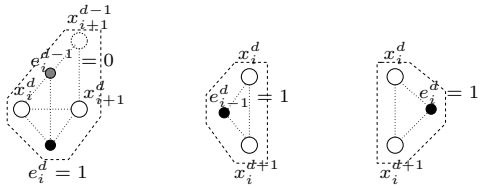


Figure 8.5: Sub-graphs for state transition (left), initialisation (middle) and ending (right).

In the HCRF we are interested in the *conditional* setting in which the entire state variables $\{x_{1:T}^{1:D}, e_{1:T}^{1:D}\}$ are conditioned on observational sequences z. For example, in computational linguistics, the observation is often the sequence of words and the state variables might be the part-of-speech tags and the phrases.

To capture the correlation between variables and such conditioning, we define a non-negative potential function $\psi(\sigma, z)$ over each contextual clique σ. Figure 8.6 shows the notations for potentials that correspond to the four contextual clique types we have identified above. Details of potential specification are described in the Section 8.4.1.

- $R_{i:j}^{d,s,z} = \psi(\sigma_{i:j}^{persist,d}, z)$ where $s = x_{i:j}^d$.
- $A_{u,v,i}^{d,s,z} = \psi(\sigma_i^{transit,d}, z)$ where $s = x_{i+1}^{d-1}$ and $u = x_i^d, v = x_{i+1}^d$.
- $\pi_{u,i}^{d,s,z} = \psi(\sigma_i^{init,d}, z)$ where $s = x_i^d, u = x_i^{d+1}$.
- $E_{u,i}^{d,s,z} = \psi(\sigma_i^{end,d}, z)$ where $s = x_i^d, u = x_i^{d+1}$.

Figure 8.6: Shorthands for contextual clique potentials.

Let $\zeta = (x_{1:T}^{1:D}, e_{1:T}^{1:D})$ denote the set of all variables that satisfies the set of hierarchical constraints in Figure 8.3. Let τ^d denote ordered set of all ending time indices at level d,

e.g. if $i \in \tau^d$ then $e_i^d = 1$. The joint potential defined for each configuration is the product of all contextual clique potentials over all ending time indices $i \in [1, T]$ and all semantic levels $d \in [1, D]$:

$$\begin{aligned}\Phi[\zeta, z] &= \left[\prod_{d\in[1,D]} \prod_{i_k, i_{k+1}\in\tau^d} R^{d,s,z}_{i_k+1:i_{k+1}}\right] \times \\ &\times \prod_{d\in[1,D-1]} \left\{\left[\prod_{i_k\in\tau^{d+1}, i_k\notin\tau^d} A^{d+1,s,z}_{u,v,i_k}\right]\left[\prod_{i_k\in\tau^{d+1}} \pi^{d,s,z}_{u,i_k+1}\right]\left[\prod_{i_k\in\tau^{d+1}} E^{d,s,z}_{u,i_k}\right]\right\} \quad (8.1)\end{aligned}$$

The conditional distribution is given as

$$\Pr(\zeta|z) = \frac{1}{Z(z)}\Phi[\zeta, z] \quad (8.2)$$

where $Z(z) = \sum_\zeta \Phi[\zeta, z]$ is the partition function for normalisation.

In what follows we omit z for clarity, and implicitly use it as part of the partition function Z and the potential $\Phi[.]$. It should be noted that in the unconditional formulation, there is only a single Z for all data instances. In conditional setting there is a $Z(z)$ for each data instance z.

Remarks: The temporal model of HCRFs presented here is not a standard graphical model (Lauritzen, 1996) since the connectivity (and therefore the clique structures) is not fixed. The potentials are defined on-the-fly depending on the context of assignments of ending indicators. Although the model topology is identical to that of shared structure HHMMs (Bui *et al.*, 2004), the unrolled temporal representation is an undirected graph and the model distribution is formulated in a discriminative way. Furthermore, the state persistence potentials capture duration information that is not available in the dynamic DBN representation of the HHMMs in (Murphy and Paskin, 2002).

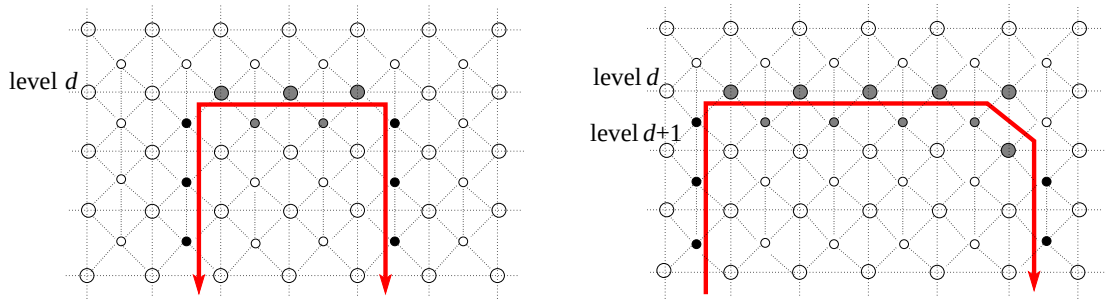
In the way the potentials are introduced it may first appear to resemble the clique templates in the discriminative relational Markov networks (RMNs) (Taskar *et al.*, 2002). It is, however, different because cliques in the HCRFs are dynamic and context-dependent.

8.3 Asymmetric Inside-Outside Algorithm

This section describes a core inference engine called Asymmetric Inside-Outside (AIO) algorithm, which is partly adapted from the generative, directed counter part of HHMMs in (Bui *et al.*, 2004). We now show how to compute the building blocks that are needed in

most inference and learning tasks.

8.3.1 Building Blocks and Conditional Independence



(a) (b)

Figure 8.7: (a) Symmetric Markov blanket, and (b) Asymmetric Markov blanket.

8.3.1.1 Contextual Markov blankets

In this subsection we define elements that are building blocks for inference and learning. These building blocks are identified given the corresponding *boundaries*. Let us introduce two types of boundaries: the contextual *symmetric* and *asymmetric Markov blankets*.

Definition 2. *A symmetric Markov blanket at level d for a state s starting at i and ending at j is the following set*

$$\Pi_{i:j}^{d,s} = (x_{i:j}^{d} = s, e_{i-1}^{d:D} = 1, e_{j}^{d:D} = 1, e_{i:j-1}^{d} = 0) \tag{8.3}$$

Definition 3. *Let $\Pi_{i:j}^{d,s}$ be a symmetric Markov blanket, we define $\zeta_{i:j}^{d,s}$ and $\underline{\zeta}_{i:j}^{d,s}$ as follows*

$$\zeta_{i:j}^{d,s} = (x_{i:j}^{d+1:D}, e_{i:j-1}^{d+1:D}) \tag{8.4}$$

$$\underline{\zeta}_{i:j}^{d,s} = \zeta \backslash (\zeta_{i:j}^{d,s}, \Pi_{i:j}^{d,s}) \tag{8.5}$$

subject to $x_{i:j}^{d} = s$. Further, we define

$$\hat{\zeta}_{i:j}^{d,s} = (\zeta_{i:j}^{d,s}, \Pi_{i:j}^{d,s}) \tag{8.6}$$

$$\hat{\underline{\zeta}}_{i:j}^{d,s} = (\underline{\zeta}_{i:j}^{d,s}, \Pi_{i:j}^{d,s}) \tag{8.7}$$

Figure 8.7a shows an example of a symmetric Markov blanket (represented by a double-arrowed line).

Definition 4. *A asymmetric Markov blanket at level d for a parent state s starting at i and a child state u ending at j is the following set*

$$\Gamma^{d,s}_{i:j}(u) = (x^d_{i:j} = s, x^{d+1}_j = u, e^{d:D}_{i-1} = 1, e^{d+1:D}_j = 1, e^d_{i:j-1} = 0) \tag{8.8}$$

Definition 5. *Let $\Gamma^{d,s}_{i:j}(u)$ be an asymmetric Markov blanket, we define $\zeta^{d,s}_{i:j}(u)$ and $\underline{\zeta}^{d,s}_{i:j}(u)$ as follows*

$$\zeta^{d,s}_{i:j}(u) = (x^{d+1:D}_{i:j-1}, x^{d+2:D}_j, e^{d+1:D}_{i:j-1}) \tag{8.9}$$
$$\underline{\zeta}^{d,s}_{i:j}(u) = \zeta \backslash (\zeta^{d,s}_{i:j}(u), \Gamma^{d,s}_{i:j}(u)) \tag{8.10}$$

subject to $x^d_{i:j} = s$ and $x^{d+1}_j = u$. Further, we define

$$\hat{\zeta}^{d,s}_{i:j}(u) = (\zeta^{d,s}_{i:j}(u), \Gamma^{d,s}_{i:j}(u)) \tag{8.11}$$
$$\hat{\underline{\zeta}}^{d,s}_{i:j}(u) = (\underline{\zeta}^{d,s}_{i:j}(u), \Gamma^{d,s}_{i:j}(u)) \tag{8.12}$$

Figure 8.7b shows an example of asymmetric Markov blanket (represented by an arrowed line).

Remark: The concepts of contextual Markov blankets (or Markov blankets for short) are different from those in traditional Markov random fields and Bayesian networks because they are specific assignments of a subset of variables, rather than a collection of variables.

8.3.1.2 Conditional independence

Recall that conditional independence refers to the situation in which two subsets of variables A and B are independent given the the subset C. Generally, C consists of separating variables that block any paths between A and B. If C is also the boundary of A, for example, then the C is a Markov blanket of A. Given the separating boundary we can safely ignore any variables outside the boundary. This often greatly simplifies computation.

As our the *symmetric* and *asymmetric Markov blankets* are the boundaries, we have important conditional independence occurrences, which are summarised in Propositions 3 and 4.

Proposition 3. *$\zeta^{d,s}_{i:j}$ and $\underline{\zeta}^{d,s}_{i:j}$ are conditionally independent given $\Pi^{d,s}_{i:j}$*

$$\Pr(\zeta^{d,s}_{i:j}, \underline{\zeta}^{d,s}_{i:j} | \Pi^{d,s}_{i:j}) = \Pr(\zeta^{d,s}_{i:j} | \Pi^{d,s}_{i:j}) \Pr(\underline{\zeta}^{d,s}_{i:j} | \Pi^{d,s}_{i:j}) \tag{8.13}$$

In words, Propositions 3 says that variables falling inside and outside the symmetric Markov

blanket $\Pi_{i:j}^{d,s}$ are conditionally independent given $\Pi_{i:j}^{d,s}$. This proposition gives rise to the following factorisation

$$\Pr(\zeta) = \Pr(\Pi_{i:j}^{d,s}) \Pr(\zeta_{i:j}^{d,s}, \underline{\zeta}_{i:j}^{d,s} | \Pi_{i:j}^{d,s}) = \Pr(\Pi_{i:j}^{d,s}) \Pr(\zeta_{i:j}^{d,s} | \Pi_{i:j}^{d,s}) \Pr(\underline{\zeta}_{i:j}^{d,s} | \Pi_{i:j}^{d,s}) \tag{8.14}$$

Although in the following development, we will not use this factorisation directly, it does offer some insight how we should proceed in computing $\Pr(\zeta)$ and the partition function. Here, we at each step, we can work with the separate components $\Pr(\zeta_{i:j}^{d,s} | \Pi_{i:j}^{d,s})$ and $\Pr(\underline{\zeta}_{i:j}^{d,s} | \Pi_{i:j}^{d,s})$ as functions of the Markov blanket. Thus we can avoid dealing with all variables ζ at the same time. Since we do not know $\Pi_{i:j}^{d,s}$ for sure, we have to examine all possible enumerations, which are about $\frac{1}{2}|S|T^2$ for each level.

We also have similar argument and insight for the asymmetric Markov blankets.

Proposition 4. *$\zeta_{i:j}^{d,s}(u)$ and $\underline{\zeta}_{i:j}^{d,s}(u)$ are conditionally independent given $\Gamma_{i:j}^{d,s}(u)$*

$$\Pr(\zeta_{i:j}^{d,s}(u), \underline{\zeta}_{i:j}^{d,s}(u) | \Gamma_{i:j}^{d,s}(u)) = \Pr(\zeta_{i:j}^{d,s}(u) | \Gamma_{i:j}^{d,s}(u)) \Pr(\underline{\zeta}_{i:j}^{d,s}(u) | \Gamma_{i:j}^{d,s}(u)) \tag{8.15}$$

The following factorisation is a consequence of Proposition 4

$$\begin{aligned} \Pr(\zeta) &= \Pr(\Gamma_{i:j}^{d,s}(u)) \Pr(\zeta_{i:j}^{d,s}(u), \underline{\zeta}_{i:j}^{d,s}(u) | \Gamma_{i:j}^{d,s}(u)) \\ &= \Pr(\Gamma_{i:j}^{d,s}(u)) \Pr(\zeta_{i:j}^{d,s}(u) | \Gamma_{i:j}^{d,s}(u)) \Pr(\underline{\zeta}_{i:j}^{d,s}(u) | \Gamma_{i:j}^{d,s}(u)) \end{aligned} \tag{8.16}$$

The proof of Propositions 3 and 4 is given in Appendix A.3.1.

8.3.1.3 Symmetric Inside/Outside Masses

From Equation 8.5 we have $\zeta = (\zeta_{i:j}^{d,s}, \Pi_{i:j}^{d,s}, \underline{\zeta}_{i:j}^{d,s})$. Since $\Pi_{i:j}^{d,s}$ separates $\zeta_{i:j}^{d,s}$ from $\underline{\zeta}_{i:j}^{d,s}$, we can group local potentials in Equation 8.1 into three parts: $\Phi[\hat{\zeta}_{i:j}^{d,s}[$, $\Phi[\hat{\underline{\zeta}}_{i:j}^{d,s}[$, and $\Phi[\Pi_{i:j}^{d,s}]$. By 'grouping' we mean to multiply all the local potentials belonging to a certain part, in the same way that we group all the local potentials belonging to the model in Equation 8.1. Note that although $\hat{\zeta}_{i:j}^{d,s}$ contains $\Pi_{i:j}^{d,s}$ we do not group $\Phi[\Pi_{i:j}^{d,s}]$ into $\Phi[\hat{\zeta}_{i:j}^{d,s}]$. The same holds for $\Phi[\hat{\underline{\zeta}}_{i:j}^{d,s}]$.

By definition of the state-persistence clique potential (Figure 8.6), we have $\Phi[\Pi_{i:j}^{d,s}] = R_{i:j}^{d,s}$. Thus Equation 8.1 can be replaced by

$$\Phi[\zeta] = \Phi[\hat{\zeta}_{i:j}^{d,s}] R_{i:j}^{d,s} \Phi[\hat{\underline{\zeta}}_{i:j}^{d,s}] \tag{8.17}$$

There are two special cases: (1) when $d = 1$, $\Phi[\hat{\underline{\zeta}}^{1,s}_{1:T}] = 1$ for $s \in S^1$, and (2) when $d = D$, $\Phi[\hat{\zeta}^{D,s}_{i:i}] = 1$ for $s \in S^D$ and $i \in [1, T]$. This factorisation plays an important role in efficient inference.

We know define a quantity called *symmetric inside mass* $\Delta^{d,s}_{i:j}$, and another called *symmetric outside mass* $\Lambda^{d,s}_{i:j}$.

Definition 6. *Given a symmetric Markov blanket* $\Pi^{d,s}_{i:j}$*, the symmetric inside mass* $\Delta^{d,s}_{i:j}$ *and the symmetric outside mass* $\Lambda^{d,s}_{i:j}$ *are defined as*

$$\Delta^{d,s}_{i:j} = \sum_{\zeta^{d,s}_{i:j}} \Phi[\hat{\zeta}^{d,s}_{i:j}] \tag{8.18}$$

$$\Lambda^{d,s}_{i:j} = \sum_{\underline{\zeta}^{d,s}_{i:j}} \Phi[\hat{\underline{\zeta}}^{d,s}_{i:j}] \tag{8.19}$$

As special cases we have $\Lambda^{1,s}_{1:T} = 1$ *and* $s \in S^1$*, and* $\Delta^{D,s}_{i:i} = 1$ *for* $i \in [1, T]$*,* $s \in S^D$*. For later use let us introduce the 'full' symmetric inside mass* $\hat{\Delta}^{d,s}_{i:j}$ *and the 'full' symmetric outside mass* $\hat{\Lambda}^{d,s}_{i:j}$ *as*

$$\hat{\Delta}^{d,s}_{i:j} = R^{d,s}_{i:j} \Delta^{d,s}_{i:j} \tag{8.20}$$

$$\hat{\Lambda}^{d,s}_{i:j} = R^{d,s}_{i:j} \Lambda^{d,s}_{i:j} \tag{8.21}$$

In the rest of the thesis, when it is clear in the context, we will use *inside mass* as a shorthand for symmetric inside mass, *outside mass* for symmetric outside mass, *full-inside mass* for full-symmetric inside mass, and *full-outside mass* for full-symmetric outside mass.

Thus, from Equation 8.17 the partition function can be computed from the full-inside mass at the top level ($d = 1$)

$$\begin{aligned} Z &= \sum_{\zeta} \Phi[\zeta] \\ &= \sum_{\zeta^{1,s}_{1:T}} \sum_{s \in S^1} \Phi[\hat{\zeta}^{1,s}_{1:T}] R^{1,s}_{1:T} \\ &= \sum_{s \in S^1} \Delta^{d,s}_{1:T} R^{d,s}_{1:T} \\ &= \sum_{s \in S^1} \hat{\Delta}^{1,s}_{1:T} \end{aligned} \tag{8.22}$$

With the similar derivation the partition function can also be computed from the full-outside mass at the bottom level ($d = D$)

$$Z = \sum_{s \in S^D} \hat{\Lambda}^{D,s}_{i:i}, \text{ for any } i \in [1, T] \tag{8.23}$$

In fact, we will prove a more general way to compute Z in Appendix A.4

$$Z = \sum_{s\in S^d} \sum_{i\in[1,t]} \sum_{j\in[t,T]} \Delta_{i:j}^{d,s} \Lambda_{i:j}^{d,s} R_{i:j}^{d,s} \tag{8.24}$$

for any $t \in [1, T]$ and $d \in [2, D-1]$. These relations are summarised in Figure 8.8.

- $Z = \sum_{s\in S^1} \hat{\Delta}_{1:T}^{1,s}$
- $Z = \sum_{s\in S^D} \hat{\Lambda}_{i:i}^{D,s}$ for any $i \in [1, T]$
- $Z = \sum_{s\in S^d} \sum_{i\in[1,t]} \sum_{j\in[t,T]} \Delta_{i:j}^{d,s} \Lambda_{i:j}^{d,s} R_{i:j}^{d,s}$ for any $t \in [1, T]$ and $d \in [2, D-1]$

Figure 8.8: Computing the partition function from the full-inside mass and full-outside mass.

Given the fact that $\zeta_{i:j}^{d,s}$ is separated from the rest of variables by the symmetric Markov blanket $\Pi_{i:j}^{d,s}$, we have Proposition 5.

Proposition 5. *The following relations hold*

$$\Pr(\zeta_{i:j}^{d,s}|\Pi_{i:j}^{d,s}) = \frac{1}{\Delta_{i:j}^{d,s}} \Phi[\hat{\zeta}_{i:j}^{d,s}] \tag{8.25}$$

$$\Pr(\underline{\zeta}_{i:j}^{d,s}|\Pi_{i:j}^{d,s}) = \frac{1}{\Lambda_{i:j}^{d,s}} \Phi[\hat{\underline{\zeta}}_{i:j}^{d,s}] \tag{8.26}$$

$$\Pr(\Pi_{i:j}^{d,s}) = \frac{1}{Z} \Delta_{i:j}^{d,s} R_{i:j}^{d,s} \Lambda_{i:j}^{d,s} \tag{8.27}$$

The proof of this proposition is given in Appendix A.3.2.

8.3.1.4 Asymmetric Inside/Outside Masses

Recall that we have introduced the concept of asymmetric Markov blanket $\Gamma_{i:j}^{d,s}(u)$ which separates $\zeta_{i:j}^{d,s}(u)$ and $\underline{\zeta}_{i:j}^{d,s}(u)$. Let us group all the local contextual clique potentials associated with $\zeta_{i:j}^{d,s}(u)$ and $\Gamma_{i:j}^{d,s}(u)$ into a joint potential $\Phi[\hat{\zeta}_{i:j}^{d,s}(u)]$. Similarly, we group all local potentials associated with $\underline{\zeta}_{i:j}^{d,s}(u)$ and $\Gamma_{i:j}^{d,s}(u)$ into a joint potential $\Phi[\hat{\underline{\zeta}}_{i:j}^{d,s}(u)]$. Note that $\Phi[\hat{\underline{\zeta}}_{i:j}^{d,s}(u)])$ includes the state-persistence potential $R_{i:j}^{d,s}$.

Definition 7. *Given the asymmetric Markov blanket $\Gamma_{i:j}^{d,s}(u)$, the asymmetric inside mass $\alpha_{i:j}^{d,s}(u)$ and the asymmetric outside mass $\lambda_{i:j}^{d,s}(u)$ are defined as follows*

$$\alpha_{i:j}^{d,s}(u) = \sum_{\zeta_{i:j}^{d,s}(u)} \Phi[\hat{\zeta}_{i:j}^{d,s}(u)] \tag{8.28}$$

$$\lambda_{i:j}^{d,s}(u) = \sum_{\underline{\zeta}_{i:j}^{d,s}(u)} \Phi[\hat{\underline{\zeta}}_{i:j}^{d,s}(u)] \tag{8.29}$$

The relationship between the asymmetric outside mass and asymmetric inside mass is analogous to that between the outside and inside masses. However, there is a small difference, that is, the asymmetric outside mass 'owns' the segment $x^d_{i:j} = s$ and the associated state-persistence potential $R^{d,s}_{i:j}$, whilst the outside mass $\Lambda^d_{i:j}(s)$ does not.

8.3.2 Computing Symmetric/Asymmetric Inside Masses

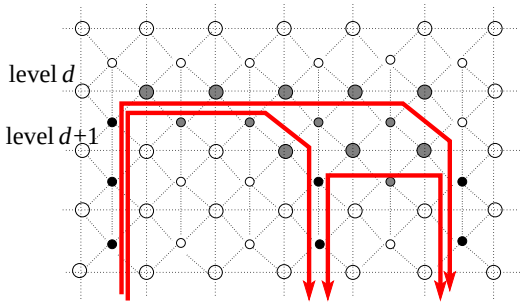


Figure 8.9: Decomposition with respect to symmetric/asymmetric Markov blankets.

In this subsection we show how to recursively compute the pair: inside mass and asymmetric inside mass. The key idea here is to exploit the decomposition within the asymmetric Markov blanket. As shown in Figure 8.9, an outer asymmetric Markov blanket can be decomposed into a sub-asymmetric Markov blanket and a symmetric blanket.

8.3.2.1 Computing asymmetric inside mass from inside mass

Assume that within the asymmetric Markov blanket $\Gamma^{d,s}_{i:j}(u)$, the child u starts somewhere at $t \in [i, j]$ and ends at j, i.e. $x^{d+1}_{t:j} = u$, $e^{d+1}_{t:j-1} = 0$ and $e^{d+1:D-1}_{t-1} = 1$. Let us consider two cases: $t > i$ and $t = i$.

Case 1. For $t > i$, denote by $v = x^{d+1}_{t-1}$. We have two smaller blankets within $\Gamma^{d,s}_{i:j}(u)$: the symmetric blanket $\Pi^{d+1,u}_{t:j}$ associated with the child $u = x^{d+1}_{t:j}$, and the asymmetric blanket $\Gamma^{d,s}_{i:t-1}(v)$ associated with the child v ending at $t-1$ under the parent s. Figure 8.9 illustrates the blanket decomposition. The assignment $\zeta^{d,s}_{i:j}(u)$ can be decomposed as

$$\zeta^{d,s}_{i:j}(u) = (\zeta^{d,s}_{i:t-1}(v), \zeta^{d+1,u}_{t:j}, u = x^{d+1}_{t:j}, e^d_{t-1:j-1} = 0, e^{d+1:D}_{t-1} = 1) \tag{8.30}$$

Thus, the joint potential $\Phi[\hat{\zeta}^{d,s}_{i:j}(u)]$ can be factorised as follows

$$\Phi[\hat{\zeta}^{d,s}_{i:j}(u)] = \Phi[\hat{\zeta}^{d,s}_{i:t-1}(v)]\Phi[\hat{\zeta}^{d+1,u}_{t:j}]A^{d+1,s}_{v,u,t-1}R^{d+1,u}_{t:j} \tag{8.31}$$

The transition potential $A^{d+1,s}_{v,u,t-1}$ is enabled in the context $c = (e^d_{t-1} = 0, e^{d+1}_{t-1} = 1, x^d_t =$

$s, x_{t-1}^{d+1} = v, x_t^{d+1} = u)$, and the state-persistence potential $R_{t:j}^{d+1,u}$ in the context $c = (e_{t:j-1}^{d+1} = 0, e_{t-1}^{d+1:D} = 1, e_j^{d+1:D} = 1, x_{t:j}^{d+1} = u)$.

Case 2. For $t = i$, the asymmetric blanket $\Gamma_{i:t-1}^{d,s}(v)$ does not exist since $i > t - 1$. We have the following decompositions of assignment $\hat{\zeta}_{i:j}^{d,s}(u) = (\hat{\zeta}_{i:j}^{d+1,u}, e_{i-1}^{d} = 1, e_{i:j-1}^{d} = 0)$. In the context $c = (e_{i-1}^{d} = 1)$, the state-initialisation potential $\pi_{u,i}^{d,s}$ is activated. Thus we have

$$\Phi[\hat{\zeta}_{i:j}^{d,s}(u)] = \pi_{u,i}^{d,s} \Phi[\hat{\zeta}_{i:j}^{d+1,u}] R_{i:j}^{d+1,u} \tag{8.32}$$

Substituting Equations 8.31 and 8.32 into Equation 8.28, and together with the fact that t can take any value in the interval $[i, j]$, and v can take any value in S^{d+1}, we have the following relation

$$\begin{aligned}
\alpha_{i:j}^{d,s}(u) &= \sum_{t\in[i+1,j]} \sum_{v\in S^{d+1}} \sum_{\zeta_{i:t-1}^{d,s}(v)} \sum_{\zeta_{t:j}^{d+1,u}} \Phi[\hat{\zeta}_{i:t-1}^{d,s}(v)] \Phi[\hat{\zeta}_{t:j}^{d+1,u}] A_{v,u,t-1}^{d+1,s} R_{t:j}^{d+1,u} + \\
&\qquad + \sum_{\zeta_{i:j}^{d+1,u}} \pi_{u,i}^{d,s} \Phi[\hat{\zeta}_{i:j}^{d+1,u}] R_{i:j}^{d+1,u} \\
&= \sum_{t\in[i+1,j]} \sum_{v\in S^{d+1}} \alpha_{i:t-1}^{d,s}(v) \hat{\Delta}_{t:j}^{d+1,u} A_{v,u,t-1}^{d+1,s} + \hat{\Delta}_{i:j}^{d+1,u} \pi_{u,i}^{d,s}
\end{aligned} \tag{8.33}$$

As we can see, the asymmetric inside mass α plays the role of a *forward message* starting from the starting time i to the ending time j. There is a recursion where the asymmetric inside mass ending at time j is computed from all the asymmetric inside masses ending at time $t - 1$, for $t \in [i + 1, j$.

There are special cases for the asymmetric inside mass: (1) when $i = j$, we only have

$$\alpha_{i:i}^{d,s}(u) = \hat{\Delta}_{i:i}^{d+1,s} \pi_{u,i}^{d,s} \tag{8.34}$$

and (2) when $d = D - 1$, the sum over the index t as in Equation 8.33 is not allowed since at level D the inside mass only spans a single index. We have the following instead

$$\begin{aligned}
\alpha_{i:j}^{D-1,s}(u) &= \sum_{v\in S^{d+1}} \alpha_{i:j-1}^{D-1,s}(v) \hat{\Delta}_{j:j}^{D,u} A_{v,u,j-1}^{D,s} \\
&= \sum_{v\in S^{d+1}} \alpha_{i:j-1}^{D-1,s}(v) R_{j:j}^{D,u} A_{v,u,j-1}^{D,s}
\end{aligned} \tag{8.35}$$

8.3.2.2 Computing inside mass from asymmetric inside mass

Notice the relationship between the asymmetric Markov blanket $\Gamma^{d,s}_{i:j}(u)$ and the symmetric blanket $\Pi^{d,s}_{i:j}$, where $d < D$. When $e^d_j = 1$, i.e. the parent s ends at j, and $\Gamma^{d,s}_{i:j}(u)$ will become $\Pi^{d,s}_{i:j}$ with $u = x^{d+1}_j$. Then we have decompositions $\zeta^{d,s}_{i:j} = (\zeta^{d,s}_{i:j}(u), u = x^{d+1}_j)$ and $\hat{\zeta}^{d,s}_{i:j} = (\hat{\zeta}^{d,s}_{i:j}(u), e^d_j = 1, u = x^{d+1}_j)$. These lead to the factorisation

$$\Phi[\hat{\zeta}^{d,s}_{i:j}] = \Phi[\hat{\zeta}^{d,s}_{i:j}(u)]E^{d,s}_{u,j} \tag{8.36}$$

where the state-ending potential $E^{d,s}_{u,j}$ is activated in the context $c = (e^d_j = 1)$. Thus, the inside mass in Equation 8.18 can be rewritten as

$$\begin{aligned}
\Delta^{d,s}_{i:j} &= \sum_{u \in S^{d+1}} \sum_{\zeta^{d,s}_{i:j}(u)} \Phi[\hat{\zeta}^{d,s}_{i:j}(u)]E^{d,s}_{u,j} \\
&= \sum_{u \in S^{d+1}} E^{d,s}_{u,j} \sum_{\zeta^{d,s}_{i:j}(u)} \Phi[\hat{\zeta}^{d,s}_{i:j}(u)] \\
&= \sum_{u \in S^{d+1}} E^{d,s}_{u,j} \alpha^{d,s}_{i:j}(u)
\end{aligned} \tag{8.37}$$

This equation holds for $d < D$. When $d = D$, we set $\Delta^{D,s}_{i:i} = 1$ for all $s \in S^D$ and $i \in [1, T]$, and when $d = 1$, we must ensure that $i = 1$ and $j = T$.

Remark: Equations 8.33, 8.34, 8.35 and 8.37 specify a *left-right* and *bottom-up* algorithm to compute both the inside and asymmetric inside masses. Initially, at the bottom level $\Delta^{D,s}_{i:i} = 1$ for $i \in [1, T]$ and $s \in S^D$. A pseudo-code of the dynamic programming algorithm to compute all the inside and asymmetric inside masses and the partition function is given in Figure 8.10.

8.3.3 Computing Symmetric/Asymmetric Outside Masses

In this subsection we show how to recursively compute the symmetric outside mass and the asymmetric outside mass. We use the same blanket decomposition as in Section 8.3.2. However, this time the view is reversed as we are interested in quantities outside the blankets. For example, outside the inner symmetric Markov blanket in Figure 8.9, there exists an outer asymmetric blanket and another sub-asymmetric blanket on the left.

Input: D, T, all the potential function values.
Output: partition function Z;
$\Delta_{1:T}^{1,s}$, for $s \in S^1$;
$\Delta_{i:j}^{d,s}$, for $d \in [2, D-1]$, $s \in S^d$ and $1 \leq i \leq j \leq T$;
$\Delta_{i:i}^{D,s}$ for $s \in S^D$ and $i \in [1, T]$;
$\alpha_{i:j}^{d,s}(u)$ for $d \in [1, D-1]$, $u \in S^{d+1}$ and $1 \leq i \leq j \leq T$

/ Initialisation */*
$\Delta_{i:i}^{D,s} = 1$ for all $i \in [1, T]$ and $s \in S^D$
/ At the level d=D-1 */*
For $i = 1, 2, ..., T$
  For $j = i, i+1, ..., T$
    Compute $\alpha_{i:j}^{D-1,s}(u)$ using Equation 8.35
    Compute $\Delta_{i:j}^{D-1,s}$ using Equation 8.37
  EndFor
EndFor
/ The main recursion loops: bottom-up and forward */*
For $d = D-2, D-3, ..., 1$
  For $i = 1, 2, ..., T$
    For $j = i, i+1, ..., T$
      Compute $\alpha_{i:i}^{d,s}(u)$ using Equation 8.34 **If** $j = i$
      Compute $\alpha_{i:j}^{d,s}(u)$ using Equation 8.33 **If** $j > i$
      Compute $\Delta_{i:j}^{d,s}$ using Equation 8.37 **If** $d > 1$
    EndFor
  EndFor
EndFor
Compute Z using Equation 8.22.

Figure 8.10: Computing the set of inside/asymmetric inside masses and the partition function.

8.3.3.1 Computing asymmetric outside mass from outside mass

Let us examine the variables $\underline{\zeta}_{i:j}^{d,s}(u)$ associated with the asymmetric Markov blanket $\Gamma_{i:j}^{d,s}(u)$, for $d \in [1, D-1]$ and $1 \leq i \leq j \leq T$ (see Definition 5). For $j < T$, assume that there exists an outer asymmetric Markov blanket $\Gamma_{i:t}^{d,s}(v)$ for some $v \in S^{d+1}$ and $t \in [j+1, T]$, and a symmetric Markov blanket $\Pi_{j+1:t}^{d+1,v}$ right next to $\Gamma_{i:j}^{d,s}(u)$. Given these blankets we have the decomposition $\underline{\hat{\zeta}}_{i:j}^{d,s}(u) = (\underline{\hat{\zeta}}_{i:t}^{d,s}(v), \hat{\zeta}_{j+1:t}^{d+1,v}, x_j^{d+1} = u)$, which leads to the following factorisation

$$\Phi[\underline{\hat{\zeta}}_{i:j}^{d,s}(u)] = \Phi[\underline{\hat{\zeta}}_{i:t}^{d,s}(v)]\Phi[\hat{\zeta}_{j+1:t}^{d+1,v}]R_{j+1:t}^{d+1,v}A_{u,v,j}^{d+1,s} \tag{8.38}$$

The state transition potential $A_{u,v,j}^{d+1,s}$ is enabled in the context $c = (e_j^d = 0, e_j^{d+1} = 1)$, and the state persistence potential $R_{j+1:t}^{d+1,v}$ in the context $c = (e_j^{d+1} = 1, e_{j+1:t-1}^{d+1} = 0, e_t^{d+1} = 1)$.

In addition, there exists a special case where the state s ends at j. We have the decomposi-

tion $\hat{\underline{\zeta}}^{d,s}_{i:j}(u) = (\hat{\underline{\zeta}}^{d,s}_{i:j}, u = x^{d+1}_j)$ and the following factorisation

$$\Phi[\hat{\underline{\zeta}}^{d,s}_{i:j}(u)] = \Phi[\hat{\underline{\zeta}}^{d,s}_{i:j}] R^{d,s}_{i:j} E^{d,s}_{u,j} \tag{8.39}$$

The ending potential $E^{d,s}_{u,j}$ appears here because of the context $c = (e^d_j = 1)$, i.e. s ends at j.

Now we relax the assumption of t, v and allow them to receive all possible values, i.e. $t \in [j, T]$ and $v \in S^{d+1}$. Thus we can replace Equation 8.29 by

$$\begin{aligned}
\lambda^{d,s}_{i:j}(u) &= \sum_{v \in S^{d+1}} \sum_{t \in [j+1,T]} \sum_{\underline{\zeta}^{d,s}_{i:t}(v)} \sum_{\zeta^{d+1,v}_{j+1:t}} \Phi[\hat{\underline{\zeta}}^{d,s}_{i:t}(v)] \Phi[\hat{\zeta}^{d+1,v}_{j+1:t}] R^{d+1,v}_{j+1:t} A^{d+1,s}_{u,v,j} \\
&\qquad + \sum_{\underline{\zeta}^{d,s}_{i:j}(u)} \Phi[\hat{\underline{\zeta}}^{d,s}_{i:j}] R^{d,s}_{i:j} E^{d,s}_{u,j} \\
&= \sum_{v \in S^{d+1}} \sum_{t \in [j+1,T]} \lambda^{d,s}_{i:t}(v) \hat{\Delta}^{d+1,v}_{j+1:t} A^{d+1,s}_{u,v,j} + \hat{\Lambda}^{d,s}_{i:j} E^{d,s}_{u,j}
\end{aligned} \tag{8.40}$$

for $d \in [2, D-2]$, and $1 \le i \le j \le T$. Thus, the $\lambda^{d,s}_{i:j}(u)$ can be thought as a message passed *backward* from $j = T$ to $j = i$. Here, the asymmetric outside mass ending at j is computed by using all the asymmetric outside masses ending at t for $t \in [j+1, T]$.

There are two special cases. At the top level, i.e. $d = 1$, then $\lambda^{d,s}_{i:j}(u)$ is only defined at $i = 1$, and the second term of the RHS of Equation 8.40 is included only if $i = 1, j = T$. At the second lowest level, i.e. $d = D-1$, we cannot sum over t as in Equation 8.40 since $\hat{\Delta}^{D,v}_{j+1:t}$ is only defined for $t = j+1$. We have the following relation instead

$$\lambda^{D-1,s}_{i:j}(u) = \sum_{v \in S^D} \lambda^{D-1,s}_{i:j+1}(v) \hat{\Delta}^{D,v}_{j+1:j+1} A^{D,s}_{u,v,j} + \hat{\Lambda}^{D-1,s}_{i:j} E^{D-1,s}_{u,j} \tag{8.41}$$

8.3.3.2 Computing outside mass from asymmetric outside mass

Given a symmetric Markov blanket $\Pi^{d+1,u}_{i:j}$ for $d \in [1, D-1]$, assume that there exists an asymmetric Markov blanket $\Gamma^{d,s}_{t:j}(u)$ at the parent level d, where $t \in [1, i]$. Clearly, for $t \in [1, i-1]$ there exists some sub-asymmetric Markov blanket $\Gamma^{d,s}_{t:i-1}(v)$. See Figure 8.9 for an illustration.

Let us consider two cases: $t < i$ and $t = i$.

Case 1. For $t < i$, this enables the decomposition $\hat{\underline{\zeta}}^{d+1,u}_{i:j} = (\hat{\underline{\zeta}}^{d,s}_{t:j}(u), \hat{\zeta}^{d,s}_{t:i-1}(v), u = x^{d+1}_{i:j})$, which leads to the following factorisation

$$\Phi[\hat{\underline{\zeta}}^{d+1,u}_{i:j}] = \Phi[\hat{\underline{\zeta}}^{d,s}_{t:j}(u)] \Phi[\hat{\zeta}^{d,s}_{t:i-1}(v)] A^{d,s}_{v,u,i-1} \tag{8.42}$$

The state transition potential $A^{d,s}_{v,u,i-1}$ is activated in the context $c = (e^d_{i-1} = 0, e^{d+1}_{i-1} = 1)$.

Case 2. For $t = i$, the decomposition reduces to $\hat{\underline{\zeta}}^{d+1,u}_{i:j} = (\hat{\underline{\zeta}}^{d,s}_{i:j}(u), u = x^{d+1}_{i:j})$, which leads to the following factorisation

$$\Phi[\hat{\underline{\zeta}}^{d+1,u}_{i:j}] = \Phi[\hat{\underline{\zeta}}^{d,s}_{i:j}(u)]\pi^{d,s}_{u,i} \tag{8.43}$$

The state-initialisation potential $\pi^{d,s}_{u,i}$ plays the role in the context $c = (e^d_{i-1} = 1)$

However, these decompositions and factorisations only hold given the assumption of specific values of $s \in S^d$, $v \in S^{d+1}$, and $t \in [1, i]$. Without further information we have to take all possibilities into account. Substituting these relations into Equation 8.19, we have

$$\begin{aligned}
\Lambda^{d+1,u}_{i:j} &= \sum_{s\in S^d}\sum_{v\in S^{d+1}}\sum_{t\in[1,i-1]}\sum_{\underline{\zeta}^{d,s}_{t:j}(u)}\sum_{\zeta^{d,s}_{t:i-1}(v)} \Phi[\hat{\underline{\zeta}}^{d,s}_{t:j}(u)]\Phi[\hat{\zeta}^{d,s}_{t:i-1}(v)]A^{d+1,s}_{v,u,i-1} + \\
&\qquad + \sum_{s\in S^d}\sum_{\underline{\zeta}^{d,s}_{i:j}(u)} \Phi[\hat{\underline{\zeta}}^{d,s}_{i:j}(u)]\pi^{d,s}_{u,i} \\
&= \sum_{s\in S^d}\sum_{t\in[1,i-1]} \lambda^{d,s}_{t:j}(u) \sum_{v\in S^{d+1}} \alpha^{d,s}_{t:i-1}(v)A^{d+1,s}_{v,u,i-1} + \sum_{s\in S^d}\lambda^{d,s}_{i:j}(u)\pi^{d,s}_{u,i}
\end{aligned} \tag{8.44}$$

for $d \in [2, D-2]$.

There are three special cases. The first is the base case where $d = 0$ and $\Lambda^{1,s}_{1:T} = 1$ for all $s \in S^1$. In the second case, for $d = 1$, we must fix the index $t = 1$ since the asymmetric inside mass $\alpha^{d,s}_{t:i-1}$ is only defined at $t = 1$. Also the second term in the RHS is included only if $i = 1$ for the asymmetric outside mass $\lambda^{d,s}_{i:j}(u)$ to make sense. In the second case, for $d + 1 = D$, we only have $i = j$.

Remark: Equations 8.40, 8.41 and 8.44 show a recursive *top-down* and *outside-in* approach to compute the symmetric/asymmetric outside masses. We start from the top with $d = 1$ and $\Lambda^{1,s}_{1:T} = 1$ for all $s \in S^1$ and proceed downward until $d = D$. The pseudo-code is given in Figure 8.11. Figure 8.12 summarises the quantities computed in Section 8.3.2 and 8.3.3.

Figure 8.13 summarises the AIO algorithm for computing all building blocks and the partition function.

8.4 Parameter Estimation

In this section, we tackle the problem of parameter estimation by maximising the (conditional) data likelihood. Typically we need some parametric form to be defined for a

Input: D, T, all the potential function values, all inside/asymmetric inside masses.
Output: all outside/asymmetric outside masses

Initialise: $\Lambda^{1,s}_{1:T} = 1$,
$\lambda^{1,s}_{1:T}(u) = E^{1,s}_{u,T}$ for $s \in S^1, u \in S^2$
/ the main recursive loops: top-down and inside-out */*
For $d = 1, 2, ..., D-1$
  For $i = 1, 2, ..., T$
    For $j = T, T-1, ..., i$
      Compute the asymmetric outside mass $\lambda^{d,s}_{i:j}(u)$ using Equations 8.40,8.41
      Compute the outside mass $\Lambda^{d,s}_{i:j}$ using Equation 8.44
    EndFor
  EndFor
EndFor

Figure 8.11: Computing the set of outside/asymmetric outside masses.

- $\Delta^{1,s}_{1:T}, \Lambda^{1,s}_{1:T}$ for $s \in S^1$
- $\Delta^{d,s}_{i:j}, \Lambda^{d,s}_{i:j}$ for $d \in [2, D-1], s \in S^d, 1 \leq i \leq j \leq T$
- $\Delta^{D,s}_{i:i}, \Lambda^{D,s}_{i:i}$ for $i \in [1, T], s \in S^D$
- $\alpha^{d,s}_{1:j}(u), \lambda^{d,s}_{1:j}(u)$ for $d = 1, s \in S^1, u \in S^2, j \in [1, T]$
- $\alpha^{d,s}_{i:j}(u), \lambda^{d,s}_{i:j}(u)$ for $d \in [2, D-1], s \in S^d, u \in S^{d+1}, 1 \leq i \leq j \leq T$

Figure 8.12: Summary of basic building blocks computed in Section 8.3.2 and 8.3.3.

particular problem and we need some numerical method to do the optimisation task.

Here we employ the log-linear parameterisation, which is commonly used in the CRF setting. Recall from Section 3.2 that estimating parameters of the log-linear models using gradient-based methods requires the computation of feature expectation, or expected sufficient statistics (ESS). For our HCRFs we need to compute four types of ESS corresponding to the state-persistence, state-transition, state-initialisation and state-ending.

8.4.1 Log-Linear Parameterisation

In our HCRF setting there is a feature vector $\mathbf{f}^d_\sigma(\sigma, z)$ associated with each type of contextual clique σ, in that $\phi(\sigma^d, z) = \exp(\mathbf{w}^\top_{\sigma^d} \mathbf{f}^d_\sigma(\sigma, z))$. Thus, the features are active only in the

Input: D, T, all the potential function values
Output: all building blocks and partition function

Compute all inside/asymmetric inside masses using the algorithm in Figure 8.10
Compute all outside/asymmetric outside masses using the algorithm in Figure 8.11

Figure 8.13: The AIO algorithm.

context in which the corresponding contextual cliques appear.

For the state-persistence contextual clique, the features incorporate *state-duration*, start time i and end time j of the state. Other feature types incorporate the time index in which the features are triggered. Specifically,

$$R^{d,s,z}_{i:j} = \exp(\mathbf{w}^{\top}_{\sigma^{persist},d}\mathbf{f}^{d,s}_{\sigma^{persist}}(i,j,z)) \quad (8.45)$$

$$A^{d,s,z}_{u,v,i} = \exp(\mathbf{w}^{\top}_{\sigma^{transit},d}\mathbf{f}^{d,s}_{\sigma^{transit},u,v}(i,z) \quad (8.46)$$

$$\pi^{d,s,z}_{u,i} = \exp(\mathbf{w}^{\top}_{\sigma^{init},d}\mathbf{f}^{d,s}_{\sigma^{init},u}(i,z) \quad (8.47)$$

$$E^{d,s,z}_{u,i} = \exp(\mathbf{w}^{\top}_{\sigma^{end},d}\mathbf{f}^{d,s}_{\sigma^{end},u}(i,z) \quad (8.48)$$

Denote by $\mathbf{F}^{d}_{\sigma}(\zeta,z)$ the global feature, which is the sum of all active features $\mathbf{f}^{d}_{\sigma}(z)$ at level d in the duration $[1,T]$ for a given assignment of ζ and a clique type σ. Recall that $\tau^{d} = \{i_k\}_{k=1}^{m}$ is the set of ending time indices (i.e. $e^{d}_{i_k} = 1$). The four feature types are given in Equations 8.49-8.52.

$$\mathbf{F}^{d,s}_{\sigma^{persist}}(\zeta,z) = \mathbf{f}^{d,s}_{\sigma^{persist}}(1,i_1,z) + \sum_{i_k\in\tau^{d},k>1}\mathbf{f}^{d,s}_{\sigma^{persist}}(i_k+1,i_{k+1},z) \quad (8.49)$$

$$\mathbf{F}^{d,s}_{\sigma^{transit},u,v}(\zeta,z) = \sum_{i_k\notin\tau^{d-1},i_k\in\tau^{d}}\mathbf{f}^{d,s}_{\sigma^{transit},u,v}(i_k,z) \quad (8.50)$$

$$\mathbf{F}^{d,s}_{\sigma^{init},u}(\zeta,z) = \mathbf{f}^{d,s}_{\sigma^{init},u,v}(1,z) + \sum_{i_k\in\tau^{d}}\mathbf{f}^{d,s}_{\sigma^{init},u,v}(i_k+1,z) \quad (8.51)$$

$$\mathbf{F}^{d,s}_{\sigma^{end},u}(\zeta,z) = \sum_{i_k\in\tau^{d}}\mathbf{f}^{d,s}_{\sigma^{end},u,v}(i,z) \quad (8.52)$$

Substituting the global features into potentials in Equation. 8.1 and 8.2 we obtain the following log-linear model:

$$\Pr(\zeta|z) = \frac{1}{Z(z)}\exp(\sum_{c\in C}\mathbf{w}^{\top}_{\sigma^{c}}\mathbf{F}_{\sigma^{c}}(\zeta,z)) \quad (8.53)$$

where $C = \{persist, transit, init, exit\}$.

Again, for clarity of presentation we will drop the notion of z but implicitly assume that it is still in the each quantity.

8.4.2 ESS for State-Persistence Features

Recall from Section 8.4.1 that the feature function for the state-persistence $\mathbf{f}^{d,s}_{\sigma^{persist}}(i,j)$ is active only in the context where $\Pi^{d,s}_{i:j} \in \zeta$. Thus, Equation 8.49 can be rewritten as

$$\mathbf{F}^{d,s}_{\sigma^{persist}}(\zeta) = \sum_{i\in[1,T]} \sum_{j\in[i,T]} \mathbf{f}^{d,s}_{\sigma^{persist}}(i,j)\delta[\Pi^{d,s}_{i:j} \in \zeta] \tag{8.54}$$

The indicator function in the RHS ensures that the feature $\mathbf{f}^{d,s}_{\sigma^{persist}}(i,j)$ is only active if there exists a symmetric Markov blanket $\Pi^{d,s}_{i:j}$ in the assignment of ζ. Consider the following expectation

$$\begin{aligned}
\mathbb{E}[\mathbf{f}^{d,s}_{\sigma^{persist}}(i,j)\delta[\Pi^{d,s}_{i:j} \in \zeta]] &= \sum_{\zeta} \Pr(\zeta)\mathbf{f}^{d,s}_{\sigma^{persist}}(i,j)\delta[\Pi^{d,s}_{i:j} \in \zeta] &(8.55)\\
&= \frac{1}{Z}\sum_{\zeta} \Phi[\zeta]\mathbf{f}^{d,s}_{\sigma^{persist}}(i,j)\delta[\Pi^{d,s}_{i:j} \in \zeta] &(8.56)
\end{aligned}$$

Using the factorisation in Equation 8.17 we can rewrite

$$\mathbb{E}[\mathbf{f}^{d,s}_{\sigma^{persist}}(i,j)\delta[\Pi^{d,s}_{i:j} \in \zeta]] = \frac{1}{Z}\sum_{\zeta} \Phi[\hat{\zeta}^{d,s}_{i:j}]\Phi[\hat{\underline{\zeta}}^{d,s}_{i:j}]R^{d,s}_{i:j}\mathbf{f}^{d,s}_{\sigma^{persist}}(i,j)\delta[\Pi^{d,s}_{i:j} \in \zeta] \tag{8.57}$$

Note that the elements inside the sum of the RHS are only non-zeros for those assignment of ζ that respect the persistent state $s^d_{i:j}$ and the factorisation in Equation 8.17, i.e. $\zeta = (\zeta^{d,s}_{i:j}, \underline{\zeta}^{d,s}_{i:j}, \Pi^{d,s}_{i:j})$. Thus, the equation can be simplified to

$$\begin{aligned}
\mathbb{E}[\mathbf{f}^{d,s}_{\sigma^{persist}}(i,j)\delta[\Pi^{d,s}_{i:j} \in \zeta]] &= \frac{1}{Z}\sum_{\zeta^{d,s}_{i:j}} \sum_{\underline{\zeta}^{d,s}_{i:j}} \Phi[\hat{\zeta}^{d,s}_{i:j}]\Phi[\hat{\underline{\zeta}}^{d,s}_{i:j}]R^{d,s}_{i:j}\mathbf{f}^{d,s}_{\sigma^{persist}}(i,j) &(8.58)\\
&= \frac{1}{Z}\Delta^{d,s}_{i:j}\Lambda^{d,s}_{i:j}R^{d,s}_{i:j}\mathbf{f}^{d,s}_{\sigma^{persist}}(i,j) &(8.59)
\end{aligned}$$

Using Equation 8.54 we obtain the ESS for the state-persistence features

$$\begin{aligned}
\mathbb{E}[F^{d,s}_{k}(\zeta)] &= \sum_{i\in[1,T]} \sum_{j\in[i,T]} \mathbb{E}[\mathbf{f}^{d,s}_{\sigma^{persist}}(i,j)\delta[\Pi^{d,s}_{i:j} \in \zeta]]\\
&= \frac{1}{Z}\sum_{i\in[1,T]} \sum_{j\in[i,T]} \Delta^{d,s}_{i:j}\Lambda^{d,s}_{i:j}R^{d,s}_{i:j}\mathbf{f}^{d,s}_{\sigma^{persist}}(i,j) &(8.60)
\end{aligned}$$

There are two special cases: (1) when $d = 1$, we do not sum over i, j but fix $i = 1, j = T$, and (2) when $d = D$ then we keep $j = i$.

8.4.3 ESS for Transition Features

Recall that in Section 8.4.1 we define $\mathbf{f}^{d,s}_{\sigma^{transit},u,v}(t)$ as a function that is active in the context $c^{transit} = (e^{d-1}_t = 0, e^d_t = 1)$, in which the child state u^d finishes its job at time t and transits to the child state v^d under the same parent s^{d-1} (that is s^{d-1} is still running). Thus Equation 8.50 can be rewritten as

$$\mathbf{F}^{d,s}_{\sigma^{transit},u,v}(\zeta) = \sum_{t\in[1,T-1]} \mathbf{f}^{d,s}_{\sigma^{transit},u,v}(t)\delta[c^{transit} \in \zeta] \tag{8.61}$$

We now consider the following expectation

$$\begin{aligned}\mathbb{E}[\mathbf{f}^{d,s}_{\sigma^{transit},u,v}(t)\delta[c^{transit} \in \zeta]] &= \sum_{\zeta} \Pr(\zeta)\mathbf{f}^{d,s}_{\sigma^{transit},u,v}(t)\delta[c^{transit} \in \zeta] &(8.62)\\ &= \frac{1}{Z}\sum_{\zeta} \Phi[\zeta]\mathbf{f}^{d,s}_{\sigma^{transit},u,v}(t)\delta[c^{transit} \in \zeta] &(8.63)\end{aligned}$$

Assume that the parent s starts at i. Since $e^d_t = 1$, the child v must starts at $t+1$ and ends some time later at $j \geq t+1$. We have the following decomposition of the configuration ζ that respects this assumption

$$\zeta = (\hat{\underline{\zeta}}^{d-1,s}_{i:j}(v), \hat{\zeta}^{d-1,s}_{i:t}(u), \hat{\zeta}^{d,v}_{t+1:j}) \tag{8.64}$$

and the following factorisation of the joint potential

$$\Phi[\zeta] = \Phi[\hat{\underline{\zeta}}^{d-1,s}_{i:j}(v)]\Phi[\hat{\zeta}^{d-1,s}_{i:t}(u)]\Phi[\hat{\zeta}^{d,v}_{t+1:j}]R^{d,v}_{t+1:j}A^{d,s}_{u,v,t} \tag{8.65}$$

The state persistent potential $R^{d,v}_{t+1:j}$ is enabled in the context $c = (e^d_t = 1, e^d_{t+1:j-1} = 0, e^d_j = 1)$ and the state transition potential $A^{d,s}_{u,v,t}$ in the context $c^{transit}$.

Substituting this factorisation into the RHS of Equation 8.63 gives us

$$\frac{1}{Z}\sum_{i\in[1,t]}\sum_{j\in[t+1,T]}\sum_{\zeta^{d-1,s}_{i:t}(u)}\sum_{\underline{\zeta}^{d-1,s}_{i:j}(v)}\sum_{\zeta^{d,v}_{t+1:j}} \Phi[\hat{\underline{\zeta}}^{d-1,s}_{i:j}(v)]\Phi[\hat{\zeta}^{d-1,s}_{i:t}(u)]\Phi[\hat{\zeta}^{d,v}_{t+1:j}]R^{d,v}_{t+1:j}A^{d,s}_{u,v,t}\mathbf{f}^{d,s}_{\sigma^{transit},u,v}(t)$$

which can be simplified to

$$\frac{1}{Z}\sum_{i\in[1,t]}\sum_{j\in[t+1,T]} \lambda^{d-1,s}_{i:j}(v)\alpha^{d-1,s}_{i:t}(u)\hat{\Delta}^{d,v}_{t+1:j}A^{d,s}_{u,v,t}\mathbf{f}^{d,s}_{\sigma^{transit},u,v}(t) \tag{8.66}$$

Using Equations 8.61 and 8.66 we obtain the ESS for the state-transition features

$$\begin{aligned}\mathbb{E}[\mathbf{F}^{d,s}_{\sigma^{transit},u,v}(\zeta)] &= \sum_{t\in[1,T-1]} \mathbb{E}[\mathbf{f}^{d,s}_{\sigma^{transit},u,v}(t)\delta[c^{transit}\in\zeta]] \\ &= \frac{1}{Z}\sum_{t\in[1,T-1]} A^{d,s}_{u,v,t}\mathbf{f}^{d,s}_{\sigma^{transit},u,v}(t)\sum_{i\in[1,t]}\sum_{j\in[t+1,T]}\alpha^{d-1,s}_{i:t}(u)\lambda^{d-1,s}_{i:j}(v)\hat{\Delta}^{d,v}_{t+1:j} \quad (8.67)\end{aligned}$$

When $d=2$ we must fix $i=1$ since $\alpha^{1,s}_{i:t}(u)$ and $\lambda^{1,s}_{i:j}(v)$ are only defined at $i=1$.

8.4.4 ESS for Initialisation Features

Recall that in Section 8.4.1 we define $\mathbf{f}^{d,s}_{\sigma^{init},u}(i)$ as a function at level d that is triggered at time i when a parent s at level d initialises a child u at level $d+1$. In this event, the context $c^{init}=(e^d_{i-1}=1)$ must be activated for $i>1$. Thus, Equation 8.51 can be rewritten as

$$\mathbf{F}^{d,s}_{\sigma^{init},u}(\zeta) = \sum_{i\in[1,T]}\mathbf{f}^{d,s}_{\sigma^{init},u}(i)\delta[c^{init}\in\zeta] \quad (8.68)$$

Now we consider the following feature expectation

$$\begin{aligned}\mathbb{E}[\mathbf{f}^{d,s}_{\sigma^{init},u}(i)\delta[c^{init}\in\zeta]] &= \sum_{\zeta}\Pr(\zeta)\mathbf{f}^{d,s}_{\sigma^{init},u}(i)\delta[c^{init}\in\zeta] \\ &= \frac{1}{Z}\sum_{\zeta}\Phi[\zeta]\mathbf{f}^{d,s}_{\sigma^{init},u}(i)\delta[c^{init}\in\zeta] \quad (8.69)\end{aligned}$$

For each assignment of ζ that enables $\mathbf{f}^{d,s}_{\sigma^{init},u}(i)$, we have the following decomposition

$$\zeta = (\hat{\underline{\zeta}}^{d,s}_{i:j}(u), \hat{\zeta}^{d+1,u}_{i:j}) \quad (8.70)$$

where the context c^{init} activates the emission from s to u and the feature function $\mathbf{f}^{d,s}_{\sigma^{init},u}(i)$. Thus the joint potential $\Phi[\zeta]$ can be factorised as

$$\Phi[\zeta] = \Phi[\hat{\underline{\zeta}}^{d,s}_{i:j}(u)]\Phi[\hat{\zeta}^{d+1,u}_{i:j}]R^{d+1,u}_{i:j}\pi^{d,s}_{u,i} \quad (8.71)$$

Using this factorisation and noting that the elements within the summation in the RHS of Equation 8.69 are only non-zeros with such assignments, we can simplify the RHS of

Equation 8.69 to

$$\frac{1}{Z}\sum_{j\in[i,T]}\sum_{\underline{\zeta}_{i:j}^{d,s}(u)}\sum_{\zeta_{i:j}^{d+1,u}}\Phi[\hat{\underline{\zeta}}_{i:j}^{d,s}(u)]\Phi[\hat{\zeta}_{i:j}^{d+1,u}]R_{i:j}^{d+1,u}\pi_{u,i}^{d,s}\mathbf{f}_{\sigma^{init},u}^{d,s}(i)$$
$$=\frac{1}{Z}\sum_{j\in[i,T]}\lambda_{i:j}^{d,s}(u)\hat{\Delta}_{i:j}^{d+1,u}\pi_{u,i}^{d,s}\mathbf{f}_{\sigma^{init},u}^{d,s}(i) \tag{8.72}$$

The summation over $j\in[i,T]$ is due to the fact that we do not know this index.

Using Equation 8.68 and 8.72 we obtain the ESS for the initialisation features

$$\begin{aligned}\mathbb{E}[\mathbf{F}_{\sigma^{init},u}^{d,s}(\zeta)] &= \sum_{i\in[1,T]}\mathbb{E}[\mathbf{f}_{\sigma^{init},u}^{d,s}(i)\delta[c^{init}\in\zeta]]\\ &= \frac{1}{Z}\sum_{i\in[1,T]}\pi_{u,i}^{d,s}\mathbf{f}_{\sigma^{init},u}^{d,s}(i)\sum_{j\in[i,T]}\lambda_{i:j}^{d,s}(u)\hat{\Delta}_{i:j}^{d+1,u}\end{aligned} \tag{8.73}$$

There are two special cases: (1) when $d=1$, there must be no scanning of i but fix $i=1$ since there is only a single initialisation at the beginning of sequence, (2) when $d=D-1$, we fix $j=i$ for $\hat{\Delta}_{i:j}^{D,u}$ is only defined at $i=j$.

8.4.5 ESS for Ending Features

Recall that in Section 8.4.1 we define $\mathbf{f}_{\sigma^{end},u}^{d,s}(j)$ as a function that is activated when a child u at level $d+1$ returns the control to its parent s at level d and time j. This event also enables the context $c^{end}=(e_j^d=1)$. Thus Equation 8.52 can be rewritten as

$$\mathbf{F}_{\sigma^{end},u}^{d,s}(\zeta)=\sum_{j\in[1,T]}\mathbf{f}_{\sigma^{end},u}^{d,s}(j)\delta[c^{end}\in\zeta] \tag{8.74}$$

Now we consider the following feature expectation

$$\begin{aligned}\mathbb{E}[\mathbf{f}_{\sigma^{end},u}^{d,s}(j)\delta[c^{end}\in\zeta]] &= \sum_{\zeta}\Pr(\zeta)\mathbf{f}_{\sigma^{end},u}^{d,s}(j)\delta[c^{end}\in\zeta]\\ &= \frac{1}{Z}\sum_{\zeta}\Phi[\zeta]\mathbf{f}_{\sigma^{end},u}^{d,s}(j)\delta[c^{end}\in\zeta]\end{aligned} \tag{8.75}$$

Assume that the state s starts at i and ends at j. For each assignment of ζ that enables $\mathbf{f}_{\sigma^{end},u}^{d,s}(j)$ and respects this assumption, we have the following decomposition

$$\zeta=(\hat{\underline{\zeta}}_{i:j}^{d,s},\hat{\zeta}_{i:j}^{d,s}(u)) \tag{8.76}$$

This assignment has the context c^{end} that activates the ending of u. Thus the joint potential $\Phi[\zeta]$ can be factorised as

$$\Phi[\zeta] = \Phi[\hat{\underline{\zeta}}^{d,s}_{i:j}]\Phi[\hat{\zeta}^{d,s}_{i:j}(u)]R^{d,s}_{i:j}E^{d,s}_{u,j} \tag{8.77}$$

Substituting this factorisation into the summation of the RHS of Equation 8.75 yields

$$\sum_{i\in[1,j]}\sum_{\underline{\zeta}^{d,s}_{i:j}}\sum_{\zeta^{d,s}_{i:j}(u)}\Phi[\hat{\underline{\zeta}}^{d,s}_{i:j}]\Phi[\hat{\zeta}^{d,s}_{i:j}(u)]R^{d,s}_{i:j}E^{d,s}_{u,j}\mathbf{f}^{d,s}_{\sigma^{end},u}(j) = \sum_{i\in[1,j]}\hat{\Lambda}^{d,s}_{i:j}\alpha^{d,s}_{i:j}(u)E^{d,s}_{u,j}\mathbf{f}^{d,s}_{\sigma^{end},u}(j) \tag{8.78}$$

Using Equations 8.74 and 8.78 we obtain the ESS for the exiting features

$$\begin{aligned}\mathbb{E}[\mathbf{F}^{d,s}_{\sigma^{end},u}(\zeta)] &= \sum_{j\in[1,T]}\mathbb{E}[\mathbf{f}^{d,s}_{\sigma^{end},u}(j)\delta[e^{d}_{i-1}\in\zeta]] \\ &= \frac{1}{Z}\sum_{j\in[1,T]}E^{d,s}_{u,j}\mathbf{f}^{d,s}_{\sigma^{end},u}(j)\sum_{i\in[1,j]}\hat{\Lambda}^{d,s}_{i:j}\alpha^{d,s}_{i:j}(u)\end{aligned} \tag{8.79}$$

There is a special case: when $d = 1$ there must be no scanning of i, j but fix $i = 1, j = T$.

8.5 Generalised Viterbi Algorithm

By definition the MAP assignment is the maximiser of the conditional distribution given an observation sequence z

$$\begin{aligned}\zeta^{MAP} &= \arg\max_{\zeta}\Pr(\zeta|z) \\ &= \arg\max_{\zeta}\Phi[\zeta, z]\end{aligned} \tag{8.80}$$

For clarity, let us drop the notation z and assume that it is implicitly there.

The process of computing the MAP assignment is very similar to that of computing the partition function. This similarity comes from the relation between the sum-product and max-product algorithm (a generalisation of the Viterbi algorithm) of Pearl (1988), and from the fact that inside/asymmetric inside procedures described in Section 8.3.2 are essentially a sum-product version. What we need to do is to just convert all the summations into corresponding maximisations. The algorithm is a two-step procedure:

- In the first step the maximum joint potential is computed and local maximum states and ending indicators are saved along the way. These states and ending indicators

are maintained in a *bookkeeper*.

- In the second step we decode the best assignment by *backtracking* through saved local maximum states.

We make use of the contextual decompositions and factorisations from Section 8.3.2.

Notations

This section, with some abuse, uses some slight modifications to the notations used in the rest of the chapter. See Table 8.2 for reference.

Notation	Description
$\Delta_{i:j}^{\max,d,s}$	The optimal potential function of the subset of variables $\zeta_{i:j}^{d,s}$
$\hat{\Delta}_{i:j}^{\max,d,s}$	The 'full' version of $\Delta_{i:j}^{\max,d,s}$
$\alpha_{i:j}^{\max,d,s}(u)$	The optimal potential function of the subset of variables $\zeta_{i:j}^{d,s}(u)$
$\Delta_{i:j}^{\arg,d,s}$	The optimal child u_j^{d+1} of s
$\alpha_{i:j}^{\arg,d,s}(u)$	The optimal child v_{t-1}^{d+1} that transits to $u_{t:j}^{d+1}$ and the time index t.
$\mathcal{I}^d$	The set of optimal 'segments' at each level d.

Table 8.2: Notations used in this section.

We now describe the first step.

8.5.1 Computing the Maximum Joint Potential, Maximal States and Time Indices

As $\Phi[\zeta] = \Phi[\hat{\zeta}_{1:T}^{1,s}]R_{1:T}^{1,s}$ for $s \in S^1$ we have

$$\max_{\zeta} \Phi[\zeta] = \max_{s \in S^1} R_{1:T}^{1,s} \max_{\zeta_{1:T}^{1,s}} \Phi[\hat{\zeta}_{1:T}^{1,s}] \tag{8.81}$$

Now, for a sub-assignment $\zeta_{i:j}^{d,s}$ for $1 \in [1, D-1]$, Equation 8.36 leads to

$$\max_{\zeta_{i:j}^{d,s}} \Phi[\hat{\zeta}_{i:j}^{d,s}] = \max_{u \in S^{d+1}} E_{u,j}^{d,s} \max_{\zeta_{i:j}^{d,s}(u)} \Phi[\hat{\zeta}_{i:j}^{d,s}(u)] \tag{8.82}$$

With some slight abuse of notation we introduce $\Delta_{i:j}^{\max,d,s}$ as the optimal potential function of the subset of variables $\zeta_{i:j}^{d,s}$, and $\alpha_{i:j}^{\max,d,s}(u)$ as the optimal potential function of the subset of variables $\zeta_{i:j}^{d,s}(u)$.

Definition 8. *We define* $\Delta_{i:j}^{\max,d,s}$ *and* $\alpha_{i:j}^{\max,d,s}(u)$ *as follows*

$$\Delta_{i:j}^{\max,d,s} = \max_{\zeta_{i:j}^{d,s}} \Phi[\hat{\zeta}_{i:j}^{d,s}] \tag{8.83}$$

$$\hat{\Delta}_{i:j}^{\max,d,s} = \Delta_{i:j}^{\max,d,s} R_{i:j}^{d,s} \tag{8.84}$$

$$\alpha_{i:j}^{\max,d,s}(u) = \max_{\zeta_{i:j}^{d,s}(u)} \Phi[\hat{\zeta}_{i:j}^{d,s}(u)] \tag{8.85}$$

The Equations 8.81 and 8.82 can be rewritten more compactly as

$$\Phi[\zeta^{MAP}] = \max_{s \in S^1} \hat{\Delta}_{1:T}^{\max,1,s} \tag{8.86}$$

$$\Delta_{i:j}^{\max,d,s} = \max_{u \in S^{d+1}} E_{u,j}^{d,s} \alpha_{i:j}^{\max,d,s}(u) \tag{8.87}$$

for $d \in [1, D-1]$. When $d = D$, we simply set $\Delta_{i:i}^{\max,D,s} = 1$ for all $s \in S^D$ and $i \in [1, T]$.

From the factorisation in Equation 8.31 and 8.32, we have

$$\begin{aligned} \max_{\zeta_{i:j}^{d,s}(u)} \Phi[\hat{\zeta}_{i:j}^{d,s}(u)] = \max \Bigg\{ \Bigg(& \max_{v \in S^{d+1}} \max_{t \in [i+1,j]} R_{t:j}^{d+1,u} A_{v,u,t-1}^{d+1,s} \max_{\zeta_{i:t-1}^{d,s}(v)} \Phi[\hat{\zeta}_{i:t-1}^{d,s}(v)] \times \\ & \times \max_{\zeta_{t:j}^{d+1,u}} \Phi[\hat{\zeta}_{t:j}^{d+1,u}] \Bigg) ; \left(R_{i:j}^{d+1,u} \max_{\zeta_{i:j}^{d+1,u}} \pi_{u,i}^{d,s} \Phi[\hat{\zeta}_{i:j}^{d+1,u}] \right) \Bigg\} \end{aligned} \tag{8.88}$$

and

$$\begin{aligned} \alpha_{i:j}^{\max,d,s}(u) = \max \Bigg\{ & \left(\max_{v \in S^{d+1}} \max_{t \in [i+1,j]} \alpha_{i:t-1}^{\max,d,s}(v) \hat{\Delta}_{t:j}^{\max,d+1,u} A_{v,u,t-1}^{d,s} \right) ; \\ & \left(\hat{\Delta}_{i:j}^{\max,d+1,u} \pi_{u,i}^{d+1,s} \right) \Bigg\} \end{aligned} \tag{8.89}$$

for $d \in [1, D-2]$ and $i < j$. For $d = D-1$, we cannot scan the index t in the interval $[i+1, j]$ because the maximum inside $\Delta_{t:j}^{\max,D,u}$ is only defined at $t = j$. We have the following instead

$$\alpha_{i:j}^{\max,D-1,s}(u) = \max_{v \in S^D} \alpha_{i:j-1}^{\max,D-1,s}(v) \hat{\Delta}_{j:j}^{\max,D,u} A_{v,u,j-1}^{D,s} \tag{8.90}$$

There is a base case for $i = j$, where the context $c = (e_{i-1}^d = 1)$ is active, then

$$\alpha_{i:i}^{\max,d,s}(u) = \hat{\Delta}_{i:i}^{\max,d+1,u} \pi_{u,i}^{d,s} \tag{8.91}$$

Of course, what we are really interested in is not the maximum joint potentials but the optimal states and time indices (or ending indicators). We need some bookkeepers to

hold these quantities along the way. With some abuse of notation let us introduce the symmetric inside bookkeeper $\Delta_{i:j}^{\arg,d,s}$ associated with Equation 8.87, and the asymmetric inside bookkeeper $\alpha_{i:j}^{\arg,d,s}(u)$ associated with Equations 8.89, 8.90 and 8.91.

Definition 9. *We define the symmetric inside bookkeeper* $\Delta_{i:j}^{\arg,d,s}$ *as follows*

$$\Delta_{i:j}^{\arg,d,s} \quad = \quad u^* = \arg\max_{u \in S^{d+1}} E_{u,j}^{d,s} \alpha_{i:j}^{\max,d,s}(u) \tag{8.92}$$

Similarly, we define the asymmetric inside bookkeeper $\alpha_{i:j}^{\arg,d,s}(u)$ *associated with Equation 8.89 for* $d \in [1, D-2]$ *as*

$$\alpha_{i:j}^{\arg,d,s}(u) = (v,t)^* = \arg\max_{t \in [i+1,j], v \in S^{d+1}} \alpha_{i:t-1}^{\max,d,s}(v) \hat{\Delta}_{t:j}^{\max,d+1,u} A_{v,u,t-1}^{d,s} \tag{8.93}$$

if $\max_{v \in S^{d+1}, t \in [i+1,j]} \alpha_{i:t-1}^{\max,d,s}(v) \hat{\Delta}_{t:j}^{\max,d+1,u} A_{v,u,t-1}^{d,s} > \hat{\Delta}_{i:j}^{\max,d+1,u} \pi_{u,i}^{d+1,s}$ *and* $i < j$*; and*

$$\alpha_{i:j}^{\arg,d,s}(u) = \textit{ undefined} \tag{8.94}$$

otherwise. For $d = D-1$*, the* $\alpha_{i:j}^{\arg,d,s}(u)$ *is associated with Equation 8.90*

$$\alpha_{i:j}^{\arg,D-1,s}(u) = \arg\max_{v \in S^D} \alpha_{i:j-1}^{\max,d,s}(v) \hat{\Delta}_{j:j}^{\max,D,u} A_{v,u,j-1}^{d,s} \tag{8.95}$$

The Equations 8.86,8.87,8.89,8.90 and 8.91 provide a recursive procedure to compute maximum joint potential in a bottom-up and left-right manner. Initially we just set $\Delta_{i:i}^{\max,D,s} = 1$ for all $s \in S^D$ and $i \in [1, T]$. The procedure is summarised in Figure 8.14.

8.5.2 Decoding the MAP Assignment

The proceeding of the backtracking process is opposite to that of the max-product. Specifically, we start from the root and proceed in a *top-down* and *right-left* manner. The goal is to identify the right-most segment at each level. Formally, a segment is a triple (s, i, j) where s is the segment label, and i and j are start and end time indices, respectively. From the maximum inside $\Delta_{i:j}^{\max,d,s}$ at level d, we identify the best child u and its ending time j from Equation 8.87. This gives rise to the maximum asymmetric inside $\alpha_{i:j}^{\max,d,s}(u)$. Then we seek for the best child v that transits to u under the same parent s using Equation 8.89. Since the starting time t for u has been identified the ending time for v is $t-1$. We now have a right-most segment (u, t, j) at level $d+1$. The procedure is repeated until we reach the starting time i of the parent s. The backtracking algorithm is summarised in Figure 8.15.

Finally, the generalised Viterbi algorithm is given in Figure 8.16.

Working in log-space to avoid numerical overflow

With long sequence and complex topology we may run into the problem of numerical overflow, i.e. when the numerical value of the maximum joint potential is beyond the number representation of the machine. To avoid this, we can work in the log-space instead, using the monotonic property of the log function. The equations in the log-space are summarised in Table 8.3.

Log-space equations	Ref. equations
$\log \Delta_{i:j}^{\max,d,s} = \max_{u \in S^{d+1}} \{\log E_{u,j}^{d,s} + \log \alpha_{i:j}^{\max,d,s}(u)\}$	Equation 8.87
$\log \alpha_{i:j}^{\max,d,s}(u) = \max \Big\{ \max_{t \in [i+1,j]} \max_{v \in S^{d+1}} \{\log \alpha_{i:t-1}^{\max,d,s}(v) + \log \hat{\Delta}_{t:j}^{\max,d+1,u} + \log A_{v,u,t-1}^{d,s}\}; \log \hat{\Delta}_{i:j}^{\max,d+1,u} + \log \pi_{u,i}^{d+1,s} \Big\}$	Equation 8.89
$\log \alpha_{i:j}^{\max,D-1,s}(u) = \max_{v \in S^D} \{\log \alpha_{i:j-1}^{\max,D-1,s}(v) + \log \hat{\Delta}_{j:j}^{\max,D,u} + \log A_{v,u,j-1}^{D,s}\}$	Equation 8.90
$\log \alpha_{i:i}^{\max,d,s}(u) = \log \hat{\Delta}_{i:i}^{\max,d+1,u} + \log \pi_{u,i}^{d,s}$	Equation 8.91

Table 8.3: MAP equations in the log-space.

8.6 Complexity Analysis

It can be seen from Figure 8.10 and 8.11 that the AIO algorithm takes $\mathcal{O}(T^3)$ time to compute all the inside and outside masses and the partition function for $D > 3$. For $D = 3$, the complexity is $\mathcal{O}(T^2)$.

For the ESS (Section 8.4), using the calculated building blocks, it is not difficult to see that the the ESS for state-persistence features takes $\mathcal{O}(T^2)$ times (see Equation 8.60), the transition $\mathcal{O}(T^3)$ (see Equation 8.67), the initialisation $\mathcal{O}(T^2)$ (see Equation 8.73) and the ending $\mathcal{O}(T^2)$ (see Equation 8.79). The overall complexity is therefore $\mathcal{O}(T^3)$.

The MAP estimation in Section 8.5 is basically the max-product version of the sum-product algorithm used in the AIO, thus it has the same cubic time complexity as the partition function.

8.7 Closing Remarks

In this chapter, we have presented a major extension to the theory of CRFs by proposing a novel model called Hierarchical Conditional Random Field to deal with recursive sequential data. The model is capable of representing complex hierarchy with flexible structures

and encoding rich domain knowledge in a discriminative framework.

We have developed a graphical model-like dynamic representation of the HCRF. This appears similar to the DBN representation of the HHMMs in (Murphy and Paskin, 2002), and somewhat resembles a dynamic factor graph (Kschischang *et al.*, 2001). However, it is not exactly the standard graphical model because the contextual cliques in HCRFs are not fixed during inference. In addition, the ability to represent state duration in the HCRFs is not replicated in the HHMMs.

For learning and inference we have introduced an efficient Asymmetric Inside Outside algorithm that exhibits cubic time complexity. We have shown how to compute various essential quantities such as the partition function, the MAP assignment and the expected sufficient statistics of feature functions. There are other quantities and special cases we have not covered in the main text of the chapter, but they are included as appendices. These include state marginals $\Pr(x_t^d)$ (Appendix A.4), the 'mirrored' AIO (Appendix A.5), and the proof that the semi-Markov CRF of (Sarawagi and Cohen, 2004) is a special case of our HCRF (Appendix A.6).

In the next chapter we will address many practical issues including numerical overflow, partial labels and efficiency and demonstrate the HCRFs framework in some applications.

Input: D, T, all the potential function values.
Output: the bookkeepers;
$\Delta_{1:T}^{\arg,1,s}$, for $s \in S^1$ and $1 \leq i \leq j \leq T$;
$\Delta_{i:j}^{\arg,d,s}$, for $d \in [2, D-1]$, $s \in S^d$;
$\Delta_{i:i}^{\arg,D,s}$ for $s \in S^D$ and $i \in [1, T]$;
$\alpha_{i:j}^{\arg,d,s}(u)$ for $d \in [1, D-1]$, $u \in S^{d+1}$ and $1 \leq i \leq j \leq T$

/ Initialisation */*
$\Delta_{i:i}^{\max,D,s} = 1$ for all $i \in [1, T]$ and $s \in S^D$
/ At the level d=D-1 */*
For $i = 1, 2, ..., T$
  For $j = i, i+1, ..., T$
    Compute $\alpha_{i:j}^{\max,D-1,s}(u)$ using Equation 8.90 and
      $\alpha_{i:j}^{\arg,D-1,s}(u)$ using Equation 8.95
    Compute $\Delta_{i:j}^{\max,D-1,s}$ using Equation 8.87 and
      $\Delta_{i:j}^{\arg,D-1,s}$ using Equation 8.92
  EndFor
EndFor
/ The main recursion loops: bottom-up and forward */*
For $d = D-2, D-3, ..., 1$
  For $i = 1, 2, ..., T$
    For $j = i, i+1, ..., T$
      If $j = i$
        Compute $\alpha_{i:i}^{\max,d,s}(u)$ using Equation 8.91
      Else
        Compute $\alpha_{i:j}^{\max,d,s}(u)$ using Equation 8.89 and
          $\alpha_{i:i}^{\arg,d,s}(u)$ using Equation 8.93
      EndIf
      If $d > 1$
        Compute $\Delta_{i:j}^{\max,d,s}$ using Equation 8.87 and
          $\Delta_{i:j}^{\arg,d,s}$ using Equation 8.92
      EndIf
    EndFor
  EndFor
EndFor
Compute $\Delta_{1:T}^{\max,1,s}$ using Equation 8.87 and
  $\Delta_{1:T}^{\arg,1,s}$ using Equation 8.92

Figure 8.14: Computing the bookkeepers.

Input: D, T, all the filled bookkeepers.
Output: the optimal assignment ζ^{MAP}

$s^* = \arg\max_{s \in S^1} \hat{\Delta}_{1:T}^{\max,1,s}$
Initialise triple buckets $\mathcal{I}^1 = \{(s^*, 1, T)\}$ and $\mathcal{I}^d = \{\}$ for $d \in [2, D]$
For $d = 1, 2, ..., D-1$
  For each triple (s^*, i, j) in $\mathcal{I}^d$
    Let $u^* = \Delta_{i:j}^{\arg,d,s^*}$
    For $i \leq j$
      If $\alpha_{i:j}^{\arg,d,s^*}(u^*)$ is defined **Then**
        $(t^*, v^*) = \alpha_{i:j}^{\arg,d,s^*}(u^*)$
        Add the triple (v^*, t^*, j) to $\mathcal{I}^{d+1}$ and Set $j = t^* - 1$ and $u^* = v^*$
      Else
        Add the triple (u^*, i, j) to $\mathcal{I}^{d+1}$ and Break this loop
      EndIf
    EndFor
  EndFor
EndFor
For each stored triple (s^*, i, j) in the bucket $\mathcal{I}^d$, for $d \in [1, D]$,
create a corresponding set of variables $(x_{i:j}^d = s^*, e_{i-1}^d = 1, e_j^d = 1, e_{i:j-1}^d = 0)$.
The joining of these sets is the optimal assignment ζ^{MAP}

Figure 8.15: Backtracking for optimal assignment (nested Markov blankets).

Input: D, T, all the potential function values.
Output: the optimal assignment ζ^{MAP}

Run the bottom-up discrete optimisation procedure described in Figure 8.14.
Run the top-down backtracking procedure described in Figure 8.15.

Figure 8.16: The generalised Viterbi algorithm.

Chapter 9

Extensions to HCRF and Applications

9.1 Introduction

In Chapter 8 we have introduced a novel model for recursive sequential data and derived a polynomial time algorithm called Asymmetric Inside Outside (AIO) for learning and inference. However, the AIO has some important drawbacks that prevent scalability.

First, the computation may be unstable because the magnitude of the partition function, which is the sum of exponentially many positive potentials, increases exponentially fast in the sequence length T, and thus goes beyond the numerical capacity of most machines for moderate T.

Second, the AIO algorithm cannot deal with the situation when the training data is partially labeled. On the other hand, the generalised Viterbi is based on the assumption that the test data does not have any labels. It does not make use of partial labels that may be obtained externally. We term the process of training with partial labels *partial-supervision*, and the process of inference with partial labels *constrained inference*. Both the processes require the construction of appropriate constrained inference algorithms.

The third problem is that the AIO generally takes $\mathcal{O}(T^3)$ time, which quickly becomes impractical for non-trivial problems with large T (e.g. about 100 or larger). Approximation techniques that trade some accuracy for speed must be formulated.

In this chapter we present a number of extensions to the basic HCRF to address these three problems

- Following the work of (Bui *et al.*, 2004) we derive in Section 9.2 a scaling algorithm that is effective in reducing numerical overflow.

- In Section 9.3 we extend the AIO algorithm and the generalised Viterbi algorithm to cope with arbitrary partial labels.
- In Section 9.4 we derive an efficient approximate inference scheme based on Rao-Blackwellisation (e.g. see (Casella and Robert, 1996)), Gibbs sampling (e.g. see Section 2.4.6).
- An approximate learning algorithm based on pseudo-likelihood (see Section 3.5.1.2) is presented in Section 9.5.
- Section 9.6 exploits a special case of exponential distribution of state duration, and introduces a factor-graph representation which enables efficient sum-product inference (see Section 2.4.8).

The rest of the chapter is organised as follows

- In Section 9.7.2 we will discuss the unconditional case of HCRF, and how it relates to the HHMM.
- Section 9.8 will evaluate the effectiveness of HCRFs in two applications: human activity recognition using the same data as in Chapters 6 and 7, and noun-phrase chunking (Sang and Buchholz, 2000). The HCRFs are run under varying conditions and tested against several competitive CRFs.
- Section 9.9 concludes the chapter.

Notations

The current chapter makes use of several mathematical notations beside those introduced in the previous chapter. These are included in Table 9.1 for reference.

Notation	Meaning
l_t	Level which the transition occurs at time t
κ	Scaling factors
$\psi(.)$	Potential functions
$\alpha_t(.)$	Rao-Blackwellised forward
$\beta_t(.)$	Rao-Blackwellised backward

Table 9.1: Notations used in this chapter.

9.2 Numerical Overflow and the Scaling Algorithm

In this section we present a scaling method to reduce numerical overflow. The idea can be traced back to belief propagation by normalising (or reducing) messages at each step (see Equation 2.75). In the context of HHMMs with which the numerical *underflow* problem is associated, a similar idea has been proposed in (Bui *et al.*, 2004). Fortunately, the overflow problem in the undirected models is closely related to the underflow issue of the directed counterparts and thus, a similar strategy can be used.

9.2.1 Scaling the Symmetric/Asymmetric Inside Masses

Before proceeding to algorithmic details let us revisit Equation 8.37. If we scale down the asymmetric inside mass $\alpha^{d,s}_{i:j}(u)$ by a factor $\kappa_j > 1$, i.e.

$$\alpha'^{d,s}_{i:j}(u) \leftarrow \frac{\alpha^{d,s}_{i:j}(u)}{\kappa_j} \tag{9.1}$$

then the symmetric inside mass $\Delta^{d,s}_{i:j}$ is also scaled down by the same factor. Similarly, as we can see from Equation 8.33 that

$$\alpha^{d,s}_{i:j}(u) = \sum_{t=i+1}^{j} \sum_{v \in S^{d+1}} \alpha^{d,s}_{i:t-1}(v) \hat{\Delta}^{d+1,u}_{t:j} A^{d,s}_{v,u,t-1} + \hat{\Delta}^{d+1,u}_{i:j} \pi^{d,s}_{u,i}$$

where $\hat{\Delta}^{d+1,u}_{t:j} = \Delta^{d+1,u}_{t:j} R^{d+1,u}_{t:j}$, if $\Delta^{d+1,u}_{t:j}$ for $t \in [1, j]$ is reduced by κ_j, then $\alpha^{d,s}_{i:j}$ is also reduced by the same factor. In addition, using the set of recursive relations in Equations 8.33 and 8.37, any reduction at the bottom level of $\Delta^{D,s}_{j:j}$ will result in the reduction of the symmetric inside mass $\Delta^{d,s}_{i:j}$ and of the asymmetric inside mass $\alpha^{d,s}_{i:j}(u)$, for $d < D$, by the same factor.

Suppose $\Delta^{D,s}_{i:i}$ for all $i \in [1, j]$ is reduced by a factor of $\kappa_i > 1$, the quantities $\Delta^{d,s}_{1:j}$ and $\alpha^{d,s}_{1:j}(u)$ will be reduced by a factor of $\prod_{i=1}^{j} \kappa_i$. That is

$$\hat{\Delta}'^{d,s}_{1:j} \leftarrow \frac{\hat{\Delta}^{d,s}_{1:j}}{\prod_{i=1}^{j} \kappa_i} \tag{9.2}$$

$$\alpha'^{d,s}_{1:j}(u) \leftarrow \frac{\alpha^{d,s}_{1:j}(u)}{\prod_{i=1}^{j} \kappa_i} \tag{9.3}$$

It follows immediately from Equation 8.22 that the partition function is scaled down by a

factor of $\prod_{i=1}^{T} \kappa_i$

$$Z' = \sum_{s \in S^1} \hat{\Delta}^{'1,s}_{1:T} = \frac{Z}{\prod_{j=1}^{T} \kappa_j} \tag{9.4}$$

where $\hat{\Delta}^{'1,s}_{1:T} = \Delta^{'1,s}_{1:T} B^{1,s}_{1:T}$. Clearly, we should deal with the log of this quantity to avoid numerical overflow. Thus, the log-partition function can be computed as

$$\log(Z) = \log \sum_{s \in S^1} \hat{\Delta}^{'1,s}_{1:T} + \sum_{j=1}^{T} \log \kappa_j \tag{9.5}$$

where $\Delta^{'1,s}_{1:T}$ has been scaled appropriately.

One question is how to choose the set of meaningful scaling factors $\{\kappa_j\}_1^T$. The simplest way is to choose a relatively large number for all scaling factors but making the right choice is not straightforward. Here we describe a more natural way to do so. Assume that we have chosen all the scaling factors $\{\kappa_i\}_1^{j-1}$. Using the original Equations 8.33, 8.34, and 8.35, where all the sub-components have been scaled appropriately, we compute the *partially-scaled* inside mass $\Delta^{''d,s}_{i:j}$ for $d \in [2, D]$ and asymmetric inside mass $\alpha^{''d,s}_{i:j}(u)$, for $d \in [1, D-1]$ and $i \in [1, j]$. Then the scaling factor at time j is computed as

$$\kappa_j = \sum_{s,u} \alpha^{''1,s}_{1:j}(u) \tag{9.6}$$

The next step is to rescale all the partially-scaled variables:

$$\alpha^{'d,s}_{i:j}(u) \leftarrow \frac{\alpha^{''d,s}_{i:j}(u)}{\kappa_j} \text{ for } s \in S^d, d \in [1, D-1] \tag{9.7}$$

$$\Delta^{'d,s}_{i:j} \leftarrow \frac{\Delta^{''d,s}_{i:j}}{\kappa_j} \text{ for } s \in S^d, d \in [2, D-1] \tag{9.8}$$

$$\Delta^{'D,s}_{j:j} \leftarrow \frac{\Delta^{''D,s}_{j:j}}{\kappa_j} \text{ for } s \in S^D \tag{9.9}$$

where $i \in [1, j]$.

9.2.2 Scaling the Symmetric/Asymmetric Outside Masses

In a similar fashion we can work out the set of factors from the derivation of symmetric/asymmetric outside masses since these masses solely depend on the inside masses as building blocks. In other words, after we finish scaling the inside masses we can compute the scaled outside masses directly, using the same set of equations described in Sec-

tion 8.3.3.

The algorithm is summarised in Figure 9.1. Note that the order of performing the loops in this case is different from that in Figure 8.10.

Input: D, T and all the contextual potentials.
Output: Scaled quantities: inside/asymmetric inside masses, outside/asymmetric outside masses.

For $j = 1, 2, .., T$
  Compute $\alpha_{1:j}^{d,s}(u)$, $d \in [1, D-1]$ using Equations 8.33, 8.34 and 8.35
  Compute κ_j using Equation 9.6
  Rescale $\alpha_{1:j}^{1,s}(u)$ using Equation 9.7
  For $i = 1, 2, .., j$
    For $d = 2, 3, .., D-1$
      Rescale $\alpha_{i:j}^{d,s}(u)$ using Equation 9.7
      Rescale $\Delta_{i:j}^{d,s}$ using Equation 9.8
    EndFor
  EndFor
  Rescale $\Delta_{j:j}^{D,s}$ using Equation 9.9
EndFor
Compute true log-partition function using Equation 9.5.
Compute the outside/asymmetric outside masses using the scaled inside/asymmetric inside masses instead of the original inside/asymmetric inside in Equations 8.40 and 8.44.

Figure 9.1: Scaling algorithm to avoid numerical overflow.

9.3 Partially Observed Data and Algorithms with Constraints

So far we have assumed that training data is fully labeled, and that testing data does not have any labels. In this section we extend the AIO to handle the cases in which these assumptions do not hold. Specifically, it may happen that the training data is not completely labeled, possibly due to lack of labeling resources. In this case, the learning algorithm should be robust enough to handle missing labels. On the other hand, during inference, we may partially obtain high quality labels from external sources. This requires the inference algorithm to be responsive to that data.

9.3.1 The Constrained AIO algorithm

In this section we consider the general case when $\zeta = (\vartheta, h)$, where ϑ is the visible set labels, and h the hidden set. Since our HCRF is also an exponential model it shares the same computation required for general CRFs (Equations 3.11 and 3.12). We have to compute four quantities: the partial log-partition function $Z(\vartheta, z)$, the partition function $Z(z)$, the 'constrained' ESS $\mathbb{E}_{h|\vartheta,z}[\mathbf{F}(\vartheta, h, z)]$, and the 'free' ESS $\mathbb{E}_{\zeta|z}[\mathbf{F}(\zeta, z)]$. The partition function and the 'free' ESS has been computed in Sections 8.3 and 8.4, respectively. This section describes the other two quantities.

Let the set of visible labels be $\vartheta = (\widetilde{x}, \widetilde{e})$ where $\widetilde{x}$ is the visible set of state variables and $\widetilde{e}$ is the visible set of ending indicators. The basic idea is that we have to modify procedures for computing the building blocks such as $\Delta_{i:j}^{d,s}$ and $\alpha_{i:j}^{d,s}(u)$, to address constraints imposed by the labels. For example, $\Delta_{i:j}^{d,s}$ implies that the state s at level d starts at i and persists till terminating at j. Then, if any labels (e.g. there is an $\widetilde{x}_k^d \neq s$ for $k \in [i, j]$) are seen, causing this assumption to be inconsistent, $\Delta_{i:j}^{d,s}$ will be zero. Therefore, in general, the computation of each building block is multiplied by an identity function that enforces the consistency between these labels and the required constraints for computation of that block. As an example, we consider the computation of $\Delta_{i:j}^{d,s}$ and $\alpha_{i:j}^{d,s}(u)$.

The symmetric inside mass $\Delta_{i:j}^{d,s}$ is consistent only if all of the following conditions are satisfied:

1. If there are state labels $\widetilde{x}_k^d$ at level d within the interval $[i, j]$, then $\widetilde{x}_k^d = s$,
2. If there is any label of ending indicator $\widetilde{e}_{i-1}^d$, then $\widetilde{e}_{i-1}^d = 1$,
3. If there is any label of ending indicator $\widetilde{e}_k^d$ for some $k \in [i, j-1]$, then $\widetilde{e}_k^d = 0$, and
4. If any ending indicator $\widetilde{e}_j^d$ is labeled, then $\widetilde{e}_j^d = 1$.

These conditions are captured by using the following identity function:

$$\mathbb{I}[\Delta_{i:j}^{d,s}] = \delta[\widetilde{x}_{k\in[i,j]}^d = s]\delta[\widetilde{e}_{i-1}^d = 1]\delta[\widetilde{e}_{k\in[i:j-1]}^d = 0]\delta[\widetilde{e}_j^d = 1] \tag{9.10}$$

When labels are observed, Equation 8.37 is thus replaced by

$$\Delta_{i:j}^{d,s} = \mathbb{I}[\Delta_{i:j}^{d,s}]\left(\sum_{u \in S^{d+1}} \alpha_{i:j}^{d,s}(u) E_{u,j}^{d,s}\right) \tag{9.11}$$

Note that we do not need to explicitly enforce the state consistency in the summation over u since in the bottom-up and left-right computation, $\alpha_{i:j}^{d,s}(u)$ is already computed and contributes to the sum only if it is consistent.

Analogously, the asymmetric inside mass $\alpha_{i:j}^{d,s}(u)$ is consistent if all of the following conditions are satisfied:

1. The first three conditions for the symmetric inside mass $\Delta_{i:j}^{d,s}$ hold,
2. If the state at level d at time j is labeled, it must be u, and
3. If any ending indicator $\widetilde{e}_j^{d+1}$ is labeled, then $\widetilde{e}_j^{d+1} = 1$.

These conditions are captured by the identity function

$$\mathbb{I}[\alpha_{i:j}^{d,s}(u)] = \delta[\widetilde{x}_{k\in[i,j]}^{d} = s]\delta[\widetilde{e}_{i-1}^{d} = 1]\delta[\widetilde{e}_{k\in[i:j-1]}^{d} = 0]\delta[\widetilde{x}_{j}^{d+1} = u]\delta[\widetilde{e}_{j}^{d+1} = 1] \quad (9.12)$$

Thus Equation 8.33 becomes

$$\alpha_{i:j}^{d,s}(u) = \mathbb{I}[\alpha_{i:j}^{d,s}(u)]\left(\sum_{k=i+1}^{j}\sum_{v\in S^{d+1}} \alpha_{i:k-1}^{d,s}(v)\hat{\Delta}_{k:j}^{d+1,u}A_{v,u,k-1}^{d,s} + \hat{\Delta}_{i:j}^{d+1,u}\pi_{u,i}^{d+1,s}\right) \quad (9.13)$$

Note that we do not need to explicitly enforce the state consistency in the summation over v and time consistency in the summation over k since in bottom-up computation, $\alpha_{i:j}^{d,s}(u)$ and $\Delta_{k:j}^{d+1,u}$ are already computed and contribute to the sum only if they are consistent. Finally, the constrained partition function $Z(\vartheta, z)$ is computed using Equation 8.22 given that the inside mass is consistent with the observations.

Other building blocks, such as the symmetric outside mass $\Lambda_{i:j}^{d,s}$ and the asymmetric outside mass $\lambda_{i:j}^{d,s}(u)$, are computed in an analogous way. Since $\Lambda_{i:j}^{d,s}$ and $\Delta_{i:j}^{d,s}$ are complementary and they share (d, s, i, j), the same indicator function $\mathbb{I}[\Delta_{i:j}^{d,s}]$ can be applied. Similarly, the pair asymmetric inside mass $\alpha_{i:j}^{d,s}(u)$ and asymmetric outside mass $\lambda_{i:j}^{d,s}(u)$ are complementary and they share d, s, i, j, u, thus the same indicator function $\mathbb{I}[\alpha_{i:j}^{d,s}(u)]$ can be applied.

Once all constrained building blocks have been computed they can be used to calculate constrained ESS as in Section 8.4 without any further modifications. The only difference is that we need to replace the partition function $Z(z)$ by the constrained version $Z(\vartheta, z)$.

9.3.2 The Constrained Viterbi Algorithm

Recall that in the Generalised Viterbi Algorithm described in Section 8.5 we want to find the most probable configuration $\zeta^{MAP} = \arg\max_{\zeta} \Pr(\zeta|z)$. When some variables ϑ of ζ are labeled, it is not necessary to estimate them. The task is now to estimate the most

probable configuration of the hidden variables h given the labels:

$$\begin{aligned} h^{MAP} &= \arg\max_{h} \Pr(h|\vartheta, z) \\ &= \arg\max_{h} \Pr(h, \vartheta|z) \\ &= \arg\max_{h} \Phi[h, \vartheta, z] \end{aligned} \tag{9.14}$$

It turns out that the constrained MAP estimation is identical to the standard MAP except that we have to respect the labeled variables ϑ.

Since the Viterbi algorithm is just the max-product version of the AIO, the constrained Viterbi can be modified in the same manner as in the constrained AIO (Section 9.3.1). Specifically, for each auxiliary quantities such as $\Delta_{i:j}^{max,s}$ and $\alpha_{i:j}^{max,s}(u)$, we need to maintain a set of indicator functions that ensures the consistency with labels. Equations 9.10 and 9.11 become

$$\begin{aligned} \mathbb{I}[\Delta_{i:j}^{max,d,s}] &= \delta[\widetilde{x}_{k\in[i,j]}^{d} = s]\delta[\widetilde{e}_{i-1}^{d} = 1]\delta[\widetilde{e}_{k\in[i:j-1]}^{d} = 0]\delta[\widetilde{e}_{j}^{d} = 1] \\ \Delta_{i:j}^{max,d,s} &= \mathbb{I}[\Delta_{i:j}^{max,d,s}]\left(\max_{u\in S^{d+1}} \alpha_{i:j}^{max,d,s}(u) E_{u,j}^{d,s}\right) \end{aligned} \tag{9.15}$$

Likewise, we have the modifications to Equation 9.12 and Equation 9.13, respectively.

$$\begin{aligned} \mathbb{I}[\alpha_{i:j}^{max,d,s}(u)] &= \delta[\widetilde{x}_{k\in[i,j]}^{d} = s]\delta[\widetilde{e}_{i-1}^{d} = 1]\delta[\widetilde{e}_{k\in[i:j-1]}^{d} = 0]\delta[\widetilde{x}_{j}^{d+1} = u]\delta[\widetilde{e}_{j}^{d+1} = 1] \\ \alpha_{i:j}^{max,d,s}(u) &= \mathbb{I}[\alpha_{i:j}^{max,d,s}(u)] \max\left\{ \max_{k\in[i+1,j]} \max_{v\in S^{d+1}} \alpha_{i:k-1}^{max,d,s}(v) \hat{\Delta}_{k:j}^{max,d+1,u} A_{v,u,k-1}^{d,s}; \right. \\ &\qquad \left. \hat{\Delta}_{i:j}^{max,d+1,u} \pi_{u,i}^{d+1,s} \right\} \end{aligned} \tag{9.16}$$

Other tasks in the Viterbi algorithm including bookkeeping and backtracking are identical to those described in Section 8.5.

9.3.3 Complexity Analysis

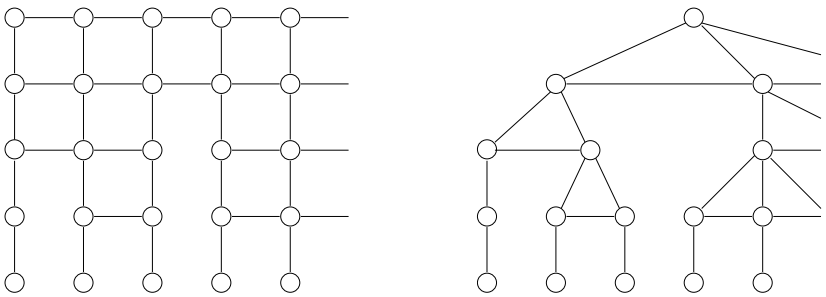
The complexity of the constrained AIO and constrained Viterbi has an upper bound of $\mathcal{O}(T^3)$, when no labels are given. It also has a lower bound of $\mathcal{O}(T)$ when all ending indicators are known and the model reduces to the standard tree-structured graphical model. In general, the complexity decreases as more labels are available, and we can expect a sub-cubic time behaviour.

9.4 Approximate Inference using Rao-Blackwellised Gibbs Sampling

Recall that the AIO algorithm derived in Chapter 8 generally takes $\mathcal{O}(T^3)$ time. This quickly becomes impractical for non-trivial problems with large T (e.g. about 100 or larger). In this section we develop an approximation scheme for inference of the HCRFs that may help improve the speed. The main idea is to combine *Gibbs sampling* (e.g. see Section 2.4.6) and *Rao-Blackwellisation* (e.g. see (Casella and Robert, 1996)). Before proceeding into algorithmic details let us make several observations that simplify the HCRF computation

- The set of ending indicators $e_{1:T}^{1:D}$ can be made simpler by noticing that for each time step there is only one transition at a certain level because all the states above it must remain unchanged, and all the states below it must finish. Thus, the entire slice of indicators $e_t^{1:D}$ can be replaced by a single variable $l_t \in [2, D]$, where l_t is the level at which a transition occurs at time t.
- The complete free variable set is now $(x_{1:T}^{1:D}, l)$ which has two components, the subset of state variables $x_{1:T}^{1:D}$ and the subset of transition indicators $l_{1:T-1}$. Under the restrictions of hierarchical consistency, *when all the transition indicators are known, the entire $x_{1:T}^{1:D}$ can be collapsed into a Markov tree* (Figure 9.2b).
- It is well-known that inference in tree structures is efficient (see Section 2.4.5), and marginalising out all the state variables takes linear time with respect to number of tree edges. In our HCRF case with observed ending indicators, the time is $\mathcal{O}(DT)$.

For presentation clarity, we will use x as a shorthand for $x_{1:T}^{1:D}$, x_t as a shorthand for $x_t^{1:D}$, and l as a shorthand for $l_{1:T-1}$.



(a) (b)

Figure 9.2: An HCRF with known l: (a) links between unrelated states are removed, and (b) the collapsed version into a Markov tree.

9.4.1 Rao-Blackwellised Gibbs Sampling

Rao-Blackwellisation is a technique that can improve the quality of sampling methods by only sampling some variables and marginalising out the rest. In our HCRFs, for example, we can sample l and marginalise out x

$$\begin{aligned} l &\sim \Pr(l) \\ &= \sum_x \Pr(x, l) \\ &= \frac{1}{Z} Z(l) \end{aligned} \tag{9.17}$$

where $Z(l) = \sum_x \Phi[x, l]$. However, there are problems with this strategy. First, computing $\Pr(l)$ still requires Z, which is expensive. Second, the size of state space of $\Pr(l)$ is $(D-1)^{T-1}$, which is difficult to sample directly.

Fortunately, Gibbs sampling allows us to sample l not from $\Pr(l)$ but from the local distributions

$$l_t \sim \Pr(l_t|l_{-t}) \text{ for } t \in [1, T-1], l_t \in [2, D] \tag{9.18}$$

where $l_{-t} = \{l \backslash l_t\} = l_{1:t-1,t+1:T}$, and

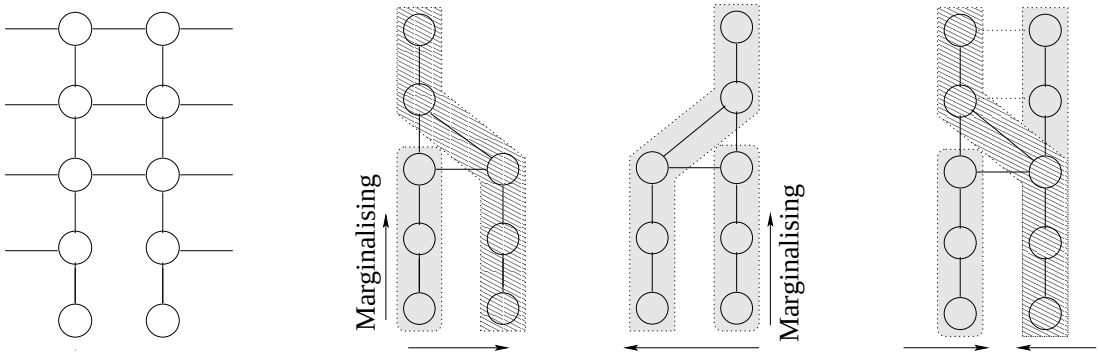
$$\begin{aligned} \Pr(l_t|l_{-t}) &= \frac{\Pr(l)}{\sum_{l'_t} \Pr(l'_t, l_{-t})} \\ &= \frac{Z(l)}{\sum_{l'_t} Z(l'_t, l_{-t})} \end{aligned} \tag{9.19}$$

The main efficiency comes from the fact that, as we will show in Section 9.4.3, *$Z(l)$ and therefore, $\Pr(l_t|l_{-t})$, can be computed exactly in linear time*.

In what follows we borrow the idea of *walking chain* from (Bui *et al.*, 2002) in the context of the Abstract Hidden Markov Model (AHMM) and adapt it to our HCRFs. The main source of complication in the adaption is that the HCRF is capable of modelling duration whilst the AHMM is not. Moreover, the HCRF is strictly nested whilst the AHMM is not. As the HCRF is undirected its factorisation of potentials does not have any probabilistic interpretation as in the AHMM.

Given l, the temporal evolution of the HCRFs can be visualised as a walking chain, either moving forward (Figure 9.3b), or backward (Figure 9.3c). The height of the forward leg at time t corresponds to the transition level l_t. Since the states above l_t are unchanged at t, the 'body' of the walking chain is copied from the previous time index.

In the proposition below we show that the walking chain can be flattened into a standard



(a) A HCRF fragment (b) Forward (c) Backward (d) Merge

Figure 9.3: The walking chains. Given a fragment of the HCRF at time t, the states above the transition level l_t stay unchanged so they can be collapsed into a single node.

sequential chain.

Proposition 6. *Given l, the joint potential $\Phi[x, l]$ can be factorised as follows*

$$\Phi[x, l] = \prod_{t \in [1, T-1]} \varphi_t(x_t, x_{t+1}|l) \tag{9.20}$$

where $\varphi_t(x_t, x_{t+1}|l)$ are non-negative functions.

Proof: This can be derived by construction. Indeed there is more than one way to do this. Let

$$\hat{\varphi}_t(x_t, x_{t+1}|l) = \left[\prod_{d=l_t}^{D-1} E_{u,t}^{d,s}\right] A_{u,v,t}^{l_t,s} \left[\prod_{d=l_t}^{D-1} \pi_{v,t+1}^{d,s}\right] \tag{9.21}$$

where we have used $E_{u,t}^{d,s}$ as a shorthand for $E_{x_t^{d+1},t}^{d,x_t^d}$, $A_{u,v,t}^{l_t,s}$ as a shorthand for $A_{x_t^{l_t},x_{t+1}^{l_t},t}^{l_t,x_{t+1}^{l_t-1}}$, and $\pi_{v,t+1}^{d,s}$ for $\pi_{x_{t+1}^{d+1},t+1}^{d,x_{t+1}^d}$. Now let

$$\overrightarrow{\varphi}_1(x_1, x_2|l) = \hat{\varphi}_1(x_1, x_2|l) \left[\prod_{d=l_1}^{D} R_{1:1}^{d,s}\right] \left[\prod_{d=1}^{D-1} \pi_{u,1}^{d,s}\right] \tag{9.22}$$

$$\begin{aligned} \overrightarrow{\varphi}_{T-1}(x_{T-1}, x_T|l) &= \hat{\varphi}_{T-1}(x_{T-1}, x_T|l) \left[\prod_{d=1}^{D-1} E_{v,T}^{d,s}\right] \times \\ &\quad \times \left[\prod_{d=l_{T-1}}^{D} R_{i:T-1}^{d,s}\right] \left[\prod_{d=1}^{l_{T-1}-1} R_{i:T}^{d,s}\right] \left[\prod_{d=l_{T-1}}^{D} R_{T:T}^{d,s}\right] \end{aligned} \tag{9.23}$$

$$\overrightarrow{\varphi}_t(x_t, x_{t+1}|l) = \hat{\varphi}_t(x_t, x_{t+1}|l) \left[\prod_{d=l_t}^{D} R_{i:t}^{d,s}\right], \forall t \in [2, T-2] \tag{9.24}$$

where we have used $R_{i:t}^{d,s}$ as a shorthand for $R_{i:t}^{d,x_t^d}$, and i is the starting time of the seg-

ment of level d that ends at t. We have assumed that the computation proceeds from left to right and the index i is recorded for each level d along the way (e.g. whenever the initialisation potential $\pi_{u,i}^{d,s}$ is triggered). Then the first construction is completed by setting $\varphi_t(x_t, x_{t+1}|l) = \overrightarrow{\varphi}_t(x_t, x_{t+1}|l)$.

Alternatively, let

$$\begin{aligned}
\overleftarrow{\varphi}_1(x_1, x_2|l) &= \hat{\varphi}_1(x_1, x_2|l)\Big[\prod_{d=1}^{D-1} \pi_{u,1}^{d,s}\Big] \times \\
&\quad \times \Big[\prod_{d=l_1}^{D} R_{2:j}^{d,s}\Big]\Big[\prod_{d=1}^{l_1-1} R_{1:j}^{d,s}\Big]\Big[\prod_{d=l_1}^{D} R_{1:1}^{d,s}\Big] \qquad (9.25)\\
\overleftarrow{\varphi}_{T-1}(x_{T-1}, x_T|l) &= \hat{\varphi}_{T-1}(x_{T-1}, x_T|l)\Big[\prod_{d=l_{T-1}}^{D} R_{T:T}^{d,s}\Big]\Big[\prod_{d=1}^{D-1} E_{v,T}^{d,s}\Big] \qquad (9.26)\\
\overleftarrow{\varphi}_t(x_t, x_{t+1}|l) &= \hat{\varphi}_t(x_t, x_{t+1}|l), \Big[\prod_{d=l_t}^{D} R_{t+1:j}^{d,s}\Big] \forall t \in [2, T-2] \qquad (9.27)
\end{aligned}$$

where j in $R_{t+1:j}^{d,s}$ is the ending time of the segment that starts at $t+1$ and level d. We have assumed that the computation proceeds from right to left and the index j is recorded for each level d along the way (e.g. whenever the state ending potential $E_{u,j}^{d,s}$ is triggered). Then we have another construction by setting $\varphi_t(x_t, x_{t+1}|l) = \overleftarrow{\varphi}_t(x_t, x_{t+1}|l)$ ■

Remark: Proposition 6 suggests that computation in the HCRFs, when l is known, should be be similar to that on Markov chains, because the factorisation of Equation 9.20 is a case of Markov chain factorisation of Equation 2.47. The complication is that the state space of x_t in the case of HCRFs is not usually small and that the straightforward forward-backward procedure described in Section 2.4.3 is not applicable. This will be the subject of the next subsection.

9.4.2 Rao-Blackwellised Forward/Backward

Let $\Phi_{\alpha,t}[x_{1:t}|l_{1:t-1}]$ be the product of all contextual clique potentials that are enabled in the corresponding contexts caused by $l_{1:t-1}$. More specifically, we have

$$\Phi_{\alpha,t}[x_{1:t}|l_{1:t-1}] = \prod_{i \in [1,t-1]} \overrightarrow{\varphi}_i(x_i, x_{i+1}|l) \qquad (9.28)$$

where $\overrightarrow{\varphi}_i(x_i, x_{i+1}|l)$ is defined in Equations 9.22-9.24.

Similarly, let $\Phi_{\beta,t}[x_{t:T}|l_{t:T-1}]$ be the product of all contextual clique potentials that are

enabled in the corresponding contexts caused by $l_{t:T-1}$. More specifically,

$$\Phi_{\beta,t}[x_{t:T}|l_{t:T-1}] = \prod_{j\in[t,T-1]} \overleftarrow{\varphi}_j(x_j, x_{j+1}|l) \tag{9.29}$$

where $\overleftarrow{\varphi}_j(x_j, x_{j+1}|l)$ is defined in Equations 9.25-9.27.

With some abuse of notations, denote by $\alpha_t(x_t|l_{1:t-1})$ the *RB-Forward*, and $\beta_t(x_t|l_{t:T-1})$ the *RB-Backward*.

Definition 10. *Let $\alpha_1(x_1) = 1$ and $\beta_T(x_T) = 1$. For general time indices the RB-Forward and RB-Backward are defined as follows*

$$\alpha_t(x_t|l_{1:t-1}) = \sum_{x_{1:t-1}|\{x^d_{i^d:t-1}=x^d_t\}_{d=1}^{l_{t-1}-1}} \Phi_{\alpha,t}[x_{1:t}|l_{1:t-1}] \tag{9.30}$$

$$\beta_t(x_t|l_{t:T-1}) = \sum_{x_{t+1:T}|\{x^d_{t+1:j^d}=x^d_t\}_{d=1}^{l_{t-1}-1}} \Phi_{\beta,t}[x_{t:T}|l_{t:T-1}] \tag{9.31}$$

where i^d and j^d are the start and end time for the segment of state x^d_t.

In performing these two sums one must keep these states at level $d \in [1, l_{t-1} - 1]$ stay the same in the interval $[i, j]$ for some $i \leq t$ and $t \leq j$. These states are fixed to $x_t^{1:l_{t-1}-1}$. The indices $\{i\}$ are known when we scan forward and the indices $\{j\}$ are known when we scan backward. In the following we will develop a recursive procedure that finds and stores these indices as we compute the RB-Forward and RB-Backward sequentially. The main results of this section are summarised in Proposition 7.

Proposition 7. *The RB-Forward and RB-Backward can be expressed compactly as follows*

$$\alpha_t(x_t|l_{1:t-1}) = \prod_{d=l_{t-1}}^{D-1} \psi^d_{\alpha,t}(x^d_t, x^{d+1}_t) \tag{9.32}$$

$$\beta_t(x_t|l_{t:T-1}) = \prod_{d=l_t}^{D-1} \psi^d_{\beta,t}(x^d_t, x^{d+1}_t) \tag{9.33}$$

where $\psi^d_{\alpha,t}(.)$ and $\psi^d_{\beta,t}(.)$ are some positive function of (x^d_t, x^{d+1}_t). In addition, the computation of $\alpha_t(.)$ and $\beta_t(.)$ for all $t \in [1, T]$ costs $\mathcal{O}(DT)$ time and space.

Remark: The significance of Proposition 7 is that the vertical chain x_t is enough to represent the RB-Forward and RB-Backward without worrying about the past. More importantly the factorisation in Equations 9.32 and 9.33 implies that we never need to explicitly store $\alpha_t(.)$ and $\beta_t(.)$ as a function of x_t, which is expensive with $\mathcal{O}(|S|^D)$ memory. We only need to store the local potential functions $\psi^d_{\alpha,t}(.)$ and $\psi^d_{\beta,t}(.)$ which require $\mathcal{O}((D-1)|S|^2)$ space.

Proof: Let us first work with the RB-Forward and prove Equation 9.32. We will proceed by induction.

Base Case. At $t = 2$, from the definition of the RB-Forward in Equation 9.30

$$\alpha_2(x_2|l_1) = \sum_{x_1|\{x_1^d = x_2^d\}_{d=1}^{l_1-1}} \Phi_{\alpha,2}[x_{1:2}|l_1] \tag{9.34}$$

$$= \sum_{x_1^{l_1:D}} \Phi_{\alpha,2}[x_{1:2}|l_1] \tag{9.35}$$

subject to $\{x_1^d = x_2^d\}_{d=1}^{l_1-1}$. Consider the forward 'walking chain' in Figure 9.3b. It consists of three elements: (1) the upper body $x_1^{1:l_1}$, where the variables at the first slice are copied forward to those at the second slice, i.e. $x_1^{1:l_1} = x_2^{1:l_1}$, (2) the 'left-leg' $x_1^{l_1+1:D}$, and (3) the 'right-leg' $x_2^{l_1+1:D}$. $\Phi_{\alpha,2}[x_{1:2}|l_1]$ is the product of local potentials distributed along the body, the left-leg and the right-leg. Thus summing over $x_1^{l_1:D}$ is equivalent to marginalising out the left-leg of the walking chain. This can be done in $D - l_1 + 1$ steps. The result after marginalisation are the body and the right-leg, which are parts of the vertical chain at time $t = 2$. Thus, $\alpha_2(x_2|l_1)$ is a product of local potentials along the vertical chain x_2.

Induction. The argument runs in a similar fashion to the base case. Assume that the RB-Forward $\alpha_{t-1}(x_{t-1}|l_{1:t-2})$ is a product of local potentials along the vertical chain x_{t-1}. From Equation 9.30, we have

$$\alpha_t(x_t|l_{1:t-1}) = \sum_{x_{1:t-1}|\{x_{i^d:t-1}^d = x_t^d\}_{d=1}^{l_{t-1}-1}} \Phi_{\alpha,t-1}[x_{1:t-1}|l_{1:t-2}]\overrightarrow{\varphi}_{t-1}(x_{t-1}, x_t|l) \tag{9.36}$$

$$= \sum_{x_{t-1}^{l_{t-1}:D}} \overrightarrow{\varphi}_{t-1}(x_{t-1}, x_t|l) \sum_{x_{1:t-2}|\{x_{k^d:t-2}^d = x_{t-1}^d\}_{d=1}^{l_{t-2}-1}} \Phi_{\alpha,t-1}[x_{1:t-1}|l_{1:t-2}]$$

$$= \sum_{x_{t-1}^{l_{t-1}:D}} \overrightarrow{\varphi}_{t-1}(x_{t-1}, x_t|l)\alpha_{t-1}(x_{t-1}|l_{1:t-2}) \tag{9.37}$$

subject to $\{x_{i^d:t-1}^d = x_t^d\}_{d=1}^{l_{t-1}-1}$. In Equation 9.36, we have used the following factorisation

$$\Phi_{\alpha,t}[x_{1:t}|l_{1:t-1}] = \Phi_{\alpha,t-1}[x_{1:t-1}|l_{1:t-2}]\overrightarrow{\varphi}_{t-1}(x_{t-1}, x_t|l) \tag{9.38}$$

which is a result of Equation 9.28. Recall that $\alpha_{t-1}(x_{t-1}|l_{1:t-2})$ and $\overrightarrow{\varphi}_{t-1}(x_{t-1}, x_t|l)$ are products of local potentials along the left-leg $x_{t-1}^{l_{t-1}+1:D}$, the right-leg $x_t^{l_{t-1}+1:D}$ and the body $x_{t-1:t}^{1:l_t}$. Summarising over $\sum_{x_{t-1}^{l_{t-1}:D}}$ is equivalent to marginalising over the left-leg, which can be done efficiently in $D - l_t + 1$ steps. After marginalisation, the body and the right-leg form a vertical chain at time t. Thus $\alpha_t(x_t|l_{1:t-1})$ is a product of local potentials along the vertical chain x_t.

The forward process is illustrated as a forward walking chain in Figure 9.3b. As the tran-

sition occurs at level l_{t-1}, the left-leg is marginalised out, the right-leg is created while the body stays the same.

Overall it is now clear that the computation of the RB-Forward costs $\mathcal{O}(DT)$ time and space.

The proof of Equation 9.33 in the case of RB-Backward is analogous to that of RB-Forward. The process is illustrated as a backward walking chain in Figure 9.3c, where the right-leg is a summarisation of the $\beta_{t+1}(.)$. As the transition occurs at level l_t, only the left-leg below it is created, while the part above it stays the same. This completes the proof ■

The implementation of forward-walking and backward-walking is summarised in Figure 9.4 and Figure 9.5, respectively.

Input: $\psi_{\alpha,t-1}(.)$ and l_{t-1}
Output: $\psi_{\alpha,t}(.)$

/ Integrating out the left leg using upward message passing */*
$\mu^D(s) = 1$
For $d = D-1, .., l_t$
    $\mu^d(s) \leftarrow \sum_u \psi^d_{\alpha,t-1}(s,u) E^{d,s}_{u,t-1} R^{d+1,u}_{i:t-1} \mu^{d+1}(u)$
EndFor

/ Integrating over u at l_{t-1} */*
$\psi^{l_{t-1}-1}_{\alpha,t}(s,v) = \sum_u \mu^{l_{t-1}}(u) R^{l_t,u}_{i:t} A^{l_{t-1},s}_{u,v,t-1}$

/ Creating the forward-walk (the right-leg) */*
For $d = l_t, .., D-1$
    $\psi^d_{\alpha,t}(s,u) \leftarrow \pi^{d,s}_{u,t}$
EndFor

/ Keeping the higher potentials */*
For $d = 1, .., l_t - 1$
    $\psi^d_{\alpha,t}(s,u) \leftarrow \psi^d_{\alpha,t-1}(s,u)$
EndFor

Figure 9.4: Forward-walking chain.

Of course, computing the RB-Forward and RB-Backward is not the main point. As we can see by analogy to the Markov chains (Section 2.4.3), they are essential ingredients for inference, which we will cover next.

Input: $\psi_{\beta,t+1}(.)$ and l_t
Output: $\psi_{\beta,t}(.)$

/ Integrating out the right leg using upward message passing */*
$\mu^D(s) = 1$
For $d = D-1, .., l_t$
    $\mu^d(s) \leftarrow \sum_v \psi^d_{\beta,t+1}(s,u)\pi^{d,v}_{u,t+1} R^{d+1,v}_{t+1:j}\mu^{d+1}(v)$
EndFor

/ Integrating over at l_t */*
$\psi^{l_t-1}_t(s,u) = \sum_u \mu^{l_t}(v) R^{l_t,v}_{t+1:j} A^{l_t,s}_{u,v,t}$

/ Creating the backward-walk (the left-leg) */*
For $d = l_t, .., D-1$
    $\psi^d_t(s,u) \leftarrow E^{d,s}_{u,t}$
EndFor

/ Keeping the higher potentials */*
For $d = 1, .., l_t - 1$
    $\psi^d_t(s,u) \leftarrow \psi^d_{t+1}(s,u)$
EndFor

Figure 9.5: Backward-walking chain.

9.4.3 Efficient Computation of $\Pr(l_t|l_{-t})$

Now we show how to compute the quantity of interest for Gibbs sampling $\Pr(l_t|l_{-t})$. Proposition 8 summaries the computation.

Proposition 8. *We can express* $\Pr(l_t|l_{-t})$ *as follows*

$$\Pr(l_t|l_{-t}) \propto \sum_{x^{1:D}_t} \prod_{d=l_t}^{D-1} \psi^d_t(x^d_{t+1}, x^{d+1}_{t+1}) \tag{9.39}$$

where $\psi^d_t(.)$ *is some positive function of* $(x^d_{t+1}, x^{d+1}_{t+1})$. *In addition, the computation of* $\Pr(l_t|l_{-t})$ *for all* $t \in [1, T-1]$ *costs* $\mathcal{O}(DT)$ *time and space.*

Proof: Recall from Equation 9.19 that we just have to compute $Z(l)$ in order to estimate $\Pr(l_t|l_{-t})$.

Recall that the joint potential $\Phi[x,l]$ (see Equation 8.1) is the product of all local potentials that are enabled in the contexts caused by l. Thus the joint potential can be factorised as follows

$$\Phi[x,l] = \Phi_{\alpha,t+1}[x_{1:t+1}|l_{1:t}]\Phi_{\beta,t+1}[x_{t+1:T}|l_{t+1:T-1}]\left[\prod_{d=1}^{1} R^{d,s}_{i:j}\right]\left[\prod_{d=l_t}^{D} R^{d,s}_{t+1:j}\right] \tag{9.40}$$

for $t \in [1, T-1]$, where $d > l_{i-1}$, $d > l_j$, and $i \leq t \leq j$. To visualise this relation, imagine a merge between the forward-walk and the backward-walk, as depicted in Figure 9.3d. As we walk in both directions we already know the starting time i and the ending time j of the segment with persistence potential $R^{d,s}_{i:j}$.

Consider the decomposition: $x = (x_{1:t}, x_{t+1}, x_{t+2:T})$. Given the factorisation in Equation 9.40 and the definition of RB-Forward/Backward in Equations 9.30 and 9.31 we have

$$\begin{aligned} Z(l) &= \sum_{x} \Phi[x,l] = \sum_{x_{1:t}} \sum_{x_{t+1}} \sum_{x_{t+2:T}} \Phi[x,l] \\ &= \sum_{x_{t+1}} \left(\left[\prod_{d=1}^{l_t - 1} R^{d,s}_{i:j} \right] \left[\prod_{d=l_t}^{D} R^{d,s}_{t+1:j} \right] \alpha_{t+1}(x_{t+1}|l_{1:t}) \beta_{t+1}(x_{t+1}|l_{t+1:T-1}) \right) \end{aligned} \tag{9.41}$$

Notice that by Proposition 7 the RB-Forward and RB-Backward have the factorised form along the vertical chain $x_{t+1} = (x^1_{t+1}, x^2_{t+1}, ..., x^D_{t+1})$. Hence Equation 9.41 has the form of the sum-product along the chain, which can be computed in D steps (see Section 2.4.3). We can even compute and store all the values of $\Pr(l)$ for all $l_t \in [2, D]$ in D time using the same dynamic algorithm along chain. This implies a $\mathcal{O}(2DT|S|^2)$ time complexity to compute all the conditional probabilities $\Pr(l_t|l_{-t})$ for $t \in [1, T-1]$ and $l_t \in [2, D]$.

Substituting Equations 9.32 and 9.33 into Equation 9.41 and grouping local quantities into appropriate function of $(x^d_{t+1}, x^{d+1}_{t+1})$, for $d \in [1, D-1]$ will give us Equation 9.39 ■

9.4.4 Estimating State Marginals

We have shown that Gibbs sampling the time indices l can be carried out efficiently. In this section, we show how to approximately estimate the probability of a state at given level d and time t, a quantity often used in smoothing:

$$\hat{\Pr}(x^d_t) = \frac{1}{N} \sum_{n \in [1,N]} \Pr(x^d_t | l^{(n)}_{1:T-1}) \tag{9.42}$$

where N is the number of samples.

We have

$$\begin{aligned} \Pr(x^d_t, l) &= \sum_{x \setminus x^d_t} \Pr(x,l) \\ &= \frac{1}{Z} \sum_{x \setminus x^d_t} \Phi[x,l] \\ &= \frac{Z(x^d_t, l)}{Z} \end{aligned} \tag{9.43}$$

where $Z(x_t^d, l) = \sum_{x \setminus x_t^d} \Phi[x, l]$. Using Equation 9.17, we have

$$\begin{aligned} \Pr(x_t^d|l) &= \frac{\Pr(x_t^d, l)}{\Pr(l)} \\ &= \frac{Z(x_t^d, l)}{Z(l)} \end{aligned} \tag{9.44}$$

We have computed $Z(l)$ in Section 9.4.3. Using the same logic we have an expression similar to Equation 9.41

$$Z(x_t^d, l) = \sum_{x_t \setminus x_t^d} \left(\left[\prod_{d=1}^{l_{t-1}-1} R_{i:j}^{d,s} \right] \left[\prod_{d=l_{t-1}}^{D} R_{t:j}^{d,s} \right] \alpha_t(x_t|l_{1:t-1}) \beta_t(x_t|l_{t:T-1}) \right) \tag{9.45}$$

This equation is almost identical to Equation 9.41 except that now we sum over $x_t \setminus x_t^d$ instead of x_t. Thus, like $Z(l)$ in Equation 9.41, we can compute $Z(x_t^d, l)$ in $2D|S|^2$ steps.

We propose the following sampling procedure to estimate $\hat{\Pr}(x_t^d)$. First we compute all the RB-Backward $\beta_{t=1}^T$. Then we proceed from left to right to estimate the RB-Forward α_t. Since α_t only depends on $l_{1:t-1}$, we can sample l_t using Equation 9.18, x_t^d for $d \in [1, D]$ using Equation 9.42 and update α_t as we go. Then the process is repeated until convergence criteria are met. It means the states x_t^d are only updated after every TD Gibbs samples. The intuition is that since successive states sampled by the Gibbs sampler are highly correlated and the marginals may not change significantly after each step, we can also wait for quite a bit of time before picking a value. Finally, the complete procedure to compute Rao-Blackwellised smoothing probability $\hat{\Pr}(x_t^d)$ is summarised in Figure 9.6.

Input: Model parameters
Output: Smooth marginals $\hat{\Pr}(x_t^d)$ $t \in [1, T]$, $d \in [1, D]$

/ Initialisation */*
Sample l
/ Main MCMC loop */*
For $i = 1, ., N$
    For $t = T, T-1, ..., 1$
        Compute β_t using the backward-chain of Figure 9.5
    EndFor
    For $t = 1, 2, ..., T-1$
        Resample l_t
        Compute α_{t+1} using the forward-chain of Figure 9.4
        Update the smooth marginals $\hat{\Pr}(x_{t+1}^d)$
    EndFor
EndFor

Figure 9.6: Computing the Rao-Blackwellised smoothing probability.

9.5 Learning based on Pseudo-Likelihood

Learning in HCRFs using the standard maximum likelihood with the AIO algorithm as the underlying inference is expensive. This subsection investigates the application of Besag's pseudo-likelihood (see Section 3.5.1.2) as an alternative to the true likelihood for our HCRFs:

$$\begin{aligned} \mathcal{L}_{pseudo} &= \log\left(\Pr(x|l,z)\prod_{t\in[t,T-1]}\Pr(l_t|l_{-t},z)\right) && (9.46)\\ &\approx \log\Pr(x,l|z) && (9.47)\end{aligned}$$

Again, let us drop z for clarity. In Section 9.4 we have shown how to efficiently compute $\Pr(l_t|l_{-t})$ for all $t \in [1, T-1]$ in linear time. Similarly, for the $\Pr(x|l)$ we have

$$\begin{aligned} \Pr(x|l) &= \frac{\Pr(x,l)}{\sum_x \Pr(x,l)}\\ &= \frac{\Phi[x,l]}{\sum_x \Phi[x,l]}\\ &= \frac{1}{Z(l)}\Phi[x,l] && (9.48)\end{aligned}$$

Here, $Z(l)$ has been computed in Equation 9.41 (Section 9.4) in linear time.

Now we need to compute the gradient of the pseudo-likelihood for parameter estimation:

$$\nabla\mathcal{L}_{pseudo} = \nabla\log\Pr(x|l) + \sum_{t\in[1,T-1]}\nabla\log\Pr(l_t|l_{-t}) \quad (9.49)$$

Using Equation 9.48, the first term of the RHS reads

$$\nabla\log\Pr(x|l) = \nabla\log\Phi[x,l] - \nabla\log Z(l) \quad (9.50)$$

The term $\nabla\log\Phi[x,l]$ is straightforward due to the factorisation in Proposition 6.

$$\nabla\log\Phi[x,l] = \sum_{t\in[1,T-1]}\nabla\log\varphi_t(x_{t:t+1}|l) \quad (9.51)$$

Now we proceed to estimate $\nabla\log\Pr(l_t|l_{-t})$ in the RHS of Equation 9.49. Recall from

Equation 9.19 that $\Pr(l_t|l_{-t}) = Z(l)/\sum_{l'_t} Z(l'_t, l_{-t})$, we have

$$\begin{aligned}
\nabla \log \Pr(l_t|l_{-t}) &= \nabla \log Z(l) - \nabla \log \sum_{l'_t} Z(l'_t, l_{-t}) \\
&= \nabla \log Z(l) - \frac{1}{\sum_{l'_t} Z(l'_t, l_{-t})} \sum_{l'_t} \nabla Z(l'_t, l_{-t}) \\
&= \nabla \log Z(l) - \sum_{l'_t} \frac{Z(l'_t, l_{-t})}{\sum_{l'_t} Z(l'_t, l_{-t})} \frac{\nabla Z(l'_t, l_{-t})}{Z(l'_t, l_{-t})} \\
&= \nabla \log Z(l) - \sum_{l'_t} \Pr(l'_t|l_{-t}) \nabla \log Z(l'_t, l_{-t}) \qquad (9.52)
\end{aligned}$$

Thus, both Equations 9.50 and 9.52 require estimation of $\nabla \log Z(l)$. Recall the factorisation in Proposition 6, so $\Pr(x|l)$ is a standard sequential CRF (see Chapter 3) with local clique potentials $\varphi_t(x_{t:t+1}|l)$. Then from Proposition 1 we have

$$\nabla \log Z(l) = \sum_{t \in [1,T-1]} \sum_{x_{t:t+1}} \Pr(x_{t:t+1}|l) \nabla \log \varphi_t(x_{t:t+1}|l) \qquad (9.53)$$

Estimating the local expectation in Equation 9.53

$$\sum_{x_{t:t+1}} \Pr(x_{t:t+1}|l) \nabla \log \varphi_t(x_{t:t+1}|l) \qquad (9.54)$$

by directly summing over $x_{t:t+1}$ is not a good choice because we end up with the sum of $|S|^{2D}$ terms. Fortunately, the factorisation given in Proposition 9 below greatly simplifies the computation.

Proposition 9. $\Pr(x_{t:t+1}|l)$ *can be factorised as follows*

$$\Pr(x_{t:t+1}|l) = \frac{1}{Z(l)} \prod_{d=1}^{D-1} \psi_t(x_{t:t+1}^{d:d+1}|l) \qquad (9.55)$$

where $\psi_t(x_{t:t+1}^{d:d+1}|l)$ *are some non-negative functions.*

Proof: We have

$$\begin{aligned}
\Pr(x_{t:t+1}|l) &= \sum_{x \setminus x_{t:t+1}} \Pr(x|l) \\
&= \frac{1}{Z(l)} \sum_{x \setminus x_{t:t+1}} \Phi[x, l] \\
&= \frac{Z(x_{t:t+1}, l)}{Z(l)} \qquad (9.56)
\end{aligned}$$

where $Z(x_{t:t+1}, l) = \sum_{x \setminus x_{t:t+1}} \Phi[x, l]$. As $(x, l) = (x_{1:t-1}, x_{t:t+1}, x_{t+2:T}, l_{1:t-1}, l_t, l_{t+1:T-1})$, $\Phi[x, l]$ can be factorised as

$$\begin{aligned}\Phi[x,l] &= \Phi_{\alpha,t}[x_{1:t}|l_{1:t-1}]\Phi_{\beta,t+1}[x_{t+1:T}|l_{t+1:T-1}] \times \\ &\quad \times A^{d,s}_{u,v,t} \prod_{d=1}^{l_t-1} R^{d,s}_{i:j} \prod_{d=l_t}^{D-1} E^{d,s}_{u,t} \prod_{d=l_t}^{D} R^{d,s}_{i:t} \prod_{d=l_t}^{D-1} \pi^{d,s}_{u,t+1} \prod_{d=l_t}^{D} R^{d,s}_{t+1:j} \quad (9.57)\end{aligned}$$

This gives rise to

$$\begin{aligned}Z(x_{t:t+1}, l) &= \alpha_t(x_t|l_{1:t-1})\beta_{t+1}(x_{t+1}|l_{t+1:T-1}) \times \\ &\quad \times A^{d,s}_{u,v,t} \prod_{d=1}^{l_t-1} R^{d,s}_{i:j} \prod_{d=l_t}^{D-1} E^{d,s}_{u,t} \prod_{d=l_t}^{D} R^{d,s}_{i:t} \prod_{d=l_t}^{D-1} \pi^{d,s}_{u,t+1} \prod_{d=l_t}^{D} R^{d,s}_{t+1:j} \quad (9.58)\end{aligned}$$

Due to Proposition 7, the RB-Forward $\alpha_t(x_t|l_{1:t-1})$ and the RB-Backward $\beta_{t+1}(x_{t+1}|l_{t+1:T-1})$ are factorisable along the vertical chains of x_t and x_{t+1}, respectively. As a result $Z(x_{t:t+1}, l)$ is also factorisable in the same manner. With appropriate arrangement of the local factors of $Z(x_{t:t+1}, l)$ into potentials of the form $\psi_t(x^{d:d+1}_{t:t+1}|l)$, Proposition 9 follows. This completes the proof ■

As a result of Proposition 9 the local marginal $\Pr(x_{t:t+1}|l)$ can be represented by a vertical chain. More precisely, since below level l_t, there are no links between nodes, we have a three-branch tree. The expectation in Equation 9.54 is therefore efficient to compute in $\mathcal{O}(D)$ time. Overall we can compute the gradient of the log-pseudo-likelihood in $\mathcal{O}(DT^2)$ time due to Equation 9.53.

9.6 Representing HCRF with Exponential Duration using Dynamic Factor Graphs

This subsection describes a method to represent the idea of hierarchical topology of the HCRF in the form of a dynamic CRF (Sutton *et al.*, 2007), analogous to what Murphy and Paskin (2002) have done to convert the HHMMs (Fine *et al.*, 1998) into DBNs (see also Section 2.5.1). The dynamic CRF is a standard graphical model with fixed cliques and connectivity. This allows many efficient approximate inference methods, which may require sub-cubic time in practice.

The main source of difficulty is that in general, the HCRF does not have fixed cliques and connectivity, as we have already seen in previous sections. Fortunately, there is a special case of HCRFs which gives us a way to represent the HCRF using a dynamic CRF. This is when *the duration distribution is precisely exponential*, so we can factorise the state-

persistence potential $R_{i:j}^{d,s}$ into product of node potentials at each time step $t \in [i, j]$:

$$R_{i:j}^{d,s} = \prod_{t \in [i,j]} \phi_t^d(s) \tag{9.59}$$

We show how to create a factor graph (see Section 2.4.8) with this special case. This can be considered as an undirected version of the DBN/HHMM. Approximate inference in factor graphs such as loopy sum-product can therefore be used. This method (possibly) allows linear time inference, as opposed to cubic time using the exact AIO method, so it may scale better when the sequence is long.

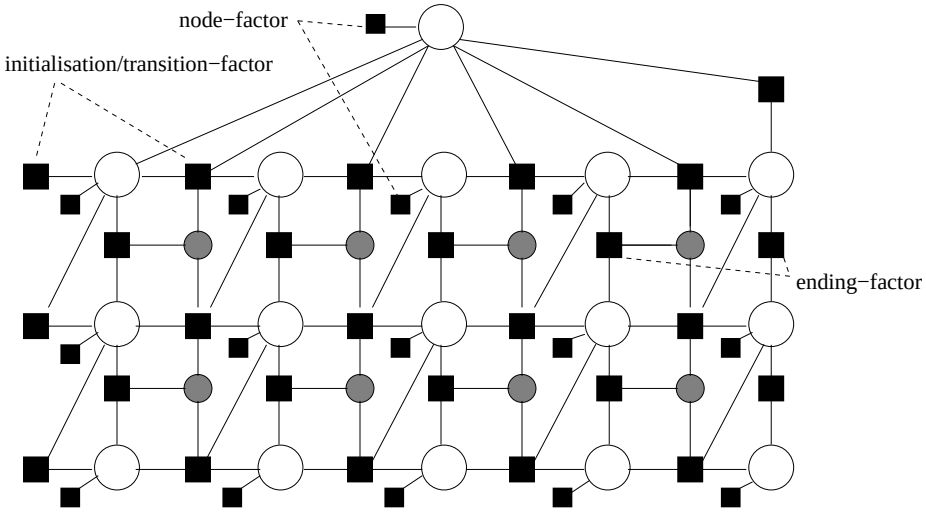


Figure 9.7: Dynamic factor graph representation of HCRFs. Filled squares represent factor nodes, big circles represent variable nodes, and small circles represent ending indicators. We ignore the observation for presentation clarity since it can be thought as being absorbed into the node potentials.

Figure 9.7 depicts the resulting factor graph. There is a root node to represent the top level because the top state persists during the whole sequence. Ending indicators at the bottom level are not used since they are always triggered. There are three types of factors: *node, initialisation/transition* and *ending*. The last three capture the corresponding events, and more importantly, ensure the hierarchical consistency of the model. Associated with these factor types are corresponding potential functions:

- Node potential $\phi_t^d(x_t^d)$, for $d \in [1, D], t \in [1, T]$
- Initialisation/transition potential $\psi_t^d(x_{t+1}^{d-1}, x_{t:t+1}^d, e_t^{d-1:d})$ for $d \in [2, D], t \in [1, T-1]$ with an additional $\psi_0^d(x_1^{d-1}, x_1^d)$ for $d \in [2, D]$ at the beginning of sequence.
- ending potential $\psi_t^d(x_t^{d:d+1}, e_t^d)$ for $d \in [1, D-1], t \in [1, T]$.

Now we describe the potentials in more detail.

Initialisation/Transition Factor: At the beginning of the sequence: $\psi_0^d(x_1^{d-1}, x_1^d) = \pi_{u,1}^{d-1,s}$ for $d \in [2, D]$, where $s = x_1^{d-1}, u = x_1^d$. For other time indices, for $t \in [1, T-1]$, there are three sub-cases:

- At the second level the potential is
  - $A_{u,v,t}^{2,r}$ if $e_t^2 = 1$, where r is the root variable, $u = x_t^2, v = x_{t+1}^2$.
  - $\delta[u = v]$, otherwise
- At the bottom level the potential is
  - $A_{u,v,t}^{D,s}$ if $e_{D-1,t} = 0$, where $s = x_{t+1}^{D-1}, u = x_t^D, v = x_{t+1}^D$.
  - $\pi_{u,t}^{D-1,s}$, otherwise.
- At other levels, for $d \in [3, D-1]$, the potential is
  - 0 if $e_t^{d-1:d} = (1, 0)$. The constraint here is that if a parent finishes then its child must also finish.
  - $\delta[x_t^d = x_{t+1}^d]$ if $e_t^{d-1:d} = (0, 0)$. Here the both the parent and child continues so at least the child state must stay the same.
  - $\pi_{u,t+1}^{d-1,s}$ if $e_t^{d-1:d} = (1, 1)$, where $s = x_{t+1}^{d-1}, u = x_{t+1}^d$,
  - $A_{u,v,t}^{d,s}$, otherwise, where $v = x_{t+1}^d$.

Ending Factor: At the end of sequence all states end so the potential is $E_{u,T}^{d,s}$ for $d \in [1, D-1]$, where $s = x_T^d, u = x_T^{d+1}$. For $t \in [1, T-1]$, the potential is

- $E_{u,t}^{d,s}$ if $e_t^d = 1$, where $s = x_t^d, u = x_t^{d+1}$,
- 1, otherwise.

9.7 Hierarchical HMMs as HCRFs

In this section we show that with slight modification to the HCRF it covers the HHMM (see Section 2.5.1 for a general review, and see (Phung, 2005, Chapter 5) for elaborated details) as a special case.

9.7.1 From HCRFs to *Un*conditional HCRFs

We have worked exclusively with the conditional distribution $\Pr(\zeta|z)$ of Equation 8.2, where we simply ignore modelling $\Pr(z)$. Now, let us modify the HCRF in the way that each state at the bottom level (also called production level in HHMMs) x_t^D is associated with an observable z_t. States at other levels are not directly associated with z. For simplicity we only consider the discrete case, where $z_t \in \mathcal{Z} = \{1, 2, ..., |\mathcal{Z}|\}$. We turn our attention to the unconditional case where we want to model $\Pr(\zeta, z)$

$$\Pr(\zeta, z) = \frac{1}{Z}\Phi[\zeta, z] \tag{9.60}$$

Note that we have only a single partition function $Z = \sum_{\zeta,z} \Phi[\zeta, z]$ for all data instances.

We shall use the same contextual cliques as in the definition of the HCRF in Section 8.2. However, the potentials associated with those contextual cliques listed in Figure 8.6 are not functions of the observational sequence z except for the the persistence potential at the bottom level $R_{t:t}^{D,s,z_t}$, with $t \in [1, T]$.

Like in most undirected graphical models, the most important quantity is the partition function. For any ζ we alway have the following factorisation

$$\Phi[\zeta, z] = \Phi[\zeta \backslash x_{1:T}^D] \prod_{1 \in [1,T]} R_{t:t}^{D,s,z_t} \tag{9.61}$$

where $\Phi[\zeta \backslash x_{1:T}^D]$ is the product of all local potentials other than the state persistence at the bottom level and s is a shorthand for x_t^D. Then the partition function can be computed as

$$\begin{aligned} Z &= \sum_{\zeta} \left(\Phi[\zeta \backslash x_{1:T}^D] \sum_{z} \prod_{1 \in [1,T]} R_{t:t}^{D,s,z_t} \right) && (9.62) \\ &= \sum_{\zeta} \left(\Phi[\zeta \backslash x_{1:T}^D] \prod_{1 \in [1,T]} \sum_{z_t} R_{t:t}^{D,s,z_t} \right) && (9.63) \\ &= \sum_{\zeta} \left(\Phi[\zeta \backslash x_{1:T}^D] \prod_{1 \in [1,T]} \bar{R}_{t:t}^{D,s} \right) && (9.64) \end{aligned}$$

where $\bar{R}_{t:t}^{D,s} = \sum_{z_t} R_{t:t}^{D,s,z_t}$. Indeed $\bar{R}_{t:t}^{D,s}$ plays the role of a state persistence potential. Then the computation of Z can proceed in the same way as in the AIO algorithm (Section 8.3).

9.7.2 From Unconditional HCRFs to HHMMs

Now we turn into converting this unconditional HCRF to an HHMM in a similar way to converting a chain MRF into an HMM (Section 2.4.3.1). We reuse the concepts of symmetric and asymmetric Markov blankets, which are depicted in Figure 9.8. .

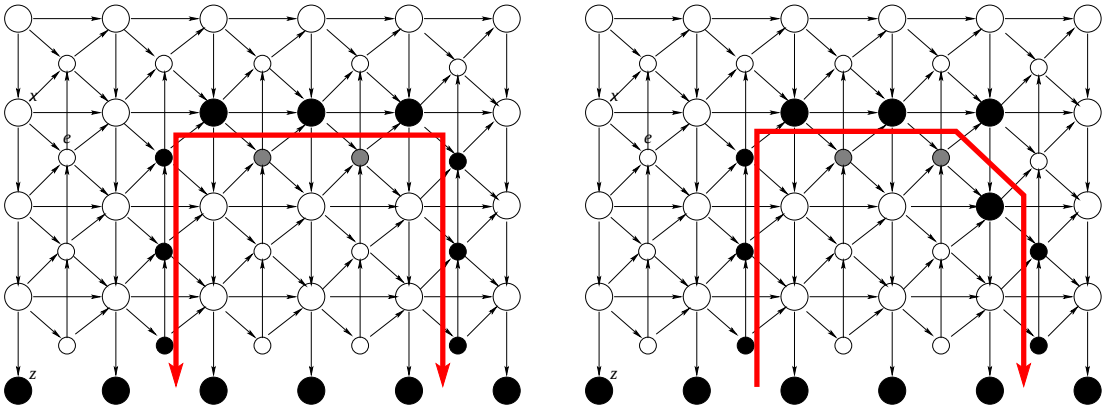


Figure 9.8: Symmetric (left) and asymmetric (right) Markov blankets for HHMMs.

Note that, the unconditional HCRFs are strictly more general than the HHMMs in that HCRFs allow arbitrary *duration modelling*, and the local potentials are *time dependent*, which are not present in HHMMs.

Let us drop all these extra elements, and an HHMM is an unconditional HCRF with following constraints

$$\sum_{u \in S^{d+1}} \pi_u^{d,s} = 1, \pi_u^{d,s} \geq 0 \text{ for all } s \in S^d, d \in [1, D-1] \tag{9.65}$$

$$\sum_{v \in S^d} A_{u,v}^{d,s} + E_u^{d-1,s} = 1, A_{u,v}^{d,s} \geq 0, E_u^{d-1,s} \geq 0$$

$$\text{for all } s \in S^d, u \in S^{d+1}, d \in [2, D] \tag{9.66}$$

$$\sum_{z_t \in \mathcal{Z}} R_{t:t}^{D,s,z_t} = 1, R_{t:t}^{D,s,z_t} \geq 0 \text{ for all } s \in S^D \tag{9.67}$$

and other state persistence potentials are not modelled. In other words, $\pi_u^{d,s}$ plays the role of the initialisation probability of a child u given the parent s at level d; $A_{u,v}^{d,s}$ the transition probability from state u to state v at level d under the parent s; $E_u^{d-1,s}$ the probability that the child u will exit under the parent s; and $R_{t:t}^{D,s,z_t}$ the emission probability of an observable at time t by the production state s. Notice the relationship of the transition and ending potentials which says that under the parent s, the child u has two choices: to transit to a new child v or to return the control to its parent. Formally, these potentials are defined

as follows (Phung, 2005, Chapter 5)

$$\pi_u^{d,s} = \Pr(x_t^{d+1} = u | x_t^d = s, e_t^d = 1) \tag{9.68}$$

$$A_{u,v}^{d,s} = \Pr(x_{t+1}^d = v, e_t^{d-1} = 0 | x_t^{d-1} = s, x_t^d = u, e_t^d = 1) \tag{9.69}$$

$$E_u^{d,s} = \Pr(e_t^d = 1 | x_t^d = s, x_t^{d+1} = u, e_t^{d+1} = 1) \tag{9.70}$$

$$R_{t:t}^{D,s,z_t} = \Pr(z_t | x_t^D = s) \tag{9.71}$$

It is straightforward to verify that with these definitions the set of constraints in Equations 9.65-9.67 are satisfied.

In the HHMM setting these potentials (or probabilities) themselves are parameters. In what follows we draw the probabilistic interpretation of the building blocks under this setting.

The concept of Markov blankets and conditional independence need to be modified to incorporate z. Denote by $\underline{z}_{i:j} = z_{1:i-1,j+1:T}$, i.e. $z = (z_{i:j}, \underline{z}_{i:j})$. With some abuse of notation, let us define $(\zeta_{i:j}^{d,s}, z_{i:j})$ as the set of variables falling inside the symmetric Markov blanket $\Pi_{i:j}^{d,s}$, and $(\underline{\zeta}_{i:j}^{d,s}, \underline{z}_{i:j})$ falling outside.

The symmetric inside mass now becomes

$$\begin{aligned}
\Delta_{i:j}^{d,s} &= \sum_{\zeta_{i:j}^{d,s}} \Phi[\hat{\zeta}_{i:j}^{d,s}, z_{i:j}] \\
&= \sum_{\zeta_{i:j}^{d,s}} \Pr(\zeta_{i:j}^{d,s}, z_{i:j}, e_{i:j-1}^d = 0, e_j^{d:D} = 1 | e_{i-1}^{d:D} = 1, x_{i:j}^d = s) \\
&= \Pr(z_{i:j}, e_{i:j-1}^d = 0, e_j^{d:D} = 1 | e_{i-1}^{d:D} = 1, x_{i:j}^d = s)
\end{aligned} \tag{9.72}$$

The symmetric outside mass can be expressed as

$$\begin{aligned}
\Lambda_{i:j}^{d,s} &= \sum_{\underline{\zeta}_{i:j}^{d,s}} \Phi[\hat{\underline{\zeta}}_{i:j}^{d,s}, \underline{z}_{i:j}] \\
&= \sum_{\underline{\zeta}_{i:j}^{d,s}} \Pr(\underline{\zeta}_{i:j}^{d,s}, \underline{z}_{i:j}, e_{i-1}^{d:D} = 1, x_{i:j}^d = s | e_j^{d:D} = 1, e_{i:j-1}^d = 0) \\
&= \Pr(\underline{z}_{i:j}, e_{i-1}^{d:D} = 1, x_{i:j}^d = s | e_j^{d:D} = 1, e_{i:j-1}^d = 0)
\end{aligned} \tag{9.73}$$

Similarly, we define $(\zeta_{i:j}^{d,s}(u), z_{i:j})$ as the set of variables falling inside the asymmetric Markov blanket $\Gamma_{i:j}^{d,s}(u)$, and $(\underline{\zeta}_{i:j}^{d,s}(u), \underline{z}_{i:j})$ falling outside. The new asymmetric inside

mass is defined as

$$\begin{aligned}
\alpha_{i:j}^{d,s}(u) &= \sum_{\zeta_{i:j}^{d,s}(u)} \Phi[\hat{\zeta}_{i:j}^{d,s}(u), z_{i:j}] \\
&= \sum_{\zeta_{i:j}^{d,s}(u)} \Pr(\zeta_{i:j}^{d,s}(u), z_{i:j}, e_{i:j-1}^{d} = 0, x_j^{d+1} = u, e_j^{d+1:D} = 1 | e_{i-1}^{d:D} = 1, x_{i:j}^{d} = s) \\
&= \Pr(z_{i:j}, e_{i:j-1}^{d} = 0, x_j^{d+1} = u, e_j^{d+1:D} = 1 | e_{i-1}^{d:D} = 1, x_{i:j}^{d} = s) \qquad (9.74)
\end{aligned}$$

Coupled with this is the asymmetric outside mass

$$\begin{aligned}
\lambda_{i:j}^{d,s}(u) &= \sum_{\underline{\zeta}_{i:j}^{d,s}(u)} \Phi[\hat{\underline{\zeta}}_{i:j}^{d,s}(u)] \\
&= \sum_{\underline{\zeta}_{i:j}^{d,s}(u)} \Pr(\underline{\zeta}_{i:j}^{d,s}(u), \underline{z}_{i:j}, e_{i-1}^{d:D} = 1, x_{i:j}^{d} = s | e_j^{d+1:D} = 1, e_{i:j-1}^{d} = 0, x_j^{d+1} = u) \\
&= \Pr(\underline{z}_{i:j}, e_{i-1}^{d:D} = 1, x_{i:j}^{d} = s | e_j^{d+1:D} = 1, e_{i:j-1}^{d} = 0, x_j^{d+1} = u) \qquad (9.75)
\end{aligned}$$

We now examine several relations between those building blocks. Suppose we want to infer the conditional probability $\Pr(x_t^d = s|z)$ of certain state at time $t \in [1, T]$ and level $d \in [2, D-1]$ given the observation sequence z

$$\Pr(x_t^d = s|z) = \frac{\Pr(x_t^d = s, z)}{\Pr(z)} \qquad (9.76)$$

Naturally, the state $s_{i:j}^d$ must start and end somewhere so that $t \in [i, j]$, so we can expand $\Pr(x_t^d = s, z)$ as

$$\begin{aligned}
\Pr(x_t^d = s, z) &= \sum_{i \in [1,t], j \in [t,T]} \Pr(x_{i:j}^d = s, e_{i-1}^{d:D} = 1, e_{i:j-1}^{d} = 0, e_j^{d:D} = 1, z_{i:j}, \underline{z}_{i:j}) \\
&= \sum_{i \in [1,t], j \in [t,T]} \Pr(z_{i:j}, e_{i:j-1}^{d} = 0, e_j^{d:D} = 1 | e_{i-1}^{d:D} = 1, x_{i:j}^{d} = s) \times \\
&\qquad \times \Pr(\underline{z}_{i:j}, e_{i-1}^{d:D} = 1, x_{i:j}^{d} = s | e_j^{d:D} = 1, e_{i:j-1}^{d} = 0) \\
&= \sum_{i \in [1,t], j \in [t,T]} \Delta_{i:j}^{d,s} \Lambda_{i:j}^{d,s} \qquad (9.77)
\end{aligned}$$

which naturally leads to the data likelihood

$$\begin{aligned}
\Pr(z) &= \sum_{s \in S^d} \Pr(x_t^d = s, z) \\
&= \sum_{s \in S^d} \sum_{i \in [1,t], j \in [t,T]} \Delta_{i:j}^{d,s} \Lambda_{i:j}^{d,s} \qquad (9.78)
\end{aligned}$$

for any $t \in [1, T]$ and $d \in [2, D-1]$.

Another relation is between the inside and asymmetric inside masses. Expanding the RHS of Equation 9.72 we arrive at

$$\begin{aligned}
\Delta_{i:j}^{d,s} &= \Pr(z_{i:j}, e_{i:j-1}^{d} = 0, e_{j}^{d:D} = 1 | e_{i}^{d:D} = 1, x_{i:j}^{d} = s) \\
&= \sum_{u \in S^{d+1}} \Pr(z_{i:j}, e_{i:j-1}^{d} = 0, x_{j}^{d+1} = u, e_{j}^{d+1:D} = 1, e_{j}^{d} = 1 | e_{i}^{d:D} = 1, x_{i:j}^{d} = s) \\
&= \sum_{u \in S^{d+1}} \Pr(z_{i:j}, e_{i:j-1}^{d} = 0, x_{j}^{d+1} = u, e_{j}^{d+1:D} = 1 | e_{i}^{d:D} = 1, x_{i:j}^{d} = s) \times \\
&\qquad \times \Pr(e_{j}^{d} = 1 | x_{j}^{d} = s, x_{j}^{d+1} = u, e_{j}^{d+1} = 1) \\
&= \sum_{u \in S^{d+1}} \alpha_{i:j}^{d,s}(u) E_{u}^{d,s} \qquad (9.79)
\end{aligned}$$

where the asymmetric inside mass is from Equation 9.74 and the exiting potential from Equation 9.70. This result is identical to Equation 8.37 as expected.

9.8 Evaluation

9.8.1 Recognising Indoor Activities

In this experiment we evaluate the HCRFs on the home video surveillance dataset (see Chapter 6 and Chapter 7). Recall that the data has a hierarchy of activities: each complex activity is a sequence of simpler activities. Thus, we build a three-level HCRF in which the top level is just a dummy node, the second level has 3 states (representing complex activities), and the bottom level has 12 states (representing simple activities).

At the bottom level (simple activities), we reuse the feature set used in Chapter 6 and Chapter 7. At the second level (complex activities), we use average velocities and a vector of positions visited in the state duration. To encode the duration into the state-persistence potentials, we employ the sufficient statistics of the *gamma* distribution as features $f_k(s,i,j) = \mathbb{I}(s)\log(j-i+1)$ and $f_{k+1}(s,i,j) = \mathbb{I}(s)(j-i+1)$.

At each level d and time t we count an error if the predicted state is not the same as the ground-truth. Firstly, we examine the fully observed case where the HCRF is compared against the grid-structured CRF (known as Factorial CRF (FCRF) (Sutton *et al.*, 2007)) at both data levels, and against the sequential CRF (SCRF) (Lafferty *et al.*, 2001) at the bottom level. Table 9.2 (the left half) shows that (a) both the multilevel models significantly outperform the flat model and (b) the HCRF outperforms the FCRF.

We also test the ability of the model to learn the hierarchical topology and state transitions. We find that it is very informative to examine parameters which correspond to the state

Alg.	$d=2$	$d=3$	Alg.	$d=2$	$d=3$
HCRF	100	93.9	Po-HCRF	80.2	90.4
FCRF	96.5	89.7	Po-SCRF	-	83.5
SCRF	-	82.6	-	-	-

Table 9.2: Accuracy (%) for fully observed data (left), and partially observed (Po) data (right).

transition features. Typically, negative entries in the transition parameter matrix means that the transition is impossible. This is because state features are non-negative, so negative parameters mean the probabilities of these transitions are very small, compared to the positive ones. For the transition at the second level (the complex activity level), we obtain all negative entries. This clearly matches the training data where each sequence already belongs to one of three complex activities. With this method we are able to construct the correct hierarchical topology as in Figure 9.9. The state transition model is presented in Figure 9.10. There is only one wrong transition, from state 12 to state 10, which is not presented in the training data. The rest is correct.

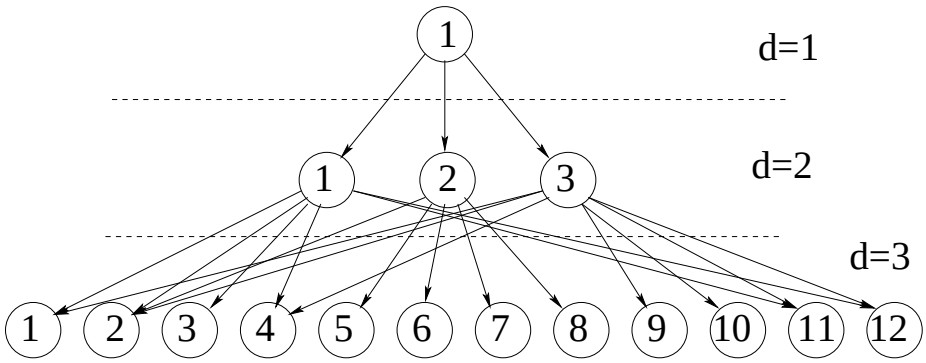


Figure 9.9: The topo learned from data.

Next we consider partially-supervised learning in that about 50% of start/end times of a state and state labels are observed at the second level. All ending indicators are known at the bottom level. The results are reported in Table 9.2 (the right half). As can be seen, although only 50% of the state labels and state start/end times are observed, the model learned is still performing well with accuracy of 80.2% and 90.4% at levels 2 and 3, respectively.

We now consider the issue of partially observing labels during decoding and test the effect using degraded learned models. Such degraded models (emulating noisy training data or lack of training time) are extracted from the 10th iteration of the fully observed data case. The labels are provided at random times. Figure 9.11 shows the decoding accuracy as a function of available labels. It is interesting to observe that a moderate amount of observed labels (e.g. $20-40\%$) causes the accuracy rate to go up considerably.

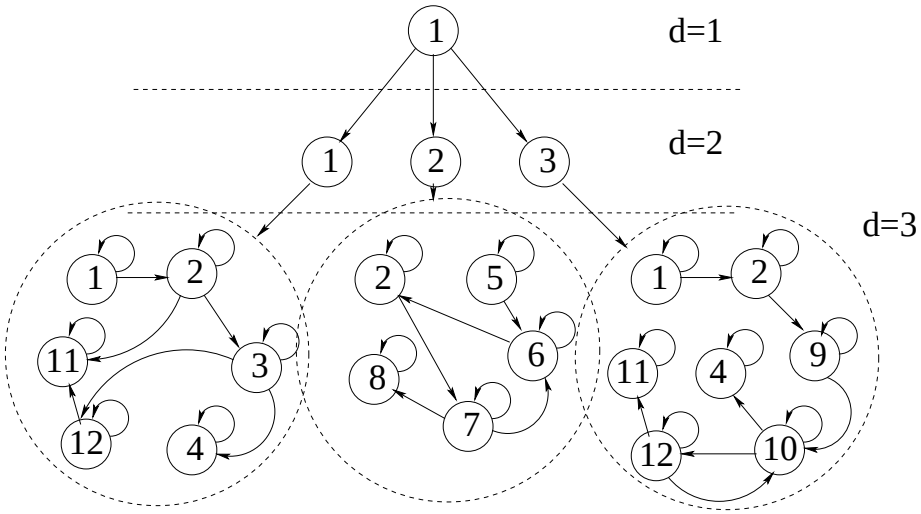


Figure 9.10: The state transition model learned from data. Primitive states are duplicated for clarity only. They are shared among complex states.

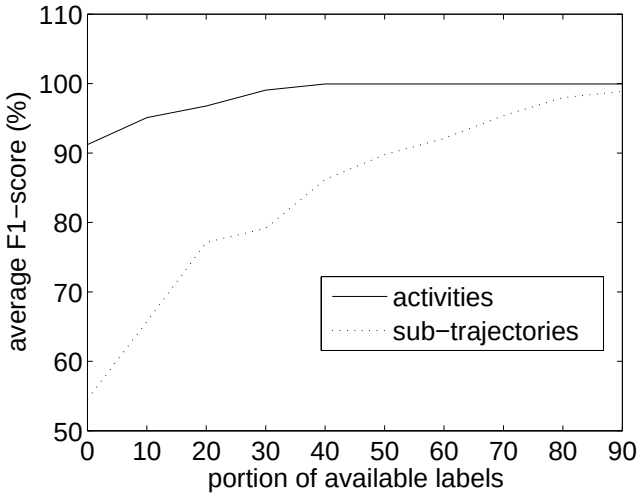


Figure 9.11: Performance of the constrained max-product algorithm described in Section 9.3.2 as a function of available information on label/start/end time.

9.8.2 POS Tagging and Noun-Phrase Chunking

In this experiment we apply the HCRF to the task of noun-phrase chunking. The data is from the CoNLL-2000 shared task (Sang and Buchholz, 2000), in which 8926 English sentences from the Wall Street Journal corpus are used for training and 2012 sentences are for testing. Each word in a pre-processed sentence is labeled by two labels: the part-of-speech (POS) and the noun-phrase (NP). There are 48 POS different labels and 3 NP labels (B-NP for beginning of a noun-phrase, I-NP for inside a noun-phrase or O for others). Each noun-phrase generally has more than one word. To reduce the computational burden, we reduce the POS tag-set to 5 groups: *noun*, *verb*, *adjective*, *adverb and others*. Since in our HCRFs we do not have to explicitly indicate which node is at the beginning of a segment,

the NP label set can be reduced further into NP for noun-phrase, and O for anything else.

The POS tags are actually the output of the Brill's tagger (Brill, 1995), while the NPs are manually labeled. We extract raw features from the text in the way similar to that in (Sutton *et al.*, 2007). However, we consider only a limited vocabulary extracted from the training data in that we only select words with more than 3 occurrences. This reduces the vocabulary and the feature size significantly. We also make use of bi-grams with similar selection criteria. Furthermore, we use the contextual window of 5 instead of 7 as in (Sutton *et al.*, 2007). This setting gives rise to about 32K raw features. The model feature is factorised as $f(x_c, z) = \mathbb{I}(x_c)g_c(z)$, where $\mathbb{I}(x_c)$ is a binary function on the assignment of the clique variables x_c, and $g_c(z)$ are the raw features.

We build an HCRF topology of 3 levels where the root is just a dummy node, the second level has 2 NP states and the bottom level has 5 POS states. For comparison, we implement a FCRF, a SCRF, and a semi-Markov CRF (SemiCRF) (Sarawagi and Cohen, 2004). The FCRF has grid structure of depth 2, one for modelling the NP process and another for the POS process. Since the state spaces are relatively small, we are able to run exact inference in the FCRF by collapsing both the NP and POS state spaces to a combined state space of size $3 \times 5 = 15$. The SCRF and SemiCRF model only the NP process, taking the POS tags as input.

The raw feature set used in the FCRF is identical to those in our HCRF. However, the set shared by the SCRF and the SemiCRF is a little more elaborate since it takes the POS tags into account (Sutton *et al.*, 2007).

Although both the HCRF and the SemiCRF are capable of modelling arbitrary segment durations, we use a simple exponential distribution as it can be processed sequentially and thus is very efficient. For learning, we use a simple online stochastic gradient ascent method since it has been shown to work relatively well and fast in CRFs (Vishwanathan *et al.*, 2006). At test time, as the SCRF and the SemiCRF are able to use the Brill's POS tags as input, it is not fair for the FCRF and HCRF to predict those labels during inference. Instead, we also give the POS tags to the FCRF and HCRF and perform constrained inference to predict *only* the NP labels. This boosts the performance of the two multi-level models significantly.

The performance of these models is depicted in Figure 9.12 and we are interested in only the prediction of the noun-phrases since this data has Brill's POS tags. Without Brill's POS tags given at test time, both the HCRF and the FCRF perform worse than the SCRF. This is not surprising because the Brill's POS tags are always given in the case of SCRF. However, with POS tags the HCRF consistently works better than all other models. The FCRF does worse than the SCRF, even with POS tags given. This does not share the observation made in (Sutton *et al.*, 2007). However, we use a much smaller POS tag set than (Sutton *et al.*,

2007) does. Our explanation is that the SCRF is able to make use of wider context of the given POS tags (here, within the window of 5 tags) than the FCRF (limited to 1 POS tag per NP chunk). The SemiCRF, although in theory it is more expressive than the SCRF, does not show any advantage under current setting. Recall that the SemiCRF is a special case of HCRF in that the POS level is not modelled, it is possible to conclude that joint modelling of NP and POS levels is important.

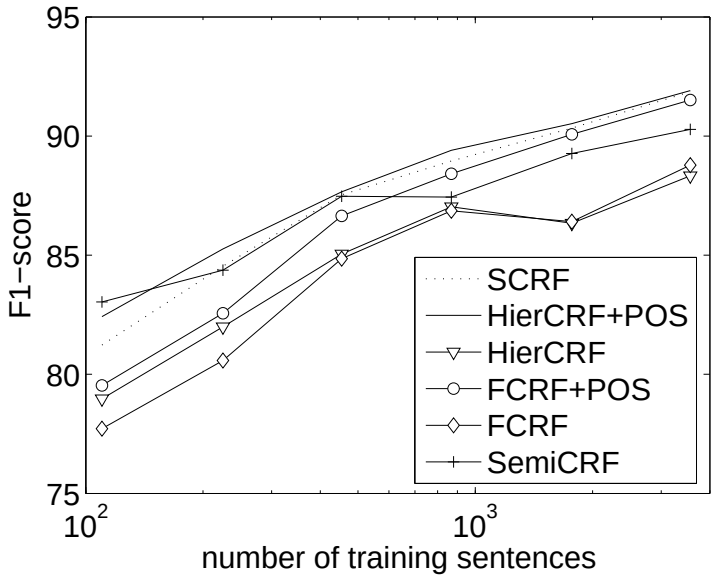


Figure 9.12: Performance of various models on Conll2000 noun-phrase chunking. HCRF+POS and FCRF+POS mean HCRF and FCRF with POS given at test time, respectively.

9.9 Closing Remarks

In this chapter we have presented a number of extensions to the HCRF to address three important issues: numerical overflow in computing the partition function, learning and inference with partial labels (partial supervision and constrained inference, respectively), and the cubic time complexity of the AIO family. We have derived a scaling algorithm that helps to minimise the overflow. For the second issue the AIO algorithm is extended so that it is consistent with the known labels. And for the last issue we have proposed a number of approximation techniques based on Rao-Blackwellisation and Gibbs sampling for inference, and pseudo-likelihood for learning. We have also shown how to represent an HCRF with exponential duration by a factor graph, in which inference can be carried out approximately by using the sum-product algorithm. We have demonstrated the capacity of the HCRFs on home video surveillance data and the shallow parsing of English text, in which the hierarchical information inherent in the context helps to increase the recognition.

Chapter 10

Conclusions

10.1 Summary

This thesis has presented a study of various aspects of the recently introduced Conditional Random Field, a probabilistic and discriminative framework for modelling and learning structured outputs. First, we have demonstrated the strength of CRFs in the area of Natural Language Processing with the application of *statistical Vietnamese accent restoration* (Chapter 4), and in the area of *recommendation systems* (Chapter 5). Motivated by these applications we have provided a deeper investigation into the theory of CRF in the area of *feature selection* (Chapter 6), *learning with arbitrary structures* (Chapter 7), and modelling *recursive sequential data* (Chapter 8 and Chapter 9). The common theme of these theoretical contributions is learning with missing labels in a *partially supervised* manner.

In Chapter 4, the thesis contributes to the existing literature of CRFs by a novel application in the area of accent restoration with the focus in Vietnamese. Given a sequence of accent-less Vietnamese words, the problem is to restore the original accents without further information. This is a common problem in computational linguistics of Vietnamese, in which texts may not have appropriate accents, causing a comprehension problem to readers. This problem is challenging as the raw texts are highly ambiguous even with native speakers. To the best of our knowledge there has been no publicly reported work to solve this problem. We propose the use of a second-order chain CRF to model the output space of the restored sequence of accents. The result is excellent with 94% word accuracy when tested in a diverse news domain.

The second motivating application of the CRFs is reported in Chapter 5 where we propose to model the *entire rating database* of recommendation systems by a single novel CRF-based framework called *Preference Networks*. Different from most existing work in the literature of automatic recommenders, our modelling is formal and can seamlessly incor-

porate varieties of domain knowledge, including the content of the products and services, the user attributes like demographics and historical profiles and the correlations between users and between products. More importantly we have empirically shown that our Preference Networks outperform well-known methods. Our study also clearly demonstrates efficiency issues associated with large-scale and densely connected CRFs. Specifically, the network in our study has hundreds of thousands of nodes and each node has thousands of neighbours.

Chapter 6 investigates the issue of feature selection as an embedded process of partially supervised learning CRFs. Feature selection, as evidenced in our empirical work Chapter 4 and Chapter 5, is an issue of great practical importance and it critically affects the final performance of the CRF-based systems. To this end we have proposed a boosting-based method called AdaBoost.CRF that extends the current boosting methodology to structured output with missing labels. Experiments have shown that the proposed algorithm is capable of selecting a small subset of features from a large feature pool with little loss in performance.

Chapter 7 presents an attempt to deal with the efficiency issue with arbitrary structures, as evidenced in the empirical study in our Preference Networks (Chapter 5), and in the assumption of the underlying inference made in Chapter 6. Based on the boosting methodology we propose an alternative loss that requires only inference over a set of spanning trees. The trees are co-learned in an iterative fashion and finally re-combined to recover the original network. The result is a scalable algorithm called AdaBoost.MRF that can handle missing training labels and exhibits linear time complexity.

The third theoretical contribution is presented in Chapter 8 where we introduce a major extension to the theory of CRFs for modelling recursive sequential data. This data type is inherent in many domains such as signal and image processing, human activities and natural language processing, where the semantics can be decomposed in different resolutions. Motivated by the early work of HHMMs we propose a novel Hierarchical CRF to address the problem. We introduce a graphical model based representation that helps to visualise the temporal evolution of the model and to encode varieties of domain knowledge into the system. Finally, an efficient algorithm based on the Asymmetric Inside Outside family is derived for learning and inference.

Chapter 9 continues the framework outlined in Chapter 8 through several important extensions for practical application of HCRFs. First, we derive a scaling procedure to avoid numerical overflow in the computation of the partition function. Second, the AIO algorithm is modified to handle partial labels occurring in learning (*partial supervision*) and inference (*constrained inference*). Third, based on Rao-Blackwellisation and Gibbs sampling we propose a sub-cubic time approximate procedure for inference. Likewise, a sub-cubic time

pseudo-likelihood learning style is also offered in the chapter. For the HCRF with exponential state duration distributions, we have shown that it can be represented as a standard factor-graph which allows fast approximate inference based on the Pearl's sum-product algorithm. Finally, the HCRFs with partial labels are evaluated on two datasets: the human activity recognition, and noun-phrase chunking. Experimental results validate the expressiveness and usefulness of the HCRF formulation in these domains when compared with rival methods.

10.2 Future Directions

There are issues associated with the CRFs which have not been addressed in this thesis and are left for future investigation. The most important one is perhaps efficient inference and learning algorithms to compute various aspects of the CRF-based systems with arbitrary structures. In Chapter 7 we have put forward an effort into fast parameter learning of the CRFs by decomposing the network into superimposing spanning trees. However, the trees are still manually specified and thus the algorithm may be only effective for network with highly regular structures (e.g. grids, as in our study). It is best to automatically select the best spanning tree at each boosting step.

Most parameter learning work in CRFs, including this thesis, only deals with non-Bayesian setting. Although regularisation is often used we do not integrate over the parameters. *Bayesian learning* may be important for controlling overfitting and incorporating prior knowledge of the parameters. The only work addressing this problem that we are aware of is Bayesian CRF by Qi *et al.* (2005).

Beside parameter learning, *structure learning* of CRFs has not received adequate attention, although it is much more popular in the directed Bayesian networks. From the connection of this problem with the spanning tree selection in our study of AdaBoost.MRF, we conjecture that the main difficulty is associated with the conditional nature of the CRFs. More specifically, the structures of the CRFs sometimes depend on the conditioning variables even in the same domain.

The main argument for sole use of conditional distribution $\Pr(x|z)$ is that we do not waste effort in modelling $\Pr(z)$. However, while $\Pr(x|z)$ addresses the output directly, a significant amount of information is thrown away with $\Pr(z)$. When labeled training data is sufficient, learning only $\Pr(x|z)$ is of great practical advantage. In many real world situations, unfortunately, labels are too expensive while unlabeled data is cheap. In this case $\Pr(z)$ provides important information to infer about the nature of data, i.e. the *intrinsic* manifold of the data. Recent advances in *semi-supervised*, *active* and *transductive* learning

have proven that unlabeled data can be very valuable to improve the classification performance. Current work is mostly carried out with unstructured output models. We expect that the graphical structure of CRFs will offer new insights into the problem.

Regarding HCRFs, we have introduced several approximation methods for inference and learning in Chapter 9. It remains to investigate into their behaviour and effectiveness for real applications. In addition, we have shown in Section 9.7.2 that the HHMM can be derived from the unconditional version of HCRF. The HHMM can be shown to be a special case of the Probabilistic Context-Free Grammar (PCFG). The conditional version of PCFG has been investigated elsewhere in the literature of Nature Language Processing. Thus, it may be interesting to study the relationship between the HCRF and the conditional PCFG. To the best of our knowledge, the numerical scaling issue, which we have addressed in the HCRF, has not been explicitly raised and solved.

As a modelling tool, HCRFs are designed for the recursive sequential data. This gives rise to the question that to what extent can HCRFs be still applicable for generic hierarchical data, i.e. the assumption of nested Markov chains does not strictly hold? For example, in noun-phrase chunking the POS tags do not belong exclusively to any noun and non-noun phrases. Another interesting issue is that given the data is inherently spatial as in images, how can we convert it into a sequential form to which HCRFs can be applied?

Bibliography

Altun, Y., Hofmann, T., and Johnson, M. (2003a). Discriminative learning for label sequences via boosting. In S. T. S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 977–984. MIT Press, Cambridge, MA.

Altun, Y., Johnson, M., and Hofmann, T. (2003b). Investigating loss functions and optimization methods for discriminative learning of label sequences. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 145–152.

Altun, Y., Hofmann, T., and Smola, A. (2004). Gaussian process classification for segmenting and annotating sequences. In *Proceedings of 21st International Conference on Machine Learning (ICML)*, Banff, Alberta, Canada.

Altun, Y., McAllester, D., and Belkin, M. (2006). Maximum margin semi-supervised learning for structured variables. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18, pages 33–40. MIT Press, Cambridge, MA.

Ando, R. and Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, **6**, 1817–1853.

Andrieu, C., de Freitas, N., Doucet, A., and Jordan, M. I. (2003). An introduction to MCMC for machine learning. *Machine Learning*, **50**(1-2), 5–43.

Balabanović, M. and Shoham, Y. (1997). Fab: content-based, collaborative recommendation. *Communications of the ACM*, **40**(3), 66–72.

Banerjee, O. and Natsoulis, G. (2006). Convex optimization techniques for fitting sparse Gaussian graphical models. In *Proceedings of the 23rd International Conference on Machine learning (ICML)*, pages 89–96. ACM Press New York, NY, USA.

Basilico, J. and Hofmann, T. (2004). Unifying collaborative and content-based filtering. In *Proceedings of the 21st International Conference on Machine learning (ICML)*, pages 65–72. ACM Press New York, NY, USA.

Basu, C., Hirsh, H., and Cohen, W. (1998). Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI)*, pages 174–720, Madison, WI.

Baxter, R. (1982). *Exactly Solved Models in Statistical Mechanics*. Academic Press, London.

Berger, A. L., Pietra, S. A. D., and Pietra, V. J. D. (1996). A Maximum Entropy approach to natural language processing. *Computational Linguistics*, **22**(1), 39–71.

Bertsekas, D. (1999). *Nonlinear Programming*. Athena Scientific, Belmont, Massachussets, 2 edition.

Besag, J. (1975). Statistical analysis of non-lattice data. *The Statistician*, **24**(3), 179–195.

Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society B*, **48**(3), 259–302.

Billsus, D. and Pazzani, M. (1998). Learning collaborative information filters. In *Proceedings of the 15th International Conference on Machine Learning (ICML)*, pages 46–54, Madison, WI.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

Blunsom, P. and Cohn, T. (2006). Discriminative word alignment with conditional random fields. In *Proceedings of the 21st International Conference on Computational Linguistics (COLING) and the 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 65–72. Association for Computational Linguistics, Morristown, NJ, USA.

Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **23**(11), 1222–1239.

Breese, J., Heckerman, D., Kadie, C., *et al.* (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 461, pages 43–52. Morgan Kaufman, Madison, WI.

Brefeld, U. and Scheffer, T. (2006). Semi-supervised learning for structured output variables. In *Proceedings of the 23rd International Conference on Machine learning (ICML)*, pages 145–152. ACM Press New York, NY, USA.

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, **21**(4), 543–566.

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, **19**(2), 263–296.

Bui, H. H., Venkatesh, S., and West, G. (2002). Policy recognition in the abstract hidden Markov model. *Journal of Artificial Intelligence Research*, **17**, 451–499.

Bui, H. H., Phung, D. Q., and Venkatesh, S. (2004). Hierarchical hidden Markov models with general state hierarchy. In D. L. McGuinness and G. Ferguson, editors, *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI)*, pages 324–329, San Jose, CA.

Buntine, W. L. (1995). Chain graphs for learning. In P. Besnard and S. Hanks, editors, *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 46–54. Morgan Kaufmann, San Francisco, CA.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, **2**(2), 121–168.

Byrd, R. H., Nocedal, J., and Schnabel, R. B. (1994). Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, **63**, 129–156.

Carreira-Perpiñán, M. A. and Hinton, G. E. (2005). On contrastive divergence learning. In R. G. Cowell and Z. Ghahramani, editors, *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics (AISTATS)*, pages 33–40, Barbados. Society for Artificial Intelligence and Statistics.

Casado, A., Griot, M., and Wesel, R. (2007). Improving LDPC decoders via informed dynamic scheduling. In *Proceedings of the IEEE Information Theory Workshop (ITW)*, pages 208–213.

Casella, G. and Robert, C. (1996). Rao-Blackwellisation of sampling schemes. *Biometrika*, **83**(1), 81.

Chapelle, O., Schölkopf, B., and Zien, A. (2006). *Semi-supervised Learning*. MIT Press.

Chow, C. and Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, **14**(3), 462–467.

Clark, S. and Curran, J. R. (2003). Log-linear models for wide-coverage CCG parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 97–104.

Cohn, T. and Blunsom, P. (2005). Semantic role labelling with tree conditional random fields. In *Proceedings of the 9th Conference on Natural Language Learning (CoNLL)*, pages 169–172. Association for Computational Linguistics.

Collins, M. (2002). Discriminative training methods for hidden Markov models: Theory and experiments with the perceptron algorithm. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Collins, M. (2005). Discriminative reranking for natural language parsing. *Computational Liguistics*, **31**(1), 25–70.

Collins, M., Schapire, R., and Singer, Y. (2002). Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, **48**(1), 253–285.

Coughlan, J. and Shen, H. (2006). Dynamic quantization for belief propagation in sparse spaces. *Computer Vision and Image Understanding*, **106**(1), 47–58.

Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory*. John Wiley & Sons.

Cowans, P. J. and Szummer, M. (2005). A graphical model for simultaneous partitioning and labeling. In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics (AISTATS)*.

Cowie, J. and Lehnert, W. (1996). Information extraction. *Communications of the ACM*, **39**(1), 80–91.

Darroch, J. and Ratcliff, D. (1972). Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, **42**, 1470–1480.

Dechter, R. and Mateescu, R. (2003). A simple insight into iterative belief propagation's success. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 175–183.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society*, **39**(1), 1–38.

Deshpande, M. and Karypis, G. (2004). Item-based top-N recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, **22**(1), 143–177.

Dietterich, T. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, **40**(2), 139–157.

Dietterich, T. G., Ashenfelter, A., and Bulatov, Y. (2004). Training conditional random fields via gradient tree boosting. In *Proceedings of the 21 st International Conference on Machine Learning (ICML)*, pages 217–224, Banff, Canada.

Drucker, H., Schapire, R., and Simard, P. (1992). Improving performance in neural networks using a boosting algorithm. In *Advances in Neural Information Processing Systems 5*, pages 42–49. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

Duchi, J., Tarlow, D., Elidan, G., and Koller, D. (2007). Using combinatorial optimization within max-product belief propagation. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 369–376. MIT Press, Cambridge, MA.

Dudík, M., Phillips, S. J., and Schapire, R. E. (2007). Maximum Entropy density estimation with generalized regularization and an application to species distribution modeling. *Journal of Machine Learning Research*, **8**, 1217–1260.

Elidan, G., McGraw, I., and Koller, D. (2006). Residual belief propagation: Informed scheduling for asynchronous message passing. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (UAI)*, Boston, Massachussetts.

Elkan, C. (2001). The foundations of cost-sensitive learning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, volume 17, pages 973–978.

Felzenszwalb, P. F. and Huttenlocher, D. P. (2006). Efficient belief propagation for early vision. *International Journal of Computer Vision*, **70**(1), 41–54.

Fine, S., Singer, Y., and Tishby, N. (1998). The hierarchical hidden Markov model: Analysis and applications. *Machine Learning*, **32**(1), 41–62.

Finkel, J., Manning, C., and Ng, A. Y. (2006). Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 618–626, Sydney, Australia. Association for Computational Linguistics.

Freund, Y. and Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, **55**(1), 119–139.

Freund, Y. and Schapire, R. (1999). Large margin classification using the perceptron algorithm. *Machine Learning*, **37**(3), 277–296.

Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 148–156.

Frey, B. J. (2003). Extending factor graphs so as to unify directed and undirected graphical models. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, CA, Acapulco, Mexico. Morgan Kaufmann.

Friedman, J., Hastie, T., and Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, **28**(2), 337–374.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, **29**(5), 1189–1232.

Garg, A., Pavlović, V., and Rehg, J. (2003). Boosted learning in dynamic Bayesian networks for multimodal speaker detection. *Proceedings of the IEEE*, **91**(9).

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **6**(6), 721–742.

Golding, A. and Roth, D. (1999). Winnow-based approach to context-sensitive spelling correction. *Machine Learning*, **34**(1/3), 107–130.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, **82**(4), 711–732.

Gregory, M. L. and Altun, Y. (2004). Using conditional random fields to predict pitch accents in conversational speech. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL)*, pages 677–683, Morristown, NJ, USA. Association for Computational Linguistics.

Gross, S., Russakovsky, O., Do, C., and Batzoglou, S. (2007). Training conditional random fields for maximum labelwise accuracy. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19, pages 529–536. MIT Press, Cambridge, MA.

Gunawardana, A., Mahajan, M., Acero, A., and Platt, J. (2005). Hidden conditional random fields for phone classification. In *Proceedings of the International Conference on Speech Communication and Technology*, pages 1117–1120, Lisbon, Portugal.

Gupta, R., Diwan, A., and Sarawagi, S. (2007). Efficient inference with cardinality-based clique potentials. In *Proceedings of the 24th international conference on Machine learning (ICML)*, pages 329–336. ACM Press New York, NY, USA.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, **3**, 1157–1182.

Hardy, G., Littlewood, J., and Pólya, G. (1952). *Inequalities*. Cambridge University Press, Cambridge, 2nd edition.

Hastie, T., Tibshirani, R., Friedman, J., *et al.* (2001). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, **57**(1), 97–109.

He, X., Zemel, R., and Carreira-Perpinan, M. (2004). Multiscale conditional random fields for image labeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 695–702.

Heckerman, D., Chickering, D., Meek, C., Rounthwaite, R., and Kadie, C. (2001). Dependency networks for inference, collaborative filtering, and data visualization. *The Journal of Machine Learning Research*, **1**, 49–75.

Heskes, T. (1998). Selecting weighting factors in logarithmic opinion pools. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10, pages 266–272. The MIT Press.

Heskes, T. (2004). On the uniqueness of loopy belief propagation fixed points. *Neural Computation*, **16**(11), 2379–2413.

Hestenes, M. and Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, **49**(6), 409–436.

Hickl, A. and Harabagiu, S. (2006). Enhanced interactive question-answering with conditional random fields. In *Proceedings of the Interactive Question Answering Workshop at HLT-NAACL*, pages 25–32.

Hinton, G. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, **14**, 1771–1800.

Hinton, G. and Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, **313**(5786), 504–507.

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, **22**(1), 89–115.

Hofmann, T. and Buhmann, J. M. (1997). Pairwise data clustering by deterministic annealing. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **19**(1), 1–14.

Ide, N. and Veronis, J. (1998). Introduction to the special issue on word sense disambiguation: The state of the art. *Computational Linguistics*, **24**(1), 1–40.

Ihler, A., Fischer III, J., and Willsky, A. (2005). Loopy belief propagation: Convergence and effects of message errors. *The Journal of Machine Learning Research*, **6**, 905–936.

Japkowicz, N. (2002). The class imbalance problem: A systematic study. *Intelligent Data Analysis*, **6**(5), 429–449.

Jaynes, E. T. (1957). Information theory and statistical mechanics. *Physical Review*, **106**, 620–630.

Jensen, D., Neville, J., and Gallagher, B. (2004). Why collective inference improves relational classification. In *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 593–598. ACM Press New York, NY, USA.

Jiao, F., Wang, S., Lee, C., Greiner, R., and Schuurmans, D. (2006). Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 21st International Conference on Computational Linguistics (COLING) and the 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 209–216. Association for Computational Linguistics Morristown, NJ, USA.

Jing, Y., Pavlovic, V., and Rehg, J. (2005). Efficient discriminative learning of Bayesian network classifiers via boosted augmented naïve Bayes. In *Proceedings of the 22nd International Conference on on Machine Learning (ICML)*, volume 119 of *ACM International Conference Proceeding Series*, pages 369–376.

Johnson, J. K., Malioutov, D., and Willsky, A. S. (2007). Lagrangian relaxation for MAP estimation in craphical models. In *45th Annual Allerton Conference on Communication, Control and Computing*.

Jordan, M., Ghahramani, Z.and Jaakkola, T., and Saul, L. (1999). An introduction to variational methods for graphical models. *Machine Learning*, **37**(2), 183–233.

Kakade, S., Teh, Y. W., and Roweis, S. (2002). An alternative objective function for Markovian fields. In *Proceedings of the 19th International Conference on Machine Learning (ICML)*, pages 275–282.

Kappen, H. and Wiegerinck, W. (2001). Mean field theory for graphical models. In M. Opper and D. Saad, editors, *Adavanced Mean Field Methods: Theory and Practice*, chapter 4, pages 37–49. MIT Press.

Kautz, H. and Allen, J. (1986). Generalized plan recognition. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 423–427, Philadelphia, PA.

Kazama, J. and Tsujii, J. (2003). Evaluation and extension of maximum entropy models with inequality constraints. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 137–144.

Kirkpatrick, S., Jr., C. D. G., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, **220**(4598), 671–680.

Knight, K. and Graehl, J. (1998). Machine transliteration. *Computational Linguistics*, **24**(4), 599–612.

Kolmogorov, V. (2005). Primal-dual algorithm for convex Markov random fields. Technical Report MSR-TR-2005-117, Microsoft Research.

Kolmogorov, V. (2006). Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **28**(10), 1568–1583.

Kolmogorov, V. and Wainwright, M. (2005). On the optimality of tree-reweighted max-product message passing. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 316–323.

Kristjannson, T., Culotta, A., Viola, P., and McCallum, A. (2004). Interactive information extraction with constrained conditional random fields. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI)*, pages 412–418, San Jose, CA.

Kschischang, F. R., Frey, B. J., and Loeliger, H. A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, **47**(2), 498–519.

Kumar, S. and Hebert, M. (2004). Discriminative fields for modeling spatial dependencies in natural images. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA.

Kumar, S. and Hebert, M. (2005). A hierarchical field framework for unified context-based classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1284–1291.

Lafferty, J. and Wasserman, L. (2006). Challenges in statistical machine learning. *Statistica Sinica*, **16**(2), 307–323.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine learning (ICML)*, pages 282–289.

Lafferty, J. D., Zhu, X., and Liu, Y. (2004). Kernel conditional random fields: Representation and clique selection. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, Banff, Canada.

Lauritzen, S. (1996). *Graphical Models*. Oxford Science Publications.

Lebanon, G. and Lafferty, J. (2002). Boosting and maximum likelihood for exponential models. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 447–454. MIT Press, Cambridge, MA.

Lee, C.-H., Greiner, R., and Schmidt, M. (2005). Support vector random fields for spatial classification. In *Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, number 3721 in Lecture Notes in Computer Science, pages 121–132.

Lee, D. and Seung, H. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, **401**(6755), 788–791.

Leordeanu, M. and Hebert, M. (2006). Efficient MAP approximation for dense energy functions. In *Proceedings of the 23rd International Conference on Machine learning (ICML)*, pages 545–552. ACM Press New York, NY, USA.

Li, H., Zhang, M., and Su, J. (2004). A joint source-channel model for machine transliteration. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 159–166, Barcelona, Spain.

Liao, L., Fox, D., and Kautz, H. (2005). Location-based activity recognition using relational Markov networks. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 773–778.

Liao, L., Fox, D., and Kautz, H. (2007). Extracting places and activities from GPS traces using hierarchical conditional random fields. *The International Journal of Robotics Research*, **26**(1), 119–134.

Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization methods. *Mathematical Programming*, **45**, 503–528.

Liu, Y., Stolcke, A., Shriberg, E., and Harper, M. (2005). Using conditional random fields for sentence boundary detection in speech. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL)*, pages 451–458. Association for Computational Linguistics Morristown, NJ, USA.

Loe, K.-F., Wu, J.-K., and Wang, Y. (2006). A dynamic conditional random field model for foreground and shadow segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **28**(2), 279–289.

MacKay, D. (1996). Introduction to Monte Carlo methods. In M. Jordan, editor, *Proceedings of NATO Advanced Study Institute on Learning in Graphical Models*, pages 175–204, Cambridge, MA. MIT Press.

MacKay, D. J. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 6.0 edition.

Macskassy, S. and Provost, F. (2007). Classification in networked data: A toolkit and a univariate case study. *The Journal of Machine Learning Research*, **8**, 935–983.

Malouf, R. (2002). A comparison of algorithms for Maximum Entropy parameter estimation. In D. Roth and A. van den Bosch, editors, *Proceedings of the 6th Conference on Natural Language Learning (CoNLL)*, pages 49–55, Taipei.

Mann, G. and McCallum, A. (2007). Efficient computation of entropy gradient for semi-supervised conditional random fields. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 109–112. Association for Computational Linguistics.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.

Mao, Y. and Lebanon, G. (2007). Isotonic conditional random fields and local sentiment flow. In *Advances in Neural Information Processing Systems 19*, pages 961–968. MIT Press, Cambridge, MA.

Marlin, B. (2004). Modeling user rating profiles for collaborative filtering. In *Advances in Neural Information Processing Systems*, volume 16, pages 627–634. MIT Press, Cambridge, MA.

Mason, L., Baxter, J., Bartlett, P., and Frean, M. (2000). Functional gradient techniques for combining hypotheses. In *Advances in Large Margin Classifiers*, pages 221–247. MIT Press.

McCallum, A. (2003). Efficiently inducing features of conditional random fields. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 403–410.

McCallum, A. and Nigam, K. (1998). A comparison of event models for naïve Bayes text classification. In *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization*, pages 41–48. AAAI Press.

McCallum, A., Freitag, D., and Pereira, F. (2000). Maximum Entropy Markov models for information extraction and segmentation. In *Proceedings of the 17th International Conference on on Machine Learning (ICML)*, pages 591–598. Morgan Kaufmann, San Francisco, CA.

McDonald, R. and Pereira, F. (2005). Identifying gene and protein mentions in text using conditional random fields. *BMC Bioinformatics*, **6**(S6).

McEliece, R. and Cheng, D. (1998). Turbo decoding as an instance of Pearl's beliefpropagation algorithm. *IEEE Journal on Selected Areas in Communications*, **16**(2), 140–152.

Meinshausen, N. and Buhlmann, P. (2006). High dimensional graphs and variable selection with the Lasso. *Annals of Statistics*, **34**(3), 1436–1462.

Meltzer, T., Yanover, C., and Weiss, Y. (2005). Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 428–435.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, N., Teller, A. H., and Teller, E. (1953). Equation of state calculation by fast computing machines. *Journal of Chemical Physics*, **21**, 1087–1092.

Minka, T. (2001a). *A Family of Algorithms for Approximate Bayesian Inference*. Ph.D. thesis, MIT.

Minka, T. (2001b). Algorithms for maximum-likelihood logistic regression. Technical report, Carnegie Mellon, http://www.stat.cmu.edu/˜minka/papers/logreg.html.

Minka, T. (2005a). Discriminative models, not discriminative training. Technical Report MSR-TR-2005-144, Microsoft Research.

Minka, T. (2005b). Divergence measures and message passing. Technical Report MSR-TR-2005-173, Microsoft Research.

Miyao, Y. and Tsujii, J. (2002). Maximum entropy estimation for feature forests. In *Proceedings of Human Language Technology Conference (HLT)*.

Mooij, J. and Kappen, H. (2005a). On the properties of the Bethe approximation and loopy belief propagation on binary networks. *Journal of Statistical Mechanics: Theory and Experiment*, **11**, P11012.

Mooij, J. and Kappen, H. (2005b). Sufficient conditions for convergence of loopy belief propagation. In *Proceedings of the 21th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 396–40, Arlington, Virginia. AUAI Press.

Morris, J. and Fosler-Lussier, E. (2006). Combining phonetic attributes using conditional random fields. In *Proceedings of the 9th International Conference on Spoken Language Processing (Interspeech)*, pages 597–600, Pittsburgh. ISCA.

Murphy, K. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. thesis, Computer Science Division, University of California, Berkeley.

Murphy, K. and Paskin, M. (2002). Linear time inference in hierarchical HMMs. In *Advances in Neural Information Processing Systems (NIPS)*, volume 2, pages 833–840. MIT Press.

Murphy, K., Weiss, Y., and Jordan, M. (1999). Loopy belief propagation for approximate inference: An empirical study. In K. Laskey and H. Prade, editors, *Proceedings of the 15th Conference on on Uncertainty in Artificial Intelligence (UAI)*, pages 467–475, Stockholm.

Neal, R. (1993). Probabilistic inference using Markov-chain Monte Carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto.

Nguyen, N., Phung, D., Venkatesh, S., and Bui, H. H. (2005). Learning and detecting activities from movement trajectories using the hierarchical hidden Markov models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 955–960, San Diego, CA.

Nigam, K., Lafferty, J., and McCallum, A. (1999). Using maximum entropy for text classification. In *Proceedings of the IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67.

Nocedal, J. and Wright, S. J. (1999). *Numerical Optimisation*. Springer-Verlag New York Inc.

Oliver, N., Garg, A., and Horvitz, E. (2004). Layered representations for learning and inferring office activity from multiple sensory channels. *Computer Vision and Image Understanding*, **96**(2), 163–180.

Papineni, K., Roukos, S., Ward, T., and Zhu, W. (2001). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL)*, pages 311–318. Association for Computational Linguistics Morristown, NJ, USA.

Parise, S. and Welling, M. (2007). Bayesian model scoring in Markov random fields. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1073–1080. MIT Press, Cambridge, MA.

Pazzani, M. (1999). A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, **13**(5), 393–408.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA.

Pelillo, M. and Refice, M. (1994). Learning compatibility coefficients for relaxation labeling processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **16**(9), 933–945.

Peng, F. and McCallum, A. (2004). Accurate information extraction from research papers using conditional random fields. In D. M. Susan Dumais and S. Roukos, editors, *Proceedings of Human Language Technology (HLTNAACL)*, pages 329–336, Boston, Massachusetts, USA. Association for Computational Linguistics.

Peng, F., Feng, F., and McCallum, A. (2004). Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 562–568.

Pennock, D. and Wellman, M. (1999). Graphical representations of consensus belief. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 531–540.

Pereira, F. and Schabes, Y. (1992). Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the Meeting of the Association for Computational Linguistics (ACL)*, pages 128–135.

Phan, X.-H., Nguyen, L.-M., Ho, T.-B., and Horiguchi, S. (2005). Improving discriminative sequential learning with rare–but–important associations. In *Proceeding of the 11th ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 304–313, Chicago, Illinois, USA.

Phung, D. Q. (2005). *Probabilistic and Film Grammar based Methods for Video Content Understanding*. Ph.D. thesis, Dept. Computing, Curtin University of Technology.

Pietra, S. D., Pietra, V. D., and Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **19**(4), 380–393.

Pinto, D., McCallum, A., Wei, X., and Croft, W. (2003). Table extraction using conditional random fields. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 235–242. ACM Press New York, NY, USA.

Qi, Y., Szummer, M., and Minka, T. P. (2005). Bayesian conditional random fields. In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics (AISTATS)*.

Quattoni, A., Collins, M., and Darrell, T. (2005). Conditional random fields for object recognition. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1097–1104. MIT Press, Cambridge, MA.

Quattoni, A., Wang, S., Morency, L., Collins, M., and Darrell, T. (2007). Hidden conditional random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **29**(10), 1848–1852.

Quinlan, J. (1993). *C4. 5: Programs for Machine Learning*. Morgan Kaufmann.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, **77**(2), 257–286.

Ratnaparkhi, A. (1996). A maximum entropy part-of-speech tagger. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 17–18.

Ravikumar, P. and Lafferty, J. (2006). Quadratic programming relaxations for metric labeling and Markov random field MAP estimation. In *Proceedings of the 23rd international conference on Machine learning (ICML)*, pages 737–744. ACM Press New York, NY, USA.

Rennie, J. and Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 713–719, Bonn, Germany.

Resnick, P., Iacovou, N., Suchak, M., Bergstorm, P., and Riedl, J. (1994). GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina. ACM.

Richardson, M. and Domingos, P. (2006). Markov logic networks. *Machine Learning*, **62**, 107–136.

Roark, B., Saraclar, M., Collins, M., and Johnson, M. (2004). Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 47–54, Barcelona.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, **22**(400–407).

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol Rev*, **65**(6), 386–408.

Salakhutdinov, R., Mnih, A., and Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 791–798.

Sang, E. F. T. K. and Buchholz, S. (2000). Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning*, volume 7, pages 127–132, Lisbon, Portugal. http://www.cnts.ua.ac.be/conll2000/chunking/.

Sanghavi, S. (2007). Equivalence of LP relaxation and max-product for weighted matching in general graphs. In *Proceedings of the IEEE Information Theory Workshop (IWT)*, pages 242–247.

Sarawagi, S. and Cohen, W. W. (2004). Semi-Markov conditional random fields for information extraction. In B. L. Saul LK, Weiss Y, editor, *Advances in Neural Information Processing Systems 17*, pages 1185–1192. MIT Press, Cambridge, Massachusetts.

Sarwar, B., Karypis, G., Konstan, J., and Reidl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM Press New York, NY, USA.

Saul, L., Jaakkola, T., and Jordan, M. (1996). Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, **4**, 61–76.

Saul, L. K. and Jordan, M. I. (1995). Boltzmann chains and hidden Markov models. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 435–442. MIT Press, Cambridge, MA.

Schapire, R., Freund, Y., Bartlett, P., and Lee, W. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, **26**(5), 1651–1686.

Schapire, R. E. and Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, **37**(3), 297–336.

Schapire, R. E. and Singer, Y. (2000). BoosTexter: A boosting-based system for text categorization. *Machine Learning*, **39**(2-3), 135–168.

Scharstein, D. and Pal., C. (2007). Learning conditional random fields for stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Minneapolis.

Schmidt, M., Niculescu-Mizil, A., and Murphy, K. (2007). Learning graphical model structure using l1-regularization paths. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI)*, pages 1278–1283.

Sen, P. and Getoor, L. (2006). Cost-sensitive learning with conditional Markov networks. In *Proceedings of the 23rd International Conference on Machine learning (ICML)*, volume 148 of *ACM International Conference Proceeding Series*, pages 801–808, Pittsburgh, Pennsylvania.

Settles, B. (2004). Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA)*, pages 104–107, Geneva, Switzerland.

Sha, F. and Pereira, F. (2003). Shallow parsing with conditional random fields. In M. Hearst and M. Ostendorf, editors, *Proceedings of Human Language Technology (NAACL)*, pages 213–220, Edmonton, Alberta, Canada. Association for Computational Linguistics.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Tech. Journal*, **27**, 379–423 and 623–656.

Shen, D., Sun, J., Li, H., Yang, Q., and Chen, Z. (2007). Document summarization using conditional random fields. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, volume 7, pages 2862–2867.

Sminchisescu, C., Kanaujia, A., and Metaxas, D. (2006). Conditional models for contextual human motion recognition. *Computer Vision and Image Understanding*, **104**(2-3), 210–220.

Smith, A., Cohn, T., and Osborne, M. (2005). Logarithmic opinion pools for conditional random fields. In *Proceedings 43rd Annual Meeting of the Association for Computational Linguists (ACL)*, pages 18–25, Ann Arbor, Michigan.

Sun, J., Zheng, N.-N., and Shum, H.-Y. (2003). Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **25**(7), 787–800.

Sutton, C. and McCallum, A. (2005). Piecewise training for undirected models. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 568–575.

Sutton, C. and McCallum, A. (2006). An introduction to conditional random fields for relational learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*, chapter 4, pages 93–128. MIT Press.

Sutton, C. and McCallum, A. (2007a). Improved dynamic schedules for belief propagation. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 376–383.

Sutton, C. and McCallum, A. (2007b). Piecewise pseudolikelihood for efficient CRF training. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 863–870.

Sutton, C., McCallum, A., and Rohanimanesh, K. (2007). Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research*, **8**, 693–723.

Suzuki, J., McDermott, E., and Isozaki, H. (2006). Training conditional random fields with multivariate evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics (COLING) and the 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 217–224. Association for Computational Linguistics Morristown, NJ, USA.

Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., and Rother, C. (2006). A comparative study of energy minimization methods for

Markov random fields. In *Proceedings of the European Conference on Computer Vision (ECCV)*, number 3952 in Lecture Notes in Computer Science, pages 16–29.

Tappen, M. F., Liu, C., Adelson, E. H., and Freeman, W. T. (2007). Learning Gaussian conditional random fields for low-level vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8.

Taskar, B., Pieter, A., and Koller, D. (2002). Discriminative probabilistic models for relational data. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 485–49. Morgan Kaufmann.

Taskar, B., Guestrin, C., and Koller, D. (2004). Max-margin Markov networks. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA.

Taycher, L., Shakhnarovich, G., Demirdjian, D., and Darrell, T. (2006). Conditional random people: Tracking humans with CRFs and grid filters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 222–229. IEEE Computer Society Washington, DC, USA.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, **58**(1), 267–288.

Torralba, A., Murphy, K. P., and Freeman, W. T. (2005). Contextual models for object detection using boosted random fields. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1401–1408. MIT Press, Cambridge, MA.

Truyen, T. T., Phung, D. Q., Bui, H. H., and Venkatesh, S. (2006). AdaBoost.MRF: Boosted Markov random forests and application to multilevel activity recognition. In *Computer Vision and Pattern Recognition*, volume 2, pages 1686–1693, New York, USA.

Truyen, T. T., Phung, D. Q., and Venkatesh, S. (2007). Preference networks: Probabilistic models for recommendation systems. In P. Christen, P. Kennedy, J. Li, I. Kolyshkina, and G. Williams, editors, *The 6th Australasian Data Mining Conference (AusDM)*, volume 70 of *CRPIT*, pages 195–202, Gold Coast, Australia. ACS.

Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, **6**, 1453–1484.

Vail, D., Veloso, M., and Lafferty, J. (2007). Conditional random fields for activity recogntion. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, Honolulu, Hawaii.

Vinson, J. P., DeCaprio, D., Pearson, M. D., Luoma, S., and Galagan, J. E. (2007). Comparative gene prediction using conditional random fields. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1441–1448. MIT Press, Cambridge, MA.

Vishwanathan, S. V. N., Schraudolph, N. N., Schmidt, M. W., and Murphy, K. P. (2006). Accelerated training of conditional random fields with stochastic gradient methods. In *Proceedings of the International Conference on Machine learning (ICML)*, pages 969–976.

Wainwright, M. J. (2006). Estimating the "wrong" graphical model: Benefits in the computation-limited setting. *Journal of Machine Learning Research*, **7**, 1829–1859.

Wainwright, M. J. and Jordan, M. I. (2003). Graphical models, exponential families, and variational inference. Technical Report 649, Department of Statistics, University of California, Berkeley.

Wainwright, M. J., Jaakkola, T., and Willsky, A. S. (2003a). Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on Information Theory*, **45**(9), 1120–1146.

Wainwright, M. J., Jaakkola, T., and Willsky, A. S. (2003b). Tree-reweighted belief propagation algorithms and approximate ML estimation by pseudo-moment matching. In *Proceedings of the 9th Workshop on Artificial Intelligence and Statistics (AISTATS)*, Key West, Florida.

Wainwright, M. J., Jaakkola, T. S., and Willsky, A. S. (2005a). MAP estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches. *IEEE Transactions on Information Theory*, **51**(11), 3697–3717.

Wainwright, M. J., Jaakkola, T., and Willsky, A. S. (2005b). A new class of upper bounds on the log partition function. *IEEE Transactions on on Information Theory*, **51**, 2313–2335.

Wainwright, M. J., Ravikumar, P., and Lafferty, J. (2006). High-dimensional graphical model selection using ℓ_1-regularized logistic regression. In *Proceedings of the Neural Information Processing Systems (NIPS)*, pages 1465–1472.

Wan, S. and Verspoor, C. (1998). Automatic English-Chinese name transliteration for development of multilingual resources. In *Proceedings of 17th International Conference on Computational Linguistics (COLING) and 36th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1352–1356.

Weiss, Y. and Freeman, W. (2001a). Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Computation*, **13**(10), 2173–2200.

Weiss, Y. and Freeman, W. (2001b). On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, **47**(2), 736–744.

Welling, M. and Teh, Y. (2001). Belief optimization for binary networks: a stable alternative to loopy belief propagation. In *Proceedings of the 17th International Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 554–561.

Wiegerinck, W. (2000). Variational approximations between mean field theory and the junction tree algorithm. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 626–633.

Wiegerinck, W. and Heskes, T. (2003). Fractional belief propagation. In S. T. S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 438–445. MIT Press, Cambridge, MA.

Willsky, A. (2002). Multiresolution Markov models for signal and image processing. *Proceedings of the IEEE*, **90**(8), 1396–1458.

Winn, J. and Shotton, J. (2006). The layout consistent random field for recognizing and segmenting partially occluded objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 37–44.

Yanover, C. and Weiss, Y. (2003). Finding the M most probable configurations using loopy belief propagation. In *Advances in Neural Information Processing Systems*, volume 16. MIT Press, Cambridge, MA.

Yanover, C., Meltzer, T., and Weiss, Y. (2006). Linear programming relaxations and belief propagation–an empirical study. *The Journal of Machine Learning Research*, **7**, 1887–1907.

Yarowsky, D. (1994). Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. In *Proceedings of the Meeting of the Association for Computational Linguistics (ACL)*, volume 32, pages 88–95. Association for Computational Linguistics.

Yedidia, J., Freeman, W., and Weiss, Y. (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, **51**(7), 2282–2312.

Yin, P., Essa, I., and Rehg, J. M. (2004). Asymmetrically boosted HMM for speech reading. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 02, pages 755–761.

Yuille, A. L. (2002). CCCP algorithms to minimize the Bethe and Kikuchi free energies: Convergent alternatives to belief propagation. *Neural Computation*, **14**(7), 1691–1722.

Zhang, R., Kikui, G., and Sumita, E. (2006a). Subword-based tagging by conditional random fields for Chinese word segmentation. In *Proceedings of the Human Language Technology Conference of the NAACL (NAACL)*, pages 193–196, New York City, USA. Association for Computational Linguistics.

Zhang, S., Wang, W., Ford, J., and Makedon, F. (2006b). Learning from incomplete ratings using non-negative matrix factorization. In *Proceedings of the 6th SIAM Conference on Data Mining (SDM)*, Bethesda, MD.

Zhang, T. (2004). Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, Banff, Alberta, Canada.

Zhang, T. and Iyengar, V. (2002). Recommender systems using linear classifiers. *Journal of Machine Learning Research*, **2**(3), 313–334.

Zhu, J., Nie, Z., Wen, J., Zhang, B., and Ma, W. (2005). 2D conditional random fields for web information extraction. In *Proceedings of the 22nd International Conference on Machine learning (ICML)*, volume 119 of *ACM International Conference Proceeding Series*, pages 1044–1051, Bonn, Germany.

Zhu, S., Wu, Y., and Mumford, D. (1998). Filters, Random Fields and Maximum Entropy (FRAME): towards a unified theory for texture modeling. *International Journal of Computer Vision*, **27**(2), 107–126.

Zhu, X. (2006). Semi-supervised learning literature survey. Technical Report TR-1530, Computer Science, University of Wisconsin-Madison. This is updated over time.

Zitnick, C. and Kanade, T. (2004). Maximum entropy for collaborative filtering. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 636–643. AUAI Press Arlington, Virginia, United States.

Appendix A

Appendix

A.1 Derivation of Variational Updates

In this appendix we will show how to obtain Equations 2.85 and 2.86, which we repeat here for convenience

$$\mu_{\beta\to\alpha}(x_{c,\alpha}) = \exp\left(\sum_{x_{c,\beta}} Q_{c,\beta}(x_{c,\beta}) \log \psi_{c,\alpha,\beta}(x_{c,\alpha,\beta})\right) \tag{A.1}$$

$$Q_\alpha(x_\alpha) \propto \Phi_\alpha(x_\alpha) \prod_{\beta\in\mathcal{N}(\alpha)} \prod_{c\in\alpha\cap\beta} \mu_{\beta\to\alpha}(x_{c,\alpha}) \tag{A.2}$$

where α, β are two subnetworks that share the common clique variables $x_{c,\alpha,\beta}$, $x_{c,\alpha}$ is the part of this subset that exclusively belongs to α; $\mu_{\beta\to\alpha}(x_{c,\alpha})$ is the message sending from β to $x_{c,\alpha}$; and $\mathcal{N}(\alpha)$ is the set of neighbouring sub-networks of α.

Recall that we want to minimise the KL-divergence between the approximate distribution $Q(x)$ and the true distribution $\Pr(x)$

$$\hat{Q} = \arg\min_Q KL(Q||\Pr) \tag{A.3}$$

$$= \arg\min_Q \sum_x Q(x) \log \frac{Q(x)}{\Pr(x)} \tag{A.4}$$

where $Q(x)$ is constrained to the factorised distribution

$$Q(x) = \prod_\alpha Q_\alpha(x_\alpha) \tag{A.5}$$

Now let us expand the KL-divergence

$$KL(Q||\Pr) = \sum_x Q(x)\log Q(x) - \sum_x Q(x)\log \Phi(x) + \log Z \tag{A.6}$$

where we have used $\Pr(x) = \Phi(x)/Z$. Since we are only interested in finding $Q(x)$, the last term can be neglected. The first term is the negative entropy of the new network, which is the negative sum of entropies of independent sub-networks

$$\sum_x Q(x)\log Q(x) = \sum_\alpha \sum_{x_\alpha} Q_\alpha(x_\alpha)\log Q_\alpha(x_\alpha) \tag{A.7}$$

The factorisation in Equation A.5 implies that $Q(x) = Q(x\backslash x_\alpha)Q_\alpha(x_\alpha)$. As we need to ensure that $\sum_{x_\alpha} Q_\alpha(x_\alpha) = 1$, adding a Lagrangian term to the KL-divergence and then taking derivative of Equation A.6 with respect to $Q_\alpha(x_\alpha)$ yields

$$\frac{\partial KL(Q||\Pr)}{\partial Q_\alpha(x_\alpha)} = \log Q_\alpha(x_\alpha) + 1 - \sum_{x\backslash x_\alpha} Q(x\backslash x_\alpha)\log \Phi(x) + \lambda_\alpha \tag{A.8}$$

where λ_α is the Lagrangian factor. Setting this gradient to zero, we achieve

$$Q_\alpha(x_\alpha) \propto \exp\left\{\log \Phi_\alpha(x_\alpha) + \sum_{x\backslash x_\alpha} Q(x\backslash x_\alpha)\log \Phi(x)\right\} \tag{A.9}$$

Since α is assumed to be a tractable sub-network, it remains to efficiently compute $\sum_{x\backslash x_\alpha} Q(x\backslash x_\alpha)\log \Phi(x)$.

Due to network partitioning, $x = (x_{\alpha_1}, x_{\alpha_2}, ...)$, we can decompose x into three parts: those belong exclusively to a sub-network α, those to other sub-networks, and those at the boundary between α and other sub-networks. Thus we have the following factorisation

$$\Phi(x) = \Phi_\alpha(x_\alpha)\Phi_{-\alpha}(x\backslash x_\alpha) \prod_{\beta\in\mathcal{N}(\alpha)} \prod_{c\in\alpha\cap\beta} \psi_{c,\alpha,\beta}(x_{c,\alpha,\beta}) \tag{A.10}$$

where $\Phi_\alpha(x_\alpha)$ is the product of local clique potentials belonging the to sub-network α. and $\Phi_{-\alpha}(x\backslash x_\alpha)$ is the product of local clique potentials of other sub-networks.

Then the third term of the RHS of Equation A.8 becomes

$$\begin{aligned}\sum_{x\backslash x_\alpha} Q(x\backslash x_\alpha)\log \Phi(x) \;=\; & \log \Phi_\alpha(x_\alpha) + \sum_{x\backslash x_\alpha} Q(x\backslash x_\alpha)\log \Phi_{-\alpha}(x\backslash x_\alpha) + \\ & + \sum_{x\backslash x_\alpha} Q(x\backslash x_\alpha) \sum_{\beta\in\mathcal{N}(\alpha)} \sum_{c\in\alpha\cap\beta} \log \psi_{c,\alpha,\beta}(x_{c,\alpha,\beta})\end{aligned} \tag{A.11}$$

The second term of the RHS of Equation A.11 is a constant with respect to $Q_\alpha(x_\alpha)$, while the third term reduces to

$$\sum_{x\backslash x_\alpha} Q(x\backslash x_\alpha) \sum_{\beta\in\mathcal{N}(\alpha)} \sum_{c\in\alpha\cap\beta} \log\psi_{c,\alpha,\beta}(x_{c,\alpha,\beta}) = \sum_{\beta\in\mathcal{N}(\alpha)} \sum_{c\in\alpha\cap\beta} \sum_{x_{c,\beta}} Q_{c,\beta}(x_{c,\beta}) \log\psi_{c,\alpha,\beta}(x_{c,\alpha,\beta}) \quad \text{(A.12)}$$

In the last equation we have split $x\backslash x_\alpha$ into $x_{c,\beta}$ and the rest, which are integrated out with $Q(x\backslash x_\alpha)$.

Substituting Equation A.12 into Equation A.11 and then Equation A.11 into Equation A.9, and setting the gradient to zero, we have

$$Q_\alpha(x_\alpha) \propto \exp\left\{\log\Phi_\alpha(x_\alpha) + \sum_{\beta\in\mathcal{N}(\alpha)} \sum_{c\in\alpha\cap\beta} \sum_{x_{c,\beta}} Q_{c,\beta}(x_{c,\beta}) \log\psi_{c,\alpha,\beta}(x_{c,\alpha,\beta})\right\} \quad \text{(A.13)}$$

Rearranging the terms in the RHS into the appropriate messages, we obtain the Equations A.1 and A.2.

A.2 General Hölder's inequality

Let us start with the elementary Hölder's inequalities (Hardy *et al.*, 1952, Theorem 13). *For* $r \geq 1, a \geq 0, b \geq 0$ *and* $1/r + 1/r' = 1$*, the following holds*

$$\sum_{i=1}^{n} a_i b_i \leq (\sum_{i=1}^{n} a_i^r)^{1/r} (\sum_{i=1}^{n} b_i^{r'})^{1/r'} \quad \text{(A.14)}$$

The sign of equality hold iff $a_i^r = \alpha b_i^{r'}, \forall i$*, for some scalar* α . The case $\alpha = 0$ is trivial, thus we do not consider here. The Cauchy's inequality is a special case if $r = r' = 2$.

By induction, we can obtain the following extension to this basic inequality (Hardy *et al.*, 1952, Theorem 11). *If* $a_{ij} \geq 0$*, for* $i = 1, 2, .., n$ *and* $j = 1, 2, .., m$ *, and if* $r_j > 1$ *with* $\sum_{j=1}^{m} 1/r_j = 1$*, denoting* $A_j = \{a_{ij}\}_{i=1}^{n}$*, then*

$$\sum_{i=1}^{n} \prod_{j=1}^{m} a_{ij} \leq \prod_{j=1}^{m} (\sum_{i=1}^{n} a_{ij}^{r_j})^{1/r_j}, \quad \text{(A.15)}$$

and the sign of equality holding iff $A_j = \alpha_{jj'} A_{j'\neq j}$ *for some scalars* $\alpha_{jj'}$*. In other words, the equality sign holds iff all vectors* A_j *are proportional.*

Let us proceed by induction to prove the 'inequality' part.

Base Case: For $j = 1$, Equation A.15 holds trivially.

Induction: Assume Equation A.15 holds for any $1 \leq j \leq l$, we will prove that it also holds for $j = l + 1$. Then

$$\begin{aligned}
\sum_i \prod_{j=1}^{l+1} a_{i,j} &= \sum_i a_{i,l+1} \prod_{j=1}^{l} a_{i,j} \\
&\leq \left(\sum_i a_{i,l+1}^r \right)^{1/r} \left(\sum_i \{ \prod_{j=1}^{l} a_{i,j} \}^{r'} \right)^{1/r'} \\
&= \left(\sum_i a_{i,l+1}^r \right)^{1/r} \left(\sum_i \prod_{j=1}^{l} a_{i,j}^{r'} \right)^{1/r'} \qquad \text{(A.16)}
\end{aligned}$$

using the basic Hölder inequality (A.14) for $r > 1; r' > 1; 1/r + 1/r' = 1$. Let $\beta_{i,j} = a_{i,j}^{r'}$, then the applying our assumption that Equation A.15 holds for $j \leq l$ to the second factor in the RHS of Equation A.16 yields

$$\begin{aligned}
\left(\sum_i \prod_{j=1}^{l} \beta_{i,j} \right)^{1/r'} &\leq \left(\prod_{j=1}^{l} \{ \sum_i \beta_{i,j}^{r'_j} \}^{1/r'_j} \right)^{1/r'} \\
&= \prod_{j=1}^{l} \{ \sum_i \beta_{i,j}^{r'_j} \}^{1/r'_j r'} \\
&= \prod_{j=1}^{l} \{ \sum_i a_{i,j}^{r'_j r'} \}^{1/r'_j r'} \qquad \text{(A.17)}
\end{aligned}$$

where $\sum_{j=1}^{l} 1/r'_j = 1$. Substituting Equation A.17 back into Equation A.16, we have

$$\sum_i \prod_{j=1}^{l+1} a_{i,j} \leq \left(\sum_i a_{i,l+1}^r \right)^{1/r} \prod_{j=1}^{l} \{ \sum_i a_{i,j}^{r'_j r'} \}^{1/r'_j r'} \qquad \text{(A.18)}$$

As $\sum_{j=1}^{l} 1/r'_j = 1$, we then have $1/r + \sum_{j=1}^{l} 1/r'_j r' = 1/r + 1/r' = 1$. Now we change the notation as $r'_{l+1} \leftarrow r$ and $r'_j \leftarrow r'_j r'$, we have $\sum_{j=1}^{l+1} 1/r'_j = 1$. Thus Equation A.18 becomes

$$\sum_i \prod_{j=1}^{l+1} a_{i,j} \leq \prod_{j=1}^{l+1} \{ \sum_i a_{i,j}^{r'_j} \}^{1/r'_j} \qquad \text{(A.19)}$$

This means that the inequality Equation A.15 holds for $j = l+1$. By the induction principle the inequality Equation A.15 holds for all $j \geq 1$. This completes the proof ■

A.3 Proofs

A.3.1 Proof of Propositions 3 and 4

Before proving Proposition 3 and 4 let us introduce a lemma.

Lemma 1. *Given a distribution of the form*

$$\Pr(x) = \frac{1}{Z}\Phi[x] \tag{A.20}$$

where $x = (x_a, x_s, x_b)$*, if there exists a factorisation*

$$\Phi[x] = \Phi[x_a, x_s]\Phi[x_s]\Phi[x_s, x_b] \tag{A.21}$$

then x_a *and* x_b *are conditionally independent given* x_s*.*

Proof: We want to prove that

$$\Pr(x_a, x_b|x_s) = \Pr(x_a|x_s)\Pr(x_b|x_s) \tag{A.22}$$

Since $\Pr(x_a, x_b|x_s) = \Pr(x_a, x_b, x_s)/\sum_{x_a,x_b}\Pr(x_a, x_b, x_s)$, the LHS of Equation A.22 becomes

$$\begin{aligned}\Pr(x_a, x_b|x_s) &= \frac{\Phi[x_a, x_s]\Phi[x_s]\Phi[x_s, x_b]}{\sum_{x_a,x_b}\Phi[x_a, x_s]\Phi[x_s]\Phi[x_s, x_b]} \\ &= \frac{\Phi[x_a, x_s]}{\sum_{x_a}\Phi[x_a, x_s]}\frac{\Phi[x_s, x_b]}{\sum_{x_b}\Phi[x_s, x_b]}\end{aligned} \tag{A.23}$$

where we have used the following fact

$$\sum_{x_a,x_b}\Phi[x_a, x_s]\Phi[x_s]\Phi[x_s, x_b] = \Phi[x_s]\Big(\sum_{x_a}\Phi[x_a, x_s]\Big)\Big(\sum_{x_b}\Phi[x_s, x_b]\Big) \tag{A.24}$$

and canceled out the normalisation factor Z and $\Phi[x_s]$.

To prove $\Pr(x_a|x_s) = \Phi[x_a, x_s]/\sum_{x_a}\Phi[x_a, x_s]$, we need only to show $\Pr(x_a|x_s) \propto \Phi[x_a, x_s]$ since the normalisation over x_a is due to $\sum_{x_a}\Pr(x_a|x_s) = 1$. Using the Bayes

rule, we have

$$\begin{aligned}\Pr(x_a|x_s) &\propto \Pr(x_a, x_s) \\ &= \sum_{x_b} \Pr(x_a, x_s, x_b) \\ &= \frac{1}{Z}\Phi[x_a, x_s]\Phi[x_s] \sum_{x_b} \Phi[x_s, x_b] \\ &\propto \Phi[x_a, x_s] \end{aligned} \tag{A.25}$$

where we have ignored all the factors that do not depend on x_a.

A similar proof gives $\Pr(x_b|x_s) = \Phi[x_s, x_b]/\sum_{x_b} \Phi[x_s, x_b]$. Combining this result and Equation A.25 with Equation A.23 gives us Equation A.22. This completes the proof ■

In fact, x_s acts as a separator between x_a and x_b. In standard Markov networks there are no paths from x_a to x_b that do not go through x_s. Now we proceed to proving Proposition 3 and 4.

Given the symmetric Markov blanket $\Pi^{d,s}_{i:j}$, there are no potentials that are associated with variables belonging to both $\zeta^{d,s}_{i:j}$ and $\underline{\zeta}^{d,s}_{i:j}$. The blanket completely separates the $\zeta^{d,s}_{i:j}$ and $\underline{\zeta}^{d,s}_{i:j}$. Therefore, Lemma 1 ensures the conditional independence between $\zeta^{d,s}_{i:j}$ and $\underline{\zeta}^{d,s}_{i:j}$.

Similarly, the asymmetric Markov blanket $\Gamma^{d,s}_{i:j}(u)$ separates $\zeta^{d,s}_{i:j}(u)$ and $\underline{\zeta}^{d,s}_{i:j}(u)$ and thus these two variable sets are conditionally independent due to Lemma 1 ■

A.3.2 Proof of Proposition 5

Here we want to derive Equations 8.25, 8.26 and 8.27. With the same conditions as in Lemma 1, in Equation A.25 we have shown that $\Pr(x_a|x_s) \propto \Phi[x_a, x_s]$. Similarly, this extends to

$$\begin{aligned}\Pr(\zeta^{d,s}_{i:j}|\Pi^{d,s}_{i:j}) &\propto \Phi[\zeta^{d,s}_{i:j}, \Pi^{d,s}_{i:j}] \\ &= \Phi[\hat{\zeta}^{d,s}_{i:j}] \end{aligned} \tag{A.26}$$

which is equivalent to

$$\begin{aligned}\Pr(\zeta^{d,s}_{i:j}|\Pi^{d,s}_{i:j}) &= \frac{1}{\sum_{\zeta^{d,s}_{i:j}} \Phi[\hat{\zeta}^{d,s}_{i:j}]}\Phi[\hat{\zeta}^{d,s}_{i:j}] \\ &= \frac{1}{\Delta^{d,s}_{i:j}}\Phi[\hat{\zeta}^{d,s}_{i:j}] \end{aligned} \tag{A.27}$$

The last equation follows from the definition of the symmetric inside mass in Equation 8.18. Similar procedure will yield Equation 8.26.

To prove Equation 8.27, notice the Equation 8.14 that says

$$\Pr(\zeta) = \Pr(\Pi_{i:j}^{d,s}) \Pr(\zeta_{i:j}^{d,s}|\Pi_{i:j}^{d,s}) \Pr(\underline{\zeta}_{i:j}^{d,s}|\Pi_{i:j}^{d,s}) \tag{A.28}$$

or equivalently

$$\begin{aligned}
\Pr(\Pi_{i:j}^{d,s}) &= \Pr(\zeta) \frac{1}{\Pr(\zeta_{i:j}^{d,s}|\Pi_{i:j}^{d,s})} \frac{1}{\Pr(\underline{\zeta}_{i:j}^{d,s}|\Pi_{i:j}^{d,s})} && \text{(A.29)} \\
&= \frac{1}{Z}\Phi[\zeta] \frac{\Delta_{i:j}^{d,s}}{\Phi[\hat{\zeta}_{i:j}^{d,s}]} \frac{\Lambda_{i:j}^{d,s}}{\Phi[\underline{\hat{\zeta}}_{i:j}^{d,s}]} && \text{(A.30)} \\
&= \frac{1}{Z}\Phi[\hat{\zeta}_{i:j}^{d,s}] R_{i:j}^{d,s} \Phi[\underline{\hat{\zeta}}_{i:j}^{d,s}] \frac{\Delta_{i:j}^{d,s}}{\Phi[\hat{\zeta}_{i:j}^{d,s}]} \frac{\Lambda_{i:j}^{d,s}}{\Phi[\underline{\hat{\zeta}}_{i:j}^{d,s}]} && \text{(A.31)} \\
&= \frac{1}{Z}\Delta_{i:j}^{d,s} R_{i:j}^{d,s} \Lambda_{i:j}^{d,s} && \text{(A.32)}
\end{aligned}$$

In the proof proceeding, we have made use of the relation in Equation 8.17. This completes the proof ■

A.4 Computing the State Marginals of HCRF

We are interested in computing the marginals of state variables $\Pr(x_t^d)$. We have

$$\begin{aligned}
\Pr(x_t^d) &= \sum_{\zeta \backslash x_t^d} \Pr(x_t^d, \zeta \backslash x_t^d) \\
&= \sum_{\zeta} \Pr(\zeta) \delta(x_t^d \in \zeta) \\
&= \frac{1}{Z} \sum_{\zeta} \Phi[\zeta] \delta(x_t^d \in \zeta) && \text{(A.33)}
\end{aligned}$$

Let $s = x_t^d$ and assume that the state s starts at i and end at j, and $t \in [i, j]$. For each configuration ζ that respects this assumption, we have the factorisation of Equation 8.17 that says

$$\Phi[\zeta] = \Phi[\hat{\zeta}_{i:j}^{d,s}] \Phi[\underline{\hat{\zeta}}_{i:j}^{d,s}] R_{i:j}^{d,s} \tag{A.34}$$

Then Equation A.33 becomes

$$\begin{aligned}\Pr(x_t^d = s) &= \frac{1}{Z}\sum_{\zeta}\Phi[\hat{\zeta}_{i:j}^{d,s}]\Phi[\underline{\hat{\zeta}}_{i:j}^{d,s}]R_{i:j}^{d,s}\delta(t\in[i,j]) \\ &= \frac{1}{Z}\sum_{i\in[1,t]}\sum_{j\in[t,T]}\Delta_{i:j}^{d,s}\Lambda_{i:j}^{d,s}R_{i:j}^{d,s} \end{aligned} \tag{A.35}$$

The summing over i and j is due to the fact that we do not know these indices.

There are two special cases, (1) when $d = 1$ we cannot scan the left and right indices, the marginals are simply

$$\Pr(x_t^1 = s) = \frac{1}{Z}\hat{\Delta}_{1:T}^{1,s} \tag{A.36}$$

since $\Lambda_{1:T}^{1,s} = 1$ for all $s \in S^1$; and (2) when $d = D$, the start and end times must be the same ($i = j$), thus

$$\Pr(x_t^D = s) = \frac{1}{Z}\hat{\Lambda}_{t:t}^{D,s} \tag{A.37}$$

since $\Delta_{t:t}^{D,s} = 1$ for all $t \in [1, T]$ and $s \in S^D$.

Since $\sum_{s\in S^d}\Pr(x_t^d = s) = 1$, it follows from Equation A.35 that

$$Z = \sum_{s\in S^d}\sum_{i\in[1,t]}\sum_{j\in[t,T]}\Delta_{i:j}^{d,s}\Lambda_{i:j}^{d,s}R_{i:j}^{d,s} \tag{A.38}$$

This turns out to be the most general way of computing the partition function. Some special cases have been shown earlier. For example, when $d = 1$, $i = 1$ and $j = T$, Equation A.38 becomes Equation 8.22 since $\Lambda_{1:T}^{1,s} = 1$. Similarly, when $d = D$, $i = j = t$, Equation A.38 recovers Equation 8.23 since $\Delta_{i:i}^{D,s} = 1$.

A.5 The Mirrored Version of AIO

Due to the fact that the HCRFs are undirected there is actually no bias in the direction where the time indices are scanned. It is therefore straightforward to derive a mirrored and equivalent version of the AIO algorithm described in Section 8.3. In what follows we present only building blocks for the mirrored AIO algorithm and some variants of ESS computation. Other computation including the MAP estimation, learning and inference with partially observed state information and numerical scaling can be derived straightforwardly using these blocks and methods described in the main text.

A.5.1 Mirrored Markov Blankets

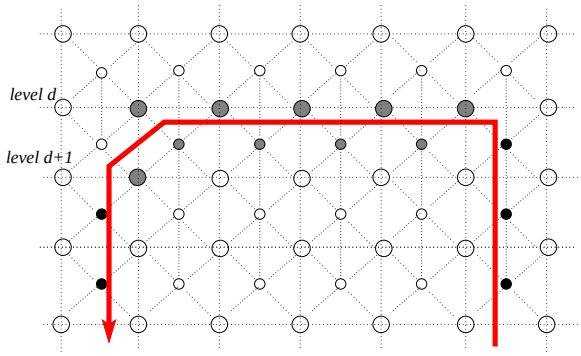


Figure A.1: A mirrored asymmetric Markov blanket.

Let us define a *mirrored asymmetric Markov blanket* (Figure A.1) as follows:

Definition 11. *A mirrored asymmetric Markov blanket at level d for a parent state s ending at j and a child state u starting at i, is the following set*

$$\mathfrak{I}^{d,s}_{i:j}(u) = (x^d_{i:j} = s, x^{d+1}_i = u, e^{d+1:D}_{i-1} = 1, e^{d:D}_j = 1, e^d_{i:j-1} = 0) \tag{A.39}$$

Further, let us define the following sets of variables that are associated with the blanket

$$\xi^{d,s}_{i:j}(u) = (x^{d+1:D}_{i+1:j}, x^{d+2:D}_i, e^{d+1:D}_{i:j-1}) \tag{A.40}$$

$$\underline{\xi}^{d,s}_{i:j}(u) = \zeta \backslash (\xi^{d,s}_{i:j}(u), \mathfrak{I}^{d,s}_{i:j}(u)) \tag{A.41}$$

We define $\hat{\xi}^{d,s}_{i:j}(u)$ and $\underline{\hat{\xi}}^{d,s}_{i:j}(u)$ as follows

$$\hat{\xi}^{d,s}_{i:j}(u) = (\xi^{d,s}_{i:j}(u), \mathfrak{I}^{d,s}_{i:j}(u)) \tag{A.42}$$

$$\underline{\hat{\xi}}^{d,s}_{i:j}(u) = (\underline{\xi}^{d,s}_{i:j}(u), \mathfrak{I}^{d,s}_{i:j}(u)) \tag{A.43}$$

Remark: $\mathfrak{I}^{d,s}_{i:j}(u)$ is a 'mirrored' $\Gamma^{d,s'}_{i:j}(v)$ in the sense that u is the starting child of s while v is the ending child of s'. We also know that s ends at j while s' starts at i. The similar relation holds for the pairs $\xi^{d,s}_{i:j}(u)$ versus $\zeta^{d,s'}_{i:j}(v)$ and $\underline{\xi}^{d,s}_{i:j}(u)$ versus $\underline{\zeta}^{d,s'}_{i:j}(v)$.

A.5.2 Mirrored Asymmetric Inside

We group all the local potentials associated with variables in $\xi^{d,s}_{i:j}(u)$ and in the blanket $\mathfrak{I}^{d,s}_{i:j}(u)$ into a joint potential $\Phi[\hat{\xi}^{d,s}_{i:j}(u)]$, and define a quantity called *mirrored asymmetric*

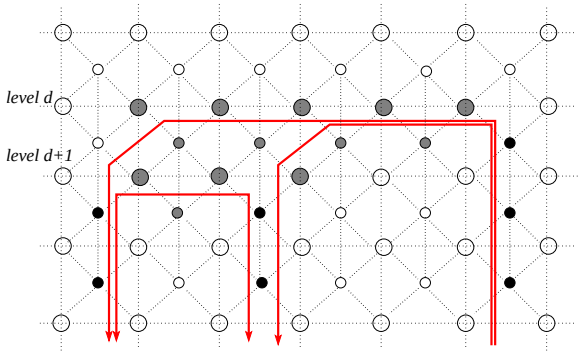


Figure A.2: Decomposition with respect to symmetric/mirrored asymmetric Markov blankets.

inside mass (or 'mirrored asymmetric inside' for short) as follows

$$\beta^{d,s}_{i:j}(u) = \sum_{\xi^{d,s}_{i:j}(u)} \Phi[\hat{\xi}^{d,s}_{i:j}(u)] \tag{A.44}$$

Analogous to the derivation in Section 8.3.2, let us examine a mirrored decomposition (see Figure A.2). Using the same argument as in Section 8.3.2 we have the following recursive relations. A mirrored version of Equation 8.37 reads

$$\Delta^{d,s}_{i:j} = \sum_{u \in S^{d+1}} \pi^{d,s}_{u,i} \beta^{d,s}_{i:j}(u) \tag{A.45}$$

A mirrored version of Equation 8.33

$$\beta^{d,s}_{i:j}(u) = \sum_{t \in [i,j-1]} \sum_{v \in S^{d+1}} \beta^{d,s}_{t+1:j}(v) \hat{\Delta}^{d+1,u}_{i:t} A^{d,s}_{u,v,t} + \hat{\Delta}^{d+1,u}_{i:j} E^{d,s}_{u,j} \tag{A.46}$$

A mirrored version of Equation 8.34

$$\beta^{d,s}_{i:i}(u) = \hat{\Delta}^{d+1,s}_{i:i} E^{d,s}_{u,i} \tag{A.47}$$

A mirrored version of Equation 8.35

$$\beta^{D-1,s}_{i:j}(u) = \sum_{v \in S^{d+1}} \beta^{D-1,s}_{i+1:j}(v) R^{D,u}_{i:i} A^{D,s}_{u,v,i} \tag{A.48}$$

The Equations A.45,A.46,A.47 and A.48 specify a *bottom-up* and *right-left* algorithm to compute the symmetric inside masses and mirrored asymmetric inside masses. Initially, at the bottom level $\Delta^{D,s}_{i:i} = 1$ for $i \in [1, T]$ and $s \in S^D$.

A.5.3 Mirrored Asymmetric Outside

Recall that in Section A.5.1 we have introduced a notion of mirrored asymmetric Markov blanket $\mathfrak{I}^{d,s}_{i:j}(u)$. Given $\mathfrak{I}^{d,s}_{i:j}(u)$, we can group all the local potentials which are defined on $\underline{\xi}^{d,s}_{i:j}(u)$ and on the blanket into a joint potential $\Phi[\hat{\underline{\xi}}^{d,s}_{i:j}(u)]$. Let's define a quantity called *mirrored asymmetric outside mass* (or 'mirrored asymmetric outside' for short) as follows

$$\mu^{d,s}_{i:j}(u) = \sum_{\underline{\xi}^{d,s}_{i:j}(u)} \Phi[\hat{\underline{\xi}}^{d,s}_{i:j}(u)] \tag{A.49}$$

The relation between the mirrored asymmetric outside and mirrored asymmetric inside is analogous to that between the asymmetric outside and asymmetric inside.

Using the same techniques used in Section 8.3.3, we have the following relations. A mirrored version of Equation 8.44 reads

$$\Lambda^{d+1,u}_{i:j} = \sum_{s\in S^d} \sum_{t\in[j+1,T]} \mu^{d,s}_{i:t}(u) \sum_{v\in S^{d+1}} \beta^{d,s}_{j+1:t}(v) A^{d+1,s}_{u,v,j} + \sum_{s\in S^d} \mu^{d,s}_{i:j}(u) E^{d,s}_{u,j} \tag{A.50}$$

for $d \in [1, D-2]$. At the bottom level, i.e. $d+1 = D$, we only have $i = j$.

A mirrored version of Equation 8.40

$$\mu^{d,s}_{i:j}(u) = \sum_{v\in S^{d+1}} \sum_{t\in[1,i-1]} \mu^{d,s}_{t:j}(v) \hat{\Delta}^{d+1,v}_{t:i-1} A^{d+1,s}_{v,u,i-1} + \hat{\Lambda}^{d,s}_{i:j} \pi^{d,s}_{u,i} \tag{A.51}$$

for $d \in [2, D-1]$.

A mirrored version of Equation 8.41

$$\mu^{D-1,s}_{i:j}(u) = \sum_{v\in S^D} \mu^{D-1,s}_{i-1:j}(v) \hat{\Delta}^{D,v}_{i-1:i-1} A^{D,s}_{v,u,i-1} + \hat{\Lambda}^{D-1,s}_{i:j} \pi^{D-1,s}_{u,i} \tag{A.52}$$

Equations A.50,A.51 and A.52 show a recursive *top-down* and *outside-in* approach to compute the symmetric outside masses and the mirrored asymmetric outside masses. We start from the top with $d = 1$ and $\Lambda^{1,s}_{1:T} = 1$ for all $s \in S^1$ and proceed downward until $d = D$.

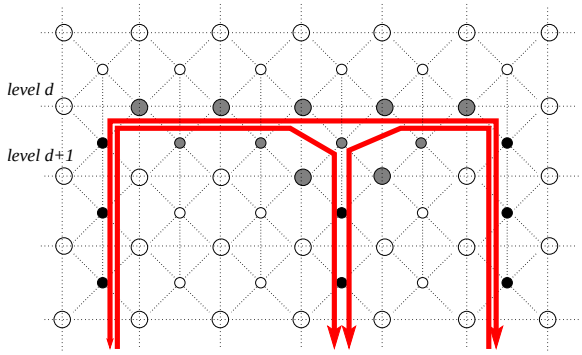


Figure A.3: The symmetric Markov blanket contains an asymmetric blanket and a mirrored asymmetric blanket.

A.5.4 Variants of Expected Sufficient Statistics

ESS for transition features:

As a mirrored version of Equation 8.67 we have

$$\mathbb{E}[\mathbf{F}^{d,s}_{\sigma^{transit},u,v}(\zeta)] = \frac{1}{Z}\sum_{t\in[1,T-1]} A^{d,s}_{u,v,t}\mathbf{f}^{d,s}_{\sigma^{transit},u,v}(t)\sum_{i\in[1,t]}\sum_{j\in[t+1,T]}\beta^{d-1,s}_{t+1:j}(v)\mu^{d-1,s}_{i:j}(u)\hat{\Delta}^{d,u}_{i:t}$$

There exists another variant that makes use of both the asymmetric inside and mirrored asymmetric inside. Given a symmetric Markov blanket $\Pi^{d-1,s}_{i:j}$, the set of variables $\zeta^{d-1,s}_{i:j}$ within the blanket can be decomposed into smaller components, which include those falling within the sub-asymmetric Markov blanket $\Gamma^{d-1,s}_{i:t}(u)$ and those within the sub-mirrored asymmetric Markov blanket $\Upsilon^{d-1,s}_{t+1:j}(v)$ (see Figure A.3). For $j > i$, there is a context $c = (e^{d-1}_t = 0, e^d_t = 1)$. Following the similar derivation as in Section 8.4, we obtain

$$\mathbb{E}[\mathbf{F}^{d,s}_{\sigma^{transit},u,v}(\zeta)] = \frac{1}{Z}\sum_{t\in[1,T-1]} A^{d,s}_{u,v,t}\mathbf{f}^{d,s}_{\sigma^{transit},u,v}(t)\sum_{i\in[1,t]}\sum_{j\in[t+1,T]}\alpha^{d-1,s}_{i:t}(u)\beta^{d-1,s}_{t+1:j}(v)\hat{\Lambda}^{d,s}_{i:j}$$

for $d \in [3, D]$. For $d = 2$, we must fix $i = 1$ and $j = T$.

Since everything here is just a mirrored version of the AIO algorithm the roles of initialisation and of ending potentials can be swapped.

ESS for initialisation features:

As a mirrored version of Equation 8.73 we have

$$\mathbb{E}[\mathbf{F}^{d,s}_{\sigma^{init},u}(\zeta)] = \frac{1}{Z}\sum_{i\in[1,T]} \pi^{d,s}_{u,i}\mathbf{f}^{d,s}_{\sigma^{init},u}(i) \sum_{j\in[i,T]} \hat{\Lambda}^{d,s}_{i:j}\beta^{d,s}_{i:j}(u) \tag{A.53}$$

ESS for ending features:

As a mirrored version of Equation 8.79 we have

$$\mathbb{E}[\mathbf{F}^{d,s}_{\sigma^{end},u}(\zeta)] = \frac{1}{Z}\sum_{i\in[1,T]}\sum_{j\in[i,T]} \mu^{d,s}_{i:j}(u)\hat{\Delta}^{d+1,u}_{i:j}E^{d,s}_{u,j}\mathbf{f}^{d,s}_{\sigma^{end},u}(j) \tag{A.54}$$

A.6 Semi-Markov CRF as a Special Case of HCRF

In this Appendix we first describe the semi-Markov CRF (SemiCRF) (Sarawagi and Cohen, 2004) in our HCRF framework and show how to convert a SemiCRF into an HCRF. Then under the light of HCRF inference we show how to modify the original SemiCRF to handle (a) partial supervision and constrained inference, and (b) numerical scaling to avoid overflow. The modifications are of interest in their own right.

A.6.1 SemiCRF as an HCRF

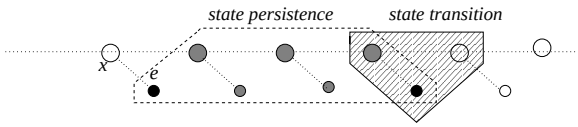


Figure A.4: The SemiCRFs in our contextual clique framework.

SemiCRF is an interesting flat segmental undirected model that generalises the chain CRF. In the SemiCRF framework the Markov process operates at the segment level, where a segment is a non-Markovian chain of nodes. A chain of segments is a Markov chain. However, since each segment can potentially have arbitrary length, inference in SemiCRFs is more involved than the chain CRFs.

Represented in our HCRF framework (Figure A.4), each node x_t of the SemiCRF is associated with an ending indicator e_t, with the following contextual cliques

- *Segmental state*, which corresponds to a single segment $s_{i:j}$ and is essentially the *state persistence* contextual clique in the context $c = (e_{i-1:j} = (1, 0, .., 0, 1))$ in the HCRF's terminology.
- *State transition*, which is similar to the state transition contextual clique in the HCRFs, corresponding to the context $c = (e_t = 1)$.

Associated with the segmental state clique is the potential $R^s_{i:j}$, and with the state transition is the potential $A_{s',s,t}$, where $s, s' \in S$, and $S = \{1, 2, ..., |S|\}$.

A SemiCRF is a three-level HCRF, where the root and bottom are dummy states. This gives a simplified way to compute the partition function, ESS, and the MAP assignment using the AIO algorithms. Thus, techniques developed in this paper for numerical scaling and partially observed data can be applied to the SemiCRF. To be more consistent with the literature of flat models such as HMMs and CRFs, we call the asymmetric inside/outside masses by the *forward/backward*, respectively. Since the model is flat, we do not need the inside and outside variables.

Forward

With some abuse of notation, let $\zeta^s_{1:j} = (x_{1:j-1}, e_{1:j-1}, x_j = s, e_j = 1)$. In other words, there is a segment of state s ending at j. We write the forward $\alpha_t(s)$ as

$$\alpha_j(s) \;=\; \sum_{\zeta^s_{1:j}} \Phi[\zeta^s_{1:j}, z] \tag{A.55}$$

As a result the partition function can be written in term of the forward as

$$\begin{aligned} Z(z) \;&=\; \sum_{\zeta_{1:T}} \Phi[\zeta_{1:T}, z] = \sum_s \sum_{\zeta^s_{1:T}} \Phi[\zeta^s_{1:T}, z] \\ &=\; \sum_s \alpha_T(s) \end{aligned} \tag{A.56}$$

We now derive a recursive relation for the forward. Assume that the segment ending at j starts somewhere at $i \in [1, j]$. Then for $i > 1$, there exists the decomposition $\zeta^s_{1:j} = (\zeta^{s'}_{1:i-1}, x_{i:j} = s, e_{i:j-1} = 0)$ for some s', which leads to the following factorisation

$$\Phi[\zeta^s_{1:j}, z] = \Phi[\zeta^{s'}_{1:i-1}] A_{s',s,i-1} R^s_{i:j} \tag{A.57}$$

The transition potential $A_{s',s,i-1}$ occurs in the context $c = (e_{i-1} = 1)$, and the segmental potential $R^s_{i:j}$ in the context $c = (x_{i:j} = s, e_{i-1} = 1, e_{i:j-1} = 0)$.

For $i = 1$, the factorisation reduces to $\Phi[\zeta^s_{1:j}, z] = R^s_{1:j}$. Since we do not know the starting i, we must consider all possible values in the interval $[1, j$. Thus, Equation A.55 can be rewritten as

$$\alpha_j(s) = \sum_{i\in[2,j]} \sum_{s'} \sum_{\zeta^{s'}_{1:i-1}} \Phi[\zeta^{s'}_{1:i-1}] A_{s',s,i-1} R^s_{i:j} + R^s_{1:j} \tag{A.58}$$

$$= \sum_{i\in[2,j]} \sum_{s'} \alpha_{i-1}(s') A_{s',s,i-1} R^s_{i:j} + R^s_{1:j} \tag{A.59}$$

Backward

The backward is the 'mirrored' version of the forward. In particular, let

$$\underline{\zeta}^s_{j:T} = (x_{j+1:T}, e_{j:T}, x_j = s, e_{j-1} = 1)$$

and we define the backward $\beta_t(s)$ as

$$\beta_j(s) = \sum_{\underline{\zeta}^s_{j:T}} \Phi[\underline{\zeta}^s_{j:T}, z] \tag{A.60}$$

Clearly, the partition function can be written in term of the backward as

$$Z(z) = \sum_s \beta_1(s) \tag{A.61}$$

The recursive relation for the backward

$$\beta_i(s) = \sum_{j\in[i,T-1]} \sum_{s'} R^s_{i:j} A_{s,s',j} \beta_{j+1}(s') + R^s_{i:T} \tag{A.62}$$

Typically, we want to limit the segment to the maximum length of $L \in [1, T]$. This limitation introduces some special cases when performing recursive computation of the the forward and backward. Equation A.58 and A.62 are rewritten as follows

$$\alpha_j(s) = \sum_{i\in[j-L+1,j], i>1} \sum_{s'} \alpha_{i-1}(s') A_{s',s,i-1} R^s_{i:j} + R^s_{1:j} \tag{A.63}$$

$$\beta_i(s) = \sum_{j\in[i,i+L-1], j<T} \sum_{s'} R^s_{i:j} A_{s,s',j} \beta_{j+1}(s') + R^s_{i:T} \tag{A.64}$$

Since it is a bit clumsy to represent a SemiCRF as a three-level HCRF, we can extend the HCRF straightforwardly by allowing the bottom level states to persist. With this relaxation

we have a *nested SemiCRF model* in the sense that each segment in a Markov chain is also a Markov chain of sub-segments.

A.6.2 Partially Supervised Learning and Constrained Inference

Following the intuition in Section 9.3.1, we require that all the forward and backward quantities and the potentials $R^s_{i:j}$ used in Equations A.63 and A.64 must be *consistent* with the labels in the case of partial supervision and constrained inference.

Specifically, any quantities that are not consistent are set to zero. Let the labels be $\vartheta = (\widetilde{x}, \widetilde{e})$. Then the potential $R^s_{i:j}$ is consistent if it satisfies the following requirements:

- if there are any labeled states in the interval $[i, j]$, they must be s,
- if there is any labeled ending indicator $\widetilde{e}_{i-1}$, then $\widetilde{e}_{i-1} = 1$,
- if there is any labeled ending indicator $\widetilde{e}_k$ for some $k \in [i, j-1]$, then $\widetilde{e}_k = 0$, and
- if any ending indicator $\widetilde{e}_j$ is labeled, then $\widetilde{e}_j = 1$.

These conditions are captured by using the following identity function:

$$\mathbb{I}[R^s_{i:j}] = \delta[\widetilde{x}_{k\in[i,j]} = s]\delta[\widetilde{e}_{i-1} = 1]\delta[\widetilde{e}_{k\in[i:j-1]} = 0]\delta[\widetilde{e}_j = 1] \tag{A.65}$$

Notice how these conditions and equation resembles those in the Equation 9.10. This is because a SemiCRF is just a simplified version of an HCRF where the potential $R^s_{i:j}$ plays the role of the inside $\Delta^{2,s}_{i:j}$.

Similarly, the forward $\alpha_j(s)$ is consistent if the following conditions are satisfied:

- if there is a labeled ending indicator at j, then $\widetilde{e}_j = 1$, and
- if there is a labeled state at j, then $\widetilde{x}_j = s$.

The consistency is captured in the following identity function:

$$\mathbb{I}[\alpha_j(s)] = \delta[\widetilde{e}_j = 1]\delta[\widetilde{x}_j = s] \tag{A.66}$$

Furthermore, the backward $\beta_i(s)$ is consistent where:

- if there is a labeled ending indicator at $i-1$, then $\widetilde{e}_{i-1} = 1$, and

- if there is a labeled state at i then $\widetilde{x}_i = s$.

And again, we have the following identity function

$$\mathbb{I}[\beta_i(s)] = \delta[\widetilde{e}_{i-1} = 1]\delta[\widetilde{x}_i = s] \tag{A.67}$$

By installing the consistency identity functions in Equations A.65, A.66 and A.67 into Equations A.63 and A.64, we now arrive at

$$\alpha_j(s) = \mathbb{I}[\alpha_j(s)] \left(\sum_{i \in [j-L+1,j], i>1} \sum_{s'} \alpha_{i-1}(s') A_{s',s,i-1} \mathbb{I}[R^s_{i:j}] R^s_{i:j} + \mathbb{I}[R^s_{1:j}] R^s_{1:j} \right) \tag{A.68}$$

$$\beta_i(s) = \mathbb{I}[\beta_i(s)] \left(\sum_{j \in [i,i+L-1], j<T} \sum_{s'} \mathbb{I}[R^s_{i:j}] R^s_{i:j} A_{s,s',j} \beta_{j+1}(s') + \mathbb{I}[R^s_{i:j}] R^s_{i:T} \right) \tag{A.69}$$

A.6.3 Numerical Scaling

We have already shown that a SemiCRF is indeed a 3-level HCRF where the top and the bottom levels are dummy states, that is, the state size is one and all the potentials associated with them have a value of one. To apply the scaling method described in Section 9.2, we notice that

- $\alpha_t(s)$ plays the role of the asymmetric inside mass $\alpha^{1,1}_{1:j}(s)$
- $\beta_t(s)$ plays the role of the asymmetric outside mass $\lambda^{1,1}_{1:j}(s)$

What we do not have here is the explicit notion of inside mass $\Delta^{2,s}_{i:j}$, but it can be considered as having a value of one. So to apply the scaling algorithm in Figure 9.1 we may scale the state-persistence potential $R^s_{i:j}$ instead. The simplified version of Figure 9.1 is given in Figure A.5.

Of course, the partial scaling step can be the source of numerical overflow with $\prod_{k=i}^{j-1} \kappa_k$. The trick here is to realise that $b/\prod_k a_k = \exp(\log b - \sum_k \log a_k)$ so that we never compute $b/\prod_k a_k$ directly but the equivalence $\exp(\log b - \sum_k \log a_k)$.

Input: T, the transition potentials and the state-persistence potentials.
Output: Scaled quantities: state-persistence potentials, forward/backward.

For $j = 1, 2, .., T$
    */*Partial scaling*/*
    For $i = j - L + 1, .., j - 1$
        Rescale $R^s_{i:j-1} \leftarrow R^s_{i:j-1} / \prod_{k=i}^{j-1} \kappa_k$
    EndFor
    Compute $\alpha_j(s)$ using Equation A.55
    Compute $\kappa_j = \sum_s \alpha_j(s)$
    */*Full scaling*/*
    Rescale $\alpha_j(s) \leftarrow \alpha_j(s)/\kappa_j$
    For $i = j - L + 1, .., j$
        Rescale $R^s_{i:j} \leftarrow R^s_{i:j}/\kappa_j$
    EndFor
EndFor
Compute true log-partition function using Equation 9.5.
Compute the backward/ESSes using the scaled potentials.

Figure A.5: Scaling SemiCRF.