Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data

John Lafferty[†*] LAFFERTY@CS.CMU.EDU
Andrew McCallum[*†] MCCALLUM@WHIZBANG.COM
Fernando Pereira[*‡] FPEREIRA@WHIZBANG.COM
[*]WhizBang! Labs–Research, 4616 Henry Street, Pittsburgh, PA 15213 USA
[†]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA
[‡]Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104 USA

Abstract

We present *conditional random fields*, a framework for building probabilistic models to segment and label sequence data. Conditional random fields offer several advantages over hidden Markov models and stochastic grammars for such tasks, including the ability to relax strong independence assumptions made in those models. Conditional random fields also avoid a fundamental limitation of maximum entropy Markov models (MEMMs) and other discriminative Markov models based on directed graphical models, which can be biased towards states with few successor states. We present iterative parameter estimation algorithms for conditional random fields and compare the performance of the resulting models to HMMs and MEMMs on synthetic and natural-language data.

1. Introduction

The need to segment and label sequences arises in many different problems in several scientific fields. Hidden Markov models (HMMs) and stochastic grammars are well understood and widely used probabilistic models for such problems. In computational biology, HMMs and stochastic grammars have been successfully used to align biological sequences, find sequences homologous to a known evolutionary family, and analyze RNA secondary structure (Durbin et al., 1998). In computational linguistics and computer science, HMMs and stochastic grammars have been applied to a wide variety of problems in text and speech processing, including topic segmentation, part-of-speech (POS) tagging, information extraction, and syntactic disambiguation (Manning & Schütze, 1999).

HMMs and stochastic grammars are generative models, assigning a joint probability to paired observation and label sequences; the parameters are typically trained to maximize the joint likelihood of training examples. To define a joint probability over observation and label sequences, a generative model needs to enumerate all possible observation sequences, typically requiring a representation in which observations are task-appropriate atomic entities, such as words or nucleotides. In particular, it is not practical to represent multiple interacting features or long-range dependencies of the observations, since the inference problem for such models is intractable.

This difficulty is one of the main motivations for looking at conditional models as an alternative. A conditional model specifies the probabilities of possible label sequences given an observation sequence. Therefore, it does not expend modeling effort on the observations, which at test time are fixed anyway. Furthermore, the conditional probability of the label sequence can depend on arbitrary, non-independent features of the observation sequence without forcing the model to account for the distribution of those dependencies. The chosen features may represent attributes at different levels of granularity of the same observations (for example, words and characters in English text), or aggregate properties of the observation sequence (for instance, text layout). The probability of a transition between labels may depend not only on the current observation, but also on past and future observations, if available. In contrast, generative models must make very strict independence assumptions on the observations, for instance conditional independence given the labels, to achieve tractability.

Maximum entropy Markov models (MEMMs) are conditional probabilistic sequence models that attain all of the above advantages (McCallum et al., 2000). In MEMMs, each source state[1] has a exponential model that takes the observation features as input, and outputs a distribution over possible next states. These exponential models are trained by an appropriate iterative scaling method in the

[1]Output labels are associated with states; it is possible for several states to have the same label, but for simplicity in the rest of this paper we assume a one-to-one correspondence.

maximum entropy framework. Previously published experimental results show MEMMs increasing recall and doubling precision relative to HMMs in a FAQ segmentation task.

MEMMs and other non-generative finite-state models based on next-state classifiers, such as discriminative Markov models (Bottou, 1991), share a weakness we call here the *label bias problem*: the transitions leaving a given state compete only against each other, rather than against all other transitions in the model. In probabilistic terms, transition scores are the conditional probabilities of possible next states given the current state and the observation sequence. This per-state normalization of transition scores implies a "conservation of score mass" (Bottou, 1991) whereby all the mass that arrives at a state must be distributed among the possible successor states. An observation can affect which destination states get the mass, but not how much total mass to pass on. This causes a bias toward states with fewer outgoing transitions. In the extreme case, a state with a single outgoing transition effectively ignores the observation. In those cases, unlike in HMMs, Viterbi decoding cannot downgrade a branch based on observations after the branch point, and models with state-transition structures that have sparsely connected chains of states are not properly handled. The Markovian assumptions in MEMMs and similar state-conditional models insulate decisions at one state from future decisions in a way that does not match the actual dependencies between consecutive states.

This paper introduces *conditional random fields* (CRFs), a sequence modeling framework that has all the advantages of MEMMs but also solves the label bias problem in a principled way. The critical difference between CRFs and MEMMs is that a MEMM uses per-state exponential models for the conditional probabilities of next states given the current state, while a CRF has a single exponential model for the joint probability of the entire sequence of labels given the observation sequence. Therefore, the weights of different features at different states can be traded off against each other.

We can also think of a CRF as a finite state model with unnormalized transition probabilities. However, unlike some other weighted finite-state approaches (LeCun et al., 1998), CRFs assign a well-defined probability distribution over possible labelings, trained by maximum likelihood or MAP estimation. Furthermore, the loss function is convex,[2] guaranteeing convergence to the global optimum. CRFs also generalize easily to analogues of stochastic context-free grammars that would be useful in such problems as RNA secondary structure prediction and natural language processing.

[2] In the case of fully observable states, as we are discussing here; if several states have the same label, the usual local maxima of Baum-Welch arise.

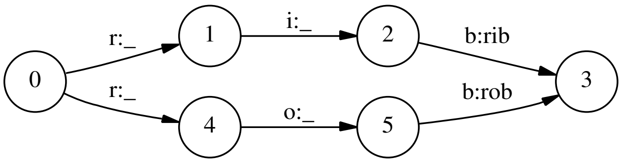

Figure 1. Label bias example, after (Bottou, 1991). For conciseness, we place observation-label pairs $o : l$ on transitions rather than states; the symbol '_' represents the null output label.

We present the model, describe two training procedures and sketch a proof of convergence. We also give experimental results on synthetic data showing that CRFs solve the classical version of the label bias problem, and, more significantly, that CRFs perform better than HMMs and MEMMs when the true data distribution has higher-order dependencies than the model, as is often the case in practice. Finally, we confirm these results as well as the claimed advantages of conditional models by evaluating HMMs, MEMMs and CRFs with identical state structure on a part-of-speech tagging task.

2. The Label Bias Problem

Classical probabilistic automata (Paz, 1971), discriminative Markov models (Bottou, 1991), maximum entropy taggers (Ratnaparkhi, 1996), and MEMMs, as well as non-probabilistic sequence tagging and segmentation models with independently trained next-state classifiers (Punyakanok & Roth, 2001) are all potential victims of the label bias problem.

For example, Figure 1 represents a simple finite-state model designed to distinguish between the two words rib and rob. Suppose that the observation sequence is r i b. In the first time step, r matches both transitions from the start state, so the probability mass gets distributed roughly equally among those two transitions. Next we observe i. Both states 1 and 4 have only one outgoing transition. State 1 has seen this observation often in training, state 4 has almost never seen this observation; but like state 1, state 4 has no choice but to pass all its mass to its single outgoing transition, since it is not generating the observation, only conditioning on it. Thus, states with a single outgoing transition effectively ignore their observations. More generally, states with low-entropy next state distributions will take little notice of observations. Returning to the example, the top path and the bottom path will be about equally likely, independently of the observation sequence. If one of the two words is slightly more common in the training set, the transitions out of the start state will slightly prefer its corresponding transition, and that word's state sequence will always win. This behavior is demonstrated experimentally in Section 5.

Léon Bottou (1991) discussed two solutions for the label bias problem. One is to change the state-transition structure of the model. In the above example we could collapse states 1 and 4, and delay the branching until we get a discriminating observation. This operation is a special case of determinization (Mohri, 1997), but determinization of weighted finite-state machines is not always possible, and even when possible, it may lead to combinatorial explosion. The other solution mentioned is to start with a fully-connected model and let the training procedure figure out a good structure. But that would preclude the use of prior structural knowledge that has proven so valuable in information extraction tasks (Freitag & McCallum, 2000).

Proper solutions require models that account for whole state sequences at once by letting some transitions "vote" more strongly than others depending on the corresponding observations. This implies that score mass will not be conserved, but instead individual transitions can "amplify" or "dampen" the mass they receive. In the above example, the transitions from the start state would have a very weak effect on path score, while the transitions from states 1 and 4 would have much stronger effects, amplifying or damping depending on the actual observation, and a proportionally higher contribution to the selection of the Viterbi path.[3]

In the related work section we discuss other heuristic model classes that account for state sequences globally rather than locally. To the best of our knowledge, CRFs are the only model class that does this in a purely probabilistic setting, with guaranteed global maximum likelihood convergence.

3. Conditional Random Fields

In what follows, $\mathbf{X}$ is a random variable over data sequences to be labeled, and $\mathbf{Y}$ is a random variable over corresponding label sequences. All components $\mathbf{Y}_i$ of $\mathbf{Y}$ are assumed to range over a finite label alphabet $\mathcal{Y}$. For example, $\mathbf{X}$ might range over natural language sentences and $\mathbf{Y}$ range over part-of-speech taggings of those sentences, with $\mathcal{Y}$ the set of possible part-of-speech tags. The random variables $\mathbf{X}$ and $\mathbf{Y}$ are jointly distributed, but in a discriminative framework we construct a conditional model $p(\mathbf{Y} \mid \mathbf{X})$ from paired observation and label sequences, and do not explicitly model the marginal $p(\mathbf{X})$.

Definition. *Let $G = (V, E)$ be a graph such that $\mathbf{Y} = (\mathbf{Y}_v)_{v \in V}$, so that $\mathbf{Y}$ is indexed by the vertices of G. Then $(\mathbf{X}, \mathbf{Y})$ is a conditional random field in case, when conditioned on $\mathbf{X}$, the random variables $\mathbf{Y}_v$ obey the Markov property with respect to the graph: $p(\mathbf{Y}_v \mid \mathbf{X}, \mathbf{Y}_w, w \neq v) = p(\mathbf{Y}_v \mid \mathbf{X}, \mathbf{Y}_w, w \sim v)$, where $w \sim v$ means that w and v are neighbors in G.*

Thus, a CRF is a random field globally conditioned on the observation $\mathbf{X}$. Throughout the paper we tacitly assume that the graph G is fixed. In the simplest and most important example for modeling sequences, G is a simple chain or line: $G = (V = \{1, 2, \ldots m\}, E = \{(i, i+1)\})$. $\mathbf{X}$ may also have a natural graph structure; yet in general it is not necessary to assume that $\mathbf{X}$ and $\mathbf{Y}$ have the same graphical structure, or even that $\mathbf{X}$ has any graphical structure at all. However, in this paper we will be most concerned with sequences $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n)$ and $\mathbf{Y} = (\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_n)$.

If the graph $G = (V, E)$ of $\mathbf{Y}$ is a tree (of which a chain is the simplest example), its cliques are the edges and vertices. Therefore, by the fundamental theorem of random fields (Hammersley & Clifford, 1971), the joint distribution over the label sequence $\mathbf{Y}$ given $\mathbf{X}$ has the form

$$p_\theta(\mathbf{y} \mid \mathbf{x}) \propto \tag{1}$$
$$\exp\left(\sum_{e \in E, k} \lambda_k\, f_k(e, \mathbf{y}|_e, \mathbf{x}) + \sum_{v \in V, k} \mu_k\, g_k(v, \mathbf{y}|_v, \mathbf{x})\right),$$

where $\mathbf{x}$ is a data sequence, $\mathbf{y}$ a label sequence, and $\mathbf{y}|_S$ is the set of components of $\mathbf{y}$ associated with the vertices in subgraph S.

We assume that the *features* f_k and g_k are given and fixed. For example, a Boolean vertex feature g_k might be true if the word $\mathbf{X}_i$ is upper case and the tag $\mathbf{Y}_i$ is "proper noun."

The parameter estimation problem is to determine the parameters $\theta = (\lambda_1, \lambda_2, \ldots; \mu_1, \mu_2, \ldots)$ from training data $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^N$ with empirical distribution $\widetilde{p}(\mathbf{x}, \mathbf{y})$. In Section 4 we describe an iterative scaling algorithm that maximizes the log-likelihood objective function $\mathcal{O}(\theta)$:

$$\begin{aligned} \mathcal{O}(\theta) &= \sum_{i=1}^N \log p_\theta(\mathbf{y}^{(i)} \mid \mathbf{x}^{(i)}) \\ &\propto \sum_{\mathbf{x},\mathbf{y}} \widetilde{p}(\mathbf{x}, \mathbf{y})\, \log p_\theta(\mathbf{y} \mid \mathbf{x}) . \end{aligned}$$

As a particular case, we can construct an HMM-like CRF by defining one feature for each state pair (y', y), and one feature for each state-observation pair (y, x):

$$\begin{aligned} f_{y',y}(<u,v>, \mathbf{y}|_{<u,v>}, \mathbf{x}) &= \delta(\mathbf{y}_u, y')\, \delta(\mathbf{y}_v, y) \\ g_{y,x}(v, \mathbf{y}|_v, \mathbf{x}) &= \delta(\mathbf{y}_v, y)\, \delta(\mathbf{x}_v, x) . \end{aligned}$$

The corresponding parameters $\lambda_{y',y}$ and $\mu_{y,x}$ play a similar role to the (logarithms of the) usual HMM parameters $p(y' \mid y)$ and $p(x|y)$. Boltzmann chain models (Saul & Jordan, 1996; MacKay, 1996) have a similar form but use a single normalization constant to yield a joint distribution, whereas CRFs use the observation-dependent normalization $Z(\mathbf{x})$ for conditional distributions.

Although it encompasses HMM-like models, the class of conditional random fields is much more expressive, because it allows arbitrary dependencies on the observation

[3]Weighted determinization and minimization techniques shift transition weights while preserving overall path weight (Mohri, 2000); their connection to this discussion deserves further study.

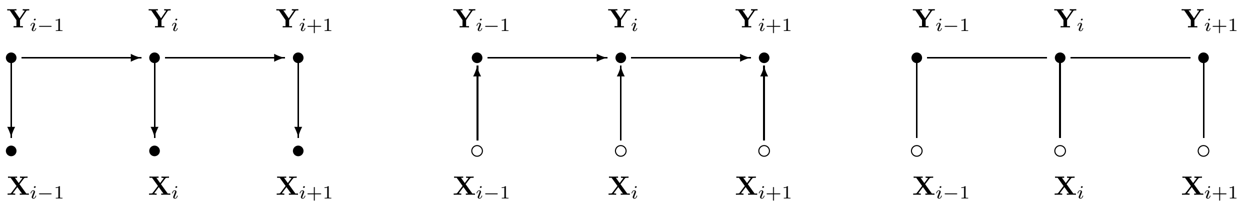

Figure 2. Graphical structures of simple HMMs (left), MEMMs (center), and the chain-structured case of CRFs (right) for sequences. An open circle indicates that the variable is not generated by the model.

sequence. In addition, the features do not need to specify completely a state or observation, so one might expect that the model can be estimated from less training data. Another attractive property is the convexity of the loss function; indeed, CRFs share all of the convexity properties of general maximum entropy models.

For the remainder of the paper we assume that the dependencies of $\mathbf{Y}$, conditioned on $\mathbf{X}$, form a chain. To simplify some expressions, we add special start and stop states $\mathbf{Y}_0 = \mathsf{start}$ and $\mathbf{Y}_{n+1} = \mathsf{stop}$. Thus, we will be using the graphical structure shown in Figure 2. For a chain structure, the conditional probability of a label sequence can be expressed concisely in matrix form, which will be useful in describing the parameter estimation and inference algorithms in Section 4. Suppose that $p_\theta(\mathbf{Y} \,|\, \mathbf{X})$ is a CRF given by (1). For each position i in the observation sequence $\mathbf{x}$, we define the $|\mathcal{Y}| \times |\mathcal{Y}|$ matrix random variable $M_i(\mathbf{x}) = [M_i(y', y \,|\, \mathbf{x})]$ by

$$\begin{aligned} M_i(y', y \,|\, \mathbf{x}) &= \exp\left(\Lambda_i(y', y \,|\, \mathbf{x})\right) \\ \Lambda_i(y', y \,|\, \mathbf{x}) &= \textstyle\sum_k \lambda_k\, f_k(e_i, \mathbf{Y}|_{e_i} = (y', y), \mathbf{x}) + \\ &\quad \textstyle\sum_k \mu_k\, g_k(v_i, \mathbf{Y}|_{v_i} = y, \mathbf{x})\,, \end{aligned}$$

where e_i is the edge with labels $(\mathbf{Y}_{i-1}, \mathbf{Y}_i)$ and v_i is the vertex with label $\mathbf{Y}_i$. In contrast to generative models, conditional models like CRFs do not need to enumerate over all possible observation sequences $\mathbf{x}$, and therefore these matrices can be computed directly as needed from a given training or test observation sequence $\mathbf{x}$ and the parameter vector θ. Then the normalization (partition function) $Z_\theta(\mathbf{x})$ is the $(\mathsf{start}, \mathsf{stop})$ entry of the product of these matrices:

$$Z_\theta(\mathbf{x}) = \left(M_1(\mathbf{x})\, M_2(\mathbf{x}) \cdots M_{n+1}(\mathbf{x})\right)_{\mathsf{start},\mathsf{stop}}\,.$$

Using this notation, the conditional probability of a label sequence $\mathbf{y}$ is written as

$$p_\theta(\mathbf{y} \,|\, \mathbf{x}) = \frac{\prod_{i=1}^{n+1} M_i(\mathbf{y}_{i-1}, \mathbf{y}_i \,|\, \mathbf{x})}{\left(\prod_{i=1}^{n+1} M_i(\mathbf{x})\right)_{\mathsf{start},\mathsf{stop}}}\,,$$

where $\mathbf{y}_0 = \mathsf{start}$ and $\mathbf{y}_{n+1} = \mathsf{stop}$.

4. Parameter Estimation for CRFs

We now describe two iterative scaling algorithms to find the parameter vector θ that maximizes the log-likelihood of the training data. Both algorithms are based on the improved iterative scaling (IIS) algorithm of Della Pietra et al. (1997); the proof technique based on auxiliary functions can be extended to show convergence of the algorithms for CRFs.

Iterative scaling algorithms update the weights as $\lambda_k \leftarrow \lambda_k + \delta\lambda_k$ and $\mu_k \leftarrow \mu_k + \delta\mu_k$ for appropriately chosen $\delta\lambda_k$ and $\delta\mu_k$. In particular, the IIS update $\delta\lambda_k$ for an edge feature f_k is the solution of

$$\begin{aligned} \widetilde{E}[f_k] &\stackrel{\text{def}}{=} \sum_{\mathbf{x},\mathbf{y}} \widetilde{p}(\mathbf{x}, \mathbf{y}) \sum_{i=1}^{n+1} f_k(e_i, \mathbf{y}|_{e_i}, \mathbf{x}) \\ &= \sum_{\mathbf{x},\mathbf{y}} \widetilde{p}(\mathbf{x})\, p(\mathbf{y} \,|\, \mathbf{x}) \sum_{i=1}^{n+1} f_k(e_i, \mathbf{y}|_{e_i}, \mathbf{x})\, e^{\,\delta\lambda_k T(\mathbf{x},\mathbf{y})}\,. \end{aligned}$$

where $T(\mathbf{x}, \mathbf{y})$ is the *total feature count*

$$T(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} \sum_{i,k} f_k(e_i, \mathbf{y}|_{e_i}, \mathbf{x}) + \sum_{i,k} g_k(v_i, \mathbf{y}|_{v_i}, \mathbf{x})\,.$$

The equations for vertex feature updates $\delta\mu_k$ have similar form.

However, efficiently computing the exponential sums on the right-hand sides of these equations is problematic, because $T(\mathbf{x}, \mathbf{y})$ is a global property of $(\mathbf{x}, \mathbf{y})$, and dynamic programming will sum over sequences with potentially varying T. To deal with this, the first algorithm, Algorithm S, uses a "slack feature." The second, Algorithm T, keeps track of partial T totals.

For Algorithm S, we define the *slack feature* by

$$\begin{aligned} &s(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} \\ &\quad S - \sum_i \sum_k f_k(e_i, \mathbf{y}|_{e_i}, \mathbf{x}) - \sum_i \sum_k g_k(v_i, \mathbf{y}|_{v_i}, \mathbf{x})\,, \end{aligned}$$

where S is a constant chosen so that $s(\mathbf{x}^{(i)}, \mathbf{y}) \geq 0$ for all $\mathbf{y}$ and all observation vectors $\mathbf{x}^{(i)}$ in the training set, thus making $T(\mathbf{x}, \mathbf{y}) = S$. Feature s is "global," that is, it does not correspond to any particular edge or vertex.

For each index $i = 0, \ldots, n+1$ we now define the *forward vectors* $\alpha_i(\mathbf{x})$ with base case

$$\alpha_0(y \,|\, \mathbf{x}) = \begin{cases} 1 & \text{if } y = \mathsf{start} \\ 0 & \text{otherwise} \end{cases}$$

and recurrence

$$\alpha_i(\mathbf{x}) \;=\; \alpha_{i-1}(\mathbf{x})\, M_i(\mathbf{x}) \,.$$

Similarly, the *backward vectors* $\beta_i(\mathbf{x})$ are defined by

$$\beta_{n+1}(y\,|\,\mathbf{x}) \;=\; \begin{cases} 1 & \text{if } y = \mathsf{stop} \\ 0 & \text{otherwise} \end{cases}$$

and

$$\beta_i(\mathbf{x})^\top \;=\; M_{i+1}(\mathbf{x})\, \beta_{i+1}(\mathbf{x}) \,.$$

With these definitions, the update equations are

$$\delta\lambda_k \;=\; \frac{1}{S} \log \frac{\widetilde{E} f_k}{E f_k} \,, \quad \delta\mu_k \;=\; \frac{1}{S} \log \frac{\widetilde{E} g_k}{E g_k} \,,$$

where

$$\begin{aligned} E f_k \;&=\; \sum_{\mathbf{x}} \widetilde{p}(\mathbf{x}) \sum_{i=1}^{n+1} \sum_{y',y} f_k(e_i, \mathbf{y}|_{e_i} = (y',y), \mathbf{x}) \;\times \\ &\qquad \frac{\alpha_{i-1}(y'\,|\,\mathbf{x})\, M_i(y',y\,|\,\mathbf{x})\, \beta_i(y\,|\,\mathbf{x})}{Z_\theta(\mathbf{x})} \\ E g_k \;&=\; \sum_{\mathbf{x}} \widetilde{p}(\mathbf{x}) \sum_{i=1}^{n} \sum_{y} g_k(v_i, \mathbf{y}|_{v_i} = y, \mathbf{x}) \;\times \\ &\qquad \frac{\alpha_i(y\,|\,\mathbf{x})\, \beta_i(y\,|\,\mathbf{x})}{Z_\theta(\mathbf{x})} \,. \end{aligned}$$

The factors involving the forward and backward vectors in the above equations have the same meaning as for standard hidden Markov models. For example,

$$p_\theta(\mathbf{Y}_i = y\,|\,\mathbf{x}) \;=\; \frac{\alpha_i(y\,|\,\mathbf{x})\, \beta_i(y\,|\,\mathbf{x})}{Z_\theta(\mathbf{x})}$$

is the marginal probability of label $\mathbf{Y}_i = y$ given that the observation sequence is $\mathbf{x}$. This algorithm is closely related to the algorithm of Darroch and Ratcliff (1972), and MART algorithms used in image reconstruction.

The constant S in Algorithm S can be quite large, since in practice it is proportional to the length of the longest training observation sequence. As a result, the algorithm may converge slowly, taking very small steps toward the maximum in each iteration. If the length of the observations $\mathbf{x}^{(i)}$ and the number of active features varies greatly, a faster-converging algorithm can be obtained by keeping track of feature totals for each observation sequence separately.

Let $T(\mathbf{x}) \stackrel{\text{def}}{=} \max_{\mathbf{y}} T(\mathbf{x},\mathbf{y})$. Algorithm T accumulates feature expectations into counters indexed by $T(\mathbf{x})$. More specifically, we use the forward-backward recurrences just introduced to compute the expectations $a_{k,t}$ of feature f_k and $b_{k,t}$ of feature g_k given that $T(\mathbf{x}) = t$. Then our parameter updates are $\delta\lambda_k = \log \beta_k$ and $\delta\mu_k = \log \gamma_k$, where β_k and γ_k are the unique positive roots to the following polynomial equations

$$\sum_{i=0}^{T_{\max}} a_{k,t}\, \beta_k^t \;=\; \widetilde{E} f_k \,, \quad \sum_{i=0}^{T_{\max}} b_{k,t}\, \gamma_k^t \;=\; \widetilde{E} g_k \,, \tag{2}$$

which can be easily computed by Newton's method.

A single iteration of Algorithm S and Algorithm T has roughly the same time and space complexity as the well known Baum-Welch algorithm for HMMs. To prove convergence of our algorithms, we can derive an auxiliary function to bound the change in likelihood from below; this method is developed in detail by Della Pietra et al. (1997). The full proof is somewhat detailed; however, here we give an idea of how to derive the auxiliary function. To simplify notation, we assume only edge features f_k with parameters λ_k.

Given two parameter settings $\theta = (\lambda_1, \lambda_2, \ldots)$ and $\theta' = (\lambda_1 + \delta\lambda_1, \lambda_2 + \delta\lambda_2, \ldots)$, we bound from below the change in the objective function with an *auxiliary function* $\mathcal{A}(\theta', \theta)$ as follows

$$\begin{aligned} \mathcal{O}(\theta') - \mathcal{O}(\theta) \;&=\; \sum_{\mathbf{x},\mathbf{y}} \widetilde{p}(\mathbf{x},\mathbf{y}) \log \frac{p_{\theta'}(\mathbf{y}\,|\,\mathbf{x})}{p_\theta(\mathbf{y}\,|\,\mathbf{x})} \\ &=\; (\theta' - \theta) \cdot \widetilde{E} f - \sum_{\mathbf{x}} \widetilde{p}(\mathbf{x}) \log \frac{Z_{\theta'}(\mathbf{x})}{Z_\theta(\mathbf{x})} \\ &\geq\; (\theta' - \theta) \cdot \widetilde{E} f - \sum_{\mathbf{x}} \widetilde{p}(\mathbf{x}) \frac{Z_{\theta'}(\mathbf{x})}{Z_\theta(\mathbf{x})} \\ &=\; \delta\lambda \cdot \widetilde{E} f - \sum_{\mathbf{x}} \widetilde{p}(\mathbf{x}) \sum_{\mathbf{y}} p_\theta(\mathbf{y}\,|\,\mathbf{x})\, e^{\delta\lambda \cdot f(\mathbf{x},\mathbf{y})} \\ &\geq\; \delta\lambda \cdot \widetilde{E} f - \sum_{\mathbf{x},\mathbf{y},k} \widetilde{p}(\mathbf{x})\, p_\theta(\mathbf{y}\,|\,\mathbf{x})\, \frac{f_k(\mathbf{x},\mathbf{y})}{T(\mathbf{x})} e^{\delta\lambda_k T(\mathbf{x})} \\ &\stackrel{\text{def}}{=}\; \mathcal{A}(\theta', \theta) \end{aligned}$$

where the inequalities follow from the convexity of $-\log$ and $\exp$. Differentiating $\mathcal{A}$ with respect to $\delta\lambda_k$ and setting the result to zero yields equation (2).

5. Experiments

We first discuss two sets of experiments with synthetic data that highlight the differences between CRFs and MEMMs. The first experiments are a direct verification of the label bias problem discussed in Section 2. In the second set of experiments, we generate synthetic data using randomly chosen hidden Markov models, each of which is a mixture of a first-order and second-order model. Competing *first-order* models are then trained and compared on test data. As the data becomes more second-order, the test error rates of the trained models increase. This experiment corresponds to the common modeling practice of approximating complex local and long-range dependencies, as occur in natural data, by small-order Markov models. Our

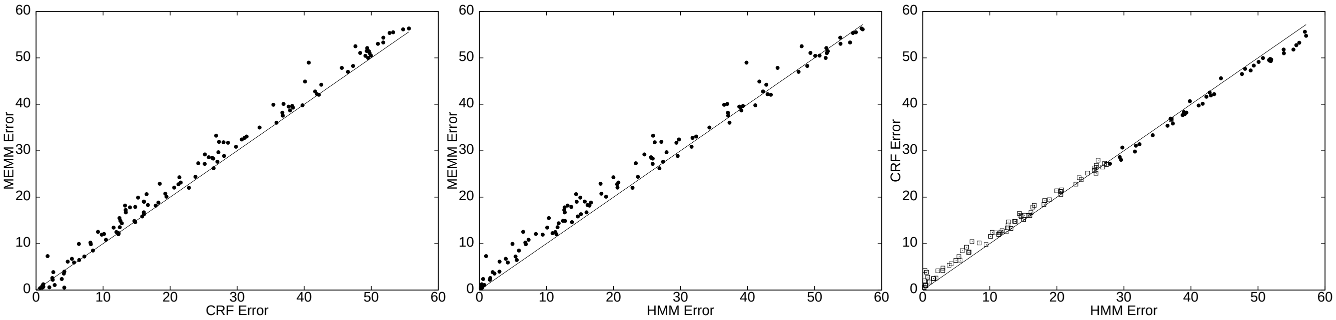

Figure 3. Plots of 2×2 error rates for HMMs, CRFs, and MEMMs on randomly generated synthetic data sets, as described in Section 5.2. As the data becomes "more second order," the error rates of the test models increase. As shown in the left plot, the CRF typically significantly outperforms the MEMM. The center plot shows that the HMM outperforms the MEMM. In the right plot, each open square represents a data set with $\alpha < \frac{1}{2}$, and a solid circle indicates a data set with $\alpha \geq \frac{1}{2}$. The plot shows that when the data is mostly second order ($\alpha \geq \frac{1}{2}$), the discriminatively trained CRF typically outperforms the HMM. These experiments are not designed to demonstrate the advantages of the additional representational power of CRFs and MEMMs relative to HMMs.

results clearly indicate that even when the models are parameterized in exactly the same way, CRFs are more robust to inaccurate modeling assumptions than MEMMs or HMMs, and resolve the label bias problem, which affects the performance of MEMMs. To avoid confusion of different effects, the MEMMs and CRFs in these experiments *do not* use overlapping features of the observations. Finally, in a set of POS tagging experiments, we confirm the advantage of CRFs over MEMMs. We also show that the addition of overlapping features to CRFs and MEMMs allows them to perform much better than HMMs, as already shown for MEMMs by McCallum et al. (2000).

5.1 Modeling label bias

We generate data from a simple HMM which encodes a noisy version of the finite-state network in Figure 1. Each state emits its designated symbol with probability $29/32$ and any of the other symbols with probability $1/32$. We train both an MEMM and a CRF with the same topologies on the data generated by the HMM. The observation features are simply the identity of the observation symbols. In a typical run using $2,000$ training and 500 test samples, trained to convergence of the iterative scaling algorithm, the CRF error is 4.6% while the MEMM error is 42%, showing that the MEMM fails to discriminate between the two branches.

5.2 Modeling mixed-order sources

For these results, we use five labels, a-e ($|\mathcal{Y}| = 5$), and 26 observation values, A-Z ($|\mathcal{X}| = 26$); however, the results were qualitatively the same over a range of sizes for $\mathcal{Y}$ and $\mathcal{X}$. We generate data from a mixed-order HMM with state transition probabilities given by $p_\alpha(\mathbf{y}_i \mid \mathbf{y}_{i-1}, \mathbf{y}_{i-2}) = \alpha\, p_2(\mathbf{y}_i \mid \mathbf{y}_{i-1}, \mathbf{y}_{i-2}) + (1-\alpha)\, p_1(\mathbf{y}_i \mid \mathbf{y}_{i-1})$ and, similarly, emission probabilities given by $p_\alpha(\mathbf{x}_i \mid \mathbf{y}_i, \mathbf{x}_{i-1}) = \alpha\, p_2(\mathbf{x}_i \mid \mathbf{y}_i, \mathbf{x}_{i-1}) + (1-\alpha)\, p_1(\mathbf{x}_i \mid \mathbf{y}_i)$. Thus, for $\alpha = 0$ we have a standard first-order HMM. In order to limit the size of the Bayes error rate for the resulting models, the conditional probability tables p_α are constrained to be sparse. In particular, $p_\alpha(\cdot \mid y, y')$ can have at most two nonzero entries, for each y, y', and $p_\alpha(\cdot \mid y, x')$ can have at most three nonzero entries for each y, x'. For each randomly generated model, a sample of 1,000 sequences of length 25 is generated for training and testing.

On each randomly generated training set, a CRF is trained using Algorithm S. (Note that since the length of the sequences and number of active features is constant, Algorithms S and T are identical.) The algorithm is fairly slow to converge, typically taking approximately 500 iterations for the model to stabilize. On the 500 MHz Pentium PC used in our experiments, each iteration takes approximately 0.2 seconds. On the same data an MEMM is trained using iterative scaling, which does not require forward-backward calculations, and is thus more efficient. The MEMM training converges more quickly, stabilizing after approximately 100 iterations. For each model, the Viterbi algorithm is used to label a test set; the experimental results do not significantly change when using forward-backward decoding to minimize the per-symbol error rate.

The results of several runs are presented in Figure 3. Each plot compares two classes of models, with each point indicating the error rate for a single test set. As α increases, the error rates generally increase, as the first-order models fail to fit the second-order data. The figure compares models parameterized as μ_y, $\lambda_{y',y}$, and $\lambda_{y',y,x}$; results for models parameterized as μ_y, $\lambda_{y',y}$, and $\mu_{y,x}$ are qualitatively the same. As shown in the first graph, the CRF generally outperforms the MEMM, often by a wide margin of 10%–20% relative error. (The points for very small error rate, with $\alpha < 0.01$, where the MEMM does better than the CRF, are suspected to be the result of an insufficient number of training iterations for the CRF.)

model	*error*	*oov error*
HMM	5.69%	45.99%
MEMM	6.37%	54.61%
CRF	5.55%	48.05%
MEMM$^+$	4.81%	26.99%
CRF$^+$	4.27%	23.76%

$^+$Using spelling features

Figure 4. Per-word error rates for POS tagging on the Penn treebank, using first-order models trained on 50% of the 1.1 million word corpus. The oov rate is 5.45%.

5.3 POS tagging experiments

To confirm our synthetic data results, we also compared HMMs, MEMMs and CRFs on Penn treebank POS tagging, where each word in a given input sentence must be labeled with one of 45 syntactic tags.

We carried out two sets of experiments with this natural language data. First, we trained first-order HMM, MEMM, and CRF models as in the synthetic data experiments, introducing parameters $\mu_{y,x}$ for each tag-word pair and $\lambda_{y',y}$ for each tag-tag pair in the training set. The results are consistent with what is observed on synthetic data: the HMM outperforms the MEMM, as a consequence of the label bias problem, while the CRF outperforms the HMM. The error rates for training runs using a 50%-50% train-test split are shown in Figure 5.3; the results are qualitatively similar for other splits of the data. The error rates on out-of-vocabulary (oov) words, which are not observed in the training set, are reported separately.

In the second set of experiments, we take advantage of the power of conditional models by adding a small set of orthographic features: whether a spelling begins with a number or upper case letter, whether it contains a hyphen, and whether it ends in one of the following suffixes: -ing, -ogy, -ed, -s, -ly, -ion, -tion, -ity, -ies. Here we find, as expected, that both the MEMM and the CRF benefit significantly from the use of these features, with the overall error rate reduced by around 25%, and the out-of-vocabulary error rate reduced by around 50%.

One usually starts training from the all zero parameter vector, corresponding to the uniform distribution. However, for these datasets, CRF training with that initialization is much slower than MEMM training. Fortunately, we can use the optimal MEMM parameter vector as a starting point for training the corresponding CRF. In Figure 5.3, MEMM$^+$ was trained to convergence in around 100 iterations. Its parameters were then used to initialize the training of CRF$^+$, which converged in 1,000 iterations. In contrast, training of the same CRF from the uniform distribution had not converged even after 2,000 iterations.

6. Further Aspects of CRFs

Many further aspects of CRFs are attractive for applications and deserve further study. In this section we briefly mention just two.

Conditional random fields can be trained using the exponential loss objective function used by the AdaBoost algorithm (Freund & Schapire, 1997). Typically, boosting is applied to classification problems with a small, fixed number of classes; applications of boosting to sequence labeling have treated each label as a separate classification problem (Abney et al., 1999). However, it is possible to apply the parallel update algorithm of Collins et al. (2000) to optimize the per-sequence exponential loss. This requires a forward-backward algorithm to compute efficiently certain feature expectations, along the lines of Algorithm T, except that each feature requires a separate set of forward and backward accumulators.

Another attractive aspect of CRFs is that one can implement efficient feature selection and feature induction algorithms for them. That is, rather than specifying in advance which features of $(\mathbf{X}, \mathbf{Y})$ to use, we could start from feature-generating rules and evaluate the benefit of generated features automatically on data. In particular, the feature induction algorithms presented in Della Pietra et al. (1997) can be adapted to fit the dynamic programming techniques of conditional random fields.

7. Related Work and Conclusions

As far as we know, the present work is the first to combine the benefits of conditional models with the global normalization of random field models. Other applications of exponential models in sequence modeling have either attempted to build generative models (Rosenfeld, 1997), which involve a hard normalization problem, or adopted local conditional models (Berger et al., 1996; Ratnaparkhi, 1996; McCallum et al., 2000) that may suffer from label bias.

Non-probabilistic local decision models have also been widely used in segmentation and tagging (Brill, 1995; Roth, 1998; Abney et al., 1999). Because of the computational complexity of global training, these models are only trained to minimize the error of individual label decisions assuming that neighboring labels are correctly chosen. Label bias would be expected to be a problem here too.

An alternative approach to discriminative modeling of sequence labeling is to use a permissive generative model, which can only model local dependencies, to produce a list of candidates, and then use a more global discriminative model to rerank those candidates. This approach is standard in large-vocabulary speech recognition (Schwartz & Austin, 1993), and has also been proposed for parsing (Collins, 2000). However, these methods fail when the correct output is pruned away in the first pass.

Closest to our proposal are gradient-descent methods that adjust the parameters of all of the local classifiers to minimize a smooth loss function (e.g., quadratic loss) combining loss terms for each label. If state dependencies are local, this can be done efficiently with dynamic programming (LeCun et al., 1998). Such methods should alleviate label bias. However, their loss function is not convex, so they may get stuck in local minima.

Conditional random fields offer a unique combination of properties: discriminatively trained models for sequence segmentation and labeling; combination of arbitrary, overlapping and agglomerative observation features from both the past and future; efficient training and decoding based on dynamic programming; and parameter estimation guaranteed to find the global optimum. Their main current limitation is the slow convergence of the training algorithm relative to MEMMs, let alone to HMMs, for which training on fully observed data is very efficient. In future work, we plan to investigate alternative training methods such as the update methods of Collins et al. (2000) and refinements on using a MEMM as starting point as we did in some of our experiments. More general tree-structured random fields, feature induction methods, and further natural data evaluations will also be investigated.

Acknowledgments

We thank Yoshua Bengio, Léon Bottou, Michael Collins and Yann LeCun for alerting us to what we call here the label bias problem. We also thank Andrew Ng and Sebastian Thrun for discussions related to this work.

References

Abney, S., Schapire, R. E., & Singer, Y. (1999). Boosting applied to tagging and PP attachment. *Proc. EMNLP-VLC*. New Brunswick, New Jersey: Association for Computational Linguistics.

Berger, A. L., Della Pietra, S. A., & Della Pietra, V. J. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, *22*.

Bottou, L. (1991). *Une approche théorique de l'apprentissage connexionniste: Applications à la reconnaissance de la parole*. Doctoral dissertation, Université de Paris XI.

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: a case study in part of speech tagging. *Computational Linguistics*, *21*, 543–565.

Collins, M. (2000). Discriminative reranking for natural language parsing. *Proc. ICML 2000*. Stanford, California.

Collins, M., Schapire, R., & Singer, Y. (2000). Logistic regression, AdaBoost, and Bregman distances. *Proc. 13th COLT*.

Darroch, J. N., & Ratcliff, D. (1972). Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, *43*, 1470–1480.

Della Pietra, S., Della Pietra, V., & Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*, 380–393.

Durbin, R., Eddy, S., Krogh, A., & Mitchison, G. (1998). *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*. Cambridge University Press.

Freitag, D., & McCallum, A. (2000). Information extraction with HMM structures learned by stochastic optimization. *Proc. AAAI 2000*.

Freund, Y., & Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*, 119–139.

Hammersley, J., & Clifford, P. (1971). Markov fields on finite graphs and lattices. Unpublished manuscript.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*, 2278–2324.

MacKay, D. J. (1996). Equivalence of linear Boltzmann chains and hidden Markov models. *Neural Computation*, *8*, 178–181.

Manning, C. D., & Schütze, H. (1999). *Foundations of statistical natural language processing*. Cambridge Massachusetts: MIT Press.

McCallum, A., Freitag, D., & Pereira, F. (2000). Maximum entropy Markov models for information extraction and segmentation. *Proc. ICML 2000* (pp. 591–598). Stanford, California.

Mohri, M. (1997). Finite-state transducers in language and speech processing. *Computational Linguistics*, *23*.

Mohri, M. (2000). Minimization algorithms for sequential transducers. *Theoretical Computer Science*, *234*, 177–201.

Paz, A. (1971). *Introduction to probabilistic automata*. Academic Press.

Punyakanok, V., & Roth, D. (2001). The use of classifiers in sequential inference. *NIPS 13*. Forthcoming.

Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. *Proc. EMNLP*. New Brunswick, New Jersey: Association for Computational Linguistics.

Rosenfeld, R. (1997). A whole sentence maximum entropy language model. *Proceedings of the IEEE Workshop on Speech Recognition and Understanding*. Santa Barbara, California.

Roth, D. (1998). Learning to resolve natural language ambiguities: A unified approach. *Proc. 15th AAAI* (pp. 806–813). Menlo Park, California: AAAI Press.

Saul, L., & Jordan, M. (1996). Boltzmann chains and hidden Markov models. *Advances in Neural Information Processing Systems 7*. MIT Press.

Schwartz, R., & Austin, S. (1993). A comparison of several approximate algorithms for finding multiple (N-BEST) sentence hypotheses. *Proc. ICASSP*. Minneapolis, MN.