Structure Learning in Sequential Data

by

Liam Stewart

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

Structure Learning in Sequential Data

Liam Stewart

Master of Science

Graduate Department of Computer Science

University of Toronto

2005

The goal of discriminative sequence learning is to learn how to classify items that can be arranged in a sequence. Many models have been proposed including logistic regression, the maximum entropy Markov model, the conditional random field, the input output Markov model, the hidden random field, and template models based on restricted Boltzmann machines. These models differ along several dimensions: whether they can be represented by a directed graphical model or an undirected one, whether or not they are chain structured, whether or not they are fully observed models, and whether or not they can incorporate knowledge about larger scale label structures. In this work, we compare these models on several synthetic problems and on a larger information extraction task.

Acknowledgements

I would first like to first thank my supervisor, Rich Zemel, for his support and assistance during my time at the University of Toronto. His knowledge, helpful discussions, patience, and comments have been invaluable.

I would also like to thank Sam Roweis, my second reader, for his comments and suggestions regarding the manuscript. I would like to thank both Xuming He and Simon Osindero for insightful discussions and help regarding the RBM-CRF model and contrastive divergence.

I would like to thank Greg Wilson for his motivation and for providing me with interesting orthogonal activities. I would like to thank Patricia Steckley and Tina Torii for helping me to focus my mind. I would also like to thank my friends in the AI group, Ben, David, Horst, Jenn, Roland, and Ted, for helping to make the lab an enjoyable place to work. I would like to thank all other members of the machine learning group who have contributed to the friendly and instructive atmosphere.

Finally, I would like to thank my mother Katrina and my fiancée Anna for their endless support and encouragement. None of this would have been possible without your help.

Research described herein has been supported by funding from the Natural Sciences and Engineering Research Council of Canada (NSERC).

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Sequential data is found in a wide variety of domains including computational biology and natural language processing, arising in tasks such as predicting protein-protein interactions, analyzing the semantics of documents, and recognizing speech. Systems designed to solve such tasks often involve multiple stages, each of which must function well as even small errors can adversely affect the overall performance. One common step is to affix descriptive labels to data in order to aid higher level processing. For example, successfully predicting protein-protein interactions requires knowledge of the secondary structures of the proteins and semantic analysis might involve annotating tokens with parts of speech tags.

The goal of sequence labeling is to classify all items in a sequence. Most classification methods make the assumption that the data are independent and identically distributed. This assumption does not necessarily hold for items that can be arranged in a sequence because there are often interactions between the observations and the labels and between the labels. Considering the joint classification of all the items is also difficult because observations are of an indeterminate dimensionality and the number of possible classes is exponential in the length of the sequences.

Generative models like the hidden Markov model (HMM) [26] were recognized as

being inappropriate for sequence labeling by McCallum, Freitag and Pereira [19], and so recent research has focused on discriminative models. The conditional random field (CRF) model proposed by Lafferty, McCallum and Pereira [16] has become the dominant methods, achieving state of the art performance on many tasks.

As noted by Altun and Hofmann [1], two challenges facing researchers are objective functions and architectures, and most recent work has focused those areas. The most commonly used objective function is the log likelihood. Kakade, Teh and Roweis [14] proposed maximizing the marginal probabilities of each label instead of the joint probability. Altun, Hofmann and Johnson [2] and Altun and Hofmann [1] have experimented with boosting and large margin methods and have achieved limited success. Following the work of Murray and Ghahramani [22], there has been interest in applying Bayesian methods to sequential models. Qi, Szummer and Minka [25] proposed Bayesian conditional random fields and reported good results on several data sets.

Discriminative models can use observations in almost any way: observations relating to a particular label may involve any part of the observation sequence and may be non-independent. However, it is typical to only look at local structures within labels as it can be difficult to extend models to handle long term dependencies and large scale structures while keeping the complexity of the model small. Sutton and McCallum [30] proposed a model in which long term dependencies are incorporated in a data-dependent manner. However, their model does not learn large scale structures. The semi-Markov CRF proposed by Sarawagi and Cohen [27] attempts to deal with structures by explicitly modeling segmentations. While their initial results seem promising, the semi-Markov CRF may not scale well since all possible segmentations need to be considered during inference.

In this thesis, we review several standard models and explore two families of models that can capture large scale structures: chain structured latent state models and template models. In particular, we look at the application of input output hidden Markov models

[4], hidden random fields [14], and parameterized label features [9] to sequence labeling.

We compare the models using several toy problems and an information extraction task. We find that template models perform quite well on toy data, often outperforming standard models in tasks where specific kinds of large scale structure exists. Latent state models, while theoretically quite powerful, suffer from several problems that will likely limit their usefulness in practice.

Chapter 2

Background

Before we begin the discussion of models for performing sequence labeling, we need to introduce some background material. We begin this chapter with an overview of some of the notation that will be used. We then give a brief summary of graphical models, both directed and undirected, and product models, outlining methods for performing inference and for learning parameters. We discuss the parameterization of undirected models in terms of features and observation features and conclude with details of the evaluation metrics that will be used to evaluate the models.

2.1 Notation

Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ be a sequence of observations. We assume that all observations $\mathbf{x}_i$ are vectors in $\mathbb{R}^d$. Categorical observations are represented using a one-hot encoding. If an observation can take on any of $|\mathcal{X}|$ distinct values, we represent the kth value with the vector $\mathbf{x}$ where $x_k = 1$ and $x_j = 0, j \neq k$. We often write $|\mathcal{X}|$ to denote the dimensionality of the observations.

Let $Y = \{y_1, \ldots, y_T\}$ be a sequence of annotations associated with X. The y_i are assumed to be discrete and scalar, with $y_i \in \mathcal{Y}$. Let $\mathcal{D} = \{D_1, \ldots, D_N\}$, $D_j = (X_j, Y_j)$, be a collection of observations with associated labels. Each (X_j, Y_j) are assumed to be

independently and identically distributed, and each may have a different length T_i. In models with latent variables, we use the notation $\mathcal{D}_c = \{D_1, \ldots, D_N\}$, $D_j = (X_j, Y_j, H_j)$, to represent a complete set of data. The tth element of a sequence Y_j will be referred to as $y_t^{(j)}$.

2.2 Graphical Models

Graphical models represent the joint distribution of a collection of random variables $U = \{u_1, \ldots, u_T\}$ with a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of edges. The vertices represent the random variables and the structure of the graph represents conditional independencies between them. The graph can either be directed or undirected, the choice of which affect both the conditional independencies and the parameterization of the joint distribution. In diagrams of graphical models, shaded vertices represent observed variables that are conditioned upon and not modeled.

In this section, we briefly introduce directed and undirected graphical models. One general method for doing probabilistic inference on trees is described. We also provide an overview of methods for learning in fully observed and partially observed graphical models. We refer the reader to Jordan [12] for more details on probabilistic inference and learning.

2.2.1 Directed Models

Factorizations of the form

$$p(u_1, \ldots, u_T) = \prod_{i=1}^{T} p(u_i | u_{\pi_i}) \tag{2.1}$$

where u_{π_i} are the parents of node u_i can be represented by a directed graph called a Bayes net. Each variable u_i is represented by a node and directed edges are drawn from a node to its children. The basic conditional independencies in the graph are $\{u_i \perp u_{\nu_i} | u_{\pi_i}\}$ where u_{ν_i} is the set of all non-descendants of node u_i.

2.2.2 Undirected Models

Undirected graphical models, or Markov random fields (MRFs), model the joint distribution as a product of potential functions defined on subsets of the variables. Let $\mathcal{I} = \{1, \ldots, T\}$ be the set of indexes of the random variables. Define $\mathcal{C} \subseteq \mathcal{P}(\mathcal{I})$, where $\mathcal{P}(\mathcal{I})$ is the power set of $\mathcal{I}$, to be the set of all indexes of cliques of nodes. The joint distribution is given by

$$p(u_1, \ldots, u_T) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \phi_c(u_c) \tag{2.2}$$

where ϕ_c are arbitrary non-negative potential functions, u_c are the variables indexed by set c, and $Z = \sum_u \prod_c \phi_c(u_c)$ is the normalization constant. A graphical representation is obtained by, for all $c \in \mathcal{C}$, linking all pairs of nodes that are indexed by c.

A random variable u_i is said to be conditionally independent of all other random variables given its neighbours. In general, the set of conditional independencies implied by an undirected graphical model is different from the set implied by a directed model. However, directed and undirected models express the same set of conditional independencies for tree structures in which a node has at most one parent.

One must explicitly compute the normalization constant in order to obtain probabilities. For conditional models, there is a normalization constant per observation sequence as the constant depends on the observations. The global normalization does have some advantages. Rather than being hidden by a local normalization, the contribution of each potential is weighed directly against the contribution of all other potentials. This effect helps undirected models overcome the *label-bias* problem (see Section 4.1.2).

2.2.3 Inference

Probabilistic inference is the process of computing probability distributions of the form $p(y_F|y_E)$ where y_F and y_E are disjoint sets of random variables. The set y_E is the evidence: it contains observed values of the random variables indexed by E and can be empty. In

most cases, we are interested in one of several possibilities: the marginal distribution of each node, the pairwise marginals, or the probability of a joint configuration. In some cases, we might also want to find a joint configuration which has maximal probability.

In order to simplify the discussion, we restrict the discussion to undirected graphical models. The directed models that we consider can be converted to undirected models by dropping the directions on the edges and defining the potentials to be the conditional distributions in the factorization implied by the Bayes net. Evidence is incorporated by defining evidence potentials. If the observed value of a node y_i is denoted by $\bar{y}_i$, the evidence potential $\phi^E(y_i)$ is defined to be:

$$\phi^E(y_i) = \begin{cases} \phi(y_i)\delta(y_i, \bar{y}_i) & i \in E, \\ \phi(y_i) & i \notin E. \end{cases} \tag{2.3}$$

The marginal distribution of a node y_i can be computed naively by

$$\begin{aligned} p(y_i) &= \sum_{y_1} \cdots \sum_{y_{i-1}} \sum_{y_{i+1}} \cdots \sum_{y_T} p(y_1, \ldots, y_T) \\ &= \frac{1}{Z} \sum_{y_1} \cdots \sum_{y_{i-1}} \sum_{y_{i+1}} \cdots \sum_{y_T} \prod_c \phi_c(y_c). \end{aligned} \tag{2.4}$$

This summation requires $O(|\mathcal{Y}|^T)$ time. In many cases, we can exploit the structure of the graph to reduce the time complexity, using exact methods such as the junction tree algorithm [12]. We describe one method, the belief propagation (BP) algorithm[1]. BP is equivalent to the junction tree algorithm for certain classes of graphs. BP is an iterative algorithm where, at each iteration, messages are sent across edges according to a message passing schedule. The message from vertex j to vertex i at value y_i is given by:

$$m_{ji}(y_i) = \sum_{y_j} \phi^E(y_i)\phi(y_i, y_j) \prod_{k \in \mathcal{N}(j) \setminus i} m_{kj}(y_j) \tag{2.5}$$

where $\mathcal{N}(j)$ denotes the set of all neighbours of node y_j. To avoid numerical problems, the messages are normalized at each step. Writing α as the normalization constant, the

[1]BP on tree structures is also known as the sum-product algorithm; on chain structures it is often referred to as the forward-backward algorithm.

message is

$$m_{ji}(y_i) = \alpha \sum_{y_i} \phi^E(y_i)\phi(y_i, y_j) \prod_{k \in \mathcal{N}(j) \setminus i} m_{kj}(y_j). \tag{2.6}$$

The marginal distribution of a node can be computed using all incoming messages:

$$p(y_i) = \alpha_i \phi^E(y_i) \prod_{k \in \mathcal{N}(i)} m_{ki}(y_i). \tag{2.7}$$

The pair-wise marginals of two nodes connected by an edge can also be easily computed:

$$p(y_i, y_j) = \alpha_{ij} \phi(y_i, y_j) \prod_{k \in \mathcal{N}(i) \setminus j} m_{ki}(y_i) \prod_{k \in \mathcal{N}(j) \setminus i} m_{kj}(y_j). \tag{2.8}$$

If the graph is a tree, the schedule obtained by running Algorithm 2.1 can be used to compute the exact marginals of all of variables in one iteration. This process takes $O(|\mathcal{E}||\mathcal{Y}|^2)$ time. The likelihood of a joint configuration can be computed by first collecting all messages to the root and then dividing by the product of the normalization constants. If the root node is unobserved, one must sum over its possible values before doing the division. If the graph is not a tree (a polytree or a more general graph), methods like the elimination algorithm or the junction tree algorithm may be used to compute marginals exactly. In cases where exact methods are intractable, approximate marginals can be computed using techniques such as Gibbs sampling. See Jordan [12] for more details. Most structures that we consider in this work are trees so BP can be used as the inference algorithm.

The process of finding the probability of the most likely, or maximum a posteriori (MAP), configuration is called the max-product algorithm and is a slight variation on standard BP. The summation in Equation 2.6 is replaced by max:

$$m_{ji}(y_i) = \alpha \max_{y_i} \phi^E(y_i)\phi(y_i, y_j) \prod_{k \in \mathcal{N}(j) \setminus i} m_{kj}(y_j). \tag{2.9}$$

A maximizing configuration can be recovered with very little extra work if the graph is a tree or a polytree. While collecting to the root, the values that give the maximum are recorded. If the root is unobserved, the value that gives its maximum is also recorded.

```
function TreeSchedule(G)
    f ← ChooseRoot(G)                          ▷ user-defined function
    s ← ⟨⟩
    for all e ∈ N(f) do                        ▷ collect to the root
        Collect(G,f,e,s)
    end for
    for all e ∈ N(f) do                        ▷ distribute to the leaves
        Distribute(G,f,e,s)
    end for
end function

procedure Collect(G,i,j,s)
    for all k ∈ N(j) \ i do
        Collect(G,j,k,s)
    end for
    append (j, i) to s
end procedure

procedure Distribute(G,i,j,s)
    append (i, j) to s
    for all k ∈ N(j) \ i do
        Distribute(G,j,k,s)
    end for
end procedure
```

Algorithm 2.1: Generate a schedule for belief propagation on a tree

Instead of computing messages when distributing messages to the leaves, we trace the values back using the maximum at the root as the initial value. This process is often called Viterbi decoding.

2.2.4 Learning

Given a set $\mathcal{D} = \{d_1, \ldots, d_N\}$ of IID data and model parameters θ, the likelihood function $L(\theta; \mathcal{D})$ is defined to be

$$\begin{aligned} L(\theta; \mathcal{D}) &= p(\mathcal{D}|\theta) \\ &= \prod_{i=1}^{N} p(d_i|\theta). \end{aligned} \tag{2.10}$$

The log likelihood function is defined to be

$$\begin{aligned} \ell(\theta; \mathcal{D}) &= \log L(\theta; \mathcal{D}) \\ &= \sum_{i=1}^{N} \log p(d_i|\theta). \end{aligned} \tag{2.11}$$

Maximum likelihood estimation (MLE) of parameters seeks to find the parameters $\theta^\star$ that maximize the likelihood, or, equivalently, the log likelihood. In penalized ML or maximum a posteriori (MAP) estimation[2], $\ell(\theta; \mathcal{D}) + r(\theta)$ is maximized with respect to θ. The function $r(\theta)$ is a penalty term that biases the optimization toward simpler models in order to reduce overfitting and improve generalization. The general strategy for finding the parameters $\theta^\star$ is to write down the log posterior, take the derivatives with respect to the parameters, set them to zero, and solve.

It is usually straight forward to find the optimal parameters for fully observed models. However, for models with latent variables, parameter estimation is more difficult. Consider the log likelihood function

$$\begin{aligned} \ell(\theta; \mathcal{D}) &= \sum_{i} \log p(d_i|\theta) \\ &= \sum_{i} \log \sum_{H} p(d_i, H|\theta). \end{aligned} \tag{2.12}$$

The parameters of the model are coupled since we have a sum inside of a log. There are two general strategies for finding the parameters. The first is to maximize $\ell(\theta; \mathcal{D})$ directly using numerical optimization techniques such as gradient descent. The second is to use the expectation-maximization (EM) algorithm proposed by Dempster, Laird and Rubin [6]. The EM algorithm is an iterative algorithm that alternates between two steps: the E step, in which the missing data is "estimated", and the M step, in which the parameters are updated given the new data. Neal and Hinton [23] show that the E step constructs a lower bound to the log likelihood and the M step optimizes the parameters to saturate the bound, thus increasing the log likelihood on each iteration.

[2]Not to be confused with MAP configurations. The context will often make it clear which one is meant.

$\theta_0 \leftarrow$ random
$l_0 \leftarrow \ell(\theta_0; \mathcal{D})$
$i \leftarrow 0$
repeat
  compute $p(h|d_j, \theta_i) \forall j$ ▷ E step
  $\theta_{i+1} \leftarrow \arg\max_\theta \ell_q(\theta; \mathcal{D})$ ▷ M step
  $l_{i+1} \leftarrow \ell(\theta_{i+1}; \mathcal{D})$
  $i \leftarrow i + 1$
until change in $l <$ tolerance

Algorithm 2.2: The expectation-maximization (EM) algorithm

If the values of the hidden variables were known, we could construct the complete log likelihood $\ell_c(\theta; \mathcal{D}_c)$. However, since the H_i are unknown, we can take the expectation of $\ell_c(\theta; \mathcal{D}_c)$ with respect to some distribution $q(H)$ to obtain the expected complete log likelihood:

$$\ell_q(\theta; \mathcal{D}) = \sum_i \sum_H q(H) \log p(Y_i, H|\theta). \tag{2.13}$$

It can be shown that the best choice for $q(H)$ is $p(H|d_i, \theta)$ [23]. The EM algorithm is summarized in Algorithm 2.2.

2.3 Product Models

A product of experts (POE) model [10] combines multiple "expert" distributions in a multiplicative fashion to form a new distribution:

$$p(Y|\theta) \propto \prod_m p_m(Y|\theta_m) \tag{2.14}$$

where m indexes the experts. Each expert can learn specific parts of a distribution. By combining the experts by first multiplying and then re-normalizing, the resulting distribution can be more peaked than the individual experts. Product models can be quite powerful when experts contain latent variables. One of the most basic types of product models with latent variables in the restricted Boltzmann machine (RBM) [29, 7] in which an expert in a single latent random variable with connections only to the visible

variables. It should be noted that individual experts do not need to be normalized since the product needs to be renormalized anyways.

2.3.1 Inference

The joint probability 2.14 can be computed up to a constant since the normalization factor can be quite difficult to evaluate. One useful property of POE models is that the observed variables are conditionally independent given the values the latent variables of the experts. Thus a block Gibbs sampler can be implemented and can be used to efficiently compute approximate marginals.

2.3.2 Learning

Consider the derivative of the log of Equation 2.14 with respect to a parameter θ_m of the mth expert:

$$\begin{aligned}
\frac{\partial \log p(Y|\theta)}{\partial \theta_m} &= \frac{\partial \log p_m(Y|\theta_m)}{\partial \theta_m} - \frac{1}{\sum_{Y'} \prod_m p_m(Y'|\theta_m)} \sum_{Y'} \frac{\partial \prod_m p_m(Y'|\theta_m)}{\partial \theta_m} \\
&= \frac{\partial \log p_m(Y|\theta_m)}{\partial \theta_m} - \frac{1}{\sum_{Y'} \prod_m p_m(Y'|\theta_m)} \sum_{Y'} \frac{\partial \exp(\sum_m \log p_m(Y'|\theta_m))}{\partial \theta_m} \\
&= \frac{\partial \log p_m(Y|\theta_m)}{\partial \theta_m} - \frac{1}{\sum_{Y'} \prod_m p_m(Y'|\theta_m)} \sum_{Y'} \prod_m \log p_m(Y'|\theta_m) \frac{\partial \log p_m(Y'|\theta_m)}{\partial \theta_m} \\
&= \frac{\partial \log p_m(Y|\theta_m)}{\partial \theta_m} - \sum_{Y'} p(Y'|\theta) \frac{\partial \log p_m(Y'|\theta_m)}{\partial \theta_m}. \qquad (2.15)
\end{aligned}$$

The last term in Equation 2.15 is the derivative of the log partition function. It is simply the expectation of the derivative of the log probability of the mth expert with respect to the model distribution. While it can be approximated by Markov chain Monte Carlo (MCMC) methods, the chains often take a very long time to reach equilibrium and the samples tend to be very noisy [11].

Maximizing the log likelihood of the observed data is equivalent to minimizing the Kullback-Leibler (KL) divergence $Q_0 \| Q_\theta$ between the data distribution Q_0 and the model,

Require: $k \geq 1$
  $d^\star \leftarrow$ sample from Q_0
  $i \leftarrow 0$
  for $i \leftarrow 1, k$ **do**
    compute $p_m(h_m|d^\star, \theta) \forall m$
    $h_m^\star \leftarrow$ sample from $p_m(h_m|d^\star, \theta) \forall m$
    compute $p_m(d|h_m^\star, \theta) \forall m$
    $d^\star \leftarrow$ sample from $p(d|h_1^\star, \ldots, h_M^\star) = \frac{\prod_m p_m(d|h_m^\star)}{\sum_d \prod_m p_m(d|h_m^\star)}$
  end for

Algorithm 2.3: Sample from Q_k

or equilibrium, distribution Q_θ. The derivative of $Q_0 \| Q_\theta$ with respect to θ_m is

$$\frac{\partial Q_0 \| Q_\theta}{\partial \theta_m} = \left\langle \frac{\partial \log p_m(d|\theta_m)}{\partial \theta_m} \right\rangle_{Q_0} - \left\langle \frac{\partial \log p_m(d|\theta_m)}{\partial \theta_m} \right\rangle_{Q_\theta}. \tag{2.16}$$

To avoid taking samples from Q_θ, the contrastive divergence (CD) procedure introduced by Hinton [11] minimizes the difference between $Q_0 \| Q_\theta$ and $Q_k \| Q_\theta$, where Q_k is the distribution obtained by running an MCMC chain for k steps. The negative gradient of the CD cost function is complicated as the derivate of Q_k depends on the parameters. However, based on extensive simulations, this dependence can be ignored [11]. The approximate negative gradient is:

$$-\frac{\partial}{\partial \theta_m} \left(Q_0 \| Q_\theta - Q_k \| Q_\theta \right) \approx \left\langle \frac{\partial \log p_m(Y|\theta_m)}{\partial \theta_m} \right\rangle_{Q_0} - \left\langle \frac{\partial \log p_m(Y|\theta_m)}{\partial \theta_m} \right\rangle_{Q_k}. \tag{2.17}$$

The CD learning rule is therefore

$$\Delta \theta_m \propto \left\langle \frac{\partial \log p_m(Y|\theta_m)}{\partial \theta_m} \right\rangle_{Q_0} - \left\langle \frac{\partial \log p_m(Y|\theta_m)}{\partial \theta_m} \right\rangle_{Q_k}. \tag{2.18}$$

A method for sampling from Q_k is given by Algorithm 2.3. The contrastive divergence learning algorithm is summarized in Algorithm 2.4.

There are several issues associated with the CD learning algorithm. First, one needs to set parameters such as the learning rate (η), the momentum (ρ), and the weight decay rate (κ) by hand. Second, identifying when CD has converged is difficult given that the CD cost function is difficult to compute. We train product models for a fixed number of iterations.

Require: $k \geq 1, n \geq 1$
  $\theta \leftarrow$ random
  for all m **do**
    $\Delta\theta_m \leftarrow 0$
  end for
  for $i \leftarrow 1, n$ **do**
    for all $d \in \mathcal{D}$ **do**
      $d_k \leftarrow k$ step reconstruction starting with data d
    end for
    for all m **do**
      $\Delta\theta_m \leftarrow \eta \left[\frac{1}{N} \sum_d \frac{\partial \log p_m(d|\theta)}{\partial \theta_m} - \frac{1}{N} \sum_d \frac{\partial \log p_m(d_k|\theta)}{\partial \theta_m} - \kappa\theta_m \right] + \rho\Delta\theta_m$
      $\theta_m \leftarrow \theta_m + \Delta\theta_m$
    end for
  end for

Algorithm 2.4: The contrastive divergence learning algorithm

2.4 Feature-Based Representations

In this work, the potential functions in undirected models are based on features. That is,

$$\phi(y_i, \mathbf{x}) = \exp\{\sum_k \theta_k f_k(y_i, \mathbf{x})\} \tag{2.19}$$

$$\phi(y_i, y_j, \mathbf{x}) = \exp\{\sum_l \theta_l f_l(y_i, y_j, \mathbf{x})\} \tag{2.20}$$

where θ_k is the weight for feature f_k. The features f are defined to be

$$f_i(y, \mathbf{x}) = [\![y = i]\!] g(\mathbf{x}) \tag{2.21}$$

$$f_{i,j}(y, y', \mathbf{x}) = [\![y = i]\!][\![y' = j]\!] g(\mathbf{x}) \tag{2.22}$$

where $[\![\cdot]\!]$ is an indicator function that is one only when the given condition is true. $g(\mathbf{x})$ is an observation feature, a scalar function of the observations only. There are many different types of observations features, including the features that pick out specific dimensions, regular expressions features, and dictionary features. For example, if $\mathbf{x}$ is numerical data, one observation feature could be $g_d(\mathbf{x}) = x_d$. If $\mathbf{x}$ is a single token of text, one might want a feature that is 1 when $\mathbf{x}$ is in a list of place names. In general, observation features may be overlapping and non-independent, and may be complex

functions that involve multiple parts of the observations. In this work, we assume that that the raw observations have been pre-processed by a static set of observation features to form the observations $\mathbf{x}$.

In most applications of CRFs, the potential functions are fully parameterized with respect to the number of values the variables can take on and the number of observation features. For example, supposing that there are $|\mathcal{X}|$ observation features, an edge potential linking two variables that can each take on $|\mathcal{Y}|$ distinct values will have $|\mathcal{Y}|^2|\mathcal{X}|$ features. Since the features f are non-zero only for a specific configuration of the the variables y, we can collect the features that are active for a configuration and re-write the potential functions:

$$\phi(y_i = v, \mathbf{x}) = \exp\{\boldsymbol{\theta}_v \cdot \mathbf{x}\} \tag{2.23}$$

$$\phi(y_i = v, y_j = v', \mathbf{x}) = \exp\{\boldsymbol{\theta}_{v,v'} \cdot \mathbf{x}\}. \tag{2.24}$$

2.5 Evaluation Metrics

When considering the performance of a model, one must consider how well it does with respect to predicting each label. With respect to a specific label l, let A be the number of true positives, B be the number of false negatives, C be the number of false positives, and D be the number of true negatives. The per-label accuracy with respect to l is defined as

$$labelAccuracy = \frac{A + D}{A + B + C + D} \tag{2.25}$$

The number of tokens having label l is often small compared with the number of tokens with other labels, and in our experience, models typically do quite well at predicting negatives. That is, D is often quite high. Therefore, comparing performance based on label accuracy can be difficult since the results can be quantitatively very close. Because of this, we do not use per-label accuracy as a performance metric.

The information retrieval community makes use of several other metrics: precision, recall, and the F1 score. Precision is the fraction of tokens identified as l that really are l.

$$P = \frac{A}{A + C}. \tag{2.26}$$

Recall of a method is the number of true positives divided by the total number of positive examples:

$$R = \frac{A}{A + B}. \tag{2.27}$$

The F1 score, or F measure, is defined to be the harmonic mean of the precision and the recall:

$$\begin{aligned} F1 &= \frac{2PR}{P + R} \\ &= \frac{2A}{2A + B + C}. \end{aligned} \tag{2.28}$$

F1 is 1 when $B = C = 0$. It essentially measures the number of true positives compared to the number of true positives plus mistakes, ignoring the number of true negatives.

The most common way of evaluating the performance of a sequence structure learning method is per-sequence accuracy, which is defined to be the number of correctly labeled tokens divided by the total number of tokens in a sequence:

$$accuracy(Y, Y') = \frac{\sum_{t=1}^{T} [\![y_t = y'_t]\!]}{T}. \tag{2.29}$$

When looking a sets of sequences, we can average the sequence accuracies and average F1 scores. Another method is to look at the percentage of sequences classified correctly. That is, the percentage of sequences whose accuracy is 1. This is a very coarse metric as it gives no indication about the distribution of the accuracies over all of the sequences.

In the remainder of the work, we use the F1 score to compare the model performance with respect to each label. We compare overall performance using average F1 scores, per-sequence accuracies, and whole instance accuracies.

Chapter 3

Structure Learning

The goal of sequence labeling is to infer a sequence of labels $Y = \{y_1, \ldots, y_T\}$ given a sequence of observations $X = \{x_1, \ldots, x_T\}$. One method of labeling a sequence of observations is to use a model, like the hidden Markov model (HMM), that assumes the existence of a probabilistic finite state machine (FSM) which generates the data using state specific observation distributions [26]. In most cases, the FSM is unobserved. In sequence labeling tasks, however, the FSM is observed and the states are defined to be the labels.

McCallum, Freitag and Pereira realized that generative models like the HMM are not appropriate for sequence labeling tasks [19]. They provide two reasons. The first is that in order to keep learning and inference tractable, generative models assume that the observations are conditionally independent given the state of the system. However, it is often beneficial to consider richer observations that include overlapping, non-independent features. The second reason that they provide is that generative models maximize the joint probability of the observations and the labels rather than the conditional distribution $P(Y|X)$, which is what we're interested in. This is clearly inappropriate when the task is to infer a labeling given observations. To deal with these issues, they proposed the maximum entropy Markov model (MEMM), a version of the HMM that models the

conditional distribution $p(Y|X)$ directly. Since their paper, most research in structure learning has focused on discriminative models.

In this chapter, we present an overview of techniques for structure learning in the discriminative framework. We also describe some of the common modeling issues and provide some suggestions on how to deal with them. Model-specific details are given in Chapters 4 and 5.

3.1 Basic Techniques

Sequence labeling is essentially a classification task, but what sets it apart from other classification tasks is that the class labels are the labellings of entire sequences. Since sequences can be of any length, the space of labellings is infinite. For this reason, and because it is not clear how to deal with observations of different sizes, standard methods cannot be applied directly. The simplest way to overcome these problems is to assume that all observation-label pairs (x_t, y_t) are independent. Any number of well-known techniques can be applied.

One problem with this method (and all other methods that we discuss) is that the mapping from observations to labels may be ambiguous. That is, the observation $\mathbf{x}_t$ may not by itself provide enough information to accurately label y_t. One way to address it is to use augmented observation vectors $\mathbf{x}'_t = [\mathbf{x}_{t-m}; \ldots; \mathbf{x}_t; \ldots; \mathbf{x}_{t+n}]$. The additional observations add context which essentially acts as short-term memory. However, as the size of the context increases, the number of model parameters increases, and although the increase is typically only linear in the dimension of $\mathbf{x}_t$, the number of parameters can become unwieldy if $\mathbf{x}_t$ is high dimensional. This can be a serious concern if the amount of training data is limited. Choosing an appropriate context may be difficult: it may be that it is unclear what the context should be or it may be that different labels or locations in the sequences need different contexts.

(a) A joint factorization.

(b) A factorization that separates label interactions from observations.

Figure 3.1: Factor graphs illustrating two methods for modeling pair-wise compatibilities.

Another, more fundamental, problem with the independence approximation is that dependencies between adjacent labels, as well as larger scale structures present in the labels, are ignored. Knowledge of these may help resolve ambiguities. Most existing models look at pair-wise interactions between the labels at adjacent times and are therefore $|\mathcal{Y}|$-state finite automata. We refer to these types of models as fully observed temporal models.

There are two ways of incorporating pair-wise, or first-order, label interactions. The more widely used approach is to consider the pair-wise interactions as coupled with the observations. The other approach is to consider the label interactions separately from the observation-label mapping. Factor graphs of the two approaches are shown in Figure 3.1. The former essentially classifies label pairs, while the latter amounts to locally correcting estimates from an independent classifier; models typically have $O(|\mathcal{Y}|^2+\mathcal{Y}\mathcal{X})$ parameters. The number of parameters needed when using the first method is $O(\mathcal{Y}^2\mathcal{X})$. If $\mathcal{X}$ is fairly small, the increase in complexity is not great, but for larger $\mathcal{X}$ the increase may be excessive and it may not necessarily lead to better performance. A recent study by Peng and McCallum [24] showed only a minimal increase in performance when compared to observation independent label interactions.

The types of structures present in real data are not limited to first-order structures. Higher-order structures may also be present; however, these are harder to model, especially for fully observed state machines as their expressiveness is limited by the size of $\mathcal{Y}$.

There may also be input-dependent relations. For example, in entity recognition tasks, words that are similar often have the same label. Sutton and McCallum [30] proposed a model in which, based on the observations, extra pair-wise dependencies are added to a temporal model. The addition of the extra dependencies increases the expressiveness of the model at the expense of complexity. We do not consider such models in our work.

3.2 Temporal Latent State Models

One way of looking at the labeling problem is to assume that there is one system which generates observations and another system that takes these observations and annotates them. The original generative models assumed that generating system is described by the labels. A labeling for a new sequence is produced by inferring the states of the generating system. Discriminative models separate the two systems and they do not model the generating one. Fully observed temporal models assume that the labeling system is a state machine whose state space is the set of labels. While these types of models have been used with some success, they are limited because the states of the labeling system are confounded with the labels of the task being modeled. In particular, it is difficult to represent higher-order structure unless the label space is expanded to $\mathcal{Y}^k$ and flexible variable-length memory is impossible unless state space is very large. One difficulty with expanding the state space is that the number of model parameters increases exponentially. It may be that some systems can only be modeled with an infinite number of states.

The extensions of fully observed models that address these issues are temporal latent state models. The state of the labeling system is modeled using unobserved random variables $h_t \in \mathcal{H}$. A temporal latent state model with discrete state space $\mathcal{H}$ is an $|\mathcal{H}|$ state finite automaton. Although it is usually the case that the labels are assumed to be conditionally independent given the state h_t, it is also possible to model pair-wise label

constraints directly; we do not consider such models in this work as they can be difficult to train.

While basic temporal latent state models are conceptually simple, it may be difficult to choose an appropriate size of $\mathcal{H}$ of the latent space. In addition to prior domain knowledge, methods such as cross-validation can be used to help make the decisions.

3.3 Template Models

Fully observed temporal models cannot represent higher-order structures in the labels unless the label space is expanded. However, the resulting increase in model complexity is exponential as all possible labellings must be considered, and it is often the case that only a small subset of the possible higher-order structures are ever seen. One way to address this problem is to use temporal models that assume the existence of a latent FSM. Another approach is to use *label features* which encode particular patterns in subsets of label variables. Label features are parameterized using a set of weights and may match different groups of configurations depending on the settings of the weights. They differ from features typically used in chain structured models in that they are not necessarily fixed before training. Training a model with label features can involve learning the weights for the label features. The label features thus learn what label structures are present in the data. Label features were introduced by He, Zemel and Carreira-Perpiñán [9] in the context of image labeling and have been used with success in that particular domain.

The label features that we consider have the form of a restricted Boltzmann machine (RBM) [7]. Each feature consists of one binary latent random variable. When the variable is turned on, it induces a probability distribution over some subset of label variables; when it is off, no information is provided regarded the labeling. These features are replicated across entire sequences to in order to produce complete labellings.

Template models are attractive because they can efficiently represent large spaces of structures. However, there are some issues that a modeler must address, such as the kind and number of label features to use. As with temporal latent state models, cross-validation or domain knowledge may be used.

In the following two chapters, we will give detailed descriptions of several temporal models and one type of template model.

Chapter 4

Temporal Models

The most popular methods for modeling sequential data are chain structures that capture interactions between adjacent label nodes. The chains gives a sense of causality, where a label at time t depends on the label at time $t-1$. For this reason, we refer to these models as temporal models. They have natural interpretations as kinds of finite state machines, are easy to implement, and often work quite well in practice. There are two broad classes of models: fully observed models and latent state models.

In this chapter, we detail several different kinds of temporal models, starting with logistic regression. We discuss the maximum entropy Markov model and its undirected cousin, the conditional random field. For both, we consider two variations: one in which the label interactions depend on the input and one where they do not. We also discuss two latent state models, the input output hidden Markov model and the hidden random field.

4.1 Fully Observed Models

Fully observed models have no latent variables. In this section, we present three such models: logistic regression, the maximum entropy Markov model, and the conditional random field. Logistic regression treats the label variables as independent whereas the

other two link adjacent label variables together to form a chain structure.

4.1.1 Logistic Regression

Logistic regression (LR) models the conditional probability $p(y_t|x_t)$ directly using either the logistic function (for binary classification) or the softmax function (for multinomial classification). We describe the latter since most problems involve more than two labels.

The softmax model is

$$\begin{aligned} p(y_i = k|x_i) &= \frac{\exp\{\boldsymbol{\theta}_k \cdot \mathbf{x}_i + b_k\}}{\sum_{k'} \exp\{\boldsymbol{\theta}_{k'} \cdot \mathbf{x}_i + b_{k'}\}} \\ &= \frac{\exp\{\boldsymbol{\theta}'_k \cdot \mathbf{x}'_i\}}{\sum_{k'} \exp\{\boldsymbol{\theta}'_{k'} \cdot \mathbf{x}'_i\}} \end{aligned} \tag{4.1}$$

where $\boldsymbol{\theta}'_k = [\boldsymbol{\theta}_k; b_k]$ and $\mathbf{x}'_i = [\mathbf{x}_i; 1]$.

The log likelihood of a set $\mathcal{D} = \{(y_1, x_1), \ldots, (y_N, x_N)\}$ of data is

$$\begin{aligned} \ell(\Theta; \mathcal{D}) &= \sum_{i=1}^{N} \log p(y_i|x_i) \\ &= \sum_{i=1}^{N} \sum_{k=1}^{|\mathcal{Y}|} [\![y_i = k]\!] \log p(y = k|x_i) \end{aligned} \tag{4.2}$$

where the indicator functions $[\![y_i = k]\!]$ are the class labels in a one-hot encoding of class. The parameters are estimated using either MLE or MAP estimation. The most common regularizer is weight decay, which is equivalent to putting a spherical Gaussian prior on the model parameters:

$$r(\Theta) = -\frac{1}{2\sigma^2} ||\Theta||^2 \tag{4.3}$$

where Θ is a vector containing all parameter vectors $\boldsymbol{\theta}'_k$ stacked together. Several methods can be used to learn the parameters including iteratively re-weighted least squares (IRLS). Since the dimensionality of the observations can be quite large, computing and storing the inverse Hessian is impractical. We use the L-BFGS memory-limited BFGS optimizer [33] written by Zhu et al.[1] to train our models. See Jordan [12] and Hastie et al. [8] for

[1] Available from `http://www.ece.northwestern.edu/~ciyou/code/lbcode.html`.

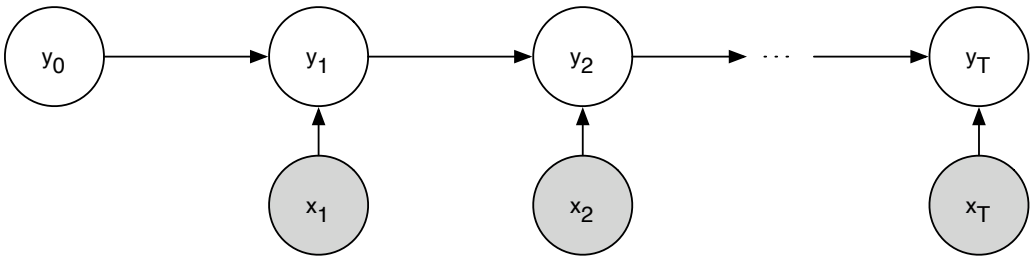


Figure 4.1: Graphical model for the MEMM.

more details on logistic regression and on how to train logistic regression models.

4.1.2 Maximum Entropy Markov Models

A maximum entropy Markov model (MEMM) [19] assumes Markovian structure of the labels variable y_t:

$$p(Y|X) = \prod_{t=1}^{T} p(y_t|y_{t-1}, \mathbf{x}_t) \tag{4.4}$$

where an additional dummy state $y_0 = start$ is used to avoid boundary conditions. The graphical model corresponding to the MEMM factorization is shown in Figure 4.1.

The standard MEMM parameterizes the compatibility, or transition, probability with a softmax model:

$$p(y_t = i|y_{t-1} = j, \mathbf{x}_t) \propto \exp\{\boldsymbol{\theta}_{ij} \cdot \mathbf{x}_t\}. \tag{4.5}$$

One way of looking at this is that there are $|\mathcal{Y}| + 1$ logistic regression classifiers, one for each $k \in \mathcal{Y} \cup \{start\}$.

One way of parameterizing the transition distribution in a MEMM model in which the transitions weights do not depend on the input is the following:

$$p(y_t = i|y_{t-1} = j, \mathbf{x}_t) \propto p(y_t = i|\mathbf{x}_t)p(y_t = i|y_{t-1} = j). \tag{4.6}$$

The distribution $p(y_t = i|\mathbf{x}_t)$ is modeled using the softmax function. For convenience, we represent the transition probabilities in an exponential form:

$$p(y_t = i|y_{t-1} = j) \propto \exp\{\theta_{ij}\}. \tag{4.7}$$

for $k \in \mathcal{Y} \cup \{start\}$ **do**
  $\mathcal{D}_k \leftarrow$ all (x_t, y_t) with $y_{t-1} = k$.
  $\Theta_k \leftarrow$ train a logistic regression classifier using $\mathcal{D}_k$.
end for

Algorithm 4.1: Train a Maximum Entropy Markov Model

Therefore,

$$p(y_t = i | y_{t-1} = j, \mathbf{x}_t) \propto \exp\{\boldsymbol{\theta}_i \cdot \mathbf{x}_t + \theta_{ij}\}. \tag{4.8}$$

We refer to this second model as the IMEMM. Dropping the initial state y_0, the complete factorization of the IMEMM is

$$p(Y|X) = p(y_1|\mathbf{x}_1) \prod_{t=2}^{T} p(y_t|y_{t-1}, \mathbf{x}_t). \tag{4.9}$$

The MEMM and the IMEMM are appropriate models when there is a Markovian relationship between the labels. However, the MEMM suffers from the "label bias" problem described by Bottou [5]. In general, low entropy transition distributions effectively ignore input, and if there is a slight imbalance in the training data with respect to the true data distribution, the model may perform poorly because of bias introduced by the skewed data.

Inference Inference in both the MEMM and the IMEMM is straight-forward since the models are trees. One can efficiently compute all of the marginals using belief propagation. Computing the probability of the most-likely label sequence and finding a sequence which achieves the maximum can be also be done efficiently by using the max-product algorithm.

Training Since a MEMM is in essence just a collection of linear regression models, training is especially simple. Algorithm 4.1 describes the algorithm used for training a MEMM. The logistic regression models typically include a regularization term like the one described in Section 4.1.1.

Training an IMEMM is a bit more difficult than training a standard MEMM since the observation and transition parameters are coupled, but numerical methods can still be used. Let

$$\eta_t(i) = \begin{cases} \boldsymbol{\theta}_i \cdot \mathbf{x}_t & \text{t} = 1, \\ \boldsymbol{\theta}_i \cdot \mathbf{x}_t + \theta_{i,y_{t-1}} & \text{t} \neq 1. \end{cases} \tag{4.10}$$

$$\mu_t(i) = \frac{\exp\{\eta_t(i)\}}{\sum_j \exp\{\eta_t(j)\}}. \tag{4.11}$$

Then

$$\nabla_{\theta_i} \eta_t(i) = \mathbf{x}_t \tag{4.12}$$

$$\frac{\partial \mu_t(i)}{\partial \eta_t(j)} = \mu_t(i)[\delta(i,j) - \mu_t(j)]. \tag{4.13}$$

The log likelihood of a set of data is

$$\begin{aligned} \ell(\theta; \mathcal{D}) &= \sum_{i=1}^{N} \log p(Y_i | X_i) \\ &= \sum_{i=1}^{N} [\log p(y_1^{(i)} | \mathbf{x}_1^{(i)}) + \sum_{t=1}^{T_i} \log p(y_t^{(i)} | \mathbf{x}_t^{(i)})]. \end{aligned} \tag{4.14}$$

The gradient of Equation 4.14 with respect to $\boldsymbol{\theta}_i$ is

$$\begin{aligned} \nabla_{\boldsymbol{\theta}_j} \ell(\theta; \mathcal{D}) &= \sum_{i=1}^{N} \sum_{t=1}^{T_i} \frac{1}{\mu_t(y_t^{(i)})} \nabla_{\theta_j} \mu_t(y_t^{(i)}) \\ &= \sum_{i=1}^{N} \sum_{t=1}^{T_i} [\delta(y_t^{(i)}, j) - \mu_t(j)] \mathbf{x}_t. \end{aligned} \tag{4.15}$$

The partial derivative of Equation 4.14 with respect to θ_{jk} is

$$\begin{aligned} \frac{\partial \log \ell(\theta; \mathcal{D})}{\partial \theta_{jk}} &= \sum_{i=1}^{N} \sum_{t=2}^{T_i} \frac{1}{\mu_t(y_t^{(i)})} \frac{\partial \mu_t(y_t^{(i)})}{\partial \theta_{jk}} \\ &= \sum_{i=1}^{N} \sum_{t=2}^{T_i} [\delta(y_t^{(i)}, j) - \mu_t(j)] \delta(y_{t-1}^{(i)}, k). \end{aligned} \tag{4.16}$$

4.1.3 Conditional Random Fields

Conditional random fields (CRFs), or more precisely linear chain CRFs, were introduced by Lafferty, McCallum, and Pereira [16]. They are undirected versions of MEMMs and

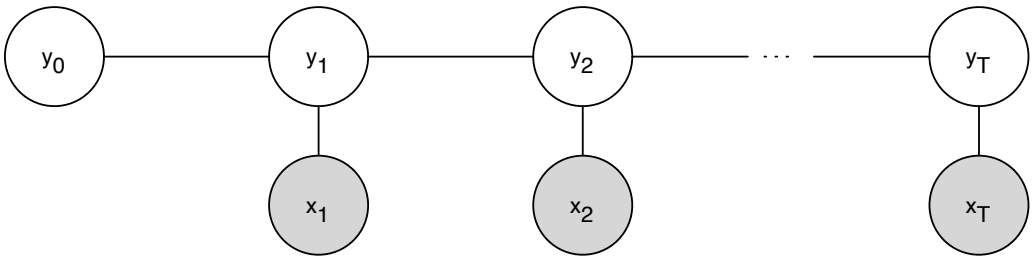


Figure 4.2: Graphical model for the CRF.

have been used extensively in structure learning tasks. Many variations on standard CRFs, including skip-chain CRFs [30], semi-Markov CRFs [27], and undirected versions of several directed models including factorial HMMs and hierarchical HMMs [31], have been proposed, but we focus on linear chain CRFs as they are the most widely used of all of the variants. Following the literature, the term CRF will refer to linear chain conditional random fields unless otherwise stated.

The linear chain CRF (Figure 4.2) makes the Markov assumptions with respect to the label variables. In particular, y_t is conditionally independent of all other labels given its neighbours y_{t-1} and y_{t+1}. Usually, only edge potentials are considered:

$$\begin{aligned} p(Y|X) &= \frac{1}{Z(X)} \prod_{t=1}^{T} \phi(y_t, y_{t-1}, \mathbf{x}_t) \\ &= \frac{1}{Z(X)} \prod_{t=1}^{T} \exp\{\boldsymbol{\theta}_{y_t, y_{t-1}} \cdot \mathbf{x}_t\}. \end{aligned} \tag{4.17}$$

We can also split the edge potentials into potentials on both the edges and the nodes. The edge potentials do not even have to depend on the observations:

$$\begin{aligned} p(Y|X) &= \frac{1}{Z(X)} \prod_{t=1}^{T} \phi(y_t, y_{t-1}) \phi(y_t, \mathbf{x}_t) \\ &= \frac{1}{Z(X)} \prod_{t=1}^{T} \exp\{\theta_{y_t, y_{t-1}} + \boldsymbol{\theta}_{y_t} \cdot \mathbf{x}_t\}. \end{aligned} \tag{4.18}$$

This parameterization is the undirected equivalent of the IMEMM; we refer to it as the ICRF.

The CRF does not suffer from the label-bias problem like the MEMM due to the global normalization property which effectively allows the CRF to look at past labels and future labels [14, 16].

Inference Since the models are trees, all marginals can be computed using belief propagation, and there exist very fast implementations of the sum-product algorithm based on matrix multiplication [16, 32]. Computing the probability of the most likely label sequence and finding a sequence which achieves that maximum can be done with the max-product algorithm.

Training As CRFs are fully observed undirected graphical models, there exist many training methods, including GIS, IIS, FIS, gradient descent, conjugate gradient methods, and second-order methods. Based on some extensive empirical studies of methods for training maximum entropy models by Malouf [17], most CRF implementations use quasi-Newton methods such as L-BFGS [33].

Our CRF implementation also uses L-BFGS to optimize the parameters. Both the log likelihood and its gradient are required. We also incorporate a Gaussian-like penalty term $r(\Theta) = -\frac{1}{2\sigma^2}||\Theta||^2$. In this section, we show the log likelihood of the CRF and the gradient of the log likelihood with respect to a parameter vector. The log likelihood of the ICRF and its gradient are similar.

The log likelihood of the CRF is

$$\begin{aligned} \ell(\Theta; \mathcal{D}) &= \sum_{i=1}^{N} \log p(Y_i|X_i) + r(\Theta) \\ &= \sum_{i=1}^{N} \sum_{t=1}^{T_i} \boldsymbol{\theta}_{y_t^{(i)}, y_{t-1}^{(i)}} \cdot \mathbf{x}_t^{(i)} - \sum_{i=1}^{N} \log Z(X_i) + r(\Theta). \end{aligned} \tag{4.19}$$

Let $m_{i,t,j}$ be an indicator variable that is 1 if and only if $y_t^{(i)} = j$. The log likelihood can be re-written as

$$\ell(\Theta; \mathcal{D}) = \sum_{i=1}^{N} \sum_{t=1}^{T_i} \sum_{j=1}^{|\mathcal{Y}|} \sum_{k=1}^{|\mathcal{Y}|} m_{i,t,j} m_{i,t-1,k} \boldsymbol{\theta}_{jk} \cdot \mathbf{x}_t^{(i)} - \sum_{i=1}^{N} \log Z(X_i) + r(\Theta). \tag{4.20}$$

The gradient with respect to $\boldsymbol{\theta}_{jk}$ is

$$\nabla_{\boldsymbol{\theta}_{jk}} \ell(\Theta; \mathcal{D}) = \sum_{i=1}^{N} \sum_{t=1}^{T_i} m_{i,t,j} m_{i,t-1,k} \mathbf{x}_t^{(i)} - \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}_{jk}} \log Z(X_i) - \frac{\boldsymbol{\theta}_{jk}}{\sigma^2}. \tag{4.21}$$

The gradient of the log partition function with respect to $\boldsymbol{\theta}_{jk}$ is

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}_{jk}} \log Z(X_i) &= \frac{1}{Z(X_i)} \sum_{Y} \exp\left\{ \sum_{t=1}^{T_i} \boldsymbol{\theta}_{y_t^{(i)}, y_{t-1}^{(i)}} \cdot \mathbf{x}_t^{(i)} \right\} \sum_{t=1}^{T_i} m_{t,j} m_{t-1,k} \mathbf{x}_t^{(i)} \\
&= \sum_{Y} p(Y|X_i) \sum_{t=1}^{T_i} m_{t,j} m_{t-1,k} \mathbf{x}_t^{(i)} \\
&= \sum_{t=1}^{T_i} \mathbf{x}_t^{(i)} \sum_{Y} p(Y|X_i) m_{t,j} m_{t-1,k} \\
&= \sum_{t=1}^{T_i} \mathbf{x}_t^{(i)} \sum_{j'=1}^{|\mathcal{Y}|} \sum_{k'=1}^{|\mathcal{Y}|} m_{t,j} m_{t-1,k} \sum_{\{Y | y_t = j', y_{t-1} = k'\}} p(Y|X_i) \\
&= \sum_{t=1}^{T_i} \mathbf{x}_t^{(i)} \sum_{j'=1}^{|\mathcal{Y}|} \sum_{k'=1}^{|\mathcal{Y}|} m_{t,j} m_{t-1,k} p(y_t = j, y_{t-1} = k|X_i) \\
&= \sum_{t=1}^{T_i} \mathbf{x}_t^{(i)} p(y_t = j, y_{t-1} = k|X_i).
\end{aligned} \tag{4.22}$$

Therefore,

$$\nabla_{\boldsymbol{\theta}_{jk}} \ell(\Theta; \mathcal{D}) = \sum_{i=1}^{N} \sum_{t=1}^{T_i} [m_{i,t,j} m_{i,t-1,k} - p(y_t = j, y_{t-1} = k|X_i)] \mathbf{x}_t^{(i)} - \frac{\boldsymbol{\theta}_{jk}}{\sigma^2}. \tag{4.23}$$

4.2 Latent State Models

Chain structured latent state models are one way of capturing complex sequential structures. Like fully observed models they can be interpreted as finite automata, but whereas the states of the machine and the labels were coupled in MEMMs and CRFs, latent state model separate the two: the latent states become the states of the machine, and a distribution over labels is associated with each of them. While their expressiveness is no longer linked to the number of labels, it is restricted by the number of latent states, the choice of which imposes a burden on the modeler. Unlike fully observed models, latent

state models are not convex and have local optima, which makes training more complex as multiple restarts typically have to be done in order to find a reasonable solution. In this section, we review two latent state models: the input output hidden Markov model (IOHMM) and the hidden random field (HRF).

4.2.1 Input Output Hidden Markov Models

The IOHMM (Figure 4.3) was originally proposed by Bengio and Frasconi as an extension of the HMM to include both inputs and outputs [4]. The model has mainly been used in speech and gesture recognition, financial time series prediction, and sequence classification [3].

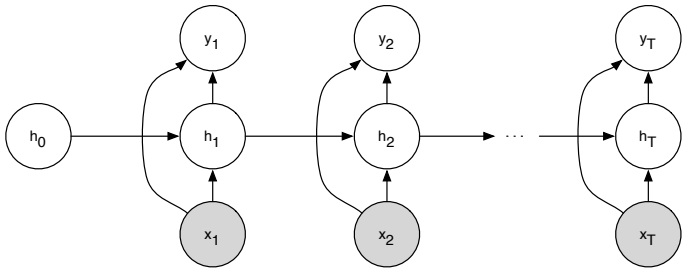


Figure 4.3: Graphical model for the IOHMM.

In addition to the observations X and the labels Y, there exists a set of latent variables $H = \{h_1, \ldots, h_T\}$ which are given a Markovian structure. The label variables are conditionally independent given H. The probability of a sequence under an IOHMM is

$$p(Y|X) = \sum_{H} \prod_{t=1}^{T} p(h_t|h_{t-1}, \mathbf{x}_t) p(y_t|h_t, \mathbf{x}_t). \tag{4.24}$$

The transition distributions have traditionally been modeled by feed-forward neural networks. We consider transition distributions $p(h_t|h_{t-1}, \mathbf{x}_t)$ parameterized by softmax functions:

$$p(h_t = j|h_{t-1} = k, \mathbf{x}_t) \propto \exp\{\boldsymbol{\lambda}_{jk} \cdot \mathbf{x}_t\}. \tag{4.25}$$

The emission distributions are also characterized by softmax functions:

$$p(y_t = j | h_t = k, \mathbf{x}_t) \propto \exp\{\boldsymbol{\theta}_{jk} \cdot \mathbf{x}_t\}. \tag{4.26}$$

Inference All marginals can be computed using belief propagation. Computing the probability of the most-likely label sequence and finding a sequence which achieves the maximum can be done with the max-product algorithm.

Training The EM algorithm to learn the parameters since the model contains latent variables. The log likelihood of a set of sequences $\mathcal{D} = \{(X_1, Y_1), \ldots, (X_N, Y_N)\}$ is

$$\begin{aligned}
\ell(\Theta; \mathcal{D}) &= \sum_{i=1}^{N} \log p(Y_i | X_i) \\
&= \sum_{i=1}^{N} \sum_{H_i} \log p(Y_i, H_i | X_i) \\
&= \sum_{i=1}^{N} \sum_{H_i} \sum_{t=1}^{T_i} \left[\log p(y_t^{(i)} | h_t^{(i)}, \mathbf{x}_t^{(i)}) + \log p(h_t^{(i)} | h_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) \right].
\end{aligned} \tag{4.27}$$

Let $m_{i,t,j}$ be an indicator random variable that is 1 if and only if $h_t^{(i)} = j$. The complete log likelihood for a complete data set $\mathcal{D}_c = \{(X_1, Y_1, H_1), \ldots, (X_N, Y_N, H_N)\}$ is

$$\begin{aligned}
\ell_c(\Theta; \mathcal{D}_c) &= \sum_{i=1}^{N} \log p(Y_i, H_i | X_i) \\
&= \sum_{i=1}^{N} \sum_{t=1}^{T_i} \left[\log p(y_t^{(i)} | h_t^{(i)}, \mathbf{x}_t^{(i)}) + \log p(h_t^{(i)} | h_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) \right] \\
&= \sum_{i=1}^{N} \sum_{t=1}^{T_i} \log p(y_t^{(i)} | h_t^{(i)}, \mathbf{x}_t^{(i)}) + \sum_{i=1}^{N} \sum_{t=1}^{T_i} \log p(h_t^{(i)} | h_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) \\
&= \sum_{i=1}^{N} \sum_{t=1}^{T_i} \sum_{j=1}^{|\mathcal{H}|} m_{i,t,j} \log p(y_t^{(i)} | h = j, \mathbf{x}_t^{(i)}) \\
&\quad + \sum_{i=1}^{N} \sum_{t=1}^{T_i} \sum_{j=1}^{|\mathcal{H}|} \sum_{k=1}^{|\mathcal{H}|} m_{i,t,j} m_{i,t-1,k} \log p(h = j | h' = k, \mathbf{x}_t^{(i)}).
\end{aligned} \tag{4.28}$$

Since the variables $m_{i,t,j}$ are unknown, we can take the conditional expectation of Equation 4.28 with respect to the posterior distribution $q = p(\{H_i\}_{i=1}^N | \mathcal{D})$. Using linearity of

expectation, we get

$$\begin{aligned}\mathrm{E}[\ell_c(\Theta;\mathcal{D}_c)|\mathcal{D}]_q &= \sum_{i=1}^{N}\sum_{t=1}^{T_i}\sum_{j=1}^{|\mathcal{H}|}\mathrm{E}[m_{i,t,j}|\mathcal{D}]_q \log p(y_t^{(i)}|h=j,\mathbf{x}_t^{(i)}) \\ &\quad + \sum_{i=1}^{N}\sum_{t=1}^{T_i}\sum_{j=1}^{|\mathcal{H}|}\sum_{k=1}^{|\mathcal{H}|}\mathrm{E}[m_{i,t,j}m_{i,t-1,k}|\mathcal{D}]_q \\ &\quad \cdot \log p(h=j|h'=k,\mathbf{x}_t^{(i)}) \qquad (4.29)\end{aligned}$$

where the expectations

$$\begin{aligned}E[m_{i,t,j}|\mathcal{D}]_q &= p(m_{i,t,j}=1|\mathcal{D}) \\ &= p(h_t^{(i)}=j|Y_i,X_i) \qquad (4.30) \\ E[m_{i,t,j}m_{i,t-1,k}|\mathcal{D}]_q &= p(m_{i,t,j}=1,m_{i,t-1,k}=1|\mathcal{D}) \\ &= p(h_t^{(i)}=j,h_{t-1}^{(i)}=k|Y_i,X_i) \qquad (4.31)\end{aligned}$$

can be computed during the E step of EM.

Collecting together the terms in Equation 4.29 that depend on $\theta_j = \{\boldsymbol{\theta}_{1j},\ldots,\boldsymbol{\theta}_{|\mathcal{Y}|j}\}$, we have:

$$J(\theta_j) = \sum_{i=1}^{N}\sum_{t=1}^{T_i} p(h_t^{(i)}=j|Y_i,X_i)\log p(y_t^{(i)}|h=j,\mathbf{x}_t^{(i)}). \qquad (4.32)$$

The problem of finding the optimal θ_j using MLE is a weighted logistic regression problem. Each data point $(y_t^{(i)},\mathbf{x}_t^{(i)})$ has an associated weight $p(h_t^{(i)}=j|Y_i,X_i)$ since the true latent state is not known. There are many methods to efficiently find the parameters including the weighted IRLS algorithm described by Jordan and Jacobs [13] and quasi-Newton methods.

Considering the optimization of the parameters $\lambda_k = \{\boldsymbol{\lambda}_{1k},\ldots,\boldsymbol{\lambda}_{|\mathcal{H}|k}\}$, we can collect the terms in Equation 4.29 that depend only on λ_k:

$$J(\lambda_k) = \sum_{i=1}^{N}\sum_{t=1}^{T_i}\sum_{j=1}^{|\mathcal{H}|} p(h_t^{(i)}=j,h_{t-1}^{(i)}=k|Y_i,X_i)\log p(h=j|h'=k,\mathbf{x}_t^{(i)}). \qquad (4.33)$$

Using the fact that

$$p(h_t^{(i)}=j,h_{t-1}^{(i)}=k|Y_i,X_i) = p(h_t^{(i)}=j|h_{t-1}^{(i)}=k,Y_i,X_i)p(h_{t-1}^{(i)}=k|Y_i,X_i) \qquad (4.34)$$

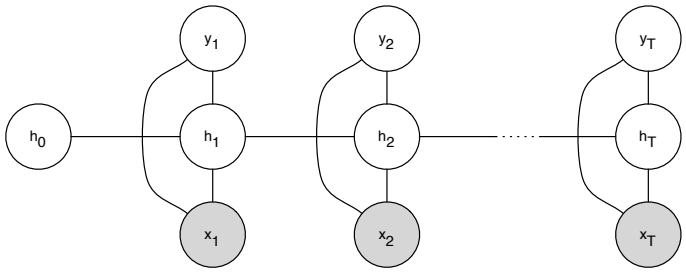


Figure 4.4: Graphical model for the HRF.

we can rewrite Equation 4.33:

$$J(\lambda_k) = \sum_{i=1}^{N}\sum_{t=1}^{T_i} p(h_{t-1}^{(i)} = k|Y_i, X_i) \cdot \sum_{j=1}^{|\mathcal{H}|} p(h_t^{(i)} = j|h_{t-1}^{(i)} = k, Y_i, X_i) \log p(h = j|h' = k, \mathbf{x}_t^{(i)}). \quad (4.35)$$

The optimization of λ_k is also a weighted logistic regression problem. The posterior probabilities $p(h_t^{(i)} = j|h_{t-1}^{(i)} = k, Y_i, X_i)$ replacing the class labels of Equation 4.2 since the true class of each data point is not known. The conditional probabilities $p(h_t^{(i)} = j|h_{t-1}^{(i)} = k, Y_i, X_i)$ can be computed easily after the E step has been run:

$$p(h_t^{(i)} = j|h_{t-1}^{(i)} = k, Y_i, X_i) = \frac{p(h_t^{(i)} = j, h_{t-1}^{(i)} = k|Y_i, X_i)}{p(h_{t-1}^{(i)} = k|Y_i, X_i)}. \quad (4.36)$$

Since the process of optimizing the parameters of logistic regression and weighted logistic regression models is iterative and can take some time to converge, the optimization procedures in the E step are often run for a fixed number of steps rather than to convergence.

4.2.2 Hidden Random Fields

The HRF (Figure 4.4) was introduced by Kakade, Teh and Roweis [14], and can be viewed as an undirected version of the IOHMM. The Markov assumption is made regarding the latent nodes and each label node is conditionally independent of all other nodes given its

associated latent node. The factorization of the joint distribution is

$$\begin{aligned} p(Y|X) &= \sum_H p(Y,H|X) \\ &= \frac{1}{Z(X)} \sum_H \prod_{t=1}^{T} \phi(h_t, h_{t-1}, \mathbf{x}_t)\phi(y_t, h_t, \mathbf{x}_t) \\ &= \frac{1}{Z(X)} \sum_H \prod_{t=1}^{T} \exp\{\boldsymbol{\lambda}_{h_t,h_{t-1}} \cdot \mathbf{x}_t + \boldsymbol{\theta}_{y_t,h_t} \cdot \mathbf{x}_t\}. \end{aligned} \tag{4.37}$$

Inference All marginals can be computed using belief propagation. The probability of the most likely label sequence and a sequence which achieves it can be found efficiently by using the max-product algorithm.

Training Since the state variables in the HRF are latent, we can use the EM algorithm to learn the parameters. The log likelihood is

$$\begin{aligned} \ell(\Theta;\mathcal{D}) &= \sum_{i=1}^{N} \log p(Y_i|X_i) \\ &= \sum_{i=1}^{N} \log \sum_{H_i} p(Y_i, H_i|X_i) \\ &= \sum_{i=1}^{N} \log \sum_{H_i} \prod_{t=1}^{T_i} \phi(h_t^{(i)}, h_{t-1}^{(i)}, \mathbf{x}_t^{(i)})\phi(y_t^{(i)}, h_t^{(i)}, \mathbf{x}_t^{(i)}) - \sum_{i=1}^{N} \log Z(X_i). \end{aligned} \tag{4.38}$$

Let $m_{i,t,j}$ be an indicator random variable that is 1 if and only if $h_t^{(i)} = j$. The complete log likelihood is

$$\begin{aligned} \ell_c(\Theta;\mathcal{D}_c) &= \sum_{i=1}^{N} \log \prod_{t=1}^{T_i} \phi(h_t^{(i)}, h_{t-1}^{(i)}, \mathbf{x}_t^{(i)})\phi(y_t^{(i)}, h_t^{(i)}, \mathbf{x}_t^{(i)}) - \sum_{i=1}^{N} \log Z(X_i) \\ &= \sum_{i=1}^{N} \sum_{t=1}^{T_i} \sum_{j=1}^{|\mathcal{H}|} \sum_{k=1}^{|\mathcal{H}|} m_{i,t,j} m_{i,t-1,k} \boldsymbol{\lambda}_{jk} \cdot \mathbf{x}_t^{(i)} \\ &\quad + \sum_{i=1}^{N} \sum_{t=1}^{T_i} \sum_{j=1}^{|\mathcal{H}|} m_{i,t,j} \boldsymbol{\theta}_{y_t^{(i)} j} \cdot \mathbf{x}_t^{(i)} - \sum_{i=1}^{N} \log Z(X_i). \end{aligned} \tag{4.39}$$

Taking the conditional expectation of Equation 4.39 with respect to the posterior distribution $q = p(\{H_i\}_{i=1}^N|\mathcal{D})$ we get

$$\begin{aligned}\ell_q(\Theta;\mathcal{D}) &= \sum_{i=1}^{N}\sum_{t=1}^{T_i}\sum_{j=1}^{|\mathcal{H}|}\sum_{k=1}^{|\mathcal{H}|}\mathrm{E}[m_{i,t,j}m_{i,t-1,k}|\mathcal{D}]_q\boldsymbol{\lambda}_{jk}\cdot\mathbf{x}_t^{(i)} \\ &\quad+\sum_{i=1}^{N}\sum_{t=1}^{T_i}\sum_{j=1}^{|\mathcal{H}|}\mathrm{E}[m_{i,t,j}|\mathcal{D}]_q\boldsymbol{\theta}_{y_t^{(i)}j}\cdot\mathbf{x}_t^{(i)}-\sum_{i=1}^{N}\log Z(X_i) \qquad (4.40)\end{aligned}$$

where

$$\mathrm{E}[m_{i,t,j}|\mathcal{D}]_q = p(h_t^{(i)}=j|Y_i,X_i) \qquad (4.41)$$

$$\mathrm{E}[m_{i,t,j}m_{i,t-1,k}|\mathcal{D}]_q = p(h_t^{(i)}=j,h_{t-1}^{(i)}=k|Y_i,X_i). \qquad (4.42)$$

These expectations are computed in the E step. The M step involves optimizing a fully-observed CRF. The following gradients need to be computed:

$$\nabla_{\boldsymbol{\lambda}jk}\ell_q(\Theta;\mathcal{D}) = \sum_{i=1}^{N}\sum_{t=1}^{T_i}[\mathrm{E}[m_{i,t,j}m_{i,t-1,k}|\mathcal{D}]_q - p(h_t^{(i)}=j,h_{t-1}^{(i)}=k)]\mathbf{x}_t^{(i)} \qquad (4.43)$$

$$\nabla_{\boldsymbol{\theta}jk}\ell_q(\Theta;\mathcal{D}) = \sum_{i=1}^{N}\sum_{t=1}^{T_i}[\delta(y_t^{(i)},j)\,\mathrm{E}[m_{i,t,k}|\mathcal{D}]_q - p(y_t^{(i)}=j,h_t^{(i)}=k)]\mathbf{x}_t^{(i)} \qquad (4.44)$$

As with the IOHMM, it can take a long time for the optimization of the parameters in the M step to converge so we usually only do a fixed number of steps.

4.3 Discussion

In this chapter, we have covered several temporal models: logistic regression, MEMMs, CRF, IOHMMs, and HRFs. Logistic regression is the most basic of the five, assuming the independence of all items in a sequence. It is a conditional model so it can make full use of features to capture some structure in the observations; however, it cannot model structure between labels. Inference is relatively inexpensive compared to the more complex models, taking roughly $O(|\mathcal{Y}||\mathcal{X}|T)$ time for an entire sequence.

Chain structured models like the MEMM and the CRF that link the label variables are the most commonly used models. In essence, they introduce soft contraints between

adjacent labels which may be either dependent on the observations or independent. The MEMM is a directed graphical model whereas the CRF is undirected. The latter is the preferred model as the global normalization allows features at different parts of the chain to compete directly against each other, thus helping to overcome some of the problems of the MEMM, namely the label-bias problem. Inference in a MEMM and a CRF takes $O(|\mathcal{Y}|^2(T + |\mathcal{X}|))$; the $|\mathcal{Y}|^2|\mathcal{X}|$ term is the time required to compute the transition matrices. Computing the likelihood of a CRF is more expensive because it requires the normalization constant to be computed.

MEMMs and CRFs are all fully observed models. In contrast, the IOHMM and the HRF are latent state models. The latent states are arranged in a chain structure; the label variables depend only on the current observations and the current latent state. The power of these models comes from the fact that labels and state are decoupled. While for some problems, it may be clear what the number of latent states should be, the choice may be more difficult for complex tasks. The IOHMM is a directed model; the HRF is its undirected equivalent. Computing marginals in IOHMM and HRF models takes $O(|\mathcal{H}|^2T + |\mathcal{H}|^2|\mathcal{X}|T + |\mathcal{Y}||\mathcal{H}||\mathcal{X}|T)$ time. Computing the likelihood takes roughly the same time for IOHMMs, but for HRFs, the normalization constant also needs to be computed. While the IOHMM and the HRF have can have more capacity than the MEMM and CRF, it may be the case that for some problems the extra time needed to choose the number of states and train the model is not worth the gain in performance when compared to a fully observed model.

Chapter 5

Template Models

Traditional CRFs make extensive use of fixed feature functions. Ignoring observations and considering groups of n labels, there is one feature for each element of

$$\underbrace{\mathcal{Y} \times \mathcal{Y} \times \cdots \times \mathcal{Y}}_{n \text{ times}}.$$

The number of features is exponential in n. For example, if the cardinality of $\mathcal{Y}$ is 10, a not unreasonable value, incorporating fourth-order features requires the addition of 10000 features and weights to the model. This is clearly wasteful given that most of the configurations will likely never be seen. To avoid overfitting, ever increasing amounts of training data are needed.

An alternative approach is to model higher-order structures by using parameterized features that detect sets of patterns. These features can be thought of as templates. In this chapter, we examine one type of template model, the RBM-CRF, which uses restricted Boltzmann machines (RBMs) [7] to represent the features. The RBM-CRF was proposed by He, Zemel, and Carreira-Perpiñán [9] in the context of multi-scale CRFs for image segmentation.

5.1 The RBM-CRF Model

A label feature h is a latent binary random variable that is connected to J label variables. Let $g = \langle n_g, \mathbf{o}_g, \{W_{g,n}\}_{n=1}^{n_g}, \mathbf{b}_g \rangle$ be a *group* that defines a set of n_g label features whose connectivity is given by the vector $\mathbf{o}_g$ of offsets; all label features in a group have the same connectivity. The $W_{g,n} = [\mathbf{w}_{g,n,1}, \ldots, \mathbf{w}_{g,n,|\mathbf{o}_g|}]$ are weight matrices where $\mathbf{w}_{g,n,j}$ is the vector of weights between label feature n and the label variable at offset j. $\mathbf{b}_g = [b_{g,1}; \ldots; b_{g,|\mathbf{o}_g|}]$ is a vector of biases.

The offset vector $\mathbf{o}_g$ does not specify the exact label variables in a sequence that the label features are connected to; rather, it is used to specify which label variables the label feature is connected to when the group is instantiated or replicated. The replication at time t is given by $g(t) = \langle n_g, \mathbf{o}_g(t), \{W_{g,n}\}_{n=1}^{n_g}, \mathbf{b}_g \rangle$ where $\mathbf{o}_g(t) = \mathbf{o}_g + t$ gives the indices of the label variables that the label features in g_t are connected to. The replications share the same weights and biases so a group can be thought of as a collection of n_g templates which are replicated across a sequence. A group can only be replicated at a time t only when all indices in $\mathbf{o}_g(t)$ specify valid label variables. When rolled out across an entire sequence, the resulting model takes the form of an RBM, a bipartite undirected graphical model. The node $h_{g,n,r}$ is the rth replication of node n of group g; it is connected to a subset of the label variables in the sequence.

The joint probability of a sequence Y with respect to group g is given by

$$
\begin{aligned}
p_g(Y, H) &\propto \exp\{\sum_r \sum_n [b_{g,n} + \sum_k \mathbf{w}_{g,n,k} \cdot \mathbf{l}_k] h_{g,n,r}\} && (5.1)\\
&= \prod_r \prod_n \exp\{[b_{g,n} + \sum_k \mathbf{w}_{g,n,k} \cdot \mathbf{l}_k] h_{g,n,r}\} && (5.2)\\
&= \prod_r \prod_n \tilde{p}_{g,n,r}(h_{g,n,r}, Y) && (5.3)
\end{aligned}
$$

where r indexes the replications, n indexes the latent variables in a replication, and k indexes the label variables that a node $h_{g,n,r}$ is connected to. The value of label y_k is represented by the vector $\mathbf{l}_k$ using a one-hot encoding. Suppose $\tilde{p}_{g,n,r}(h_{g,n,r}, Y)$ is

normalized. Then

$$p_{g,n,r}(h_{g,n,r}, Y) = \frac{\exp\{[b_{g,n} + \sum_k \mathbf{w}_{g,n,k} \cdot \mathbf{l}_k] h_{g,n,r}\}}{\sum_Y 1 + \exp\{[b_{g,n} + \sum_k \mathbf{w}_{g,n,k} \cdot \mathbf{l}_k] h_{g,n,r}\}}. \tag{5.4}$$

The probability that a label variable y_j takes on value v given $h_{g,n,r}$ is

$$p_{g,n,r}(y_j = v | h_{g,n,r}) = \frac{\exp\{[w_{g,n,k,v} \mathbf{l}_{k,v}] h_{g,n,r}\}}{\sum_{v'=1}^{|\mathcal{Y}|} \exp\{[w_{g,n,k,v'} \mathbf{l}_{k,v'}] h_{g,n,r}\}} \tag{5.5}$$

where k is the index of y_j with respect to $h_{g,n,r}$, We can view $h_{g,n,r}$ as inducing a distribution over each label variable that it is connected to. If $h_{g,n,r} = 0$, the distribution that is induced is the uniform distribution; however, if $h_{g,n,r} = 1$, the distribution that is induced depends on the weight vector $\mathbf{w}_{g,n,k}$.

The likelihood of a sequence is easy to compute up to a constant:

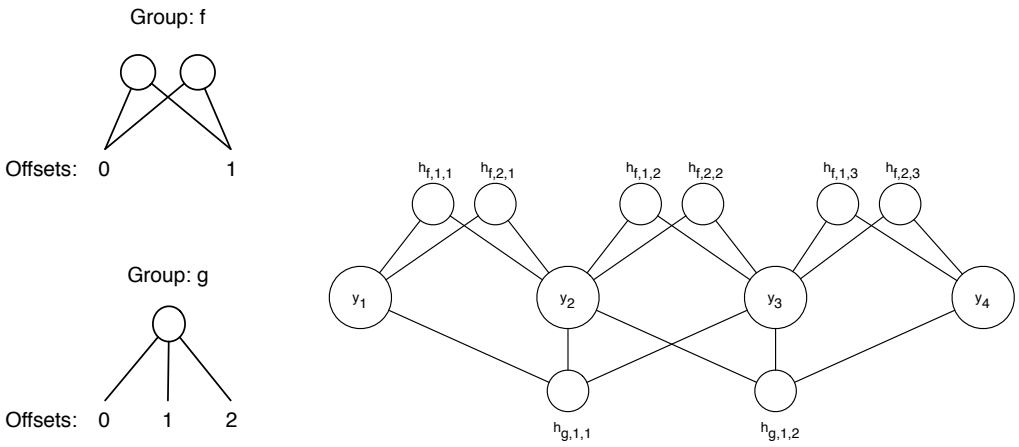
$$\begin{aligned} p_g(Y) &= \sum_H p_g(Y, H) \\ &\propto \prod_r \prod_n \sum_{h_{g,r,n}} \exp\{[b_{g,n} + \sum_k \mathbf{w}_{g,n,k} \cdot \mathbf{l}_k] h_{g,n,r}\} \\ &= \prod_r \prod_n (1 + \exp\{b_{g,n} + \sum_k \mathbf{w}_{g,n,k} \cdot \mathbf{l}_k\}). \end{aligned} \tag{5.6}$$

An RBM-CRF is a collection of K groups $G = \{g_1, \ldots, g_K\}$ which are combined multiplicatively. The resulting model is a product of experts:

$$p(Y) \propto \prod_{k=1}^{K} p_{g_k}(Y). \tag{5.7}$$

A simple RBM-CRF with two groups along with its instantiation on a sequence of length four is shown in Figure 5.1. The groups are $f = \langle n_f = 2, \mathbf{o}_f = [0; 1] \rangle$ and $g = \langle n_g = 1, \mathbf{o}_g = [0; 1; 2] \rangle$.

Inference It is very difficult to compute the exact marginals of the label features and the label variables because the graphical model is loopy. However, it is very easy to implement a Gibbs sampler to compute approximate marginals. Gibbs sampling is very



(a) The RBM-CRF. (b) The RBM-CRF rolled out.

Figure 5.1: An example of a simple RBM-CRF.

efficient because of the structure of the graph. The label variables can be sampled in parallel because they are conditionally independent given the label features. So

$$\begin{aligned} p(y_j = v|H) &= \frac{p(y_j = v, H)}{p(H)} \\ &\propto \prod_g \tilde{p}_g(y_j = v, H_g) \\ &= \sum_{y_{-j}} \prod_g \tilde{p}_g(y_{-j}, y_j = v, H_g) \\ &= \sum_{y_{-j}} \prod_g \prod_{r_g, n_g} \exp\{h_{g,n_g,r_g} \sum_k \mathbf{w}_{g,n_g,r_g,k} \cdot \mathbf{l}_k\} \\ p(y_j = v|H) &= \frac{\exp\{\sum_g \sum_{(r_g,k)=j} \sum_{n_g} h_{g,n_g,r_g} w_{g,n,k,v}\}}{\sum_{v'=1}^{|\mathcal{Y}|} \exp\{\sum_g \sum_{(r_g,k)=j} \sum_{n_g} h_{g,n_g,r_g} w_{g,n,k,v'}\}}. \end{aligned} \tag{5.8}$$

The summation $\sum_{(r_g,k)=j}$ is over replications of group g that contain y_j (which will be at offset k).

Similarly, the label features are conditionally independent given the label variables so they can also be sampled in parallel. The posterior probability $p(h_{g,n,r} = 1|Y)$ is,

dropping the g subscripts for clarity,

$$\begin{aligned} p(h_{n,r}=1|Y) &\propto \sum_{h_{-(n,r)}} p(h_{n,r}, h_{-(n,r)}, Y) \\ &= \sum_{h_{-(n,r)}} \exp\{b_n + \sum_k \mathbf{w}_{n,k}\cdot\mathbf{l}_k\} \prod_{r',n'} \exp\{b_{n'} + \sum_k \mathbf{w}_{n',k}\cdot\mathbf{l}_k\} \\ &= \exp\{b_n + \sum_k \mathbf{w}_{n,k}\cdot\mathbf{l}_k\} \prod_{r',n'} \sum_{h_{r',n'}} \exp\{b_{n'} + \sum_k \mathbf{w}_{n',k}\cdot\mathbf{l}_k\} \\ p(h_{n,r}=1|Y) &= \frac{\exp\{b_n + \sum_k \mathbf{w}_{n,k}\cdot\mathbf{l}_k\}}{1+\exp\{b_n + \sum_k \mathbf{w}_{n,k}\cdot\mathbf{l}_k\}}. \end{aligned} \tag{5.9}$$

Training Training an RBM-CRF model using standard techniques like maximum likelihood is very difficult because of the normalization constant. However, because the model is a product of experts, we can use contrastive divergence (see Section 2.3.2) to train it. Each expert in an RBM-CRF model is an RBM, and an RBM is itself a product of experts, each expert being one of the latent variables [11]. In order to calculate the approximate CD gradient we need to know the derivative of $\log \tilde{p}_{g,n,r}(Y)$ with respect to its parameters.

$$\begin{aligned} \log \tilde{p}_{g,n,r}(Y) &= \log[1+\exp\{b_{g,n} + \sum_k \mathbf{w}_{g,n,k}\cdot\mathbf{l}_k\}] \\ \nabla_{\mathbf{w}_{g,n,k}} \log \tilde{p}_{g,n,r}(Y) &= \frac{\exp\{b_{g,n} + \sum_k \mathbf{w}_{g,n,k}\cdot\mathbf{l}_k\}}{1+\exp\{b_{g,n} + \sum_k \mathbf{w}_{g,n,k}\cdot\mathbf{l}_k\}}\mathbf{l}_k \\ &= p_{g,n,r}(h_{g,n,r}=1|Y)\mathbf{l}_k \qquad (5.10) \\ \frac{\partial \log \tilde{p}_{g,n,r}(Y)}{\partial b_{g,n}} &= \frac{\exp\{b_{g,n} + \sum_k \mathbf{w}_{g,n,k}\cdot\mathbf{l}_k\}}{1+\exp\{b_{n,k} + \sum_k \mathbf{w}_{g,n,k}\cdot\mathbf{l}_k\}} \\ &= p_{g,n,r}(h_{g,n,r}=1|Y) \qquad (5.11) \end{aligned}$$

The k-step reconstructions can be produced by, at each step, first sampling $h_{g,n,r}$ according to Equation 5.9 and then sampling y_j according to Equation 5.8.

5.2 Incorporating Observations

One issue that has been tacitly ignored so far is how to incorporate observations in an RBM-CRF. There are a variety of ways of accomplishing this, and in this section, we

outline three different methods.

The most basic method is to train a local classifier first and and then incorporate it using a discount factor $0 < \gamma \leq 1$ to compensate for over-confident classifications. Writing $p_I(Y|X)$ for the distribution produced by the local classifier and $p_{RBM}(Y)$ for the distribution produced by the RBM-CRF, the final model is

$$p(Y|X) \propto p_I(Y|X)^\gamma p_{RBM}(Y). \tag{5.12}$$

This method might be good if the initial classifier is complex. A suitable value of γ must chosen, however.

The second method is to train the local classifier at the same time as the RBM-CRF model. For example, it is very easy to train a logistic regression classifier at the same time as the other experts. The model is

$$p(Y|X) \propto p_I(Y|X) p_{RBM}(Y) \tag{5.13}$$

where

$$p_I(Y|X) = \prod_{t=1}^{T} p_{LR}(y_t|\mathbf{x}_t) \tag{5.14}$$

$$p_{LR}(y_t|\mathbf{x}_t) = \frac{\exp\{\boldsymbol{\theta}_{y_t} \cdot \mathbf{x}_t\}}{\sum_{v=1}^{|\mathcal{Y}|} \exp\{\boldsymbol{\theta}_v \cdot \mathbf{x}_t\}}. \tag{5.15}$$

In this scheme, γ is effectively trained based on the magnitudes of the weights since

$$\begin{aligned} p_{LR}(y_t|\mathbf{x}_t) &\propto \exp\{\gamma \boldsymbol{\theta}_{y_t} \cdot \mathbf{x}_t\} \\ &= (\exp\{\boldsymbol{\theta}_{y_t} \cdot \mathbf{x}_t\})^\gamma. \end{aligned} \tag{5.16}$$

The gradient of the unnormalized log probability $\log \tilde{p}_{LR}(y_t|\mathbf{x}_t)$ with respect to the pa-

rameter vector $\boldsymbol{\theta}_i$ is

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}_i} \log \tilde{p}(y_t|\mathbf{x}_t) &= \sum_{v=1}^{|\mathcal{Y}|} [\![y_t = v]\!] \nabla_{\boldsymbol{\theta}_i} \log \tilde{p}(y = v|\mathbf{x}_t) \\
&= \sum_{v=1}^{|\mathcal{Y}|} [\![y_t = v]\!] \nabla_{\boldsymbol{\theta}_i} \boldsymbol{\theta}_v \cdot \mathbf{x}_t \\
&= \sum_{v=1}^{|\mathcal{Y}|} [\![y_t = v]\!] \delta(v, i) \mathbf{x}_t \\
&= l_{t,v} \mathbf{x}_t. \qquad (5.17)
\end{aligned}$$

This is the scheme that we adopt for most of the models evaluated in Chapter 6.

Rather than incorporating a simple local classifier like logistic regression, one or more latent-state temporal models, for example either IOHMMs or HRFs, can be used as experts. These models would combine the flexibility of state-based models for variable memory with the power of the RBM-CRF for recognizing larger structures. It is also possible to incorporate a CRF model, but the direct label-to-label edges complicate sampling as the label nodes can no longer be considered to be conditionally independent given the latent variables.

The third method is to incorporate observations directly into the parameterization of the label features (Figure 5.2). Such label features can then be used to map observations to label configurations and are useful for tasks that require input dependent memory. The joint distribution $p_g(Y, H)$ becomes a conditional distribution

$$p_g(Y, H|X) \propto \exp\{\sum_r \sum_n [b_{g,n} + \sum_j \boldsymbol{\theta}_{g,n,j} \cdot \mathbf{x}_j + \sum_k \mathbf{w}_{g,n,k} \cdot \mathbf{l}_k] h_{g,n,r}\}. \qquad (5.18)$$

In this approach, we basically learn a classifier with respect to the latent nodes. One drawback of this approach is that it may be difficult to choose which observations should be used for each label feature. Another issue is that the resulting model can have an prohibitive number of parameters, especially if there are many label features. An alternative setup closer to standard CRFs is to have one label feature for each dimension of the observation vector.

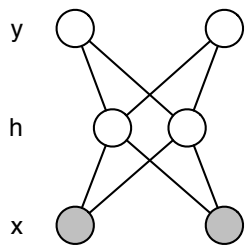


Figure 5.2: Input-dependent Label Features.

5.3 Discussion

The RBM-CRF model is a novel method of doing sequence labeling. With smaller scale label features, it can be quite similar to a standard CRF model. Ignoring observations, a feature in a stock CRF is active for a specific configuration of two label variables. An RBM-CRF with a group of $|\mathcal{Y}|$ label features that look at adjacent label variables can approximate the completely parameterized CRF. Each feature has a probability of being active based on its weights and the current configuration. A label feature's weights can be adjusted so that there is high probability that it will be active for one specific configuration and inactive for all others [7]. When it is active, it contributes a certain amount to the numerator; when it is inactive, it contributes nothing. In terms of a CRF, the contribution can be viewed as the scale factor θ_k.

The RBM-CRF is much more powerful than a CRF, especially when larger label features are considered. A label feature can model more than one structure through the probability distributions that it induces over label nodes that it is connected to. When the label feature is off, it induces a uniform distribution; when it is on, the entropy of each distribution dictates how the label node is matched. A high entropy distribution indicates that the label feature doesn't care what the label is since the probabilities for each value are roughly equal. As the entropy decreases, the label feature becomes more specific in what it is interested in through the relative probabilities of the different label values. Considering all label nodes together, the label feature can be seen as carving out

a region in the space of configurations that it matches. For this reason, it is not always necessary to have a large number of label features. It should be noted that it is always possible to include label features with fixed weights in an RBM-CRF model.

Structures of different lengths can be modeled by including parameterized features that look at different numbers of label variables. This provides another degree of freedom to the modeler – in addition to choosing arbitrary observation features, they can customize the architecture of the model. However, significant amounts domain knowledge may be needed to decide on how many experts to include and on what the connections should be. Methods such as cross-validation may also be used to decide on how many experts to include. CRFs and MEMMs can also be extended to incorporate higher-order features, but as has been noted, the resulting increase in model complexity is exponential.

We expect the RBM-CRF to work well on problems where there is ambiguity in groups of labels that can be resolved through context that is unavailable to standard models because of their local nature. For example, in the Cora references data set (see Section 6.4), there are several groups of tokens, such as title and book title, whose observations can be quite similar. The boundaries of the groups are fairly easy to identify and provide some of the context to help resolve the labeling of the interior observations. A regular CRF may have difficulty disambiguating the interior of the groups because of its local focus. An RBM-CRF with large enough label features will learn that groups of title tokens appear to together and that groups of book title tokens also appear together. Other, perhaps smaller, features, might learn that book title follows title. By putting the label features together, the RBM-CRF should be able to properly label the ambiguous observations. The RBM-CRF can be seen then as a mechanism to smooth out incorrect predictions from some more limited classifier.

An IOHMM or an HRF may also be able to resolve ambiguity by using the latent states to keep track of where it is in a sequence. The RBM-CRF does have some advantages over the HRF. For example, the RBM-CRF can be used to model fixed-length memory

easily. Consider a problem with binary labels where for a particular input, the output depends on the output from 10 steps in the past. Both the IOHMM and the HRF would require $1 + \sum_{n=1}^{10} 2^n$ states as all of the last 10 observations must be remembered. On the other hand, an RBM-CRF model would only require a couple of experts to model the memory. However, if the length of memory required is not fixed, the RBM-CRF will not be able to model the data correctly because it maintains no state information.

Chapter 6

Experimental Results

In this chapter we present experimental results on a variety of data sets, investigating issues such as ambiguous observations, larger scale structures, and memory.

Unlike most other models considered, the RBM-CRF is highly problem-dependent in that the architecture can, and often does, change for different problems. It is also possible to experiment with multiple architectures for a single problem. As such, we describe the specific architectures that we investigate for a problem in the section that details the problem. When referencing RBM-CRF architectures, we may use the same name across several problems; however, the architectures referenced by the name will be different for each problem.

At test time, labels were chosen using maximum marginals (logistic regression), approximate maximum marginals (RBM-CRF models), or Viterbi decoding (all other models). Per-label F1 scores, average F1 scores, average accuracy, and whole instance accuracy are used to compare trained models.

All models were implemented in MATLAB. Experiments were run on quad-CPU Intel Xeon (2.4 GHz) machines with 4 GB of physical memory running Red Hat Linux 7.3 and MATLAB 7 (R14).

Observation	Labels
1	A
2	B
3	C
4	Q
5	D,E

Table 6.1: Observation-label mapping for Toy Problem 1.

6.1 Toy Problem 1: Ambiguous Input

This problem illustrates the difficulty that many methods have in modeling data where a label can depend on observations outside of its immediate context. The problem was also designed to be difficult for the fully observed models by separating the ambiguity from a disambiguating label by multiple time steps.

There are five observations $\{1, 2, 3, 4, 5\}$ and six labels $\{A, B, C, D, E, Q\}$. The mapping from observations to labels, shown in Table 6.1, is mostly deterministic with the exception of an ambiguity with observation 4, which can map either to D or to E. The ambiguity can be resolved by using knowledge of the label structures present in the data: continuous Qs, ABCD, and BCE. D and E only appear after a BC so the ambiguity can be resolved when it is known whether or not an A precedes BC. Observation and label sequences are generated from the state machine illustrated in Figure 6.1. State S is the accepting state and does not generate any observations. The state transition probabilities $p(A|Q_1)$ and $p(B_2|Q_1)$ are equal.

This problem is difficult for all fully observed models because both the disambiguating input and the disambiguating label are far from the ambiguous observation. The IOHMM and the HRF should have no difficulty modeling the data as a three state FSM can model the labeling process (Figure 6.2). RBM-CRF models with larger label features should be able to learn the different structures.

6.1.1 Setup

Both the training set and the test set consisted of 100 sequences of length 50. The observations used at time t are just the raw observations (1, 2, 3, 4, or 5) in a one-hot encoding so $\mathbf{x}_t$ is 5-dimensional and $x_{t,i} = 1$ iff the observation at time t is i. Given a sufficiently large window over the observations, any of the standard methods can resolve the ambiguity present in the data; however, it may not be possible to allow large windows with real data so we kept the window small to simulate such conditions.

We trained all temporal models and three RBM-CRF models with different architectures (Table 6.2). All three had a group of label features that looked at sets of six labels and differed only in the number of features in the group: RBM(1) had one feature, RBM(2) had two, and RBM(3) had six. Each model incorporated a logistic regression expert that was trained at the same time as the label features.

For the temporal models, we used simple weight decay with $\sigma = 10$ to control capacity. This value of σ is common in CRF experiments; we did not use cross-validation to choose it. For all models except the RBM-CRF ones, training was stopped when the relative change in the log likelihood was less that 1×10^{-6}. All training runs were restarted five times with different initial parameter settings in order to get an idea of average training times, and, for the models with latent variables, to deal with problems of local optima.

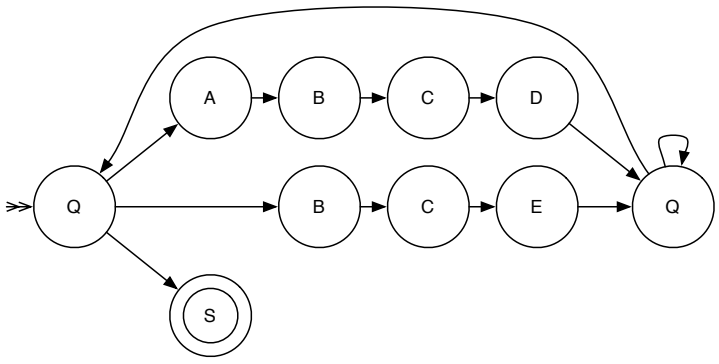


Figure 6.1: Finite state machine used to generate the data for Toy Problem 1.

Name	Observation Model	Label Features
RBM(1)	LR	$(1, 0:5)$
RBM(2)	LR	$(2, 0:5)$
RBM(3)	LR	$(6, 0:5)$

Table 6.2: Details of the RBM-CRF models used with Toy Problem 1. The notation $(n, a:b)$ is used to describe the number of label features (n) within the group and the label variables that they are connected to (at offsets $[a, b]$).

The initial parameters were chosen at random from the interval $[-1, 1]$. The model that performed best on the training data was the one used on the test data.

Because the dimensionality of the data is small, it was possible to use an implementation of IRLS to train the logistic regression classifier, although other methods like conjugate gradients or BFGS could also have been used; the results should be the same because the problem is convex. IRLS was also used as the logistic regression training algorithm in the MEMM. The L-BFGS optimizer was used to optimize the IMEMM, CRF, and the ICRF; it was also used during the M step of EM for the IOHMM and the HRF.

The number of latent states used in the IOHMM and HRF models was three. For both models, at most 100 iterations of optimization were done during the M step. The total number of EM training iterations was limited to 100.

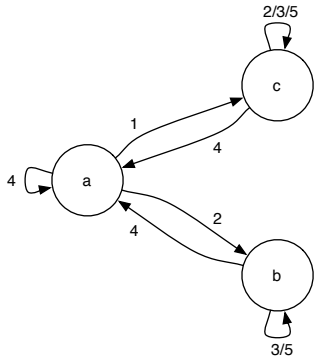


Figure 6.2: A finite state machine for Toy Problem 1.

With respect to the RBM-CRF models, the learning rate was set to 0.1, the weight decay was set to $\frac{1}{\sigma^2}$, and a momentum of 0.9 was incorporated after 25 iterations. Three reconstruction steps were done in the reconstruction phase of CD. All models were trained for 100 iterations. 50 iterations of Gibbs sampling were done to compute the approximate marginals for the RBM-CRF models at test time; none of the samples were discarded as burn-in.

Method	Mean Time (s)	Standard Deviation
LR	3.0×10^1	8.6×10^0
MEMM	1.5×10^1	1.8×10^{-1}
IMEMM	2.5×10^2	8.8×10^1
CRF	3.9×10^2	1.6×10^2
ICRF	1.2×10^2	3.9×10^1
IOHMM	2.2×10^3	2.2×10^2
HRF	3.2×10^3	4.2×10^3
RBM(1)	4.2×10^2	8.0×10^0
RBM(2)	9.5×10^2	1.0×10^2
RBM(3)	2.7×10^4	1.6×10^3

Table 6.3: Training times of the models averaged over five runs.

The mean training time of each model is shown in Table 6.3. The variance for the HRF is large because the model seems to be very susceptible to local optima. During training, it was observed that some runs were very short, only three to five iterations, and other were much longer. For both the IOHMM and the HRF, it was observed that there was typically an initial large decrease in log likelihood followed by a period of very slow convergence. The initial decrease is likely the model learning observation-label weights. With the IOHMM, a second sharp decrease can occur when the dynamics are learnt. The HRF never had a large second decrease and never reached the 100 iteration limit. These observations imply the HRF may be susceptible to local optima. We found that we could improve performance by initializing the weights of the HRF with those of an IOHMM that had been trained for a very small number of iterations. Although local optima are still a problem, we were able to train an HRF that did model the data correctly. Doing more random restarts during training could also help improve performance.

	LR	MEMM	IMEMM	CRF	ICRF	IOHMM	HRF	RBM(1)	RBM(2)	RBM(3)
A	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	99.21
B	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	99.80
C	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
D	0.00	0.00	0.00	0.00	0.00	100.00	100.00	99.19	99.60	97.23
E	66.49	66.49	66.49	66.49	66.49	100.00	100.00	99.19	99.60	96.67
Q	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	99.98
Average F1 Score	77.75	77.75	77.75	77.75	77.75	100.00	100.00	99.73	99.87	98.82
Average Accuracy	97.52	97.52	97.52	97.52	97.52	100.00	100.00	99.96	99.98	99.80
Instance Accuracy	26.00	26.00	26.00	26.00	26.00	100.00	100.00	98.00	99.00	92.00

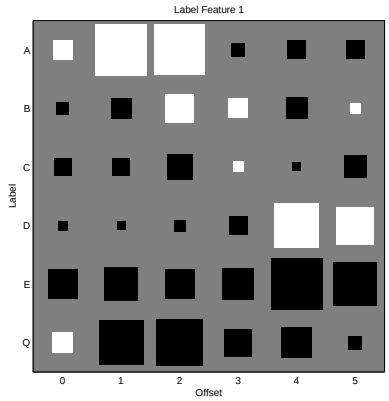
Table 6.4: Per-label F1 scores and overall performance results for the different models on test data for Toy Problem 1.

6.1.2 Results

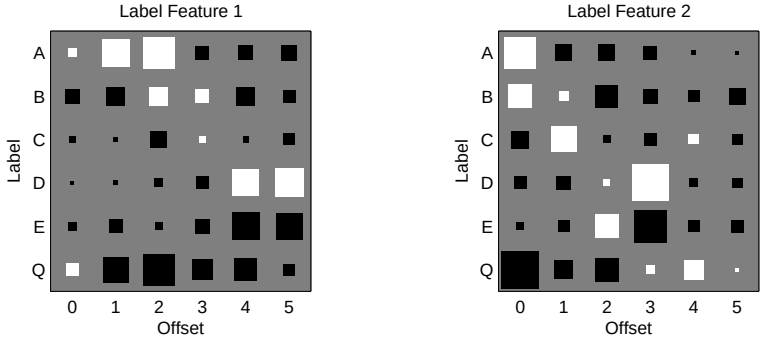
The results of running the trained models on the test data are shown in Table 6.4. All models correctly, or almost correctly, labeled all As, Bs, Cs and Qs. As expected, most were not able to distinguish between D and E. The training data contained 128 Ds and 145 Es so the best a model with limited information can do is to classify observation 4 as E. The IOHMM and the HRF were able to model the data perfectly.

All three RBM-CRF models did quite well. RBM(2) performed the best while RBM(3) did slightly worse than RBM(1) and RBM(2). RBM(3) is a complex model so the results may be due to inadequate training. The RBM-CRF models significantly outperformed the fully-observed models, and RBM(1) and RBM(2) approached the level of performance that the IOHMM achieved at a fraction of the training time.

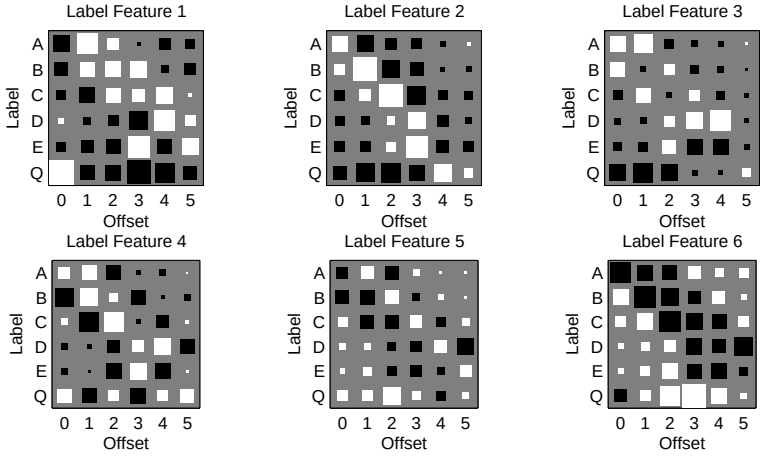
Hinton diagrams of the weights learnt by the three RBM models are shown in Figure 6.3. The columns are the offsets and the rows are the labels. Because of the slight imbalance between D and E labels in the training data, the logistic regression expert classified input 4 as an E so the label feature of RBM(1) acts as a mechanism to correct the bad classification that occurs when an A precedes the BC group. It is not known why RBM(1) has two AD correctors, although it may be that overlaying a pattern at multiple offsets may be more robust than just learning a pattern at a single offset since the label feature will be active at more than one replication for a matching pattern in the sequence.



(a) RBM(1)



(b) RBM(2)



(c) RBM(3)

Figure 6.3: Label feature weights learnt by the RBM-CRF models on data from Toy Problem 1

That is, more than one latent variable will contribute weight to a prediction. One of the label features of RBM(2) learns has weights similar those of RBM(1)'s label feature; the other matches both ABCD and BCE. RBM(3) learns several patterns including patterns that capture both structures (label features one through three) and patterns that match groups of Qs (label features five and six). It is hard to discern what label feature four learns. RBM(3) has many more label features than necessary, and it appears that the excess capacity is used to general structure. It is interesting to note that no label feature in all of the RBM-CRF models matches just the BCE structure. The reason could be that the observation-label mapping does a good job at classifying the E so there is no need to model the BCE group exclusively.

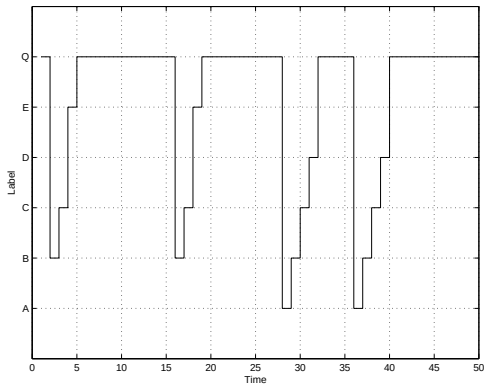
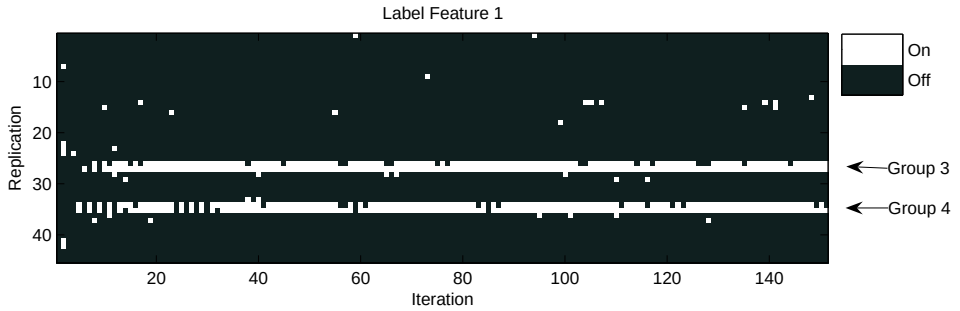
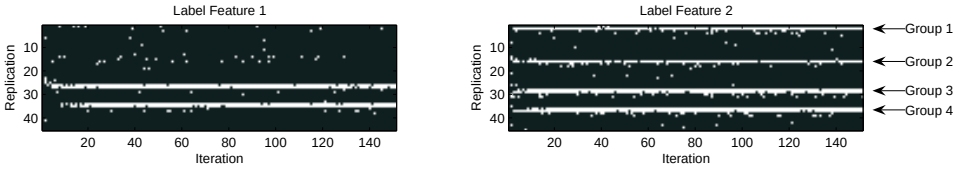


Figure 6.4: Labels of the sequence used to illustrate Gibbs sampling in the RBM-CRF models.

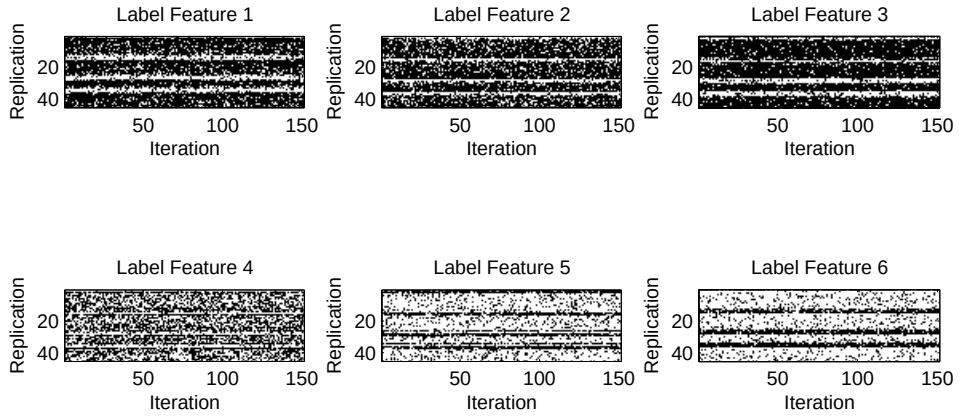
To illustrate how the RBM-CRF models work when presented with observations, we show 150 samples generated by Gibbs sampling. The labels for the sequence that we used are shown in Figure 6.4. The sequence consists of two groups of BCE (groups 1 and 2) followed by two groups of ABCD (groups 3 and 4). Samples for each label feature in each RBM-CRF model are shown in Figure 6.5. The label feature in RBM(1) is active only for groups 3 and 4, correcting the mistakes made by the logistic regression classifier. The label feature is active at two replications indicating that the ABCD pattern that it prefers it found at both offsets, each by a different instantiation of the label variable. The



(a) RBM(1)



(b) RBM(2)



(c) RBM(3)

Figure 6.5: Label feature samples produced Gibbs sampling using observations from test sequence 2 of Toy Problem 1. Within a plot, the vertical axis shows the replications of the label feature for the sequence. A white entry indicate an on sample (1) and a black entry indicates an off sample (0).

first label feature of RBM(2) is also only active for groups 3 and 4, but the second label feature is also active for the two BCE groups. The first four label features for RBM(3) are active for all four groups. The last two label features are on for everything except the structures. The samples from RBM(3) are much noisier than the samples from RBM(1) and RBM(2) since the weights for the model are not as well defined as the weights of the other two models.

6.2 Toy Problem 2: Larger Structures

To further investigate the capabilities of the RBM-CRF model, a second, more complex, toy problem was created. The motivation for this problem are tasks in which instances of a class appear in groups but where the observations for items inside the group are ambiguous. In the Cora references problem (see Section 6.4), this sort of effect is seen with some labels like Title and Book Title as well as Author, Editor, and Publisher. The observations for these fields can be very similar, but if something is known about the edges of the blocks and the structures, then the ambiguities in the middle can be resolved.

There are three labels $\{A, B, O\}$. Label sequences were generated according to the state machine shown in Figure 6.6. The a states generate A labels, the b states generate B labels, and the o states generate O labels. The s states do not generate any labels and exist only to keep the diagram clear. State $s1$ is the initial state and state $s4$ is the terminating state. When generating a sequence, the sequence length was chosen from a Poisson(30) distribution. If $s4$ was encountered before the length was reached, the generation was halted. If, when the length was reached, the system was in one of the a or b states, the length was extended and generation continued until the new length was reached or $s4$ was encountered. The length of the sequences was limited to keep training time reasonable since most of the inference algorithms take time polynomial in the length

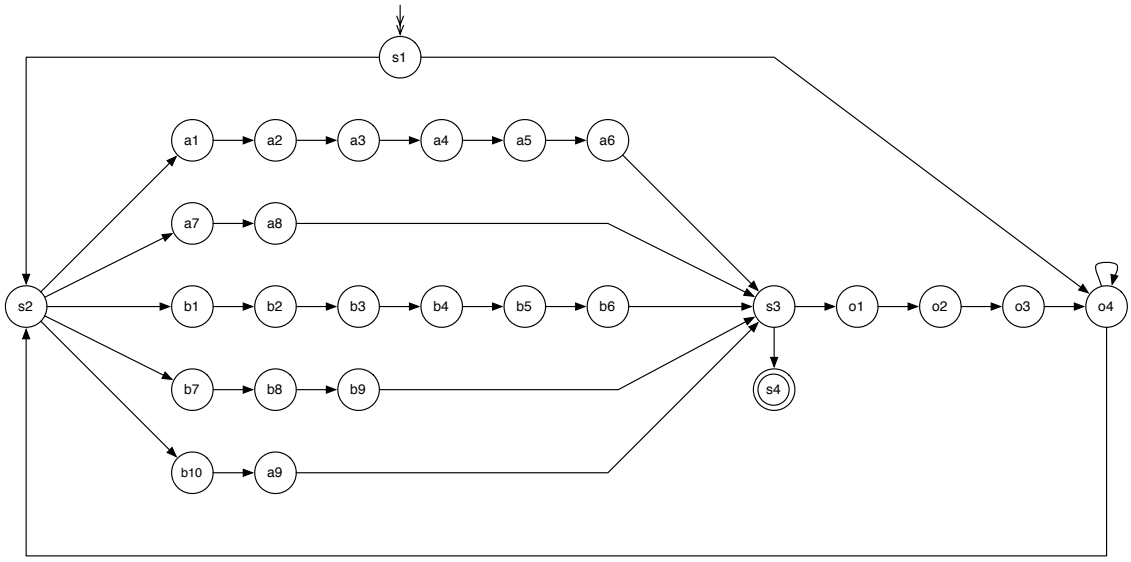


Figure 6.6: Finite state automata used to generate the data for Toy Problem 2.

of sequence.

The observations are vectors in $\mathbf{x} \in \{0,1\}^{80}$. Let x_d be the dth dimension of observation $\mathbf{x}$. All states except the s states generate observations according to the following generative model:

$$
\begin{aligned}
x_d|\alpha, p &\sim \alpha_a \text{Bernoulli}(p_{a,d}) + \alpha_b \text{Bernoulli}(p_{b,d}) + \alpha_c \text{Bernoulli}(p_{c,d}) + \alpha_o \text{Bernoulli}(p_{o,d}) \\
p_{a,d} &\sim \text{Beta}(s_{a,d}, t_{a,d}) \\
p_{b,d} &\sim \text{Beta}(s_{b,d}, t_{b,d}) \\
p_{c,d} &\sim \text{Beta}(s_{c,d}, t_{c,d}) \\
p_{o,d} &\sim \text{Beta}(s_{o,d}, t_{o,d})
\end{aligned}
$$

The $s_{i,j}$ are integers chosen at random from the interval $[1, 10]$, and the $t_{i,j}$ are integers chosen at random from the interval $[1, 100]$. The mixing proportions α depend on the state and are shown in Table 6.5.

The interior of the large blocks of a or b is meant to be fairly ambiguous, and the small patterns should provide some conflicting information about how to label observations. The observations for the o states are essentially unambiguous filler. We expect all models

States	α_a	α_b	α_c	α_o
$\{a_1, a_6, a_7, a_8, a_9\}$	1	0	0	0
$\{b_1, b_6, b_7, b_9, b_{10}\}$	0	1	0	0
$\{o_1, o_2, o_3, o_4\}$	0	0	0	1
all others	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0

Table 6.5: Mixing proportions used in the generative model for Toy Problem 2.

to do fairly well given that many observations will have an unambiguous labeling. We hypothesize that the MEMM and the CRF should perform better than logistic regression with respect to accuracy and whole instance accuracy since they can model interactions between adjacent labels; they will not, however, be perfect since it should be difficult to handle the ambiguity within the larger structures. We predict that RBM-CRF models with experts that look at larger groups of label variables should be able to improve on the performance of the other models by learning the larger structures.

6.2.1 Setup

1100 sequences were generated. The first 100 were used as the training set, and the remaining 1000 were used as the test set. We trained all temporal models along with two RBM-CRF models (Table 6.6). Because of the high dimensionality of the observations, the L-BFGS optimizer was used to train all fully-observed models. It was also used in the M step of EM for the IOHMM and the HRF. To speed up training of the CRF and the ICRF, the weights of those models were initialized using the weights of the trained MEMM and IMEMM, respectively. The parameters of all other models used were initialized using values chosen at random from the interval $[-1, 1]$. For all models except the two RBM-CRF models, training was stopped when the relative change in the log likelihood was less that 1×10^{-6}. All training runs were restarted five times and the best model with respect to the training data was chosen as the final model. All models used a weight decay regularizer with $\sigma = 10$.

The number of states in the IOHMM was chosen from two to 13 using five-fold cross-

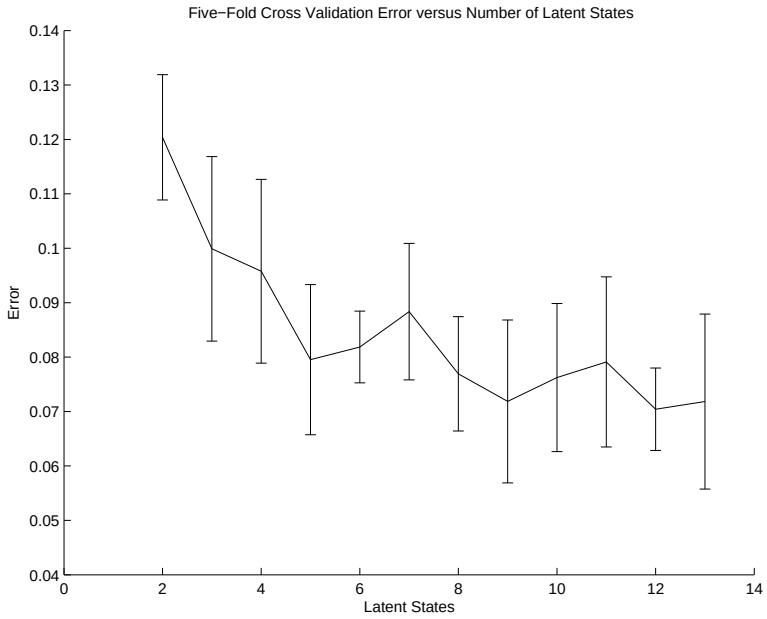


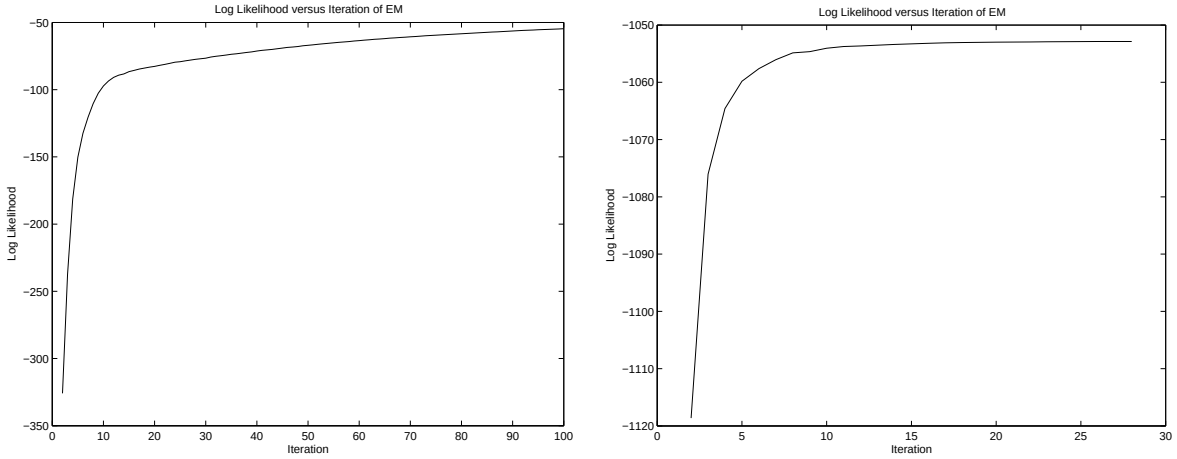
Figure 6.7: Five-fold cross-validation results for the IOHMM model on Toy Problem 2.

Name	Observation Model	Label Features
RBM(1)	LR	$(9, 0:1)$
RBM(2)	LR	$(9, 0:1)$, $(3, 0:7)$

Table 6.6: Details of the RBM-CRF models used with Toy Problem 2.

validation. The results from cross-validation are shown in Figure 6.7. Five restarts at different initial parameter settings were done for each fold to help deal with local optima. The number of states chosen for the final model was eight. The HRF also had eight states. To keep the training time reasonable, the number of iterations of EM was limited to 100 and the number of steps of optimization in the M step was limited to 100. Plots of the log likelihood during typical runs of EM are shown for the IOHMM and the HRF in Figure 6.8. The change in log likelihood on the first iteration is typically very large so the initial log likelihood is not shown. EM for the IOHMM converges quite slowly, but the HRF converges quite quickly, typically in 10 to 20 iterations.

RBM(1) had a logistic regression expert and one group of nine label features that looked at adjacent label variables. RBM(2) was like RBM(1) but also included a group



(a) IOHMM (b) HRF

Figure 6.8: Typical log likelihood versus iteration of EM curves for the IOHMM and HRF models.

of two label features that looked at eight label variables in a row. The learning rate was 0.1, the weight decay was $\frac{1}{\sigma^2}$, and a momentum of 0.9 was incorporated after 25 iterations. 1000 iterations of CD were done. For the first 500 iterations, one reconstruction step was used; after iteration 500, the number of reconstruction steps was increased to three. 150 iterations of Gibbs sampling were done to compute the approximate marginals for the RBM-CRF models at test time; 50 of the samples were discarded as burn-in.

Method	Mean Time (s)	Standard Deviation
LR	1.6×10^2	4.0×10^1
MEMM	1.1×10^1	1.5×10^0
IMEMM	4.3×10^2	7.5×10^1
CRF	2.3×10^2	4.0×10^1
ICRF	3.9×10^2	5.8×10^{-1}
IOHMM	7.3×10^3	9.4×10^2
HRF	5.8×10^3	1.5×10^3
RBM(1)	1.3×10^4	7.1×10^2
RBM(2)	2.0×10^4	3.7×10^3

Table 6.7: Training times of the models averaged over five runs.

The training times for each model are shown in Table 6.7. The time for the IOHMM does not include the time taken to run cross-validation.

	LR	MEMM	IMEMM	CRF	ICRF	IOHMM	HRF	RBM(1)	RBM(2)
A	73.87	80.93	78.34	82.19	79.91	71.71	71.28	80.37	82.98
B	75.91	79.10	77.05	84.43	82.17	73.74	73.88	82.94	84.97
O	99.20	99.43	99.30	99.20	99.55	99.09	98.91	99.59	99.64
Average F1 Score	82.99	86.49	84.90	88.61	87.21	81.51	81.36	87.63	89.20
Average Accuracy	90.80	92.71	91.82	93.73	93.16	90.01	89.84	93.41	94.23
Instance Accuracy	21.20	37.40	31.30	52.30	45.70	17.90	18.40	52.70	63.90

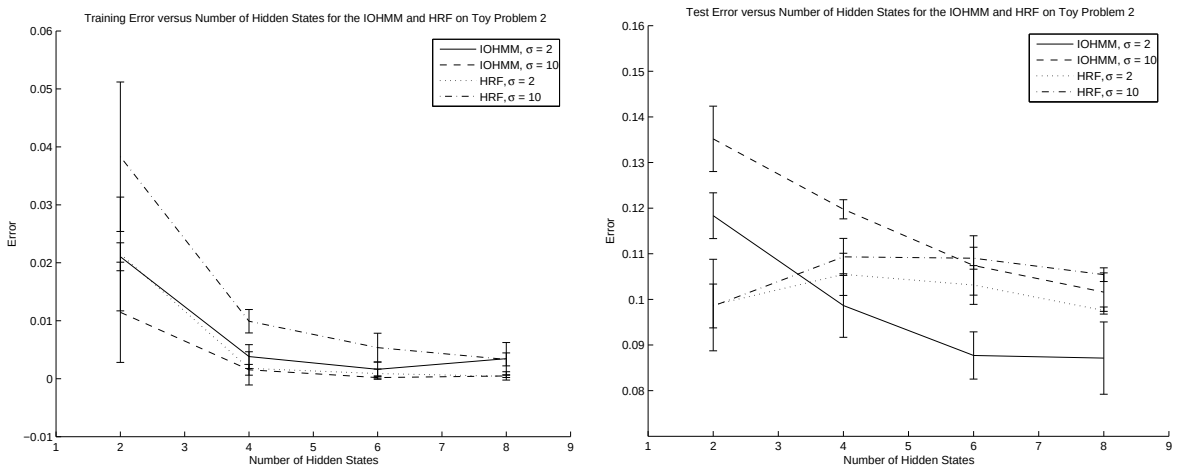
Table 6.8: Per-label F1 scores and overall performance results for the different models on test data for Toy Problem 2 All figures are in percent.

6.2.2 Results

The results of running the trained models on the test data are shown in Table 6.8. Logistic regression performed the most poorly with respect to all three overall performance metrics. The F1 scores for labels A and B are low, but, because there is very little ambiguity in the O observations, the F1 score for O is high

Adding links between adjacent label variables significantly improved both the F1 scores for the A and B labels as well as the overall performance. Using observation-dependent transitions increased the whole instance accuracy by about 6 to 8% when compared to using the separate factorization. The reason for the increase could be because observations may provide useful information about good pair-wise labellings. For example, the last item in the large blocks of A and B has a fairly unambiguous observation which may be associated with having a previous label which is also be an A or a B. The undirected models performed better than their directed version: the CRF classified about 15% more sequences correctly than the MEMM while the ICRF classified about 14% more sequences correctly than the IMEMM.

Although they performed about the same when compared to each other, the IOHMM and the HRF did not perform as well as the fully observed chain structured models. In fact, their instance accuracies were slightly lower than logistic regression. It could be that the models severely overfit the training data and were thus quite brittle. Both the IOHMM and the HRF were able to model the training data almost perfectly. This



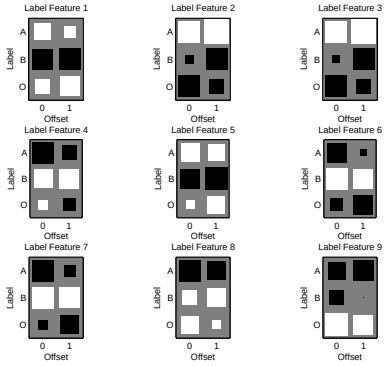
(a) Results on training data.

(b) Results of test data.

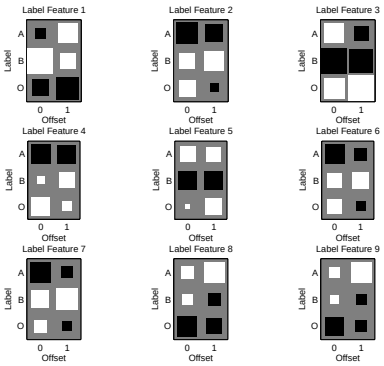
Figure 6.9: Comparison of IOHMM and HRF models with varying numbers of hidden states and settings of σ on training data and test data.

indicates that choosing the number of latent states for the latent state models can be difficult. It is usually possible to reduce overfitting by reducing the model complexity so to investigate this, we trained five IOHMM and HRF models for various numbers of latent states and two settings of σ. The results on running the trained models on the training data and on the test data are shown in Figure 6.9. The error metric is the average proportion of items misclassified. Even with a small setting of σ, all models are able to fit the training data quite well. However, they generally do quite poorly on the test data. Given enough latent states, it is likely that all models should be able to fit the test data well. A lower setting of σ, corresponding to a higher penalty for large weights also helps reduce overfitting.

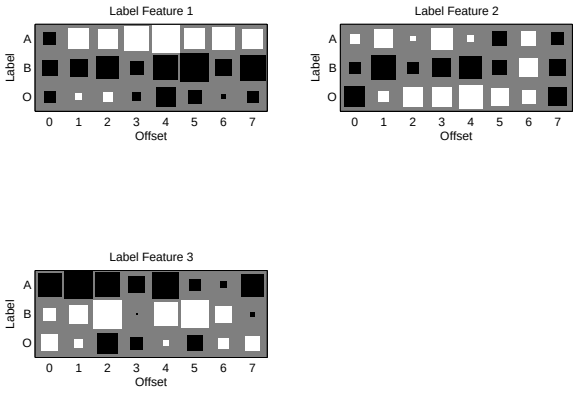
Neither of the RBM-CRF models appeared to overfit the training data and both performed well on the test data. The F1 scores of RBM(1) were close of those of the CRF, and the average accuracy was comparable. It is interesting to note that RBM(1) mimics the structure of the ICRF but performs at about the level of the CRF. RBM(2) did even better than RBM(1), classifying 64% of the sequences correctly, an increase of about 12%



(a) RBM(1)



(b) RBM(2) (pair-wise)



(c) RBM(2) (large structures)

Figure 6.10: Label feature weights learnt by the RBM-CRF models on data from Toy Problem 2

over both the CRF and RBM(1). The addition of the large label features significantly improved performance by providing the capacity to model the large structures.

Hinton diagrams of the weights learnt by the two RBM-CRF models are shown in Figure 6.10. RBM(1) learns several different kinds of label features. Most look for co-occurrences of the same label while some, like the fourth label feature, seem to specify both a co-occurrence and a transition. Several patterns are learnt by multiple label features, indicating that the set is over-complete.

The small label features of RBM(2) focus more on transitions although there are a couple (feature four and feature five) that still look for co-occurrences of the same label.

	LR	CRF	IOHMM	RBM(2)
A	77.91	96.63	91.96	82.10
B	79.95	96.87	92.43	83.49
O	99.60	99.76	99.94	99.63
Average F1 Score	85.82	97.75	94.78	88.41
Average Accuracy	92.46	98.76	97.25	93.80
Instance Accuracy	28.10	89.00	72.80	62.60

Table 6.9: Per-label F1 scores and overall performance results for the different models on training data for Toy Problem 2. All figures are in percent.

	LR	CRF	IOHMM	RBM(2)
A	42.31	25.38	37.09	38.96
B	24.48	30.19	16.73	21.43
O	48.80	8.22	50.76	45.28
Average F1 Score	38.53	21.26	34.86	35.22
Average Accuracy	42.12	22.01	40.78	38.05
Instance Accuracy	1.20	1.80	1.00	2.00

Table 6.10: Per-label F1 scores and overall performance results for the different models on test data for Toy Problem 2 All figures are in percent.

The larger label features look for larger groups of mostly contiguous labels. The first label feature matches the large group of As while the third matches the large group of Bs. Both features also match some of the Os that can occur either at the beginning or the end of the groups and appear to be matching the groups at different offsets given the longer-than-normal size of the groups and the multiple positive weights for start/end Os. The second label feature appears to match longer groups of Os but also seems to capture the edges of some A/B structural patterns.

To further investigate the issue of overfitting, we trained a logistic regression model, a CRF, an IOHMM with eight latent states, and the RBM(2) model on the test data and generated a new set of 1000 sequences to use for testing. The IOHMM model used a weight decay regularizer with $\sigma = 2$ while the other models used a weight decay regularizer with $\sigma = 10$. No random restarts were done. RBM(2) was trained for 200 iterations.

The results of running the models on the new training and testing data are shown in

Tables 6.9 and 6.10 respectively. All models appear to overfit the data as the values of all performance metrics dropped considerably on the test data. Even though the CRF was the best model with respect to the training data, it performs the worst on the test data. The IOHMM and RBM(2) were more robust than the CRF but still performed slightly worse than logistic regression. This indicates that perhaps the selection of both the model and the parameters to control the capacity of the model needs to be done very carefully.

6.3 Toy Problem 3: Memory

One of the differences between the RBM-CRF and the templates models is that there are no direct connections between the label variables or between the latent variables. Although this implies that there is no notion of state in an RBM-CRF model like there is in a MEMM/CRF or IOHMM/HRF, the RBM-CRF may still be able to use its latent variables together to model state.

We use the synthetic A/B/R/I problem proposed by Kakade, Teh and Roweis [14] to illustrate how the different models perform when state is required. There are four observations $\{A, B, R, I\}$ and two labels $\{0, 1\}$. A always maps to 0 and B always maps to 1. The observation R (resume) causes the last A or B label to be repeated. When the observation is I (interrupt), the label is the opposite of the label of the last I observation. We consider a data model in which the initial I observation always has label 0.

A standard CRF can model the Rs correctly since only one bit of information is required and there are two labels corresponding to those states. An interrupt observation does not cause any change in state. To model both the R and I observations, two bits of memory are required so a CRF cannot model the data correctly, but latent state models such as the IOHMM and the HRF can. RBM-CRF models should not be able to completely model the I observations. It may be possible for them to learn how I observations

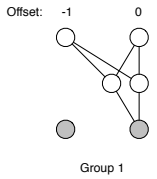


Figure 6.11: RBM(1) groups

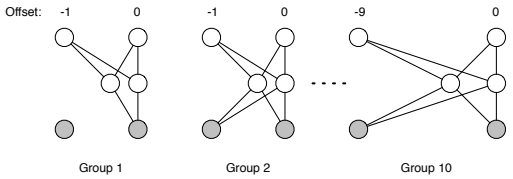


Figure 6.12: RBM(2) groups

work when restrictions are made on the maximum distance between I observations. However, it is clearly impossible for them to model I when there are no restrictions because an infinite number of label features would be required. In addition, there is no way for the RBM-CRF models to remember that the label for the first I observation in 1 because no state is maintained.

6.3.1 Setup

We generated 100 sequences of length 50 for the training set and 100 sequences, also of length 50, for the test set. About 25% of the observations were A, 25% were B, 40% were R, and 10% were I. As in Toy Problem 1, the observations are represented with a one-hot encoding.

We trained a logistic regression model, a CRF, an IOHMM, an HRF, and two RBM-CRF models. With respect to the IOHMM and the HRF, had 4 latent states, the number of optimizations in the M step was limited to 100, and the total total number of iterations of EM limited to 100. All models used a weight decay regularizer with $\sigma = 10$. Training was halted when the relative change in the log likelihood was less than 1×10^{-6}.

The architectures of the two RBM-CRF models that we trained were slightly different from those considered with the previous two problems as the label features were conditioned on the observations and did not always look at adjacent label variables. RBM(1) had a logistic regression expert and two label features that were conditional on the input (Figure 6.11). RBM(2), illustrated in Figure 6.12 was like RBM(1) but also had nine

groups of two label features which, together, approximated variable length memory by looking at pairs of labels separated by varying distances. The separations ranged from zero (adjacent) to eight. The learning rate was 0.01, the weight decay was $\frac{1}{\sigma^2}$, and a momentum of 0.9 was incorporated after 25 iterations. 1000 iterations of CD were done. For the first 500 iterations, one reconstruction step was used; after iteration 500, the number of reconstruction steps was increased to three. 150 iterations of Gibbs sampling were done to compute the approximate marginals for the RBM-CRF models at test time; 50 of the samples were discarded as burn-in.

Method	Mean Time (s)	Standard Deviation
LR	2.2×10^1	7.2×10^0
CRF	2.5×10^2	6.6×10^1
IOHMM	3.3×10^3	4.5×10^2
HRF	8.8×10^3	4.3×10^3
RBM(1)	1.4×10^4	1.2×10^3
RBM(2)	3.9×10^4	5.4×10^4

Table 6.11: Training times of the models averaged over five runs.

The average training times for the various models are shown in Figure 6.11. All models except for RBM(2) were trained five times, each time starting with a different set of initial parameters. Due to issues with server availability, only two runs of training were done for RBM(2), explaining its high variance.

6.3.2 Results

The results of running the trained models on the test data are shown in Table 6.12. The number of mistakes made when the observation was either an R or an I is shown in Figure 6.13.

Logistic regression achieved an average accuracy of 76%. It is unable to model both the R and I observations. A detailed look at the errors showed that all R observations were classified as 1 and all I observations were classified as 0. Adding links between adjacent label variables to create a state machine improves both the F1 scores and the

	LR	CRF	IOHMM	HRF	RBM(1)	RBM(2)
0	71.67	91.33	94.73	95.99	90.20	93.65
1	79.75	91.23	94.83	95.85	90.55	93.79
Average F1 Score	75.71	91.28	94.78	95.92	90.38	93.72
Average Accuracy	76.38	91.28	94.78	95.92	90.38	93.72
Instance Accuracy	0.00	1.00	5.00	7.00	1.00	4.00

Table 6.12: Per-label F1 scores and overall performance results for the different models on test data for Toy Problem 3. All figures are in percent.

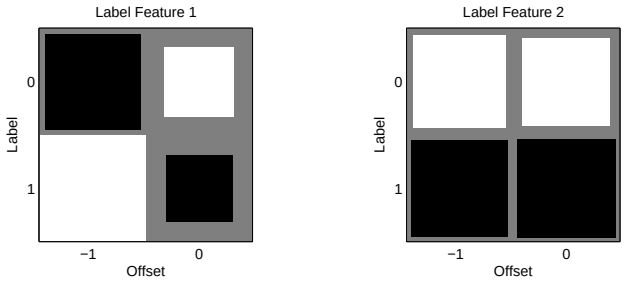
	LR	CRF	IOHMM	HRF	RBM(1)	RBM(2)
R	48.22	11.41	1.75	0.10	12.46	4.80
I	44.38	42.54	46.22	41.31	47.24	44.99

Table 6.13: Per-observation error rates (in percent) for observations R and I on Toy Problem 3 test data.

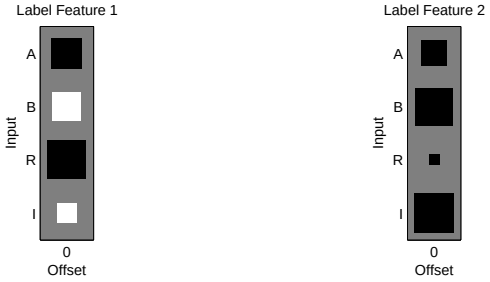
average accuracy, although the whole instance accuracy is still very low. The error rate of the CRF is not zero because the CRF appears to be trying to model both the R observations and the I observations. A CRF with hand-picked parameters can achieve the optimal instance accuracy of 95%. Kakade, Teh and Roweis [14] observed that changing the objective function to maximize the marginal probabilities of the sequence elements rather than the joint probability of the sequence can improve model performance so changing the objective function may help the performance of the CRF on this task.

The IOHMM and the HRF performed much better than the CRF. The error rate on the R observations dropped to under 2% for the IOHMM and was almost zero for the HRF. However, both did not model the joint distribution correctly even though they should be able to (hand-tuned models achieve an instance accuracy of 98.8%). There are two likely explanations. First, the models may not have been trained long enough or may not have had enough random restarts to escape local optima. Second, like with the CRF, the objective function may need to be changed.

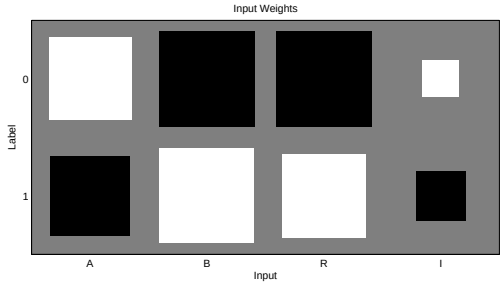
The performance of the two RBM-CRF models is interesting. RBM(1) performed just about as well as the CRF with respect to average F1 score, average accuracy and whole instance accuracy. Hinton diagrams of the weights learnt by the model are shown



(a) Label feature weights



(b) Label feature input weights



(c) Logistic regression weights

Figure 6.13: Label feature weights learnt by RBM(1) on data from Toy Problem 3

in Figure 6.13. The second label feature looks reasonable: its weights indicates that it prefers the observation R and two adjacent 0 labels. To completely model the R observations, the first label feature should prefer the observation R and two adjacent 1 labels. However, the first label feature does not do this; in fact, it's not clear what it's is modeling. Given that the logistic regression model maps R to 1 and I to 0, the label feature could be working with the logistic regression model to capture some of the dynamics of the I observations and the R observation as the observation weights bias the label feature towards being active for I and the label weights prefer the pattern 1-0.

RBM(2) performed better than RBM(1): the error rate with respect to observation R dropped significantly, indicating that additional label features were used primarily to model the effects of the R observations.

6.4 Cora: Reference Paper Citations

The Cora citations dataset[1] was developed by the Cora project [20] and used by Peng and McCallum (PM) in a recent study of CRFs [24]. It consists of 500 bibliography entries from academic papers. There are 13 possible labels for each token in each entry: author, book title, date, editor, institution, journal, location, note, pages, publisher, tech, title, and volume.

6.4.1 Setup

Following the work of PM, we divided the data set into a training set of 350 citations (chosen at random) and a test set containing the remainder. Each entry was split into tokens based on whitespace. The observation features that we used are modeled after those used by PM and are described in Table 6.14. All features are binary, and, except for the "EndsIn" features, ignore trailing commas, periods, colons, and semi-colons. None

[1] Available from `http://www.cs.umass.edu/~mccallum`.

Name	Description
InitCap	Starts with a capitalized letter.
AllCaps	All characters are capitalized.
AllDigits	All characters are digits.
ContainsDigits	Contains at least one digit.
ContainsDots	Contains at least one period.
ContainsDash	Contains at least one dash.
LonelyInitial	A single letter followed by a period.
SingleChar	One character only.
CapLetter	One capitalized character only.
URL	Regular expression for a URL
InParen	In parentheses.
Year	Regular expression for a year.
Punc	Only punctuation (period, comma, colon, semi-colon).
EndsInComma	Ends in a comma.
EndsInDot	Ends in a period.
EndsInColon	Ends in a colon.
EndsInSemiColon	Ends in a semi-colon.
EndsInQuote	Ends in a quotation mark.
StartsWithQuote	Starts with a quotation mark.
Name	Appears in a list of names.
Place	Appears in a list of places.
Acronyms	Appears in a list of acronyms.
Months	Appears in a list of month and day names.
Word	Matches the word.

Table 6.14: Input features used with the Cora-Refs data

of the list-based features are complete: there are words in both the training data and the test data that belong in a category but do not have an entry. 1374 vocabulary features ("Word") were created by extracting whole words from the training data. The words were converted to lowercase and then stemmed. The observations vectors were augmented to include the observations for the previous token and the next token. That is, $\mathbf{x}'_t = [\mathbf{x}_{t-1}; \mathbf{x}_t; \mathbf{x}_{t+1}]$. The dimensionality of the augmented observation vectors was 4191.

We trained all fully observed temporal models and seven different RBM-CRF models. We did trained one IOHMM with six latent states but no HRF because of time constraints. The number of iterations of EM was limited to 100 and the number of iterations of optimization in the M step was limited to 100. We did start cross-validation to choose the number of latent states for the IOHMM model, but it was stopped as it would have taken over three weeks to complete, even with no random restarts and with running

the cross-validation procedure for each state on a separate CPU. In general, all models took a while to train. No random restarts were done in order to keep the overall training time down. We report the time taken to train several of the models in Table 6.15. The times listed for the four RBM-CRF models that use a pre-trained classifier (RBM-LR(1), RBM-LR(2), RBM-CRF(1), and RBM-CRF(2)) do not include the time needed to train the classifier.

Method	Time (s)
LR	8.4×10^4
MEMM	1.2×10^4
IMEMM	3.6×10^4
CRF	3.6×10^5
ICRF	4.0×10^4
IOHMM	8.1×10^5
RBM(1)	1.6×10^5
RBM(2)	2.5×10^5
RBM(3)	2.9×10^5
RBM-LR(1)	1.8×10^4
RBM-LR(2)	4.2×10^4
RBM-CRF(1)	1.5×10^4
RBM-CRF(2)	4.4×10^4

Table 6.15: Training times of selected models.

All temporal models used a weight decay regularizer with $\sigma = 10$. To reduce the time taken to train the undirected models, the weights of the CRF were initialized with those from the trained MEMM and the weights of the ICRF were initialized with those from the trained IMEMM.

The details of the different RBM-CRF models are shown in Table 6.16. RBM(1), RBM(2), and RBM(3) were trained for 1000 iterations. The first 500 iterations used one-step reconstructions while the last 500 used three-step reconstructions. The other four models were trained for 500 iterations and used one-step reconstructions. With respect to the integration of the pre-trained classifiers, we used $\gamma = 0.75$ for RBM-LR(1) and RBM-LR(2) and $\gamma = 0.6$ for RBM-CRF(1) and RBM-CRF(2). These values were chosen by hand based on examinations of the entropy of the prediction distributions produced by the LR and CRF models. The low value of γ for RBM-CRF(1) and RBM-

Name	Observation Model	Label Features
RBM(1)	LR	$(15, 0:1)$
RBM(2)	LR	$(15, 0:1)$, $(20, 0:2)$
RBM(3)	LR	$(15, 0:1)$, $(20, 0:2)$, $(5, 0:9)$, $(5, 0:19)$
RBM-LR(1)	Pre-trained LR	$(10, 0:9)$
RBM-LR(2)	Pre-trained LR	$(10, 0:4)$, $(10, 0:9)$, $(10, 0:19)$
RBM-CRF(1)	Pre-trained CRF	$(10, 0:9)$
RBM-CRF(2)	Pre-trained CRF	$(10, 0:4)$, $(10, 0:9)$, $(10, 0:19)$

Table 6.16: Details of the RBM-CRF models used with the Cora references data set.

CRF(2) is because the prediction probabilities produced by the CRF had almost no entropy. All models used a learning rate of 0.1 and a weight decay of $\frac{1}{\sigma^2}$. A momentum term that added 0.9 of the previous update was added after 25 iterations. At test time, 400 iterations of Gibbs sampling were done; the first 200 iterations were discarded as burn-in.

6.4.2 Results

The per-label F1 scores and overall performance results for selected models are shown in Table 6.17. In particular, we show results for logistic regression, RBM-LR(2), our CRF, RBM-CRF(2), IOHMM, and RBM(3) as well as the results reported by PM (CRF*). Bar graphs of $-\log(1 - F1)$ for these models are shown in Figure A.1. This measure highlights the differences in the scores between the models and is useful because the F1 scores of several models on some labels are quite similar. The results for all models are given in Appendix A along with confusion matrices for LR, RBM-LR(2), CRF, and RBM-CRF(2).

In general, the models that are capable of representing label structures performed better than logistic regression. The RBM-CRF models with the integrated logistic regression expert were the exception. Even though their instance accuracies were slighly higher than the instance accuracy of logistic regression, their average F1 scores and accuracies were 10 to 15% lower. There are several possible explanations. First, RBM(1) through RBM(3) only had 15 pairwise label features which may not be enough to model

	LR	RBM-LR(2)	CRF*	CRF	RBM-CRF(2)	IOHMM	RBM(3)
Author	92.06	95.01	99.4	99.29	99.29	92.14	95.07
Book Title	78.86	88.47	93.7	91.86	91.88	83.70	42.91
Date	98.50	98.50	98.9	98.25	98.01	97.77	96.50
Editor	53.97	55.56	87.7	90.91	90.91	32.94	52.63
Institution	71.72	81.16	94.0	86.15	86.15	73.53	61.40
Journal	61.40	80.12	91.3	90.29	89.97	70.00	62.05
Location	80.85	82.54	87.2	83.98	82.87	80.90	74.39
Note	47.76	55.38	80.8	56.00	77.78	19.61	28.57
Pages	95.02	94.66	98.6	97.28	97.66	96.53	88.51
Publisher	72.58	76.56	76.1	85.04	85.71	66.67	67.92
Tech	64.10	63.29	86.7	68.42	68.42	66.67	50.00
Title	89.01	93.15	98.3	96.45	97.79	90.33	81.17
Volume	93.67	93.17	97.8	95.60	95.60	90.80	89.44
Average F1 Score	76.89	81.35	91.50	87.66	89.39	73.97	68.51
Average Accuracy	85.08	90.11	95.37	94.49	95.05	86.25	76.77
Instance Accuracy	16.00	42.00	77.33	66.00	65.33	24.67	29.33

Table 6.17: Per-label F1 scores and overall performance results for selected models on the Cora test data. All figures are in percent.

the important interactions present in the data. The addition in RBM(2) of 20 features that looked at groups of three labels did not increase overall performance that much, although the F1 score of certain labels did increase, indicating that even more small scale label features may be required. Second, the logistic regression classifier is very complex (it has on the order of 50,000 parameters). The RBM part of the models has many fewer parameters (RBM(3) has roughly 3000). It may be difficult for the training procedure to effectively deal with the difference in complexity. Third, there may be problems with local optima since the models were only trained once, although this might not be a large factor as all three models performed quite poorly.

The CRF and ICRF models outperformed their directed counterparts once again. The CRF performs slightly better than the ICRF; however, the MEMM does not perform as well as the IMEMM with respect to average F1 score and accuracy. It does, however, classify more sequences entirely correctly than the IMEMM.

PM used a feature induction scheme to learn which observation features and combinations thereof are useful for modeling the data. Feature induction is useful because it can build features that look for particular structures in the observations that are good

for classifying observations and may help to explain the high instance accuracy that their model achieved when compared to the instance accuracy of the CRF model. In general, the CRF achieved F1 scores close to what PM reported with the exception of the scores for the Institution, Note, and Tech labels. While the addition of larger scale structures with the RBM-CRF(1) and RBM-CRF(2) models did not significantly affected most of the scores, the score for Note did increase significantly.

The results of the IOHMM were disappointing, although not unexpected given our experience with the IOHMM on the synthetic problems. The average F1 score was lower the the average F1 score of logistic regression though it did manage to classify almost twice the number of test sequences entirely correctly. Like with Toy Problem 2, the IOHMM seems to be severely overfitting the data as it almost perfectly fits the training data. Regularizing the model more with a stricter weight decay parameter and choosing the number of states using cross-validation should help improve performance. However, it is not clear if the extra time required to find good settings of these parameters is worth it as the IOHMM took nine days to train, over six times longer than the combined time to train the LR and RBM-LR(2) models.

Adding the ability to learn large scale structures using an RBM-CRF can improve performance. Building a structural model on top of logistic regression improved the average F1 score, the average accuracy, and the instance accuracy when compared to plain logistic regression. In fact, RBM-LR(2) classified 163% more sequences correctly. A look at the confusion matrices (Tables A.3 and A.4) indicates that RBM-LR(2) correctly labels more tokens that belong groups whose tokens can be confused (such as Title, Book Title, and Journal).

Adding an RBM-CRF on top of the CRF model does not help as much as adding one on top of logistic regression. The reason might be that the CRF produces very low entropy predictions so even with the low value of γ the RBM-CRF may not be able to affect Gibbs sampling too much. It would be worthwhile varying γ to see if better results

can be obtained.

In Figures A.2, A.3, and A.4 we show Hinton diagrams of the weights learnt by selected label features of the RBM-LR(2) model. We observed that many of the label features in RBM-LR(2) were very similar to label features in RBM-CRF(2) indicating that the models were pulling out the common structures.

The short label features of length five tend to focus on modeling contiguous groups of labels. They often have large positive weights for several types of labels and very negative weights for others. This indicates that they might be modeling what they expect not to see as well as what they expect to see. The short label features, to a lesser extent, also model relationships between labels. For example, label feature four seems to represent the relationship "Author, Book Title" although it has some other features mixed in too.

Whereas the small label features looked mostly at groups of contiguous nodes, the label features that looked at groups of 10 labels tend to focus more on relationships between labels. For example, label feature six focuses on the relationship "Editor, (Journal,Book Title)" and label feature ten predominantly models "Author, Title". Most label features tend to look for multiple patterns. Label feature nine recognizes "Title, Journal" in addition to "Editor, Volume" and "Author, Institution". The largest label features that look at groups of twenty labels tend to model global structure, but don't seem to be as well-defined as the smaller models. They focus on structures like "Title, Book Title" as well as labels that appear near the end of an entry such Publisher, Location, Date, and Page.

Chapter 7

Conclusions

Standard models for sequence labeling have difficulty capturing large scale structures as they are completely parameterized with respect to the structures that they look for. Any attempt to incorporate large scale structure directly by expanding the set of known configurations results in an unpalatable exponential increase in model complexity.

We have discussed some common models and have described two families of models that have not been extensively studied in the context of sequence labeling: chain structured latent state models and parameterized template models. Chain structured latent state models posit the existence of a latent finite state machine whose state transition probabilities are conditioned on observations. They are more expressive than fully observed models because the states of the machine have been decoupled from the labels. We presented two specific models: the IOHMM and the HRF.

Template models define a collection of parameterized label features that look at subsets of label variables. The RBM-CRF model presented provides an elegant way of specifying problem-specific architectures that can efficiently learn label structures present in the data.

We have observed that both types of models can outperform standard techniques on data that exhibit characteristics including long term dependencies, large scale structures,

and variable term memory. However, both types of models have drawbacks. With respect to the the IOHMM and the HRF, it may hard to pick the number of latent states. The models can take a long time to train, but the gain in performance over standard methods may not be worth it. Both models are subject to local optima, but the HRF seems to be more susceptible than the IOHMM. They are also prone to overfitting, although a good regularizer and an appropriate number of latent states can result in better performance. Better initialization strategies might also make a difference and should be investigated further.

While the deciding on architectures for RBM-CRF models is generally more straightforward than choosing the number of latent states for an IOHMM or an HRF, it is easy to pick a sub-optimal architecture and may be difficult to pick a good minimal one. Certain aspects of training rely on hand picked parameters and may require some trial and error to set properly. For complex problems, directly integrating a model like logistic regression may result in poor performance, and experimentation may be necessary to determine what is most effective. Finally, the RBM-CRF is translation invariant and maintains no state so it is not well suited to tasks that require variable length memory.

In general, the choice of model for a particular problem is highly dependent on the properties of the problem. While a certain amount of power comes from being able to incorporate arbitrary observation features, problem-specific architectures may provide a significant increase in performance when compared to off-the-shelf methods.

7.1 Future Directions

Our research has opened up a large number of possible directions for future research. In the short-term, more experiments on real data need be be performed. It would be interesting to explore these models in areas such as speech recognition, specifically, in phoneme recognition, where the ability to recognize patterns of phonemes and phenomena

such as co-occurrences may be very useful. Another possible area of interest would be part-of-speech tagging. Here, it is common to see certain patterns in the parts-of-speech which may be lost with models like the CRF. Finally, it might be fruitful to apply RBM-CRF models to other tasks where CRFs have been applied successfully such as parsing [28] and coreference resolution [21].

While the task of picking the size and kind of templates is, in general, easier than picking the number of latent states in an IOHMM or an HRF, it still may be quite difficult. It would be beneficial to explore methods for automating the label feature selection process. In particular, Lafferty et al. [15] and McCallum [18] have developed methods for efficiently inducing observation features in CRFs. It may be possible to apply their methods, or variations on them, to label feature induction. Given the success of observation feature induction, label feature induction coupled with observation feature induction may be quite successful.

One issue with the RBM-CRF model is that it is flat: there is only one level of information regarding the labels. It may be desirable to have the label features interact in some way or to incorporate regional or global context. One way to address these issues is to develop hierarchical RBM-CRF models in which label features interact through meta label features, which themselves interact through meta meta label features, and so on. This might be especially useful in unsupervised settings in which the labeling of the data is not known and the goal of learning is to discover a labeling that makes sense. Specific applications of this may include DNA motif detection and music structure analysis. Another way to approach this is to use multi-scale RBM-CRF models [9] in which one gets varying levels of resolution in terms of label specificity.

Finally, template models do not maintain state like chain structured models so it can be difficult for them model data that exhibits properties that would be easy to model using state. Hierarchical models may be one way of dealing with such data. Another way would be to use product models that combine one or more latent state models like

the IOHMM or HRF with an RBM-CRF model.

Appendix A

Experimental Results

A.1 Cora: Reference Paper Citations

This appendix contains complete results on the Cora data set (Table A.2) as well as figures and tables referenced in Chapter 6. The abbreviations used in the confusion matrices (Tables A.3 to A.6) are given in Table A.1.

Label	Abbreviation
Author	au
Book Title	bo
Date	da
Editor	ed
Institution	in
Journal	jo
Location	lo
Note	no
Pages	pg
Publisher	pu
Tech	te
Title	ti
Volume	vo

Table A.1: Label Abbreviations

	CRF*	LR	MEMM	IMEMM	CRF	ICRF	IOHMM	RBM(1)	RBM(2)	RBM(3)	RBM-LR(1)	RBM-LR(2)	RBM-CRF(1)	RBM-CRF(2)
Author	99.4	92.06	97.21	97.13	99.29	99.11	92.14	91.08	92.93	95.07	95.66	95.01	99.23	99.29
Book Title	93.7	78.86	88.65	88.35	91.86	91.48	83.70	54.71	36.38	42.91	87.69	88.47	92.48	91.88
Date	98.9	98.50	93.17	98.50	98.25	99.00	97.77	97.52	97.01	96.50	98.01	98.50	98.50	98.01
Editor	87.7	53.97	70.39	73.95	90.91	91.98	32.94	4.65	47.49	52.63	70.48	55.56	90.91	90.91
Institution	94.0	71.72	74.14	77.31	86.15	84.85	73.53	40.78	52.83	61.40	83.82	81.16	86.15	86.15
Journal	91.3	61.40	74.72	76.02	90.29	82.93	70.00	47.81	54.00	62.05	81.93	80.12	90.03	89.97
Location	87.2	80.85	64.71	74.29	83.98	83.06	80.90	72.61	70.81	74.39	84.66	82.54	82.87	82.87
Note	80.8	47.76	36.84	75.00	56.00	57.97	19.61	8.51	16.33	28.57	53.12	55.38	62.16	77.78
Pages	98.6	95.02	88.61	95.38	97.28	96.55	96.53	95.31	92.72	88.51	95.02	94.66	97.66	97.66
Publisher	76.1	72.58	53.85	67.23	85.04	79.03	66.67	70.97	58.12	67.92	72.18	76.56	82.81	85.71
Tech	86.7	64.10	63.77	70.27	68.42	61.76	66.67	23.53	42.42	50.00	62.34	63.29	69.33	68.42
Title	98.3	89.01	94.83	95.79	96.45	95.70	90.33	78.90	75.79	81.17	92.78	93.15	96.60	97.79
Volume	97.8	93.67	88.75	90.68	95.60	94.41	90.80	88.62	87.01	89.44	92.59	93.17	95.60	95.60
Average F1 Score	91.50	76.89	76.13	83.07	87.66	85.99	73.97	59.62	63.37	68.51	82.33	81.35	88.03	89.39
Average Accuracy	95.37	85.08	89.35	91.34	94.49	93.66	86.25	75.66	73.67	76.77	90.33	90.11	94.66	95.05
Instance Accuracy	77.33	16.00	56.67	50.00	66.00	63.33	24.67	21.33	21.33	29.33	39.33	42.00	66.00	65.33

Table A.2: Per-label F1 scores and overall performance results for all models on the Cora test data. All figures are in percent.

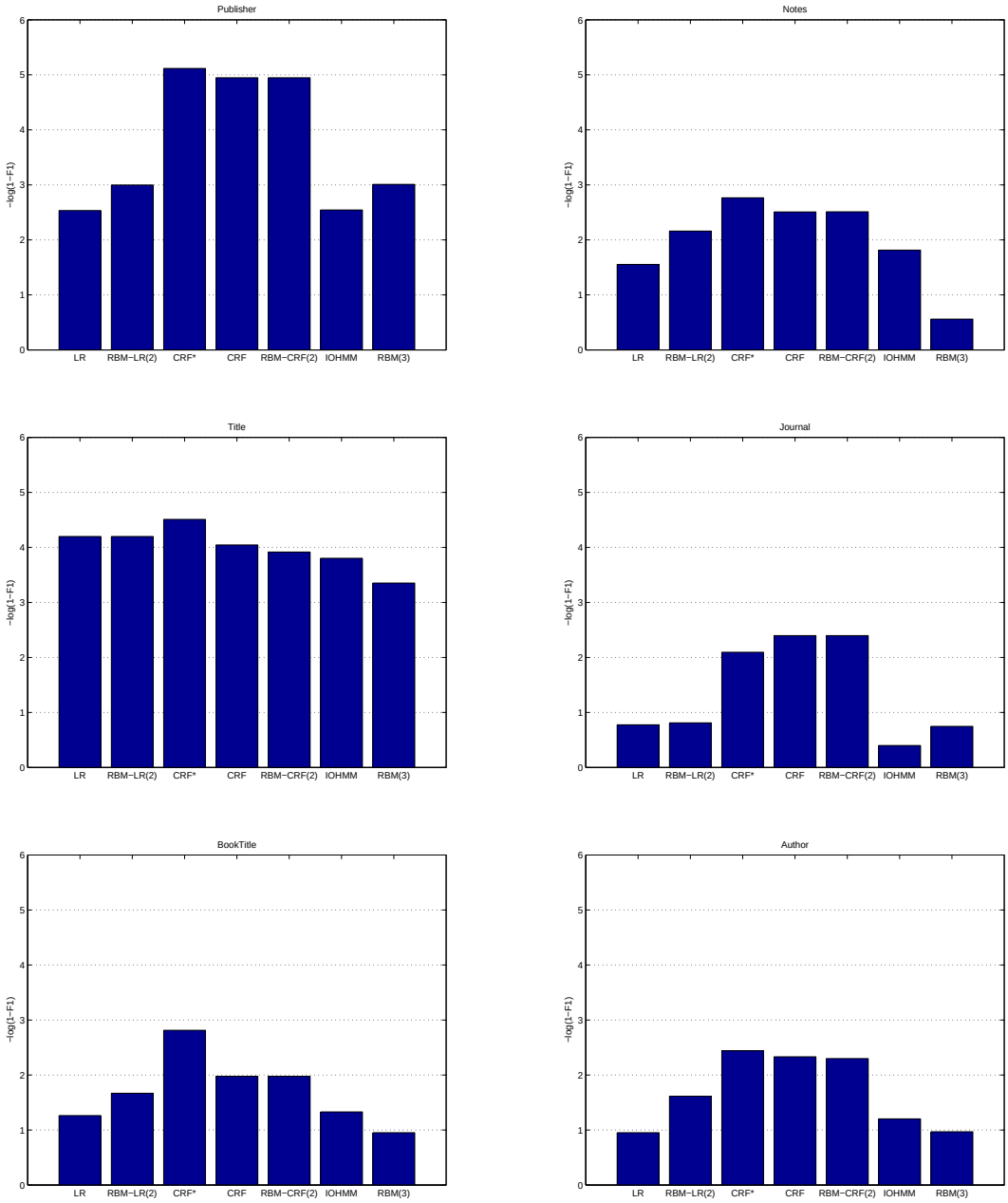


Figure A.1: Comparison of F1 scores of several models on Cora references test data.

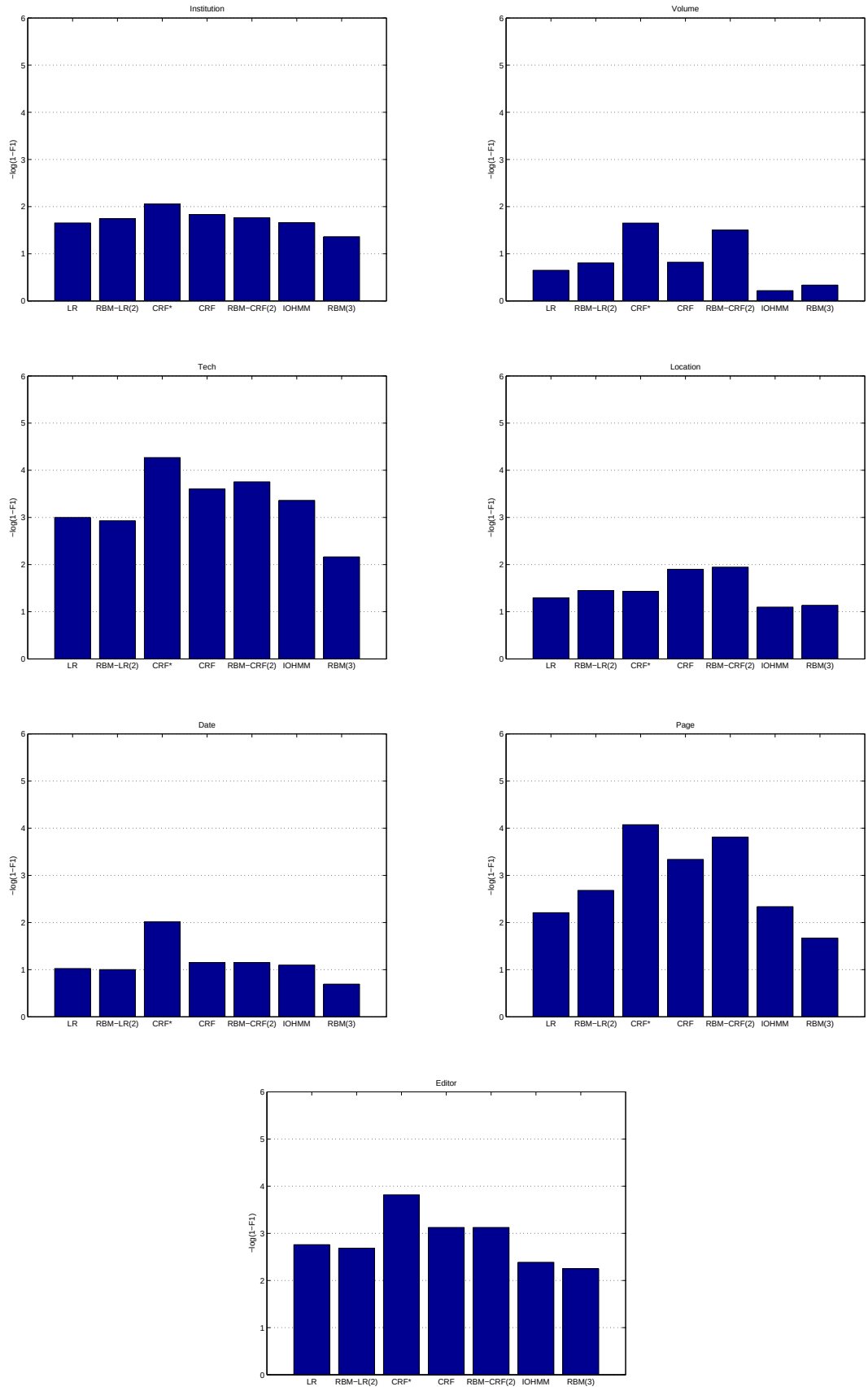


Figure A.1: Comparison of F1 scores of several models on Cora references test data.

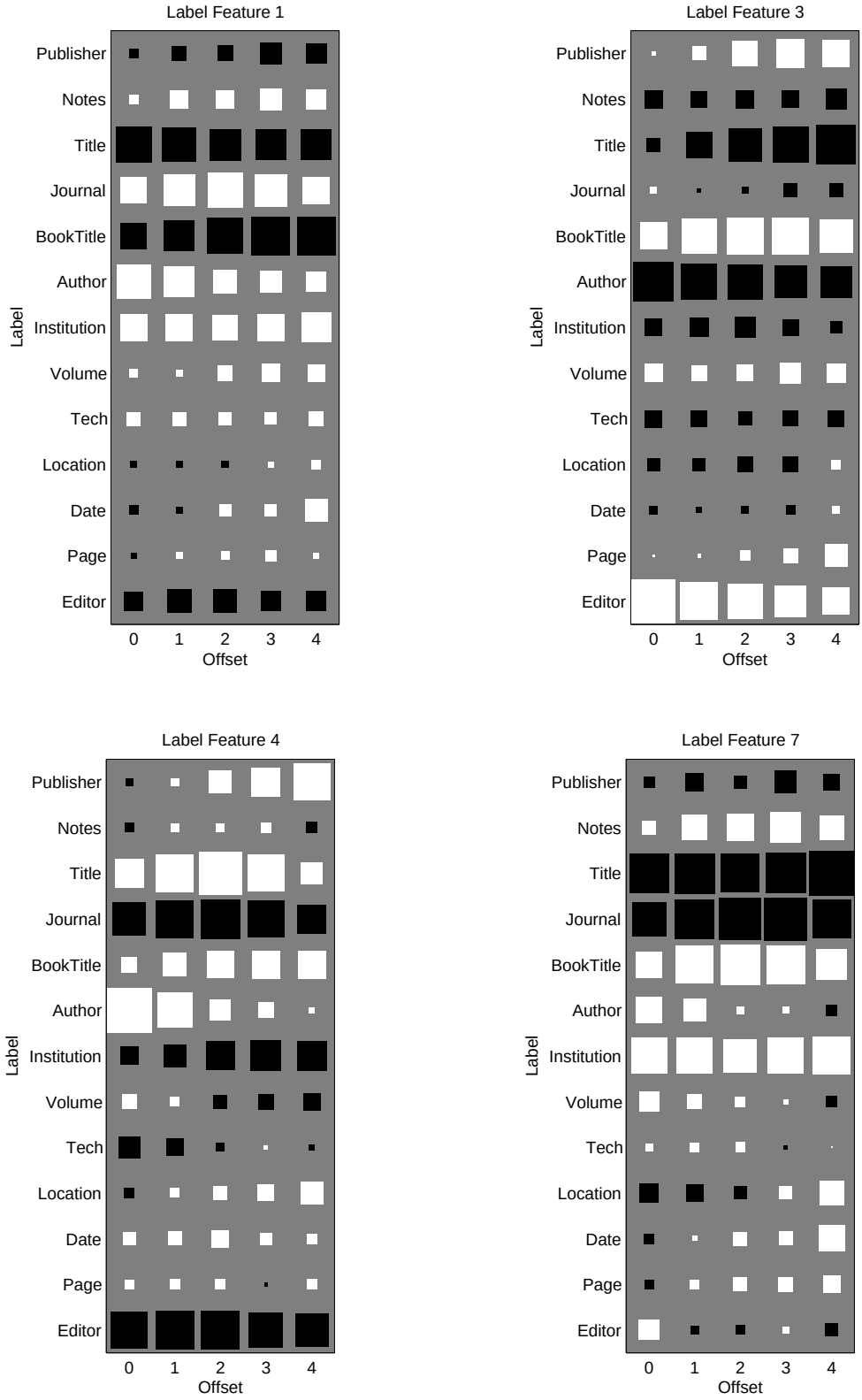


Figure A.2: Hinton diagrams of the weights learnt by selected label features of length five in RBM-LR(2).

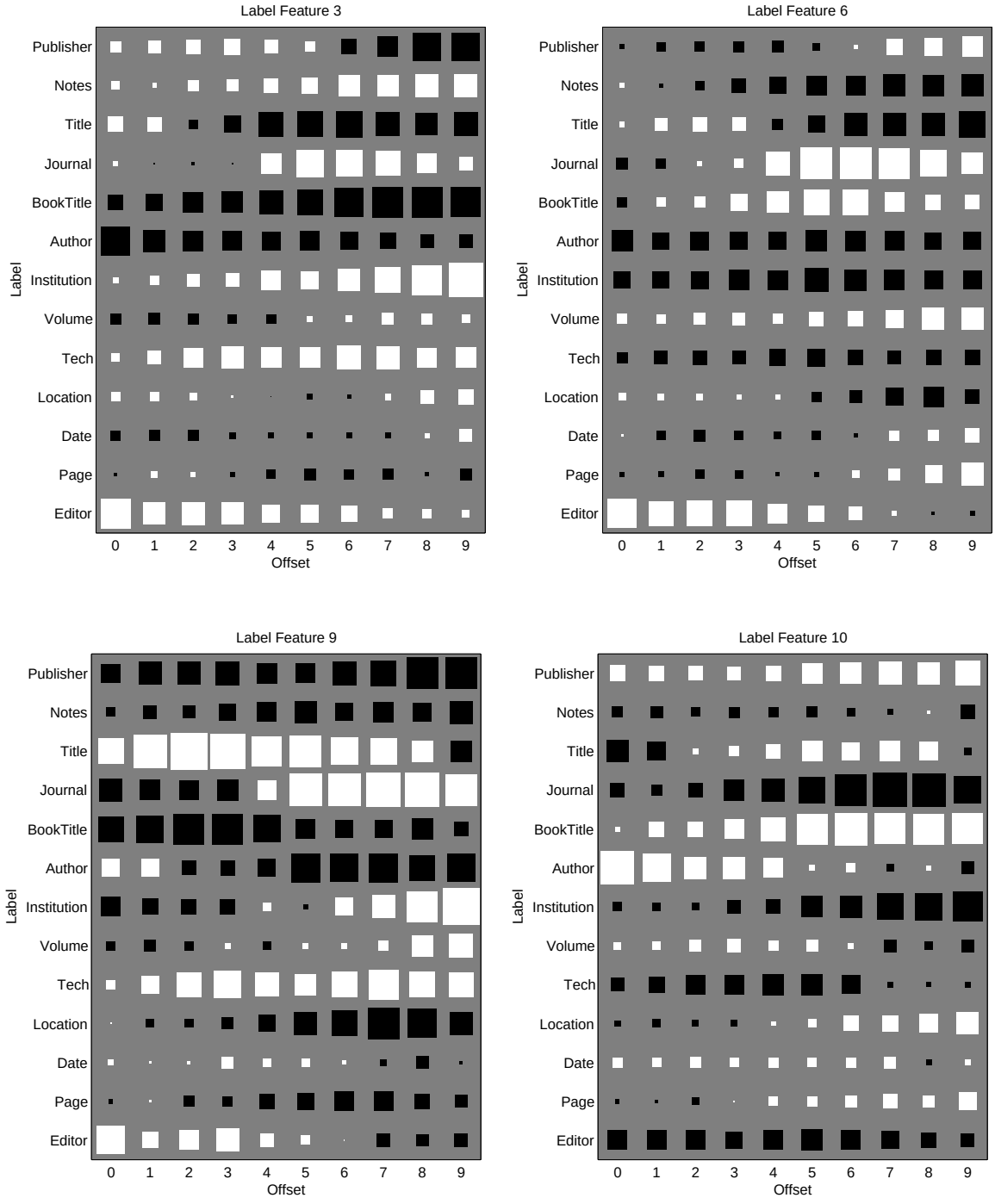


Figure A.3: Hinton diagrams of the weights learnt by selected label features of length 10 in RBM-LR(2).

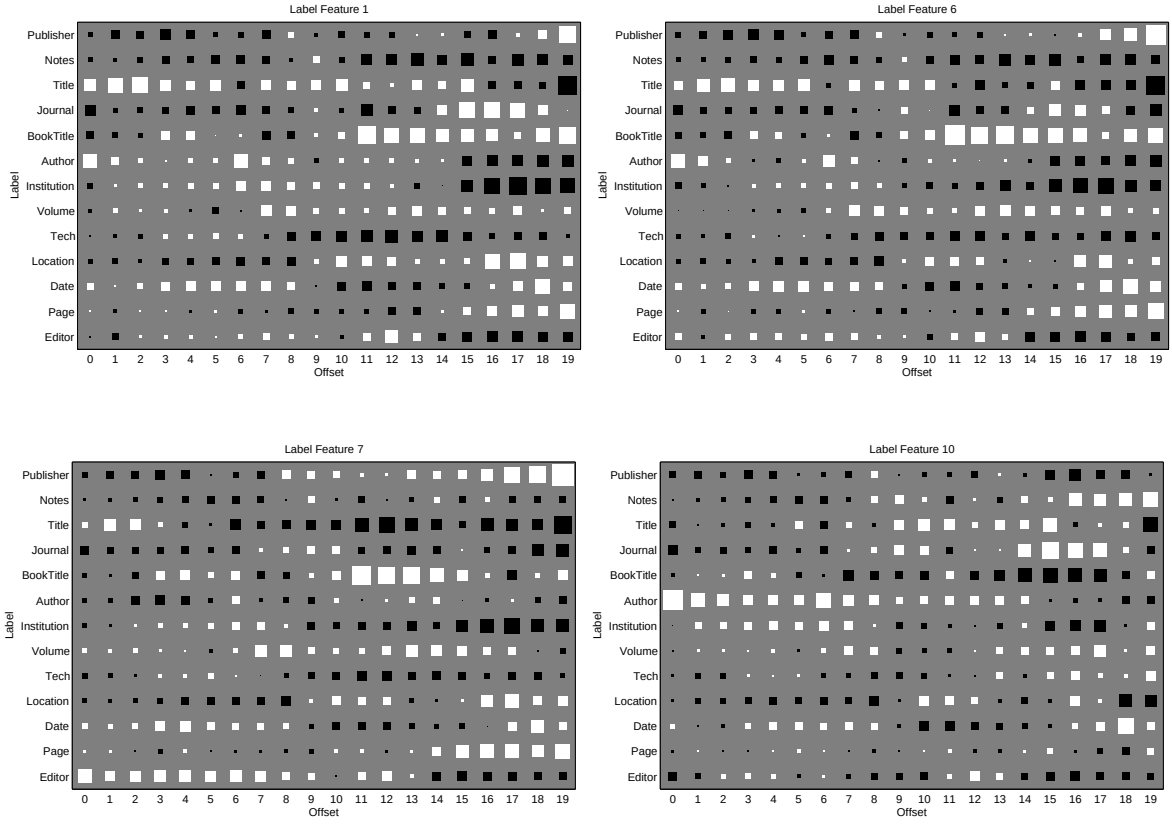


Figure A.4: Hinton diagrams of the weights learnt by selected label features of length 20 in RBM-LR(2).

	pu	no	ti	jo	bo	au	in	vo	te	lo	da	pg	ed
pu	45	0	2	2	3	9	1	0	0	3	0	0	0
no	0	16	11	5	6	1	0	0	5	1	0	0	0
ti	2	1	960	14	63	25	7	0	2	1	1	1	1
jo	4	1	8	101	48	3	4	1	1	0	0	0	0
bo	5	2	73	26	511	5	5	3	3	3	2	0	4
au	1	0	14	2	0	812	0	0	0	3	0	0	6
in	1	0	0	2	6	3	52	0	0	6	0	0	0
vo	0	0	0	0	1	0	0	74	1	0	0	3	0
te	0	1	5	1	2	3	1	0	25	0	0	2	1
lo	1	1	4	0	3	4	5	0	0	76	0	0	0
da	0	0	0	1	0	0	0	0	0	0	197	2	0
pg	0	0	0	1	3	0	0	1	0	0	0	124	0
ed	0	0	2	3	8	61	0	0	0	1	0	0	51

Table A.3: Confusion matrix for logistic regression on Cora references test data.

	pu	no	ti	jo	bo	au	in	vo	te	lo	da	pg	ed
pu	49	0	5	0	3	3	1	0	0	3	0	1	0
no	0	18	10	5	5	0	0	0	6	1	0	0	0
ti	1	0	1026	4	28	14	2	1	2	0	0	0	0
jo	2	0	10	129	25	0	3	1	1	0	0	0	0
bo	6	0	48	5	568	1	0	3	4	3	1	1	2
au	0	1	6	0	0	829	0	0	0	1	0	0	1
in	1	0	0	0	5	2	56	0	0	6	0	0	0
vo	0	0	0	1	0	0	0	75	0	0	0	3	0
te	0	0	9	1	0	1	1	0	25	0	1	2	1
lo	4	1	3	0	2	1	5	0	0	78	0	0	0
da	0	0	0	1	0	0	0	0	0	0	197	2	0
pg	0	0	1	1	1	0	0	1	0	0	1	124	0
ed	0	0	7	4	5	56	0	1	0	3	0	0	50

Table A.4: Confusion matrix for RBM-LR(2) on Cora references test data.

	pu	no	ti	jo	bo	au	in	vo	te	lo	da	pg	ed
pu	54	0	1	3	2	0	0	1	0	3	0	1	0
no	0	21	15	0	0	0	0	0	7	0	0	0	2
ti	0	4	1061	0	12	1	0	0	0	0	0	0	0
jo	0	0	0	158	11	0	0	2	0	0	0	0	0
bo	3	0	29	14	587	0	0	0	0	5	0	0	4
au	0	3	1	0	0	834	0	0	0	0	0	0	0
in	0	0	9	0	0	2	56	0	0	2	1	0	0
vo	0	0	0	1	1	0	0	76	0	0	1	0	0
te	0	0	0	2	10	0	0	1	26	0	1	1	0
lo	5	0	4	1	4	0	4	0	0	76	0	0	0
da	0	0	0	0	0	0	0	0	2	1	197	0	0
pg	0	2	2	0	0	0	0	0	0	0	0	125	0
ed	0	0	0	0	9	5	0	0	0	0	1	1	110

Table A.5: Confusion matrix for CRF on Cora references test data.

	pu	no	ti	jo	bo	au	in	vo	te	lo	da	pg	ed
pu	54	0	2	3	1	0	0	1	0	3	0	1	0
no	0	35	1	0	0	0	0	0	7	0	0	0	2
ti	0	4	1061	0	12	1	0	0	0	0	0	0	0
jo	0	0	0	157	12	0	0	2	0	0	0	0	0
bo	2	1	21	14	594	0	0	0	0	5	1	0	4
au	0	3	0	0	0	835	0	0	0	0	0	0	0
in	0	0	0	0	8	2	56	0	0	3	1	0	0
vo	0	0	0	1	1	0	0	76	0	0	1	0	0
te	0	0	1	2	9	0	0	1	26	0	1	1	0
lo	4	0	4	1	5	0	4	0	0	75	1	0	0
da	0	0	0	0	0	0	0	0	2	1	197	0	0
pg	0	2	2	0	0	0	0	0	0	0	0	125	0
ed	1	0	0	0	9	6	0	0	0	0	0	0	110

Table A.6: Confusion matrix for RBM-CRF(2) on Cora references test data.

Bibliography

[1] Yasemin Altun and Thomas Hofmann. Large margin methods for label sequence learning. In *Eighth European Conference on Speech Communication and Technology (EuroSpeech)*, 2003.

[2] Yasemin Altun, Thomas Hofmann, and Mark Johnson. Discriminative learning for label sequences via boosting. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 977–984. MIT Press, Cambridge, MA, 2003.

[3] Yoshua Bengio. Markovian Models for Sequential Data. *Neural Computing Surveys*, (2):129–162, 1999.

[4] Yoshua Bengio and Paolo Frasconi. An Input Output HMM Architecture. In Gerald Tesauro, David S. Touretzky, and Todd K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 427–433, Cambridge, MA, 1995. MIT Press.

[5] Léon Bottou. *Une Approche théorique de l'Apprentissage Connexionniste: Applications à la Reconnaissance de la Parole.* PhD thesis, Université de Paris XI, Orsay, France, 1991.

[6] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[7] Yoav Freund and David Haussler. Unsupervised Learning of Distributions on Binary Vectors Using Two Layer Networks. Technical Report UCSC-CRL-94-25, Baskin Center for Computer Engineering & Information Sciences, University of California, Santa Cruz, June 1994.

[8] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning Theory: Data Mining, Inference, and Prediction.* Springer, New York, 2001.

[9] Xuming He, Richard S. Zemel, and Miguel A. Carreira-Perpiñán. Multiscale Conditional Random Fields for Image Labeling. In *Computer Vision and Pattern Recognition (CVPR)*, 2004.

[10] Geoffrey E. Hinton. Products of experts. In *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN 99)*, volume 1, pages 1–6, 1999.

[11] Geoffrey E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. Technical Report GCNU TR 2000-004, Gatsby Computational Neuroscience Unit, University College London, 2000.

[12] Michael I. Jordan. An introduction to probabilistic graphical models. In preparation.

[13] Michael I. Jordan and Robert A. Jacobs. Hierarchical Mixtures of Experts and the EM Algorithm. Technical Report AIM-1440, 1993.

[14] Sham Kakade, Yee Whye Teh, and Sam Roweis. An Alternate Objective Function for Markovian Fields. In *International Conference on Machine Learning 19 (ICML 02)*, pages 275–282, 2002.

[15] John Lafferty, Stephen Della Pietra, and Vincent Della Pietra. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

[16] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*. Morgan Kaufmann, 2001.

[17] Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Sixth Conference on Natural Language Learning (CoNLL-2002)*, pages 49–55, 2002.

[18] Andrew McCallum. Efficiently Inducing Features of Conditional Random Fields. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2003.

[19] Andrew McCallum, Dayne Freitag, and Fernando Pereira. Maximum entropy markov models for information extraction and segmentation. In *International Conference on Machine Learning 18 (ICML 00)*, 2000.

[20] Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval Journal*, 3:127–163, 2000.

[21] Andrew McCallum and Ben Wellner. Conditional models of identity uncertainty with application to noun coreference. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 905–912. MIT Press, Cambridge, MA, 2005.

[22] Ian Murray and Zoubin Ghahramani. Bayesian learning in undirected graphical models. In *Uncertainty in Artificial Intelligence (UAI)*, 2004.

[23] R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental, sparse and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

[24] Fuchun Peng and Andrew McCallum. Accurate Information Extraction from Research Papers using Conditional Random Fields. In *Proceedings of HLT-NAACL 2004*. Association for Computational Linguistics, 2004.

[25] Yuan Qi, Martin Szummer, and Thomas P. Minka. Bayesian conditional random fields. In *Proceedings of Artificial Intelligence and Statistics (AISTATS)*, 2005.

[26] Lawrence R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Proceedings of the IEEE*, volume 77, pages 257–286, 1989.

[27] Sunita Sarawagi and William W. Cohen. Semi-markov conditional random fields for information extraction. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1185–1192. MIT Press, Cambridge, MA, 2005.

[28] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL 2003*, pages 213–220. Association for Computational Linguistics, 2003.

[29] P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 194–281. MIT Press, 1986.

[30] Charles Sutton and Andrew McCallum. Collective segmentation and labeling of distant entities in information extraction. In *ICML Workshop on Statistical Relational Learning*, 2004.

[31] Charles Sutton, Khashayar Rohanimanesh, and Andrew McCallum. Dynamic Conditional Random Fields: Factorized Probabilistic Models for Labeling and Segment-

ing Sequence Data. In *Twenty-First International Conference on Machine Learning*, pages 308–315. ACM Press, 2004.

[32] Hanna Wallach. Efficient training of conditional random fields. Master's thesis, Division of Informatics, University of Edinburgh, 2002.

[33] C. Zhu, R. H. Byrd, and J. Nocedal. L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN Routines for Large Scale Bound Constrained Optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.