Machine Learning manuscript No.
(will be inserted by the editor)

Piecewise Training for Structured Prediction

Charles Sutton[1*], **Andrew McCallum**[2]

Department of Computer Science
University of Massachusetts
Amherst, MA 01003
casutton@eecs.berkeley.edu,mccallum@cs.umass.edu

Received: date / Revised version: date

Abstract A drawback of structured prediction methods is that parameter estimation requires repeated inference, which is intractable for general structures. In this paper, we present an approximate training algorithm called *piecewise training* that divides the factors into tractable subgraphs, which we call *pieces*, that are trained independently. Piecewise training can be interpreted as approximating the exact likelihood using belief propagation, and different ways of making this interpretation yield different insights into the method. We also present an extension to piecewise training, called *piecewise pseudolikelihood*, designed for when variables have large cardinality. On several real-world NLP data sets, piecewise training performs superior to Besag's pseudolikelihood and sometimes comparably to exact maximum likelihood. In addition, PWPL performs similarly to piecewise and superior to standard pseudolikelihood, but is five to ten times more computationally efficient than batch maximum likelihood training.

1 Introduction

Fundamental to many applications is the ability to predict multiple variables that depend on each other. Such applications are as diverse as classifying regions of an image [Li, 2001], estimating the score in a game of Go [Stern et al., 2005], segmenting genes in a strand of DNA [Bernal et al., 2007], and extracting syntax from natural-language text [Taskar et al., 2004b]. In such applications, we wish to predict a vector $\mathbf{y} = \{y_0, y_1, \ldots, y_T\}$ of random variables given an observed feature vector $\mathbf{x}$. A relatively simple example from natural-language processing is part-of-speech tagging, in which each

[*] *Present address:* Computer Science Division, University of California, Berkeley CA 94720

variable y_s is the part-of-speech tag of the word at position s, and the input $\mathbf{x}$ is divided into feature vectors $\{\mathbf{x}_0, \mathbf{x}_1 \ldots \mathbf{x}_T\}$. Each $\mathbf{x}_s$ contains various information about the word at position s, such as its identity, orthographic features such as prefixes and suffixes, membership in domain-specific lexicons, and information in semantic databases such as WordNet.

An attractive approach to such problems is provided by *structured prediction* methods. Structured prediction methods are essentially a combination of classification and graphical modeling, combining the ability to compactly model multivariate data with the ability to perform prediction using large sets of input features. The idea is, for an input $\mathbf{x}$, to define a discriminant function $\Psi(\mathbf{y}, \mathbf{x})$, and predict $\mathbf{y}^* = \arg\max_{\mathbf{y}} \Psi(\mathbf{y}, \mathbf{x})$. Just as in graphical modeling, this discriminant function factorizes according to a set of local factors. But as in classification, each local factor is modeled as a linear function of $\mathbf{x}$, although perhaps in some induced high-dimensional space. Examples of structured prediction methods include conditional random fields [Lafferty et al., 2001, Sutton and McCallum, 2007a], max-margin Markov networks [Taskar et al., 2004a], MIRA [Crammer and Singer, 2003], and search-based methods [Daumé III and Marcu, 2005].

A main drawback to structured prediction methods is that parameter estimation requires repeated inference. This is because parameters are typically chosen by M-estimation: if θ are the model parameters, then the parameter estimate is $\hat{\theta} = \max_\theta \sum_i \ell(\mathbf{y}^{(i)}, \mathbf{x}^{(i)}; \theta)$ for some function ℓ, such as the likelihood or the margin. Numerical optimization of this objective typically requires performing inference over each data point at many different values in parameter space. For models with tractable structure, such as chains and parse trees, this is often feasible if sometimes computationally expensive, but for general structures inference is intractable. Therefore, the need for repeated inference during training is a significant challenge to applying structured prediction methods to large-scale data, especially when the model has complex structure. For this reason, approximate training methods for structured models are of great interest.

An attractive family of approximate training methods is *local training* methods, which are training methods that depend on sums of local functions of only a few factors rather than on global functions of the entire graph like the likelihood. In other words, the function ℓ can be written as $\ell(\mathbf{x}, \mathbf{y}, \theta) = \sum_a \ell_a(\mathbf{x}_a, \mathbf{y}_a; \theta)$, where each local function ℓ_a can be computed efficiently from only a few of the factors of the full structured model. The best-known example of a local training method is Besag's pseudolikelihood [Besag, 1975], which is a product of the conditional probability of each node given its Markov blanket. The chief attraction of this method is that it achieves consistency without requiring inference in the training procedure. Although in some situations pseudolikelihood can be very effective [Parise and Welling, 2005, Toutanova et al., 2003], in other applications, its accuracy can be poor.

In this paper, we present a novel local training method called *piecewise training*, in which the model's factors are divided into possibly overlapping sets of *pieces*, which are each trained separately. At test time, the resulting weights are used just as if they had been trained using maximum likelihood, that is, on the unseen data they are used to predict the labels using a standard approximate inference algorithm, such as max-product BP. This method has been employed occasionally throughout the literature, but to our knowledge its properties have never been systematically examined. When using piecewise training, the modeler must decide how to split the model into pieces before training. In this paper, we focus on the factor-as-piece approximation, in which each factor of the model is placed in a separate piece.

We motivate this procedure as an approximation to the likelihood of a conditional random field. In particular, this training procedure can be viewed in two ways. First, separate training of each piece can be accomplished by numerically maximizing an approximation to the likelihood. This approximate likelihood can be seen as the true likelihood on a transformation of the original graph, which we call the node-split graph, in which each of the pieces is an isolated component. The second view is based on belief propagation; namely, the objective function of piecewise training is the same as the BP approximate likelihood with uniform messages, as if BP has been stopped after zero iterations. We call this the *pseudomarginal* view of piecewise training, for reasons explained in Section 3.5. These two viewpoints will prove useful both for understanding these algorithms, and for designing extensions of these methods.

Second, we consider an extension to piecewise training, tailored for the case in which the variables have large cardinality. When variables have large cardinality, training can be computationally demanding even when the model structure is tractable. For example, consider a series of processing steps of a natural-language sentence [Sutton et al., 2004, Finkel et al., 2006], which might begin with part-of-speech tagging, continue with more detailed syntactic processing, and finish with some kind of semantic analysis, such as relation extraction or semantic entailment. This series of steps might be modeled as a simple linear chain, but each variable has an enormous number of outcomes, such as the number of parses of a sentence. In such cases, even training using forward-backward is infeasible, because it is quadratic in the variable cardinality. Thus, we also desire approximate training algorithms not only that are sub exponential in the model's treewidth, but also that scale well in the variable cardinality. The Besag pseudolikelihood (PL) is attractive here, because its running time is linear in the variable cardinality. However, piecewise training performs significantly better than pseudolikelihood on the real-world data considered here, but unlike pseudolikelihood it does not scale well in the variable cardinality.

To address this problem, we introduce and analyze a hybrid method, called *piecewise pseudolikelihood* (PWPL), that combines the advantages of both approaches. Essentially, while pseudolikelihood conditions each variable on all of its neighbors, PWPL conditions only on those neighbors within the same piece of the model, for example, that share the same factor. This is illustrated in Figure 2. PWPL can be viewed as double approximation: Rather than performing maximum likelihood on the node-split graph, as regular piecewise does, PWPL performs pseudolikelihood on the node-split graph. Remarkably, this double approximation performs better than the pseudolikelihood approximation alone on several real-world NLP data sets. In other words, in testing accuracy PWPL behaves more like piecewise than like pseudolikelihood. The training speed-up of PWPL can be significant even in linear-chain CRFs, because forward-backward training is quadratic in the variable cardinality.

In the remainder of this paper, we define piecewise training (Section 3), explaining it from the perspectives of the node-split graph (Section 3.1) and of belief propagation (Sections 3.2 and 3.3). Then we present PWPL (Section 4.1), describing it in terms of the node-split graph. This viewpoint allows us to show that under certain conditions, PWPL converges to the piecewise solution in the asymptotic limit of infinite data (Section 4.2). In addition, it provides some insight into when PWPL may be expected to do well and to do poorly, an insight that we verify on synthetic data (Section 5.2.1). Finally, we apply both piecewise and PWPL to several natural-language data sets. The model resulting from the piecewise approximation has better accuracy than pseudolikelihood and is sometimes comparable to exact maximum likelihood (Section 5.1). Finally, we evaluate PWPL on several real-world NLP data sets (Section 5.2.2), finding that it performs often comparably to piecewise training and to maximum likelihood, and on all of our data sets PWPL has higher accuracy than pseudolikelihood. Furthermore, PWPL can be as much as ten times faster than batch CRF training.

2 Background

2.1 Structured modeling and pseudolikelihood

In this paper, we are interested in estimating the conditional distribution $p(\mathbf{y}|\mathbf{x})$ of a discrete output vector $\mathbf{y}$ given an input vector $\mathbf{x}$. We model p by a factor graph G with variables $s \in S$ and factors $\{\Psi_a\}_{a=1}^{A}$ as

Normalizing F to obtain a conditional distribution yields

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{a=1}^{A} \Psi_a(\mathbf{y}_a, \mathbf{x}_a), \tag{1}$$

where $Z(\mathbf{x})$ is a normalization constant. A conditional distribution which factorizes in this way is called a *conditional random field* Lafferty et al. [2001], Sutton and McCallum [2007a]. Typically, each factor is modeled in an exponential form

$$\Psi_a(\mathbf{y}_a, \mathbf{x}_a) = \exp\{\lambda_a^\top f_a(\mathbf{y}_a, \mathbf{x}_a)\}, \tag{2}$$

where λ_a is real-valued parameter vector, and f_a returns a vector of *features* or sufficient statistics over the variables in the set a. The parameters of the model are the set $\Lambda = \{\lambda_a\}_{a=1}^{A}$, and we will be interested in estimating them given a sample of fully observed input-output pairs $D = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$.

Maximum likelihood estimation of Λ is intractable for general graphs, so parameter estimation is performed approximately. One approach is to approximate the partition function $\log Z(\mathbf{x})$ directly, such as by MCMC or variational methods. In this paper, we take a different approach, which is to use a different objective function for parameters that is more computationally tractable.

An example of a computationally convenient objective for estimation is the pseudolikelihood of Besag [1975]. Pseudolikelihood simultaneously classifies each node given its neighbors in the graph. For a variable s, let $N(s)$ be the set of all of its neighbors, not including s itself. Then the pseudolikelihood is defined as

$$\ell_{\text{PL}}(\Lambda) = \sum_{s} \log p(y_s | y_{N(s)}, \mathbf{x}),$$

where the conditional distributions are

$$p(y_s | y_{N(s)}, \mathbf{x}) = \frac{\prod_{a \ni i} \Psi_a(y_s, y_{N(s)}, \mathbf{x}_a)}{\sum_{y'_s} \prod_{a \ni s} \Psi_a(y'_s, y_{N(s)}, \mathbf{x}_a)}. \tag{3}$$

where $a \ni s$ means the set of all factors a that depend on the variable s. In other words, this is a sum of conditional log likelihoods, where for each variable we condition on the true values of its neighbors in the training data.

It is a well-known result that if the model family includes the true distribution, then pseudolikelihood converges to the true parameter setting in the limit of infinite data [Gidas, 1988, Hyvarinen, 2006]. One way to understand this is that pseudolikelihood attempts to match all of model conditional distributions to the data. If it succeeds in matching them all exactly, then a Gibbs sampler run on the model distribution will have the same invariant distribution as a Gibbs sampler run on the true data distribution.

2.2 Free energies

Piecewise training has a close connection with approximate inference methods based on the Bethe free energy, so in this section we give some background on belief propagation and approximate divergence measures.

The general idea behind any variational inference algorithm is to approximate a difficult distribution p by a distribution q drawn from some family $\mathcal{Q}$ in which all distributions are tractable. To pick q, we select the distribution that minimizes some divergence measure $D(q, p)$ over $q \in \mathcal{Q}$. A natural choice of divergence measure is the KL divergence

$$\mathcal{O}(q) = \mathrm{KL}(q\|p) - \log Z \tag{4}$$

$$= -H(q) - \sum_a q(\mathbf{y}_a) \log \Psi_a(\mathbf{y}_a). \tag{5}$$

However, minimizing this divergence function is intractable, if $\mathcal{Q}$ is taken to be to the set of all possible distributions. Therefore we make two approximations. First, we approximate the entropy term $H(q)$ of (5), which is intractable for arbitrary distributions. If q were a tree-structured distribution, then its entropy could be written exactly as

$$H_{\text{BETHE}}(q) = \sum_a q(\mathbf{y}_a) \log q(\mathbf{y}_a) + \sum_i (1 - d_i) q(y_i) \log q(y_i). \tag{6}$$

If q is not a tree, then we can still take H_{BETHE} as an approximation to H to compute the exact variational objective $\mathcal{O}$. This yields the *Bethe free energy*:

$$\mathcal{O}_{\text{BETHE}}(q) = H_{\text{BETHE}}(q) - \sum_a q(\mathbf{y}_a) \log \Psi_a(\mathbf{y}_a) \tag{7}$$

The objective $\mathcal{O}_{\text{BETHE}}$ depends on q only through its marginals, so rather than optimizing it over all probability distributions q, we can optimize over the space of all marginal vectors. Following this idea yields the result that loopy belief propagation optimizes a particular further approximation to this optimization problem [Yedidia et al., 2004], but we do not pursue the details here.

A second way of formulating BP as a variational algorithm yields a dual form of the Bethe energy that will prove particularly useful [Minka, 2005]. This dual energy arises from the expectation propagation view of BP [Minka, 2001a]. Suppose that we approximate each factor Ψ_a by a product of functions that depend on one variable:

$$\Psi_a(\mathbf{y}_a) \approx \tilde{t}_a(\mathbf{y}_a) = \prod_{i \in a} m_{ai}(y_i). \tag{8}$$

This yields an approximation to the distribution p, namely, $p(\mathbf{y}) \approx q(\mathbf{y}) = \prod_a \tilde{t}_a(\mathbf{y}_a)$. Observe that q is possibly unnormalized.

In the context of belief propagation, we can view each factor m_{ai} as the message from factor a to variable i, which have not necessarily converged. In other words, we view the outgoing messages from each factor as approximating it. As Minka [2001a] observes, each message update of loopy BP can be viewed as refining one of the terms $\tilde{t}_a$ so that q is closer, in terms of KL divergence.

Since q was therefore chosen to approximate p, it makes sense to use the mass of q to approximate the mass of p. More precisely, let p' be the unnormalized version of p, that is, $p'(\mathbf{y}) = \prod_a \Psi_a(\mathbf{y}_a)$. Then define rescaled versions of $\tilde{t}_a$ and q as

$$\bar{t}_a(\mathbf{y}_a) = s_a \tilde{t}_a(\mathbf{y}_a) \tag{9}$$

$$\bar{q}(\mathbf{y}) = \prod_a \bar{t}_a(\mathbf{y}_a) \tag{10}$$

Then the idea is to scale each of the $\bar{t}_a$ so that the resulting $\sum_{\mathbf{y}} \bar{q}(\mathbf{y})$ matches as closely as possible the partition function $\sum_{\mathbf{y}} p'(\mathbf{y})$. This can be done by optimizing local divergences in an analogous manner to EP. Define $\bar{q}^{\setminus a}$ as the approximating $\bar{q}$ without the factor $\tilde{t}_a$, that is, each s_a is separately chosen to optimize

$$\min_{s_a} \mathrm{KL}(\Psi_a(\mathbf{y}_a)\bar{q}^{\setminus a}(\mathbf{y}_a) \| s_a \tilde{t}_a(\mathbf{y}_a)\bar{q}^{\setminus a}(\mathbf{y}_a)). \tag{11}$$

Observe that because $\bar{q}^{\setminus a}$ depends on all of the scale factors s_b for all factors b, the local objective function depends on all of the other scale factors as well. The optimal s_a is given by

$$s_a = \frac{\sum_{\mathbf{y}} \frac{\Psi_a(\mathbf{y}_a)}{\tilde{t}(\mathbf{y}_a)} q(\mathbf{y})}{\sum_{\mathbf{y}} q(\mathbf{y})}. \tag{12}$$

Thus the optimal s_a actually does not depend on the other scale values. Now taking the integral $\sum_{\mathbf{y}} \bar{q}(\mathbf{y})$ yields the following approximation to the partition function

$$Z_{\text{BetheDual}} = \prod_i \left(\sum_{y_i} q_i(y_i) \right)^{1-d_i} \prod_a \left(\sum_{\mathbf{y}_a} \frac{\Psi_a(\mathbf{y}_a)}{\tilde{t}(\mathbf{y}_a)} q(\mathbf{y}_a) \right). \tag{13}$$

It can be shown [Minka, 2001b] that this is also a free energy for BP, that is, that fixed points of BP are stationary points of this objective.

A third view of BP, which is also useful in understanding piecewise training, is the reparameterization viewpoint [Wainwright et al., 2003a]. In this view, the BP updates are expressed solely in terms of the beliefs b^n at each iteration n of the algorithm. The beliefs are initialized as $b^0_a \propto \Psi_a$ for all factors, and $b^0_s \propto 1$ for all variables. The updates at iteration n are

$$\begin{aligned} b^n_s(y_s) &= \sum_{\mathbf{y}_{N(s)}} B^{n-1}_s(y_s, \mathbf{y}_{N(s)}) \\ b^n_a(\mathbf{y}_a) &= \sum_{\mathbf{y}_{N(a)}} B^{n-1}_a(\mathbf{y}_a, \mathbf{y}_{N(a)}) \end{aligned} \tag{14}$$

where the distributions B^{n-1}_a and B^{n-1}_s are defined as

$$\begin{aligned} B^{n-1}_s(y_s, \mathbf{y}_{N(s)}) &\propto b^{n-1}(y_s) \prod_{a \ni s} \frac{b^{n-1}_a(\mathbf{y}_a)}{b^{n-1}_s(y_s)} \\ B^{n-1}_a(\mathbf{y}_a, \mathbf{y}_{N(a)}) &\propto b^{n-1}_a(\mathbf{y}_a) \prod_{s \in a} \prod_{c \ni s \setminus a} \frac{b_c(\mathbf{y}_c)}{b_s(y_s)} \end{aligned} \tag{15}$$

(This notation is adapted from Rosen-Zvi et al. [2005].)

It can be shown that the beliefs from the message-based recursions are equal to those from the reparameterization-based recursions. That is, for all iterations m, $q_s^n = b_s^n$ and $q_a^n = b_a^n$. This can be seen by induction, substituting the messages corresponding to q^{n-1} into the update equations (14) for b^n.

The reason for the term "reparameterization" is that at each iteration n, we can construct a distribution $T^n(\mathbf{y})$ over the full space with factors

$$T_s^n = b_s^n, \qquad T_a^n = \frac{b_a^n}{\prod_{s \in a} b_s^n}. \tag{16}$$

This distribution is invariant under the message update, that is, $T^n = T^{n-1} = \cdots = T^0 = p$. So each T^n can be viewed as a reparameterization of the original distribution. This view of BP will prove especially useful in Section 3.3.

3 Piecewise Training

In this section, we present piecewise training. The motivation is that in some applications, the local information in each factor alone, without performing inference, is enough to do fairly well at predicting the outputs, but some amount of global information can help. Therefore, to reduce training time, it makes sense to perform less inference at training time than at test time, because at training time we loop through the examples repeatedly, whereas at test time we only need to make each prediction once. For example, suppose we want to train a loopy pairwise MRF. In piecewise estimation, what we will do is to train the parameters of each edge independently, as if each edge were a separate two-node MRF of its own. Finally, on test data, the parameters resulting from this local training become the parameters used to perform global inference, using some standard approximate inference algorithm.

Now we define the piecewise estimator more generally. Let the distribution $p(\mathbf{y}|\mathbf{x})$ be defined by a factor graph, where $\Psi_a(\mathbf{y}_a, \mathbf{x}_a, \theta)$ has the exponential form (2), and suppose that we wish to estimate θ. Assume that the model's factors are divided into a set $\mathcal{P} = \{R_0, R_1 \ldots\}$ of pieces; each piece $R \in \mathcal{P}$ is a set of factors $R = \{\Psi_a\}$. The pieces need not be disjoint. For example, in a grid-shaped MRF with unary and pairwise factors, we might isolate each factor in its own piece, or alternatively we might choose one piece for each row and each column of the MRF, in which case each unary factor would be shared between its row piece and its column piece.

To train the pieces separately, each piece R has a local likelihood

$$\ell_R(\theta) = \sum_{a \in R} \theta_a^\top f_a(\mathbf{y}_a, \mathbf{x}_a) - A_R(\theta; \mathbf{x}). \tag{17}$$

where $A_R(\theta; \mathbf{x})$ is the *local log partition function* for the piece, that is,

$$A_R(\theta; \mathbf{x}) = \log \sum_{\mathbf{y}_R} \exp\{\sum_{a \in R} \theta_a^\top f_a(\mathbf{y}_a, \mathbf{x}_a)\}, \tag{18}$$

where $\mathbf{y}_R$ is the vector of variables used anywhere in piece R. This is the likelihood for the piece R if it were a completely separate graphical model. If the pieces are disjoint, and no parameters are shared between distinct factors, then we could train each piece by separately computing parameters $\theta^R_{\text{PW}} = \max_{\theta_R} \ell_R(\theta_R)$. But in order to handle parameter tying and overlapping pieces, we instead perform a single optimization, maximizing the sum of all of the single-piece likelihoods. So for a set $\mathcal{P}$ of pieces, the piecewise likelihood becomes

$$\ell_{\text{PW}}(\theta) = \sum_{R \in \mathcal{P}} \sum_{a \in R} \theta_a f_a(\mathbf{y}_a, \mathbf{x}_a) - \sum_{R \in \mathcal{P}} A_R(\theta; \mathbf{x}). \tag{19}$$

For example, consider the special case of per-edge pieces in a pairwise MRF with no tied parameters. Then, for an edge (s,t), we have $A_{st}(\theta) = \log \sum_{y_s, y_t} \Psi(y_s, y_t)$, so that the piecewise estimator corresponds exactly to training independent probabilistic classifiers on each edge.

Now let us compare the approximate likelihood (19) to the exact likelihood. Recall that the true likelihood is

$$\ell(\theta) = \sum_a \theta_a f_a(\mathbf{y}_a, \mathbf{x}_a) - A(\theta; \mathbf{x})$$
$$A(\theta; \mathbf{x}) = \log \sum_{\mathbf{y}} \exp\{\sum_a \theta_a f_a(\mathbf{y}_a, \mathbf{x}_a)\}.$$

Notice that the first summation contains exactly the same terms as in the exact likelihood. The only difference between the piecewise objective and the exact likelihood is in the second summation of (19). So $A_{\text{PW}}(\theta; \mathbf{x}) = \sum_R A_R(\theta; \mathbf{x})$ can be viewed as an approximation of the log partition function. Standard maximum-likelihood training for CRFs can require evaluating the instance-specific log partition function $A(\theta; \mathbf{x})$ for each training instance for each iteration of an optimization algorithm. By using piecewise training, we need to compute only local normalization over small cliques, which for loopy graphs is potentially much more efficient.

A choice of pieces to which we devote particular attention is the factor-as-piece approximation, in which each factor in the model is assigned to its own piece. There is a potential ambiguity in this choice. To see, write out the summation contained within the dot product of (2)

$$\Psi_a(\mathbf{y}_a) = \exp\{\sum_k \theta_{ak} f_{ak}(\mathbf{y}_a)\},$$

so that each factor has multiple parameters and sufficient statistics. But we could just as well place each sufficient statistic in a factor all to itself, that is,

$$\Psi_{ak}(\mathbf{y}_a) = \exp\{\theta_{ak} f_{ak}(\mathbf{y}_a)\}, \tag{20}$$

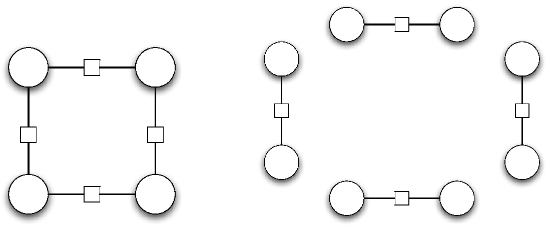

Fig. 1 Example of node splitting. Left is the original model, right is the version trained by piecewise. In this example, there are no unary factors.

and define the pieces at that level of granularity. Such a fine-grained choice of pieces could be useful. For example, in a linear-chain model, we might choose to view the model as a weighted finite-state machine, and partition the state-transition diagram into pieces. In this paper, however, when we use the factor-as-piece approximation, we will not use the fine-grained factorization of (20), that is, we will assume that the graph has been constructed so that no two factors share exactly the same support.

3.1 The Node-split Graph

The piecewise likelihood (19) can be viewed as the exact likelihood in a transformation of the original graph. In the transformed graph, we split the variables, adding one copy of each variable for each factor that it participates in, as pictured in Figure 1. We call the transformed graph the *node-split graph.*

Formally, the splitting transformation is as follows. Given a factor graph G, create a new graph G' with variables $\{y_{as}\}$, where a ranges over all factors in G and s over all variables in a. For any factor a, let π_a map variables in G to their copy in G', that is, $\pi_a(y_s) = y_{as}$ for any variable s in G. Finally, for each factor $\Psi_a(y_a, \theta)$ in G, add a factor Ψ'_a to G' as

$$\Psi'_a(\pi_a(y_a), \theta) = \Psi_a(y_a, \theta). \tag{21}$$

If we wish to use pieces that are larger than a single factor, then the definition of the node-split graph can be modified accordingly.

Clearly, piecewise training in the original graph is equivalent to exact maximum likelihood training in the node-split graph. This view of piecewise training will prove useful for understanding PWPL in Section 4.

This viewpoint also makes clear the bias in the piecewise estimate. Suppose that the model family contains the true distribution, and let $p(\mathbf{y}|\mathbf{x};\theta^*)$ be the true distribution of the data. The piecewise likelihood cannot distinguish this distribution from the distribution p_{NS} over the original space that is defined by the product of marginals on the node-split graph

$$p_{\text{NS}}(\mathbf{y}|\mathbf{x}) \propto \prod_{a \in G'} p(\pi^{-1}(y_{as})|\mathbf{x};\theta^*), \tag{22}$$

where G' is the node-split graph, and $p(\mathbf{y}_a|\mathbf{x};\theta^*)$ is the marginal distribution of the variables in factor a according to the true distribution. By that we mean that the piecewise likelihood of any parameter setting θ when the data distribution is exactly the true distribution p is equal to the piecewise likelihood of θ when the data distribution equals the distribution p_{NS}.

For example, suppose that, unknown to the modeler, all the variables in the graph are actually independent. Then $p_{\text{NS}}(\mathbf{y}|\mathbf{x}) = \prod_{s\in G} p(y_s|\mathbf{x})^{d(s)}$, where $d(s)$ is the degree of variable s, and the piecewise estimate will converge to this distribution in the limit of infinite data rather than the true distribution. Essentially, piecewise overcounts the effect of the variables that share multiple factors.

3.2 The Belief Propagation Viewpoint

Another way of understanding piecewise training arises from belief propagation. Let $M = \{m_{ai}(y_i)\}$ be a set of BP messages, not necessarily converged. We view all of a factor's outgoing messages as approximating it, that is, we define $\tilde{\Psi}_a = \prod_{i\in a} m_{ai}$. Recall from Section 2.2 that the dual energy of belief propagation yields an approximate partition function

$$Z_{\text{BP}}(\theta, M) = \prod_a \left(\sum_{\mathbf{y}_a} \frac{\Psi_a(\mathbf{y}_a,\theta)}{\tilde{\Psi}_a(\mathbf{y}_a)} q(\mathbf{y}_a) \right) \prod_i \left(\sum_{y_i} q(y_i) \right)^{1-d_i}, \qquad (23)$$

where q denotes the unnormalized beliefs

$$q(\mathbf{y}) = \prod_a \tilde{\Psi}_a(\mathbf{y}_a) = \prod_a \prod_i m_{ai}(y_i), \qquad (24)$$

with $q(\mathbf{y}_a) = \sum_{\mathbf{y}\backslash\mathbf{y}_a} q(\mathbf{y})$ and $q(y_i) = \sum_{\mathbf{y}\backslash y_i} q(\mathbf{y})$.

Now, let M_0 be the uniform message setting, that is, $m_{ai} = 1$ for all a and i. This is a common initialization for BP. Then the unnormalized beliefs are $q(\mathbf{y}) = 1$ for all $\mathbf{y}$, and the approximate partition function is

$$Z_{\text{BP}}(\theta, M_0) = \prod_a \left(C_a \sum_{\mathbf{y}_a} \Psi_a(\mathbf{y}_a,\theta) \right) \prod_i C_i^{1-d_i}, \qquad (25)$$

where C_a and C_i are constants that do not depend on θ. This approximate partition function is the same as that used by piecewise training with one factor per piece, up to a multiplicative constant that does not change the gradient. So another view is that piecewise training approximates the likelihood using belief propagation, except that we cut off BP after 0 iterations.

3.3 Pseudo-Moment Matching Viewpoint

Piecewise training is based on the intuition that if all of the local factors fit the data well, then the resulting global distribution is likely to be reasonable. An interesting way of formalizing this idea is by way of the *pseudo-moment matching* estimator of Wainwright et al. [2003b]. In this section, we show that there is a sense in which piecewise training can be viewed as an extension of the pseudo-moment matching estimator.

First, consider the case in which a discrete distribution $p(\mathbf{y})$ factorizes according to a graph G with fully-parameterized tables, that is,

$$\Psi_a(\mathbf{y}_a) = \exp\{\sum_{\mathbf{y}'_a} \theta(\mathbf{y}'_a) \mathbf{1}_{\{\mathbf{y}_a = \mathbf{y}'_a\}}\}. \tag{26}$$

Here we are not (yet) conditioning on any input variables. Let $\tilde{p}(\mathbf{y})$ be the empirical distribution, that is, $\tilde{p}(\mathbf{y}) \propto \sum_i \mathbf{1}_{\{\mathbf{y}=\mathbf{y}^{(i)}\}}$.

The pseudo-moment matching estimator chooses parameters that maximize the BP likelihood without actually computing any of the message updates. This estimator is

$$\begin{aligned} \hat{\theta}_a(\mathbf{y}_a) &= \log \frac{\tilde{p}(\mathbf{y}_a)}{\prod_{s \in a} \tilde{p}(y_s)} \\ \hat{\theta}_s(y_s) &= \log \tilde{p}(y_s). \end{aligned} \tag{27}$$

For these parameters, there exists a set of messages that (a) are a fixed-point of BP, and (b) the resulting beliefs q_a and q_s equal the empirical marginals. This can be seen using the reparameterization perspective of BP [Wainwright et al., 2003a] described in Section 2.2, because with those parameters the belief-based updates of (14) yield a fixed point immediately.

For conditional random fields, however, we are interested in estimating the parameters of conditional distributions $p(\mathbf{y}|\mathbf{x})$. A simple generalization is to require for all inputs $\mathbf{x}$ with $\tilde{p}(\mathbf{x}) > 0$ that

$$\begin{aligned} \Psi_a(\mathbf{y}_a, \mathbf{x}) &= \frac{\tilde{p}(y_a|\mathbf{x})}{\prod_{s \in A} \tilde{p}(y_s|\mathbf{x})} \\ \Psi_s(y_s, \mathbf{x}) &= \tilde{p}(y_s|\mathbf{x}). \end{aligned} \tag{28}$$

However, we can no longer expect to find parameters that satisfy these equations in closed form. This is because the factor values of $\Psi_a(\cdot, \mathbf{x})$ do not have an independent degree of freedom for each input value $\mathbf{x}$. Instead, to promote generalization across different inputs, the factors Ψ_a have some complex parameterization, such as the linear form (2), in which parameters are tied across different input values. Therefore, a natural approach is to treat the equations (28) as a nonlinear set of equations to be solved. To do this, we optimize the objective function

$$\min_\theta \sum_a D(\Psi_a \| \tilde{p}_a) + \sum_s D(\Psi_s \| \tilde{p}_s), \tag{29}$$

where $D(\cdot\|\cdot)$ is a divergence measure. By a *divergence measure* $D(p\|q)$, we simply mean a nonnegative function that is 0 if and only if $p = q$. Then if a parameter setting θ exists such that the divergence is zero, then the equations have been solved exactly, and θ optimizes the BP likelihood.

This provides another view of piecewise training, because choosing using $\mathrm{KL}(\tilde{p}_a\|\Psi_a)$ for the divergence in (29) yields an equivalent optimization problem to the piecewise likelihood (19). This provides a justification of the intuition that fitting locally can lead to a reasonable global solution: it is not the case that fitting factors locally causes the true marginals to be matched to the empirical distribution, but it does cause the BP approximation to the marginals to be matched to the empirical distribution, over the inputs that were observed in the training data.

3.4 Approximation to the likelihood

One rationale for the piecewise estimator is that it bounds the likelihood.

Proposition 1 *For any set $\mathcal{P}$ of pieces, the piecewise partition function is an upper bound on the true partition function:*

$$A(\theta; \mathbf{x}) \leq \sum_{R \in \mathcal{P}} A_R(\theta; \mathbf{x}). \tag{30}$$

Proof The bound is immediate upon expansion of $A(\theta; \mathbf{x})$.

$$A(\theta; \mathbf{x}) = \log \sum_{\mathbf{y}} \prod_{R \in \mathcal{P}} \exp \left\{ \sum_{a \in R} \theta_a f_a(\mathbf{y}_a, \mathbf{x}_a) \right\} \tag{31}$$

$$\leq \log \prod_{R \in \mathcal{P}} \sum_{\mathbf{x}_R} \exp \left\{ \sum_{a \in R} \theta_a f_a(\mathbf{y}_a, \mathbf{x}_a) \right\} \tag{32}$$

$$= \sum_{R \in \mathcal{P}} A_R(\theta; \mathbf{x}). \tag{33}$$

The bound from (31) to (32) is justified by considering the expansion of the product in (32). The expansion contains every term of the summation in (31), and all terms are nonnegative.

Therefore, the piecewise likelihood is a lower bound on the true likelihood. If the graph is connected, however, then the bound is not tight, and in practice it is extremely loose.

This bound is actually a special case of the tree-reweighted bounds of Wainwright, Jaakkola, and Willsky [2002], a connection which suggests generalizations of the simple piecewise training procedure. As in that work, we will obtain the upper bound by writing the original parameters θ as a mixture of tractable parameter vectors $\theta(T)$. Consider the set $\mathcal{T}$ of tractable subgraphs induced by single factors in $\mathcal{G}$. Precisely, for each factor Ψ_a in $\mathcal{G}$, we add a (non-spanning) tree T_R which contains only the factor Ψ_a and its associated variables. With each tree T_R we associate an exponential parameter vector $\theta(T_R)$.

Let μ be a strictly positive probability distribution over factors. To use Jensen's inequality, we will need to have the constraint

$$\theta = \sum_R \mu_R \theta(T_R). \tag{34}$$

Now, each parameter θ_i corresponds to exactly one factor of $\mathcal{G}$, which appears in only one of the T_R. Therefore, only one choice of subgraph parameter vectors $\{\theta(T_R)\}$ meets the constraint (34), namely:

$$\theta(T_R) = \frac{\theta|_r}{\mu_R}, \tag{35}$$

where $\theta|_R$ is the restriction of θ to R; that is, $\theta|_R$ has the same entries and dimensionality as θ, but with zeros in all entries that are not included in the piece R.

Therefore, using Jensen's inequality, we immediately have the bound

$$A(\theta) \leq \sum_R \mu_R A\left(\frac{\theta|_R}{\mu_R}\right). \tag{36}$$

This is the exact bounding strategy used by Wainwright et al. [2002], except that we have chosen to use the set of all single-factor trees to derive the bound, rather than the set of all spanning trees.

This *reweighted piecewise* bound is clearly related to the basic piecewise bound in (30), because $A(\theta|_R)$ differs from $A_R(\theta)$ only by an additive constant which is independent of θ. In fact, a version of Proposition 1 can be derived by considering the limit of (36) as μ approaches a point mass on an arbitrary single piece R^*.

The connection to the Wainwright et al. work suggests at least two generalizations of the basic piecewise method. The first is that the reweighted piecewise bound in (36) can itself be minimized as an approximation to $A(\theta)$, yielding a variation of the basic piecewise method.

The second is that this line of analysis can naturally handle the case when pieces overlap. For example, in an MRF with both node and edge factors, we might choose each piece to be an edge factor with its corresponding node factors, hoping that this overlap will allow limited communication between the pieces which could improve the approximation. As long as there is a value of μ for which the constraint in (35) holds, then (36) provides a bound we can minimize in an overlapping piecewise approximation.

3.5 Three Views of Approximate Training Algorithms

The connections between belief propagation and piecewise training suggest a general framework for viewing local training algorithms, which was describe by Sutton and Minka [2006]. They describe three different viewpoints on local training, which suggest different algorithms. First, many local training algorithms are straightforwardly viewed as performing exact inference on a transformed graph that cuts the global dependencies in the model. For example, standard piecewise performs maximum-likelihood training in a *node-split graph* in which variables are duplicated so that each factor is in its own connected component. Sutton and Minka refer to this viewpoint as the *neighborhood graph view* of a training algorithm.

Second, many local training algorithms can be interpreted as approximating $\log Z$ by the Bethe energy $\log Z_{\mathrm{BP}}$ at a particular message setting. Sutton and Minka call this the *pseudomarginal view*, because under this view, the estimated parameters are chosen to match the pseudomarginals to the empirical marginals. For any approximate partition function $\tilde{Z}$, the pseudomarginals are the derivatives $\partial \log \tilde{Z} / \partial \theta_{ak}$. To explain the terminology, suppose that the unary factors have the form $\Psi_i(y_i) = \exp\{\sum_{y_i'} \theta_{i,y_i'} \mathbf{1}_{\{y_i = y_i'\}}\}$. Then the derivative $\partial \log Z / \partial \theta_i(y_i)$ of the true partition function yields the marginal distribution, so the corresponding derivative of $\log \tilde{Z}$ is called a pseudomarginal.

As a third viewpoint, any approximate inference algorithm can be used to perform approximate ML training, by substituting the approximate beliefs for the exact marginals in the ML gradient. They call this the *belief view* of an approximate training algorithm. For example, this is the standard way of implementing approximate training using BP. Interestingly, although every approximate likelihood yields approximate gradients through the pseudomarginals, not all approximate gradients can themselves be obtained as the exact gradient of any single approximate objective function. Recently, training methods that have a pseudomarginal interpretation—that is, those that can be described as numerically optimizing an objective function—have received much attention, but it is not clear if training methods that have a pseudomarginal interpretation should be preferred over ones that do not.

The pseudomarginal and belief viewpoints are distinct. Explaining this requires making a distinction that is not always clear in the literature, between beliefs and pseudomarginals. By the *belief* of a node i, we mean its normalized product of messages, which is proportial to $q(y_i)$. By *pseudomarginal*, on the other hand, we mean the derivative of $\log \tilde{Z}$ with respect to θ_i. These quantities are distinct. For example, in standard piecewise, the pseudomarginal $\partial \log Z_{\mathrm{PW}} / \partial \theta_i(y_i)$ equals $p_i(y_i)$, but the belief is proportional to $q(y_i) = \sum_{\mathbf{y} \backslash y_i} q(y_i) = 1$.

This point may be confusing for several reasons. First, when the messages m are a fixed point of BP, then the pseudomarginal always equals the belief. But this does not hold before convergence, such as the all-ones message setting used in standard piecewise training. A second potential confusion arises because we define Z_{BP} using the dual Bethe energy (13) rather than the primal (7). In the primal Bethe energy, the pseudomarginal equals the belief at all message settings, but this is not true of the dual energy. The dual energy both helps in interpreting local training algorithms, as in Section 3.2, and also tends to be a better approximation to $\log Z$ at intermediate message settings.

When calculating pseudomarginals $\partial \log \tilde{Z} / \partial \theta_i$, we must recognize that the message setting is often itself a function of θ. For example, suppose we stop BP after one iteration, that is, we take $\tilde{Z}(\theta) = Z_{\text{BP}}(\theta, m^{(1)}(\theta))$, where $m^{(1)}$ are the messages after one BP iteration. Then, because the message setting is clearly a function of θ, we need to take $\partial m^{(1)} / \partial \theta_i$ into account when computing the pseudomarginals of $\tilde{Z}$.

Sutton and Minka [2006] use this viewpoint to explore several other local training algorithms based on early stopping of BP, but we do not pursue the details here.

4 Piecewise Pseudolikelhood

Although piecewise training breaks apart intractable structures, it can still be computationally expensive when the variables have large cardinality. This is because computing the local normalization functions A_R in (19) requires summing over all assignments to a piece. For example, if all factors are binary, then this summation requires quadratic time in the variable cardinality. Although this is feasible for many models, if the cardinality is large, this cost can be unacceptable.

Pseudolikelihood (Section 2.1), on the other hand, scales linearly in the variable cardinality, because each term in the pseudolikelihood conditions all but one variable in the model. However, on the data that we consider here, PL has considerably worse accuracy than piecewise training. This suggests a hybrid method, in which we apply pseudolikelihoodization to the node-split graph (Section 3.1) rather than the original graphical model. We call this training method piecewise pseudolikelihood (PWPL). Surprisingly, we find that two approximations work better than one: on the data considered here, PWPL—which applies the node-split approximation before pseudolikelihood—works better than the pseudolikelihood approximation alone.

This section proceeds as follows. In Section 4.1, we describe PWPL in terms of the node-split graph, which was presented previously in Section 3.1. This viewpoint allows us to show that under certain conditions, PWPL converges to the piecewise solution in the asymptotic limit of infinite data (Section 4.2). In addition, it provides some insight into when PWPL may be expected to do well and to do poorly, an insight that we verify on synthetic and real-world data in Section 5.2.

4.1 Definition

In this section, I define piecewise pseudolikelihood (Section 4.1), and describe its asymptotic behavior using well-known results about pseudolikelihood (Section 4.2).

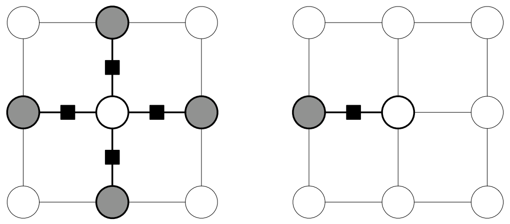

Fig. 2 Illustration of the difference between piecewise pseudolikelihood (PWPL) and standard pseudolikelihood. In standard PL, at left, the local term for a variable y_s is conditioned on its entire Markov blanket. In PWPL, at right, each local term conditions only on the neighbors within a single factor.

The main motivation of piecewise training is computational efficiency, but in fact piecewise does not always provide a large gain in training time over other approximate methods. In particular, the time required to evaluate the piecewise likelihood at one parameter setting is the same as is required to run one iteration of belief propagation (BP). More precisely, piecewise training uses $O(m^K)$ time, where m is the maximum number of assignments to a single variable y_s and K is the size of the largest factor. Belief propagation also uses $O(m^K)$ time per iteration; thus, the only computational savings over BP is a factor of the number of BP iterations required. In tree-structured graphs, piecewise training is no more efficient than forward-backward.

To address this problem, we propose piecewise pseudolikelihood. Piecewise pseudolikelihood (PWPL) is defined as:

$$\ell_{\text{PWPL}}(\Theta; \mathbf{x}, \mathbf{y}) = \sum_a \sum_{s \in a} \log p_{\text{LCL}}(y_s | \mathbf{y}_{a \setminus s}, \mathbf{x}, \theta_a), \tag{37}$$

where $(\mathbf{x}, \mathbf{y})$ are an observed data point, the index a ranges over all factors in the model, the set $a \setminus s$ means all of the variables in the domain of factor a except for s, and p_{LCL} is a locally-normalized score similar to a conditional probability and defined below.

So the piecewise pseudolikelihood is a sum of local conditional log-probabilities. Each variable s participates as the domain of a conditional once for each factor that it neighbors. As in piecewise training, the local conditional probabilities p_{LCL} are not the true probabilities according to the model, but are a quantity computed locally from a single piece (in this case, a single factor). The local probabilities p_{LCL} are defined as

$$p_{\text{LCL}}(y_s | \mathbf{y}_{a \setminus s}, \mathbf{x}, \theta_a) = \frac{\Psi_a(y_s, \mathbf{y}_{a \setminus s}, \mathbf{x}_a)}{\sum_{y'_s} \Psi_a(y'_s, \mathbf{y}_{a \setminus s}, \mathbf{x}_a)}. \tag{38}$$

Then given a data set $D = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}$, we select the parameter setting that maximizes

$$O_{\text{PWPL}}(\theta; D) = \sum_i \ell_{\text{PWPL}}(\theta; \mathbf{x}^{(i)}, \mathbf{y}^{(i)}) - \sum_a \frac{\|\theta_a\|^2}{2\sigma^2}, \tag{39}$$

where the second term is a Gaussian prior on the parameters to reduce overfitting. The piecewise pseudolikelihood is convex as a function of θ, and so its maximum can be found by standard techniques. In the experiments below, we use limited-memory BFGS [Nocedal and Wright, 1999].

For simplicity, we have presented PWPL for the case in which each piece contains exactly one factor. If larger pieces are desired, then simply take the summation over a in (37) to be over pieces rather than over factors, and generalize the definition of p_{LCL} appropriately.

Compared to standard piecewise, the main advantage of PWPL is that training requires only $O(m)$ time rather than $O(m^K)$. Compared to pseudolikelihood, the difference is that whereas in pseudolikelihood each local term conditions on the entire Markov blanket, in PWPL each local term conditions only on a variable's neighbors within a single factor. For this reason, the local terms in PWPL are not true conditional distributions according to the model. The difference between PWPL and pseudolikelihood is illustrated in Figure 2. In the next section, we discuss why in some situations this can cause PWPL to have better accuracy than pseudolikelihood.

4.2 Analysis

PWPL can be readily understood from the node-split viewpoint. In particular, the piecewise pseudolikelihood is simply the standard pseudolikelihood applied to the node-split graph. In this section, we use the asymptotic consistency of standard pseudolikelihood to gain insight into the performance of PWPL.

So equivalently, we suppose that we are given an infinite data set drawn from the distribution p_{NS}, as defined in Section 3.1. Now, the standard consistency result for pseudolikelihood is that if the model class contains the generating distribution, then the pseudolikelihood estimate converges asymptotically to the true distribution. In this setting, that implies the following statement. If the model family defined by G' contains p_{NS}, then piecewise pseudolikelihood converges in the limit to the same parameter setting as standard piecewise.

Because this is an asymptotic statement, it provides no guarantee about how PWPL will perform on real data. Even so, it has several interesting consequences that provide insight into the method. First, it may impact what sort of model is conducive to PWPL. For example, consider a Potts model with unary factors $\Psi(y_s) = [1 \quad e^{\theta_s}]^\top$ for each variable s, and pairwise factors

$$\Psi(y_s, y_t) = \begin{pmatrix} e^{\theta_{st}} & 1 \\ 1 & 1. \end{pmatrix}, \tag{40}$$

for each edge (s, t), so that the model parameters are $\{\theta_s\} \cup \{\theta_{st}\}$. Then the above condition for PWPL to converge in the infinite data limit will never be satisfied, because the pairwise piece cannot represent the marginal distribution of its variables. In this case, PWPL may be a bad choice, or it may be useful to consider pieces that contain more than one factor.

Second, this analysis provides intuition about the differences between piecewise pseudolikelihood and standard pseudolikelihood. For each variable s with neighborhood $N(s)$, standard pseudolikelihood approximates the model marginal $p(y_{N(s)})$ over the neighborhood by the empirical marginal $\tilde{p}(y_{N(s)})$. We expect this approximation to work well when the model is a good fit, and the data is ample.

In PWPL, we perform the node-splitting transformation on the graph prior to maximizing the pseudolikelihood. The effect of this is to reduce each variable's neighborhood size, that is, the cardinality of $N(s)$.

This has two potential advantages. First, because the neighborhood size is small, PWPL may converge to piecewise faster than pseudolikelihood converges to maximum likelihood. One may reasonably expect maximum likelihood to be better than piecewise, so whether to prefer standard PL or piecewise PL depends on precisely how much faster the convergence is. Second, the node-split model may be able to exactly model the marginal of its neighborhood in cases where the original graph may not be able to model its larger neighborhood. Because the neighborhood is smaller, the pseudolikelihood convergence condition may hold in the node-split model when it does not in the original model. In other words, standard pseudolikelihood requires that the original model is a good fit to the full distribution. In contrast, we expect piecewise pseudolikelihood to be a good approximation to piecewise when each individual piece fits the empirical distribution well. The performance of piecewise pseudolikelihood need not require the node-split model to represent the distribution across pieces.

Finally, this analysis suggests that we might expect piecewise pseudolikelihood to perform poorly in two regimes: First, if so much data is available that pseudolikelihood has asymptotically converged, then it makes sense to use pseudolikelihood rather than piecewise pseudolikelihood. Second, if features of the local factors cannot fit the training data well, then we expect the node-split model to fit the data quite poorly, and piecewise pseudolikelihood cannot possibly do well.

5 Experiments

5.1 Piecewise Training

In this section, we compare piecewise training to both maximum likelihood and pseudolikelihood on three real-world natural language tasks.

To be as fair as possible to pseudolikelihood, we compare to two variations of pseudolikelihood, one based on nodes and a structured version based

Method	Overall F1
Piecewise	**91.2**
Pseudolikelihood	84.7
Per-edge PL	89.7
Exact	90.6

Table 1 Comparison of piecewise training to exact and pseudolikehood training on a linear-chain CRF for named-entity recognition. On this tractable model, piecewise methods are more accurate than pseudolikelihood, and just as accurate as exact training.

Method	Noun-phrase F1
Piecewise	**88.1**
Pseudolikelihood	84.9
Per-edge PL	86.5
BP	86.0

Table 2 Comparison of piecewise training to other methods on a two-level factorial CRF for joint part-of-speech tagging and noun-phrase segmentation.

Method	Token F1	
	location	speaker
Piecewise	**87.7**	75.4
Pseudolikelihood	67.1	25.5
Per-edge PL	76.9	69.3
BP	86.6	**78.2**

Table 3 Comparison of piecewise training to other methods on a skip-chain CRF for seminar announcements.

on edges. As normally applied, pseudolikelihood is a product of per-node conditional probabilities:

$$\ell_{\text{PL}}(\theta) = \log \prod_s p(y_s | \mathbf{y}_{N(s)}),$$

where $N(s)$ are the set of variables that neighbor variable s. But this per-variable pseudolikelihood function does not work well for sequence labeling, because of the strong interactions between neighboring sequence positions. In order to have a stronger baseline, we also compare to a pairwise version of pseudolikelihood:

$$\ell_{\text{EPL}}(\theta) = \log \prod_{st} p(y_s, y_t | \mathbf{y}_{N(s,t)}), \tag{41}$$

where the variables s, t range over all variables that share a factor. That is, instead of using local conditional distributions over single variables, we use

$w_t = w$
w_t matches `[A-Z][a-z]+`
w_t matches `[A-Z][A-Z]+`
w_t matches `[A-Z]`
w_t matches `[A-Z]+`
w_t contains a dash
w_t matches `[A-Z]+[a-z]+[A-Z]+[a-z]`
The character sequence $c_0 \ldots c_n$ is a prefix of w_t (where $n \in [0, 4]$)
The character sequence $c_0 \ldots c_n$ is a suffix of w_t (where $n \in [0, 4]$)
The character sequence $c_0 \ldots c_n$ occurs in w_t (where $n \in [0, 4]$)
w_t appears in list of first names, last names, countries, locations, honorifics, etc.
$q_k(\mathbf{x}, t + \delta)$ for all k and $\delta \in [-2, 2]$

Table 4 Input features $q_k(\mathbf{x}, t)$ for the CoNLL named-entity data. In the above w_t is the word at position t, T_t is the POS tag at position t, w ranges over all words in the training data, and T ranges over all Penn Treebank part-of-speech tags. The "appears to be" features are based on hand-designed regular expressions that can span several tokens.

distributions over pairs of variables, hoping to take more of the sequential interactions into account.

We evaluate piecewise training on three models: a linear-chain CRF (Section 5.1.1), a factorial CRF (Section 5.1.2), and a skip-chain CRF (Section 5.1.3). All of these models use a large number of input features such as word identity, part-of-speech tags, capitalization, and membership in domain-specific lexicons.

In all the experiments below, we optimize ℓ_{PW} using limited-memory BFGS. We use a Gaussian prior on weights to avoid overfitting. In previous work, the prior parameter had been tuned on each data set for belief propagation, and for the local models we used the same prior parameter without change. At test time, decoding is always performed using max-product belief propagation.

5.1.1 Linear-Chain CRF First, we evaluate the accuracy of piecewise training on a tractable model, so that we can compare the accuracy to exact maximum-likelihood training. The task is named-entity recognition, that is, to find proper nouns in text. We use the CoNLL 2003 data set, consisting of 14,987 newswire sentences annotated with names of people, organizations, locations, and miscellaneous entities. We test on the standard development set of 3,466 sentences. Evaluation is done using precision and recall on the extracted chunks, and we report $F_1 = 2PR/(P + R)$. We use a linear-chain CRF, whose features are described in Table 4.

Piecewise training performs better than either of the pseudolikelihood methods. Furthermore, even though it is a completely local training method, piecewise training performs comparably to exact CRF training.

$w_{t-\delta} = w$ w_t matches `[A-Z][a-z]+` w_t matches `[A-Z]` w_t matches `[A-Z]+` w_t matches `[A-Z]+[a-z]+[A-Z]+[a-z]` w_t matches `.*[0-9].*` w_t appears in list of first names, last names, company names, days, months, or geographic entities w_t is contained in a lexicon of words with POS T (from Brill tagger)
$T_t = T$
$q_k(\mathbf{x}, t+\delta)$ for all k and $\delta \in [-3, 3]$

Table 5 Input features $q_k(\mathbf{x}, t)$ for the CoNLL data. In the above w_t is the word at position t, T_t is the POS tag at position t, w ranges over all words in the training data, and T ranges over all part-of-speech tags.

Now, in a linear-chain model, piecewise training has the same computational complexity as exact CRF training, so we do not mean to advocate the piecewise approximation for linear-chain graphs. Rather, that the piecewise approximation loses no accuracy on the linear-chain model is encouraging when we turn to loopy models, which we do next.

5.1.2 Factorial CRF The first loopy model we consider is the *factorial CRF* introduced in Sutton et al. [2007]. As in that work, we consider the task of jointly predicting part-of-speech tags and segmenting noun phrases on the CoNLL 2000 data set. The structure of this model is a pair of coupled chains. If $\mathbf{w} = (w_1, w_2, \ldots w_T)$ denotes the part-of-speech labels and $\mathbf{y} = (y_1, y_2, \ldots y_T)$ denotes the noun-phrase labels, then the model is

$$p(\mathbf{y}, \mathbf{w}|\mathbf{x}) = \prod_{t=1}^{T} \Psi_0(y_t, y_{t-1}, \mathbf{x}) \Psi_1(w_t, w_{t-1}, \mathbf{x}) \Psi_2(w_t, y_t, \mathbf{x}), \qquad (42)$$

where each type of factor Ψ_0, Ψ_1, Ψ_2 is expressed in the log-linear form of (2). The features used are described in Table 5.

We report results here on subsets of 223 training sentences, and the standard test set of 2012 sentences. Results are averaged over 5 different random subsets. There are 45 different POS labels, and the three NP labels. We report F1 on noun-phrase chunks.

In previous work, this model was optimized by approximating the partition function using belief propagation, but this was quite expensive. Training on the full data set of 8936 sentences required about 12 days of CPU time.[1]

[1] As we mentioned above, this result was obtained using batch limited-memory BFGS. It is likely that stochastic gradient methods would have performed much faster.

$w_t = w$
w_t matches `[A-Z][a-z]+`
w_t matches `[A-Z][A-Z]+`
w_t matches `[A-Z]`
w_t matches `[A-Z]+`
w_t matches `[A-Z]+[a-z]+[A-Z]+[a-z]`
w_t appears in list of first names, last names, honorifics, etc.
w_t appears to be part of a time followed by a dash
w_t appears to be part of a time preceded by a dash
w_t appears to be part of a date
$q_k(\mathbf{x}, t + \delta)$ for all k and $\delta \in [-4, 4]$

Table 6 Input features $q_k(\mathbf{x}, t)$ for the seminars data. In the above w_t is the word at position t, T_t is the POS tag at position t, w ranges over all words in the training data, and T ranges over all Penn Treebank part-of-speech tags. The "appears to be" features are based on hand-designed regular expressions that can span several tokens.

Results on this loopy data set are presented in Table 2. Again, the piecewise estimator performs better both than either version of pseudolikelihood and than maximum-likelihood estimation using belief propagation. On this small subset, approximate ML training with BP requires 1.8 h, but piecewise training is still twice as fast, using 0.83 h.

5.1.3 Skip-chain CRF Finally, we consider a model with many irregular loops, which is the skip chain model introduced in Sutton and McCallum [2004]. This model incorporates certain long-distance dependencies between word labels into a linear-chain model for information extraction. The idea is to exploit that when the same word appears multiple times in the same message, it tends to have the same label. We represent this by adding edges between output nodes (y_i, y_j) when the words x_i and x_j are identical and capitalized.

For an sentence $\mathbf{x}$, let $\mathcal{I} = \{(u, v)\}$ be the set of all pairs of sequence positions for which there are skip edges. For example, in the experiments reported here, $\mathcal{I}$ is the set of indices of all pairs of identical capitalized words. Then the probability of a label sequence $\mathbf{y}$ given an input $\mathbf{x}$ is modeled as

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \Psi_t(y_t, y_{t-1}, \mathbf{x}) \prod_{(u,v)\in\mathcal{I}} \Psi_{uv}(y_u, y_v, \mathbf{x}), \qquad (43)$$

Model	Basic	Reweighted
Linear-chain	91.2	90.4
FCRF	88.1	86.4
Skip-chain (location)	87.7	75.5
Skip-chain (speaker)	75.4	69.2

Table 7 Comparison of basic piecewise training to reweighted piecewise bound with uniform μ.

where Ψ_t are the factors for linear-chain edges, and Ψ_{uv} are the factors over skip edges. These factors are defined as

$$\Psi_t(y_t, y_{t-1}, \mathbf{x}) = \exp\left\{\sum_k \lambda_{1k} f_{1k}(y_t, y_{t-1}, \mathbf{x}, t)\right\} \tag{44}$$

$$\Psi_{uv}(y_u, y_v, \mathbf{x}) = \exp\left\{\sum_k \lambda_{2k} f_{2k}(y_u, y_v, \mathbf{x}, u, v)\right\}, \tag{45}$$

where $\theta_1 = \{\lambda_{1k}\}_{k=1}^{K_1}$ are the parameters of the linear-chain template, and $\theta_2 = \{\lambda_{2k}\}_{k=1}^{K_2}$ are the parameters of the skip template. The full set of model parameters are $\theta = \{\theta_1, \theta_2\}$.

We use a standard data set of seminar announcements [Freitag, 1998]. Consistently with the previous work on this data set, we use 10-fold cross validation with a 50/50 training/test split. We report per-token F1 on the speaker and location fields, the most difficult of the four fields. The features used are described in Table 6. Most documents contain many crossing skip-edges, so that exact maximum-likelihood training using junction tree is completely infeasible, so instead we compare to approximate training using loopy belief propagation.

Results on this model are given in Table 3. Pseudolikelihood performs particularly poorly on this model. Piecewise estimation performs much better, but worse than approximate training using BP.

Piecewise training is faster than loopy BP: in our implementation piecewise training used on average 3.5 hr, while loopy BP used 6.8 hr. To get these loopy BP results, however, we must carefully initialize the training procedure, as discussed in Sutton [2008]. For example, if instead we initialize the model to the uniform distribution, not only does loopy BP training take much longer, over 10 hours, but testing performance is much worse, because the convex optimization procedure has difficulty with noisier gradients. With uniform initialization, loopy BP does not converge for all training instances, especially at early iterations of training. Carefully initializing the model parameters seems to alleviate these issues, but this model-specific tweaking was unnecessary for piecewise training.

5.1.4 Reweighted Piecewise Training We also evaluate a reweighted piecewise training, a modification to the basic piecewise estimator discussed in

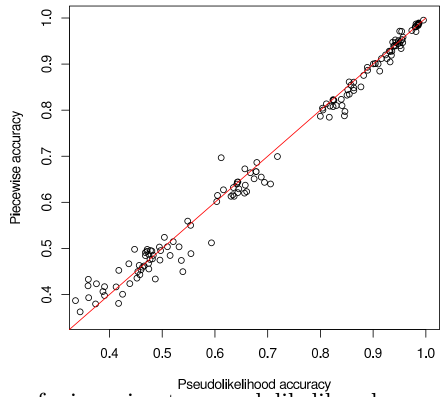

Fig. 3 Comparison of piecewise to pseudolikelihood on synthetic data. Pseudolikelihood has slightly better accuracy on training instances than piecewise. (Piecewise and PWPL perform exactly the same; this is not shown.)

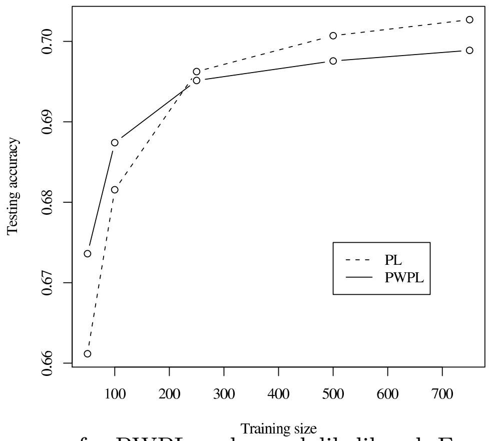

Fig. 4 Learning curves for PWPL and pseudolikelihood. For smaller amounts of training data PWPL performs better than pseudolikelihood, but for larger data sets, the situation is reversed.

Section 3.4, in which the pieces are weighted by a convex combination. The performance of reweighted piecewise training with uniform μ_R is presented in Table 7. In all cases, the reweighted piecewise method performs worse than the basic piecewise method. What seems be happening is that in each of these models, there are several hundred edges, so that the weight μ_R for each region is rather small, perhaps around 0.01. For each piece R, reweighted bound includes a term $A\left(\theta|_R/\mu_R\right)$. If μ_R is around 0.01, then this means that we multiply the log factor values by 100 before evaluating A. This multiplier is so extreme that the term $A\left(\theta|_R/\mu_R\right)$ is dominated by maximum-value weight in $\theta|_R$.

	ML	PL	PW	PWPL
POS				
Accuracy	94.4	94.4	94.2	94.4
Time (s)	33846	6705	23537	**3911**
Chunking				
Chunk F1	91.4	90.3	91.7	91.4
Time (s)	24288	1534	5708	**766**
Named-entity				
Chunk F1	90.5	85.1	90.5	90.3
Time (s)	52396	8651	6311	**4780**

Table 8 Comparison of piecewise pseudolikelihood to standard piecewise and to pseudolikelihood on real-world NLP tasks. Piecewise pseudolikelihood is in all cases comparable to piecewise, and on two of the data sets superior to pseudolikelihood.

	BP	PL	PW	PWPL
START-TIME	96.5	82.2	97.1	94.1
END-TIME	95.9	73.4	96.5	90.4
LOCATION	85.8	73.0	88.1	85.3
SPEAKER	74.5	27.9	72.7	65.0

Table 9 F1 performance of PWPL, piecewise, and pseudolikelihood on information extraction from seminar announcements. Both standard piecewise and piecewise pseudolikelihood outperform pseudolikelihood.

5.2 Piecewise Pseudolikelihood

In this section, we compare PWPL to both regular piecewise and pseudolikelihood on both synthetic and real-world data. On synthetic data (Section 5.2.1), we find that PWPL works better than PL for small amounts of training data, but PL works better when the training set is larger, confirming the intuition resulting from the asymptotic arguments of Section 4.2. Second, on real-world NLP data sets (Section 5.2.2), we find that PWPL performs often comparably to piecewise training and to maximum likelihood, and always better than pseudolikelihood. Furthermore, PWPL can be as much as ten times faster than batch CRF training.

5.2.1 Synthetic Data In the previous section, we argued intuitively that PWPL may perform better on small data sets, and pseudolikelihood on larger ones. In this section we verify this intuition in experiments on synthetic data. The general setup is replicated from Lafferty et al. [2001]. We generate data from a second-order HMM with transition probabilities

$$p_\alpha(y_t|y_{t-1}, y_{t-2}) = \alpha p_2(y_t|y_{t-1}, y_{t-2}) + (1-\alpha)p_1(y_t|y_{t-1}) \qquad (46)$$

and emission probabilities

$$p_\alpha(x_t|y_t, x_{t-1}) = \alpha p_2(x_t|y_t, x_{t-1}) + (1-\alpha)p_1(x_t|y_t). \qquad (47)$$

Thus, for $\alpha = 0$, the generating distribution p_α is a first-order HMM, and for $\alpha = 1$, it is an autoregressive second-order HMM. We compare different approximate methods for training a first-order CRF. Therefore higher values of α make the learning problem more difficult, because the model family does not contain second-order dependencies. We use five states and 26 possible observation values. For each setting of α, we sample 25 different generating distributions. From each generating distribution we sample 1,000 training instances of length 25, and 1,000 testing instances. We use $\alpha \in \{0, 0.1, 0.25, 0.5, 0.75, 1.0\}$, for 150 synthetic generating models in all.

First, we find that piecewise pseudolikelihood performs almost identically to standard piecewise training. Averaged over the 150 data sets, the mean difference in testing error between piecewise pseudolikelihood and piecewise is 0.002, and the correlation is 0.999.

Second, we compare piecewise to traditional pseudolikelihood. On this data, pseudolikelihood performs slightly better overall, but the difference is not statistically significant (paired t-test; $p > 0.1$). However, when we examine the accuracy as a function of training set size (Figure 4), we notice an interesting two-regime behavior. Both PWPL and pseudolikelihood seem to be converging to a limit, and the eventual pseudolikelihood limit is higher than PWPL, but PWPL converges to its limit faster. This is exactly the behavior intuitively predicted by the argument in Section 4.2: that PWPL can converge to the piecewise solution in less training data than pseudolikelihood to its (potentially better) solution.

Of course, the training set sizes considered in Figure 4 are fairly small, but this is exactly the case we are interested in, because on natural language tasks, even when hundreds of thousands of words of labeled data are available, this is still a small amount of data compared to the number of useful features.

5.2.2 Real-World Data Now, we evaluate piecewise pseudolikelihood on four real-world NLP tasks: part-of-speech tagging, named-entity recognition, noun-phrase chunking, and information extraction.

For *part-of-speech tagging (POS)*, we report results on the WSJ Penn Treebank data set. Results are averaged over five different random subsets of 1911 sentences, sampled from Sections 0–18 of the Treebank. Results are reported from the standard development set of Sections 19–21 of the Treebank. We use a first-order linear chain CRF. There are 45 part-of-speech labels.

In *named-entity recognition*, the task is to find proper nouns in text. We use the CoNLL 2003 data set, consisting of 14,987 newswire sentences annotated with names of people, organizations, locations, and miscellaneous entities. We test on the standard development set of 3,466 sentences. Evaluation is done using precision and recall on the extracted chunks, and we report $F_1 = 2PR/(P + R)$. We use a linear-chain CRF, whose features are described in Table 4.

For the task of *noun-phrase chunking (chunking)*, we use a loopy model, the *factorial CRF* described in Section 5.1.2, on the task of joint part-of-speech and noun-phrase prediction. We report joint accuracy on (NP, POS) pairs; other evaluation metrics show similar trends.

Finally, for the task of *information extraction*, we consider a model with many irregular loops, which is the skip chain model described in Section 5.1.3.

For all the data sets, we compare to pseudolikelihood, piecewise training, and conditional maximum likelihood with belief propagation. All of these objective functions are maximized using limited-memory BFGS. We use a Gaussian prior with variance $\sigma^2 = 10$.

5.2.3 Discussion For the first three tasks—part-of-speech tagging, chunking, and NER—piecewise pseudolikelihood and standard piecewise training have equivalent accuracy both to each other and to maximum likelihood (Table 8). Despite this, piecewise pseudolikelihood is much more efficient than standard piecewise (Table 8). On the named-entity data, which has the fewest labels, PWPL uses 75% of the time of standard piecewise, a modest improvement. On the data sets with more labels, the difference is more dramatic: on the POS data, PWPL uses 16% of the time of piecewise and on the chunking data, PWPL needs only 13%. Similarly, PWPL is also between is 5 to 10 times faster than maximum likelihood.

The training times of the baseline methods may appear relatively modest. If so, this is because for both the chunking and POS data sets, we use relatively small subsets of the full training data, to make running this comparison more convenient. This makes the absolute difference in training time even more meaningful than it may appear at first. Also, it may appear from Table 8 that PWPL is faster than standard pseudolikelihood, but the apparent difference is due to low-level inefficiencies in our implementation. In fact the two algorithms have similar complexity.

On the skip chain data (Table 9), standard piecewise performs worse than exact training using BP, and piecewise pseudolikelihood performs worse than standard piecewise. Both piecewise methods, however, perform better than pseudolikelihood.

As predicted in Section 4.2, pseudolikelihood is indeed a better approximation on the node-split graph. In Table 8, PL performs much worse than ML, but PWPL performs only slightly worse than PW. In Table 9, the difference between PWPL and PW is larger, but still less than the difference between PL and ML.

6 Related Work

Piecewise training and piecewise pseudolikelihood can both be considered types of *local training* methods, that avoid propagation throughout the graph. Such training methods have recently been the subject of much interest [Abbeel et al., 2005, Toutanova et al., 2003, Punyakanok et al., 2005]. Of course, the local training method most closely connected to the current work is pseudolikelihood itself. We are unaware of previous variants of pseudolikelihood that condition on less than the full Markov blanket.

Because the piecewise estimator is such an intuitively appealing method, it has been used in several scattered places in the literature, for tasks such as information extraction [Wellner et al., 2004], collective classification [Greiner et al., 2005], and computer vision [Freeman et al., 2000]. In these papers, the piecewise method is reported as a successful heuristic for training large models, but its performance is not compared against other training methods. We are unaware of previous work systematically studying this procedure in its own right.

As mentioned earlier, the most closely related procedure that has been studied statistically is pseudolikelihood [Besag, 1975, 1977]. The main difference is that piecewise training does not condition on neighboring nodes, but ignores them altogether during training. This is depicted schematically by the factor graphs in Figure 5. In pseudolikelihood, each locally-normalized term for a variable or edge in pseudolikelihood includes contributions from a number of factors that connect to the neighbors whose observed values are taken from labeled training data. All these factors are circled in the top section of Figure 5. In piecewise training, each factor becomes an independently, locally-normalized term in the objective function.

Huang and Ogata [1999] consider the asymptotic properties of generalized pseudolikelihood estimators whose local conditional terms predict subgraphs of the model rather than single nodes. A special case of this is the pairwise pseudolikelihood that we compare against in Section 5.1.

Also, in statistics there has been work on general families of surrogate likelihoods, called composite likelihoods, which are sums of marginal or conditional log likelihoods [Lindsay, 1988, Cox and Reid, 2004]. Such composite likelihoods are consistent and asymptotically normal under relatively general assumptions. An example of using a composite likelihood on structured models for natural-language data is Kakade et al. [2002]. But these are designed for a different situation than ours, namely when the joint likelihoods are difficult to compute but marginal likelihoods are easier to work with. An example of this situation is the multivariate Gaussian. In our context, marginal likelihoods are difficult to compute, so composite likelihoods are not as useful. Piecewise estimation is not a type of composite likelihood, because in the likelihood of each piece, the contribution of the rest of the model is ignored, not marginalized out.

Independently, Choi et al. [2007] present a node-splitting technique for upper bounds during inference, which is closely related to the piecewise upper bound that we use here for learning.

An interesting connection exists between piecewise pseudolikelihood and maximum entropy Markov models (MEMMs) [Ratnaparkhi, 1996, McCallum et al., 2000]. In a linear chain with variables $y_1 \dots y_T$, we can rewrite the piecewise pseudolikelihood as

$$\ell_{\text{PWPL}}(\theta) = \sum_{t=1}^{T} \log p_{\text{LCL}}(y_t | y_{t-1}, \mathbf{x}) p_{\text{LCL}}(y_{t-1} | y_t, \mathbf{x}). \tag{48}$$

The first part of (48) is exactly the likelihood for an MEMM, and the second part is the likelihood of a backward MEMM. Interestingly, MEMMs crucially depend on normalizing the factors at both training and test time. To include local normalization at training time but not test time performs very poorly. But by adding the backward terms, in PWPL we are able to drop normalization at test time, and therefore PWPL does not suffer from label bias.

Piecewise pseudolikelihood also has an interesting connection to search-based learning methods [Daumé III and Marcu, 2005]. Such methods learn a model to predict the next state of a local search procedure from a current state. Typically, training is viewed as classification, where the correct next states are positive examples, and alternative next states are negative examples. One view of the current work is that it incorporates backward training examples, that attempt to predict the *previous* search state given the current state.

Finally, stochastic gradient methods, which make gradient steps based on subsets of the data, have recently been shown to converge significantly faster for CRF training than batch methods, which evaluate the gradient of the entire data set before updating the parameters [Vishwanathan et al., 2006]. Stochastic gradient methods are currently the method of choice for training linear-chain CRFs, especially when the data set is large and redundant. However, stochastic gradient methods can also be used to optimize both standard piecewise likelihood and piecewise pseudolikelihood. Thus, although the training time of our baselines could likely be improved considerably, the same is true of our new approaches, so that our comparison is fair.

Also, in some cases, such as in relational learning problems, the data are not iid, and the model includes explicit dependencies between the training instances. For such a model, it is unclear how to apply stochastic gradient, but piecewise pseudolikelihood may still be useful. Finally, stochastic gradient methods do not address cases in which the variables have large cardinality, or when the graphical structure of a single training instance is intractable.

One challenge in the piecewise method is that it is not immediately clear how to extend it to the case of latent variables, because if the same

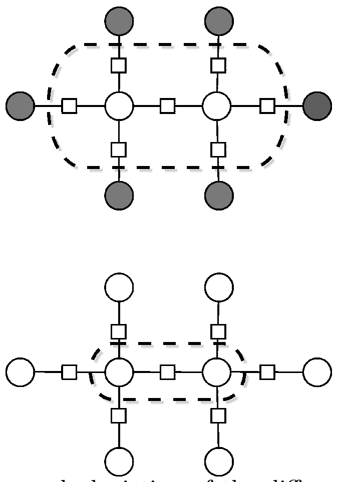
Fig. 5 Schematic factor-graph depiction of the difference between pseudolikelihood (top) and piecewise training (bottom). Each term in pseudolikelihood normalizes the product of many factors (as circled), while piecewise training normalizes over one factor at a time.

latent variable occurs in different pieces, it may have different semantics. Recently, Liang et al. [2006, 2008] have presented a method in a similar spirit that handles latent variables. They add an additional term to the objective function that encourages pieces that share a latent variable to agree on its distribution. They present two different EM-style algorithms for optimizing this objective function efficiently.

Both likelihood-based methods and max-margin methods require performing inference during training, so it is natural to wonder whether the methods in this thesis can be adapted to loopy max-margin models. A suggestive step in this direction is *factorized MIRA* [McDonald et al., 2005], in which the margin constraints are required to hold only over single edges, rather than the entire prediction. On a dependency parsing task, this method had good accuracy, but it did not improve training time because the model had special structure that made it amenable to exact inference. It may be interesting to see whether analogs of piecewise methods work well for max-margin training on loopy models.

Earlier versions of the current work have appeared in two conference papers [Sutton and McCallum, 2005, 2007b].

7 Conclusion

This paper has presented piecewise training, an intuitively appealing procedure that separately trains factor subsets, called pieces, of a loopy graph. We show that this procedure can be justified as maximizing a loose bound on the log likelihood. On three real-world language tasks with different model structures, piecewise training outperforms several versions of pseudolikelihood, a traditional local training method. On two of the data sets, in fact, piecewise training is more accurate than global training using belief propagation.

Second, we have introduced an extension called piecewise pseudolikelihood, that is especially attractive when the variables in the model have large cardinality. Because PWPL conditions on fewer variables, it can have better accuracy than standard pseudolikelihood, and is dramatically more efficient than standard piecewise, requiring as little as 13% of the training time.

Many properties of piecewise training remain to be explored. Our results indicate that in some situations piecewise training should replace pseudolikelihood as the local training method of choice. In particular, the experiments here all used conditional training, which make local training easier because of the large amount of information in the conditioning variables. In the data sets here, the local features, such as the word identity, provide a large amount of information about the labels in their own right. In generative training, there may be much less local information, making piecewise training much less effective. On the other hand, from the exponential family perspective, piecewise training does still match expected statistics of a subgraph to the empirical distribution, which still seems intuitively appealing. For this reason, it is hard to give a definitive characterization of when piecewise training is expected to work well or poorly.

A possible explanation for the performance of piecewise training is that it acts as a form of additional regularization, in that the objective function disfavors parameter settings that obtain good joint likelihood by using long-distance effects of weights. For this reason, connections to generalization bounds for feature selection, some of which take into account the amount of computation, may be interesting.

Finally, a natural question is how readily the methods here extend to the case where pieces are treated as regions that are larger than a single factor. The main challenges here seem to lie in first, how to handle the overlaps among larger pieces, and second, how to choose the pieces. The connection to the Bethe energy may be illuminating here.

Acknowledgments

We thank an anonymous reviewer, and independently Drew Bagnell and David Blei, for pointing out the simple proof of Proposition 1. An early draft had only the more complicated proof sketched in Section 3.4. Also, we thank David Rosenberg for alerting us to a discrepancy that led to my finding an error in an earlier version of the experiments in this paper. We thank Tom Minka for pointing out the connection between piecewise training and belief propagation. Finally, Sutton thanks the members of his dissertation committee—Tommi Jaakkola, Erik Learned-Miller, Jonathan Machta, and Sridhar Mahadevan—for valuable comments on this work.

References

Pieter Abbeel, Daphne Koller, and Andrew Y. Ng. Learning factor graphs in polynomial time and sample complexity. In *Twenty-first Conference on Uncertainty in Artificial Intelligence (UAI05)*, 2005.

Axel Bernal, Koby Crammer, Artemis Hatzigeorgiou, and Fernando Pereira. Global discriminative learning for higher-accuracy computational gene prediction. *PLoS Computational Biology*, 3(3), 2007.

Julian Besag. Statistical analysis of non-lattice data. *The Statistician*, 24 (3):179–195, 1975.

Julian Besag. Efficiency of pseudolikelihood estimation for simple Gaussian fields. *Biometrika*, 64(3):616–618, 1977. URL `http://links.jstor.org/sici?sici=0006-3444\%28197712\%2964\%3A3\%3C616\%3AEOPEFS\%3E2.0.CO\%3B2-Y`.

Arthur Choi, Mark Chavira, and Adnan Darwiche. Node splitting: A scheme for generating upper bounds in Bayesian networks. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007.

D. R. Cox and N. Reid. A note on pseudolikelihood constructed from marginal densities. *Biometrika*, 91:729–737, 2004.

Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, Jan 2003.

Hal Daumé III and Daniel Marcu. Learning as search optimization: Approximate large margin methods for structured prediction. In *International Conference on Machine Learning (ICML)*, Bonn, Germany, 2005. URL `http://pub.hal3.name/#daume05laso`.

Jenny Rose Finkel, Christopher D. Manning, and Andrew Y. Ng. Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines. In *Conference on Empirical Methods in Natural Language Proceeding (EMNLP)*, 2006.

William T. Freeman, Egon C. Pasztor, and Owen T. Carmichael. Learning low-level vision. *International Journal of Computer Vision*, 40(1):24–57, 2000.

Dayne Freitag. *Machine Learning for Information Extraction in Informal Domains*. PhD thesis, Carnegie Mellon University, 1998.

Basilis Gidas. Consistency of maximum likelihood and pseudolikelihood estimators for Gibbs distributions. In Wendell Fleming and Pierre-Louis Lions, editors, *Stochastic Differential Systems, Stochastic Control Theory and Applications*. Springer, New York, 1988.

Russ Greiner, Yuhong Guo, and Dale Schuurmans. Learning coordination classifiers. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.

Fuchun Huang and Yosihiko Ogata. Improvements of the maximum pseudo-likelihood estimators in various spatial statistical models. *Journal of Computational and Graphical Statistics*, 8(3):510–530, 1999. ISSN 10618600. URL `http://www.jstor.org/stable/1390872`.

Aapo Hyvarinen. Consistency of pseudolikelihood estimation of fully visible Boltzmann machines. *Neural Computation*, 18(10):2283–2292, 2006.

Sham Kakade, Yee Whye Teh, and Sam Roweis. An alternative objective function for Markovian fields. In *In Proceedings of the Nineteenth International Conference on Machine Learning*, 2002.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *International Conference on Machine Learning (ICML)*, 2001.

Stan Z. Li. *Markov Random Field Modeling in Image Analysis*. Springer-Verlag, 2001.

P. Liang, B. Taskar, and D. Klein. Alignment by agreement. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*, 2006.

P. Liang, D. Klein, and M. I. Jordan. Agreement-based learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

Bruce G. Lindsay. Composite likelihood methods. *Contemporary Mathematics*, pages 221–239, 1988.

Andrew McCallum, Dayne Freitag, and Fernando Pereira. Maximum entropy Markov models for information extraction and segmentation. In *International Conference on Machine Learning (ICML)*, pages 591–598. Morgan Kaufmann, San Francisco, CA, 2000.

Ryan McDonald, Koby Crammer, and Fernando Pereira. Spanning tree methods for discriminative training of dependency parsers. Technical Report MS-CIS-05-11, University of Pennsylvania CIS, 2005.

Thomas Minka. Expectation propagation for approximate Bayesian inference. In *17th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 362–369, 2001a.

Thomas P. Minka. The EP energy function and minimization schemes. `http://research.microsoft.com/~minka/papers/ep/minka-ep-energy.pdf`, 2001b.

Tom Minka. Divergence measures and message passing. Technical Report MSR-TR-2005-173, Microsoft Research, 2005.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer-Verlag, New York, 1999. ISBN 0-387-98793-2.

Sridevi Parise and Max Welling. Learning in Markov random fields: An empirical study. In *Joint Statistical Meeting (JSM2005)*, 2005.

Vasin Punyakanok, Dan Roth, Wen-tau Yih, and Dav Zimak. Learning and inference over constrained output. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1124–1129, 2005.

Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Conference on Empirical Methods in Natural Language Proceeding (EMNLP)*, 1996.

Michal Rosen-Zvi, Alan L. Yuille, and Michael I. Jordan. The dlr hierarchy of approximate inference. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005.

David H. Stern, Thore Graepel, and David J. C. MacKay. Modelling uncertainty in the game of go. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1353–1360. MIT Press, Cambridge, MA, 2005.

Charles Sutton. *Efficient Training Methods for Conditional Random Fields*. PhD thesis, University of Massachusetts, 2008. URL `publications/sutton-thesis.pdf`.

Charles Sutton and Andrew McCallum. Collective segmentation and labeling of distant entities in information extraction. In *ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields*, 2004.

Charles Sutton and Andrew McCallum. Piecewise training of undirected models. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005. URL `publications/tip.pdf`.

Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007a.

Charles Sutton and Andrew McCallum. Piecewise pseudolikelihood for efficient CRF training. In *International Conference on Machine Learning (ICML)*, 2007b. URL `publications/pwpl.pdf`.

Charles Sutton and Tom Minka. Local training and belief propagation. Technical Report TR-2006-121, Microsoft Research, 2006.

Charles Sutton, Khashayar Rohanimanesh, and Andrew McCallum. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *International Conference on Machine Learning (ICML)*, 2004.

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research*, 8:693–723, March 2007. URL `publications/jmlr-sutton07a.pdf`.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004a.

Ben Taskar, Dan Klein, Michael Collins, Daphne Koller, and Christopher Manning. Max-margin parsing. In *Empirical Methods in Natural Language Processing (EMNLP04)*, 2004b.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL*, 2003.

S.V.N. Vishwanathan, Nicol N. Schraudolph, Mark W. Schmidt, and Kevin Murphy. Accelerated training of conditional random fields with stochastic meta-descent. In *International Conference on Machine Learning (ICML)*, pages 969–976, 2006.

Martin J. Wainwright, Tommi Jaakkola, and Alan S. Willsky. A new class of upper bounds on the log partition function. In *Uncertainty in Artificial*

Intelligence, 2002.

Martin J. Wainwright, Tommi Jaakkola, and Alan S. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on Information Theory*, 45(9):1120–1146, 2003a.

Martin J. Wainwright, Tommi Jaakkola, and Alan S. Willsky. Tree-reweighted belief propagation and approximate ML estimation by pseudo-moment matching. In *Ninth Workshop on Artificial Intelligence and Statistics*, 2003b.

Ben Wellner, Andrew McCallum, Fuchun Peng, and Michael Hay. An integrated, conditional model of information extraction and coreference with application to citation graph construction. In *20th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2004.

Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Constructing free energy approximations and generalized belief propagation algorithms. Technical Report TR2004-040, Mitsubishi Electric Research Laboratories, 2004.