Hidden-Unit Conditional Random Fields

Laurens van der Maaten
Delft University of Technology
lvdmaaten@gmail.com

Max Welling
University of California, Irvine
welling@ics.uci.edu

Lawrence K. Saul
University of California, San Diego
saul@cs.ucsd.edu

Abstract

The paper explores a generalization of conditional random fields (CRFs) in which binary stochastic hidden units appear between the data and the labels. *Hidden-unit CRFs* are potentially more powerful than standard CRFs because they can represent nonlinear dependencies at each frame. The hidden units in these models also learn to discover latent distributed structure in the data that improves classification. We derive efficient algorithms for inference and learning in these models by observing that the hidden units are conditionally independent given the data and the labels. Finally, we show that hidden-unit CRFs perform well in experiments on a range of tasks, including optical character recognition, text classification, protein structure prediction, and part-of-speech tagging.

1 INTRODUCTION

Since their inception approximately a decade ago, Conditional Random Fields (CRFs; Lafferty et al. (2001); Sutton and McCallum (2007)) have become a popular technique for the classification and segmentation of time series. Among others, CRFs have been successfully applied to various problems in natural language processing (Collins, 2003; Sha and Pereira, 2003), speech recognition (Cheng et al., 2009), bioinformatics (Liu et al., 2005), and computer vision (He et al., 2004; Kumar and Hebert, 2003). There are three key advantages of CRFs over the traditional Hidden Markov Models (HMMs) that explain their success: (1) unlike HMMs, CRFs do not assume that the observations are conditional independent given the hidden states; (2) unlike HMMs, CRFs do not suffer from the *label bias* problems of models that do local probability normalization; (3) for certain choices of factors, the negative conditional log-likelihood that is minimized when training CRFs is a convex objective function.

The most popular CRFs are those with linear state transition factors and linear data-dependent factors. While the convexity of such linear CRFs is appealing, the use of linear factors strongly restricts the mappings that the CRFs can represent, as they can only represent linear decision boundaries. More recently, some work has been performed on non-linear data-dependent factors, such as (mixtures of) quadratic factors (Cheng et al., 2009; Sha and Saul, 2009), kernel-based factors (Lafferty et al., 2004), and factors based on multilayer perceptrons (LeCun et al., 1998; Peng et al., 2010). Quadratic factors have the disadvantage that they cannot be employed in domains with large numbers of features (such as natural language processing), whereas kernel-based factors do not scale well in the number of training instances. Factors based on multilayer perceptrons lead to non-linearities in the energy function, which makes optimization tedious. As an alternative, some studies have trained CRFs on feature representations learned by unsupervised learners (Do and Artieres, 2010; Mohamed et al., 2009); however, features discovered by unsupervised learning are not necessarily optimized for classification.

In parallel with work on non-linear data-dependent factors in CRFs, there has been some success in the discriminative training of Restricted Boltzmann Machines (RBMs; Larochelle and Bengio (2008); Gelfand et al. (2010)). Discriminative RBMs are logistic regressors that have binary stochastic hidden units, which are conditionally independent given the data and the labels. The binary hidden units model latent features of the data that improve classification. Although maximum conditional likelihood learning in discriminative RBMs is non-convex, these models performed well in several benchmarks (Larochelle and Bengio, 2008) and real-world applications (Schmah et al., 2009).

Appearing in Proceedings of the 14^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2011, Fort Lauderdale, FL, USA. Volume 15 of JMLR: W&CP 15.

In this paper, we introduce a new model, called hidden-unit CRF, that – similar to discriminative RBMs – has binary stochastic hidden units that are conditionally independent given the data and the label sequence[1]. By exploiting properties of conditional independence, we can efficiently compute (1) the exact gradient of the conditional log-likelihood, (2) the most likely label sequence for a given time series, and (3) marginal distributions over label sequences. At the same time, the binary hidden units drastically expand the decision boundaries that can be learned by the CRFs. In fact, it can be shown that the individual predictors (i.e., predictors that ignore the temporal correlations) are universal approximators for discrete data. We develop various training algorithms for hidden-unit CRFs, and we show that hidden-unit CRFs perform well in experiments on optical character recognition, sentence labeling, protein secondary structure prediction, and part-of-speech tagging.

2 HIDDEN-UNIT CRFS

In the paper, we assume we are given a time series $\mathbf{x}_{1,\ldots,T}$ of length T with observations $\mathbf{x}_t$ of dimensionality D, and a corresponding label sequence $\mathbf{y}_{1,\ldots,T}$ with label vectors $\mathbf{y}_t$ that use a so-called 1-of-K encoding (i.e., $y_{tk} \in \{0, 1\}$ and $\sum_k y_{tk} = 1$). Linear CRFs (Lafferty et al., 2001) model the conditional distribution over the label sequence given the data as

$$p(\mathbf{y}_{1,\ldots,T}|\mathbf{x}_{1,\ldots,T}) = \frac{\exp\left\{E(\mathbf{x}_{1,\ldots,T}, \mathbf{y}_{1,\ldots,T})\right\}}{Z(\mathbf{x}_{1,\ldots,T})},$$

where $Z(\mathbf{x}_{1,\ldots,T})$ is the partition function

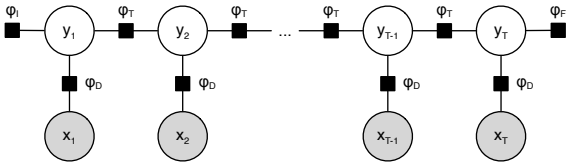
$$Z(\mathbf{x}_{1,\ldots,T}) = \sum_{\mathbf{y}'_{1,\ldots,T}} \exp\left\{E(\mathbf{x}_{1,\ldots,T}, \mathbf{y}'_{1,\ldots,T})\right\},$$

and the energy function $E(\mathbf{x}_{1,\ldots,T}, \mathbf{y}_{1,\ldots,T})$ is given by

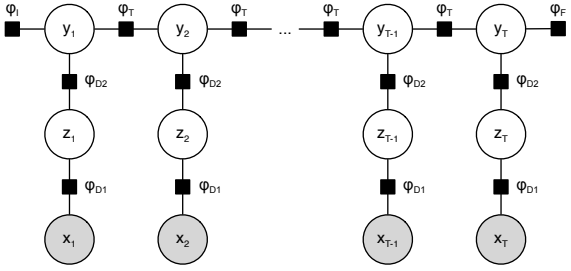
$$E(\mathbf{x}_{1,\ldots,T}, \mathbf{y}_{1,\ldots,T}) = \sum_{t=2}^{T} \left[\mathbf{y}_{t-1}^T \mathbf{A} \mathbf{y}_t\right] + \sum_{t=1}^{T} \left[\mathbf{x}_t^T \mathbf{W} \mathbf{y}_t\right].$$

In the above, $\mathbf{A}$ represents the state transition parameters and $\mathbf{W}$ represents the parameters of the data-dependent term. We left out an initial-state factor $\mathbf{y}_1^T \boldsymbol{\pi}$, a final-state factor $\mathbf{y}_T^T \boldsymbol{\tau}$, and T label bias terms $\mathbf{c}^T \mathbf{y}_t$. The factor graph of linear CRFs is depicted in Figure 1(a). One of the main disadvantages of the linear CRF is the linear nature of the data-dependent term: if the state transition factors provide a uniform input, the model reduces to a collection of simple linear logistic regressors.

[1]The reader should not confuse hidden-unit CRFs with the "hidden CRF" (Quattoni et al., 2010). Hidden CRFs only assign a single label to a complete time series.



(a) Linear CRF.



(b) Hidden-unit CRF.

Figure 1: Factor graphs of the linear CRF and the hidden-unit CRF. Shaded circles represent variables that are conditioned on.

To address the linear nature of the data-dependent factors of linear CRFs, we propose the hidden-unit CRF. At every time step t, the hidden-unit CRF employs H binary stochastic hidden units $\mathbf{z}_t$. The hidden units are conditionally independent given the data $\mathbf{x}_{1,\ldots,T}$ and the labels $\mathbf{y}_{1,\ldots,T}$. The factor graph of the hidden-unit CRF is depicted in Figure 1(b). The hidden-unit CRF models the conditional distribution over the label sequence $\mathbf{y}_{1,\ldots,T}$ given the data $\mathbf{x}_{1,\ldots,T}$ as follows

$$p(\mathbf{y}_{1,\ldots,T}|\mathbf{x}_{1,\ldots,T}) = \frac{1}{Z(\mathbf{x}_{1,\ldots,T})} \sum_{\mathbf{z}_{1,\ldots,T}} \exp\left\{E(\mathbf{x}_{1,\ldots,T}, \mathbf{z}_{1,\ldots,T}, \mathbf{y}_{1,\ldots,T})\right\},$$

where the partition function is now given by

$$Z(\mathbf{x}_{1,\ldots,T}) = \sum_{\mathbf{y}'_{1,\ldots,T}} \sum_{\mathbf{z}'_{1,\ldots,T}} \exp\left\{E(\mathbf{x}_{1,\ldots,T}, \mathbf{z}'_{1,\ldots,T}, \mathbf{y}'_{1,\ldots,T})\right\}.$$

In the hidden-unit CRF, the initial-state, final-state, and state transition factors are still standard linear factors, but the data-dependent factor now consists of two main parts, which are both defined to be linear functions. The energy function is given by

$$E(\mathbf{x}_{1,\ldots,T}, \mathbf{z}_{1,\ldots,T}, \mathbf{y}_{1,\ldots,T}) = \mathbf{y}_1^T \boldsymbol{\pi} + \mathbf{y}_T^T \boldsymbol{\tau} + \sum_{t=2}^{T} \left[\mathbf{y}_{t-1}^T \mathbf{A} \mathbf{y}_t\right] + \sum_{t=1}^{T} \left[\mathbf{x}_t^T \mathbf{W} \mathbf{z}_t + \mathbf{z}_t^T \mathbf{V} \mathbf{y}_t + \mathbf{b}^T \mathbf{z}_t + \mathbf{c}^T \mathbf{y}_t\right].$$

The aim of the hidden units is to construct a distributed representation of the data that captures the underlying structure of that data, similarly to multilayer perceptrons or (discriminative) RBMs.

An important property of the hidden-unit CRF is that the evaluation of the predictive distribution $p(\mathbf{y}_{1,\ldots,T}|\mathbf{x}_{1,\ldots,T})$ is tractable. To infer the predictive distribution, we need to marginalize out the hidden units $\mathbf{z}_{1,\ldots,T}$. Because the hidden units are conditionally independent given the data and the labels, the hidden units can be marginalized out one-by-one (Larochelle and Bengio, 2008; Salakhutdinov et al., 2007). Marginalizing out the hidden units gives

$$p(\mathbf{y}_{1\ldots T}|\mathbf{x}_{1\ldots T}) = \frac{\exp\left\{\mathbf{y}_1^T\boldsymbol{\pi} + \mathbf{y}_T^T\boldsymbol{\tau}\right\}}{Z(\mathbf{x}_{1,\ldots,T})} \prod_{t=1}^{T}\Bigg[\exp\left\{\mathbf{c}^T\mathbf{y}_t + \mathbf{y}_{t-1}^T\mathbf{A}\mathbf{y}_t\right\} \prod_{h=1}^{H}\sum_{z_{th}\in\{0,1\}}\exp\left\{z_{th}\left(b_h + \mathbf{w}_h^T\mathbf{x}_t + \mathbf{v}_h^T\mathbf{y}_t\right)\right\}\Bigg]$$

$$= \frac{\exp\left\{\mathbf{y}_1^T\boldsymbol{\pi} + \mathbf{y}_T^T\boldsymbol{\tau}\right\}}{Z(\mathbf{x}_{1,\ldots,T})} \prod_{t=1}^{T}\Bigg[\exp\left\{\mathbf{c}^T\mathbf{y}_t + \mathbf{y}_{t-1}^T\mathbf{A}\mathbf{y}_t\right\} \prod_{h=1}^{H}\left(1 + \exp\left\{b_h + \mathbf{w}_h^T\mathbf{x}_t + \mathbf{v}_h^T\mathbf{y}_t\right\}\right)\Bigg],$$

where $\mathbf{w}_h$ denotes the h-th row of $\mathbf{W}$, and $\mathbf{v}_h$ denotes the h-th row of $\mathbf{V}$. We observe that the predictive distribution for a single label $\mathbf{y}_t$ comprises a product of a number of terms. The terms outside the product over hidden units also appear in the predictive distribution of linear CRFs; they measure the prior probability of a label given the preceding label. The term inside the product over hidden units is always larger than one. As a result, none of the hidden units can "veto" a decision: if a hidden unit does not like a decision, its value in the product over hidden units is roughly one, and it will thus be ignored in the product over hidden units. At the same time, a single hidden unit can strongly suggest a particular decision if all three terms inside the exponent are large. The first of these three terms, b_h, measures the "importance" of hidden unit h. The second term, $\mathbf{w}_h^T\mathbf{x}_t$, measures to what extent hidden unit h "likes" the observed data vector. The third term, $\mathbf{v}_h^T\mathbf{y}_t$, determines to what extent hidden unit h "likes" each of the possible labels. Hence, *if* the hidden unit is important, *and* it models the data well, *and* it likes a particular label, the contribution of that hidden unit to the overall product explodes. As a result, the hidden-unit CRF can assign a label based on a small number of hidden units with high output.

This makes the hidden-unit CRF much more powerful than the linear CRF[2]. In fact, the predictors for the individual labels (i.e., ignoring the temporal correlations) are universal approximators when the data is discrete (the proof of this universal approximation theorem is similar to that presented by Roux and Bengio (2008)).

In addition to the marginalization over the hidden units, the evaluation of $p(\mathbf{y}_{1\ldots T}|\mathbf{x}_{1\ldots T})$ requires evaluation of the partition function $Z(\mathbf{x}_{1,\ldots,T})$, i.e., it requires summing over all possible label sequences. Because of the chain structure of the model, the partition function factorizes into a sum that is linear (and not exponential) in T. Defining[3] potentials

$$\Psi_t(\mathbf{x}_t, \mathbf{z}_t, \mathbf{y}_{t-1}, \mathbf{y}_t) = \exp\left\{\mathbf{y}_{t-1}^T\mathbf{A}\mathbf{y}_t + \mathbf{x}_t^T\mathbf{W}\mathbf{z}_t + \mathbf{z}_t^T\mathbf{V}\mathbf{y}_t\right\},$$

the sum over all possible label sequences factorizes as

$$\begin{aligned} Z(\mathbf{x}_{1,\ldots,T}) &= \sum_{\mathbf{y}_{1\ldots T}}\prod_{t=1}^{T}\sum_{\mathbf{z}_t}\Psi_t(\mathbf{x}_t, \mathbf{z}_t, \mathbf{y}_{t-1}, \mathbf{y}_t) \\ &= \sum_{\mathbf{y}_T}\sum_{\mathbf{y}_{T-1}}\sum_{\mathbf{z}_T}\Psi_T(\mathbf{x}_T, \mathbf{z}_T, \mathbf{y}_{T-1}, \mathbf{y}_T) \\ &\sum_{\mathbf{y}_{T-2}}\sum_{\mathbf{z}_{T-1}}\Psi_{T-1}(\mathbf{x}_{T-1}, \mathbf{z}_{T-1}, \mathbf{y}_{T-2}, \mathbf{y}_{T-1})\sum_{\mathbf{y}_{T-3}}\sum_{\mathbf{z}_{T-2}}\cdots \end{aligned}$$

The two main inferential problems that need to be solved during learning are the computation of the marginal distribution of a label, $\gamma_{tk} = p(y_{tk} = 1|\mathbf{x}_{1,\ldots,T})$, and the distribution over a label edge, $\xi_{tkl} = p(y_{t,k} = 1, y_{t+1,l} = 1|\mathbf{x}_{1,\ldots,T})$. Because of the model's chain structure, these inferential can be solved efficiently using the sum-product algorithm (Bishop, 2006, p. 402). In particular, the forward messages $\boldsymbol{\alpha}_t \propto \sum_{\mathbf{y}_{1,\ldots,t-1}} p(\mathbf{y}_{1,\ldots,T}|\mathbf{x}_{1,\ldots,T})$ and the backward messages $\boldsymbol{\beta}_t \propto \sum_{\mathbf{y}_{t+1,\ldots,T}} p(\mathbf{y}_{1,\ldots,T}|\mathbf{x}_{1,\ldots,T})$ of the sum-product algorithm are given by

$$\alpha_{tk} = \sum_{i=1}^{K}\sum_{\mathbf{z}_t}\Psi_t(\mathbf{x}_t, \mathbf{z}_t, y_{t-1,i} = 1, y_{tk} = 1)\alpha_{t-1,i}$$

$$\beta_{tk} = \sum_{i=1}^{K}\sum_{\mathbf{z}_{t+1}}\Psi_{t+1}(\mathbf{x}_{t+1}, \mathbf{z}_{t+1}, y_{t+1,k} = 1, y_{t+2,i} = 1)\beta_{t+1,i}$$

where $\alpha_1(k) = \pi_k$ and $\beta_T(k) = \tau_k$. Using these messages, the marginal distributions can be expressed as

$$\boldsymbol{\gamma}_t \propto \boldsymbol{\alpha}_t \circ \boldsymbol{\beta}_t$$

$$\boldsymbol{\xi}_t \propto \left[\boldsymbol{\alpha}_t \circ \sum_{\mathbf{z}_{t+1}}\Psi_{t+1}(\mathbf{x}_{t+1}, \mathbf{z}_{t+1}, \mathbf{y}_t, \mathbf{y}_{t+1})\right]\boldsymbol{\beta}_{t+1}^T,$$

where $\circ$ denotes an element-wise product. Note that the forward (or backward) messages can also be used to compute the partition function, as by definition, $Z(\mathbf{x}_{1,\ldots,T}) = \sum_{k=1}^{K}\alpha_{Tk} = \sum_{k=1}^{K}\beta_{1k}$.

[2] We note that when the number of hidden units H is chosen very small compared to the dimensionality D and the number of classes K, the hidden-unit CRF may be simpler than the linear CRF in terms of number of parameters.

[3] To prevent the notation from becoming too cluttered, we ignore initial-state, final-state, and bias terms.

The main inferential problem that needs to be solved at test time is the computation of the most likely label sequence given the data $\mathbf{y}^*_{1\ldots T} = \text{argmax}_{\mathbf{y}'_{1\ldots T}} p(\mathbf{y}'_{1\ldots T}|\mathbf{x}_{1\ldots T})$. Using similar messages, this inferential problem can be solved efficiently using the max-sum algorithm (Bishop, 2006, p. 411).

3 TRAINING

In the previous section, we introduced hidden-unit CRFs and showed how to efficiently solve inferential problems in these models. In this section, we present three approaches to train hidden-unit CRFs: (1) maximum conditional likelihood training, (2) perceptron training, and (3) large-margin perceptron training.

3.1 Maximum Conditional Likelihood

In training hidden-unit CRFs, our aim is to maximize[4] the conditional log-likelihood $\mathcal{L}$, which is given by

$$\begin{aligned}\mathcal{L} &= \log p(\mathbf{y}_{1,\ldots,T}|\mathbf{x}_{1,\ldots,T}) \\ &= \sum_{t=1}^{T} \log\left(\sum_{\mathbf{z}_t} \Psi_t(\mathbf{x}_t, \mathbf{z}_t, \mathbf{y}_{t-1}, \mathbf{y}_t)\right) - \log\left(Z(\mathbf{x}_{1,\ldots,T})\right).\end{aligned}$$

The gradients of the conditional log-likelihood $\mathcal{L}$ with respect to the model parameters can be computed analytically. In particular, defining $\Upsilon = \{\mathbf{W}, \mathbf{V}, \mathbf{b}, \mathbf{c}\}$, the gradient with respect to the data-dependent parameters $\upsilon \in \Upsilon$ is given by

$$\frac{\partial \mathcal{L}}{\partial \upsilon} = \sum_{t=1}^{T}\left[\sum_{k=1}^{K}\left((y_{tk} - \gamma_{tk})\sum_{h=1}^{H}\sigma(o_{hk}(\mathbf{x}_t))\frac{\partial o_{hk}(\mathbf{x}_t)}{\partial \upsilon}\right)\right],$$

where $o_{hk}(\mathbf{x}_t) = b_h + c_k + V_{hk} + \mathbf{w}_h^T\mathbf{x}_t$, and where $\sigma(x)$ represents the sigmoid function $\sigma(x) = \frac{1}{1+\exp(-x)}$. The gradient with respect to the state transition parameters $\mathbf{A}$, the initial-state parameters $\boldsymbol{\pi}$, and the final-state parameters $\boldsymbol{\tau}$ are similar to those in linear CRFs, and are omitted here because of space limitations.

Unlike linear CRFs, the negative conditional log-likelihood of the hidden-unit CRF model is a non-convex function. This implies that we are only guaranteed to converge to a local maximum of the conditional log-likelihood (depending on the selected optimization technique). In our experiments, we perform maximum-likelihood training using (1) an L-BFGS optimizer and (2) a stochastic gradient descent (SGD) optimizer. In preliminary experiments, we also experimented with a conjugate gradient optimizer, but that optimizer did not work as well.

[4] In practice, we often train on multiple time series. All derivations in this section readily extend to such a learning setting.

3.2 Perceptron Training

Perceptron-based training approaches often learn models that are comparable in performance with models learned using maximum likelihood, but at a much lower computational cost (Collins, 2002; Gelfand et al., 2010). Perceptron training is typically computationally more efficient because: (1) it learns good models in few iterations, (2) it uses the Viterbi algorithm instead of the forward-backward algorithm, and (3) parameters only need to be updated for frames that are erroneously classified. Perceptron training approaches aim to minimize the number of misclassified frames by updating the model parameters only when a frame is misclassified. This is achieved by directly performing a type of stochastic gradient descent on the energy gap between the observed label sequence and the predicted label sequence. In the case of the hidden-unit CRF, the update rule for a parameter $\theta \in \Theta = \{\boldsymbol{\pi}, \boldsymbol{\tau}, \mathbf{A}, \mathbf{W}, \mathbf{V}, \mathbf{b}, \mathbf{c}\}$ takes the form

$$\begin{aligned}\theta \leftarrow \theta + \eta_\theta \frac{\partial}{\partial \theta}\big(&E(\mathbf{x}_{1,\ldots,T}, \mathbf{z}^*_{1,\ldots,T}, \mathbf{y}_{1,\ldots,T}) - \\ &E(\mathbf{x}_{1,\ldots,T}, \mathbf{z}^{**}_{1,\ldots,T}, \mathbf{y}^{**}_{1,\ldots,T})\big),\end{aligned}$$

where η_θ is a parameter-specific step size, and where $\mathbf{z}^*_{1,\ldots,T}$ represents the most likely state of the hidden units given the data $\mathbf{x}_{1,\ldots,T}$ and the labels $\mathbf{y}_{1,\ldots,T}$. In the above, $\mathbf{y}^{**}_{1,\ldots,T}$ and $\mathbf{z}^{**}_{1,\ldots,T}$ represent the most likely label sequence and the corresponding most likely state of the hidden units given the data. Note that the model parameters are not updated when the true labels $\mathbf{y}_{1,\ldots,T}$ match the predicted labels $\mathbf{y}^{**}_{1,\ldots,T}$.

The hidden unit states $\mathbf{z}^*_{1,\ldots,T}$ can be computed in closed form (Gelfand et al., 2010), viz. $\forall t : \mathbf{z}^*_t = S(\mathbf{x}_t^T\mathbf{W} + \mathbf{V}\mathbf{y}_t + \mathbf{b}^T)$ with $S(\cdot)$ the Heaviside step function. The most likely state sequence $\mathbf{y}^{**}_{1,\ldots,T}$ can be computed using a slightly adapted version of the Viterbi algorithm that (1) for each label y_{tk} finds the most likely states of the hidden units $\mathbf{z}_t$ in closed-form, and (2) uses the Viterbi algorithm to find the optimal label sequence given the states of the hidden units. The corresponding states of the hidden units can then be found as $\forall t : \mathbf{z}^{**}_t = S(\mathbf{x}_t^T\mathbf{W} + \mathbf{V}\mathbf{y}^{**}_t + \mathbf{b}^T)$. This procedure performs an exact energy maximization assuming we constrain $\mathbf{y}_t$ to have a 1-of-K encoding; the maximization is partial if the the label vectors $\mathbf{y}_t$ are allowed to be binary K-vectors. The reader should note that the maximization procedure described above is *not* the same as the Viterbi algorithm described in Section 2: the above procedure jointly maximizes over $(\mathbf{z}_{1,\ldots,T}, \mathbf{y}_{1,\ldots,T})$, whereas the standard Viterbi algorithm for hidden-unit CRFs marginalizes over $\mathbf{z}_{1,\ldots,T}$ and maximizes over $\mathbf{y}_{1,\ldots,T}$. The two procedures may identify different "optimal" label sequences.

In perceptron training, the update rule described

above is applied repeatedly. In general, this process does not converge, but convergence can be obtained by averaging the model parameters obtained after each parameter update. We employ such parameter averaging because it generally leads to better models.

3.3 Large-Margin Perceptron Training

Perceptron training can be improved by incorporating a large-margin term in the conditional log-likelihood (Cheng et al., 2009; Sha and Saul, 2009). Such a large-margin term serves as a surrogate for the maximum-margin objective that can be used to learn the parameters of linear CRFs (Taskar et al., 2004). It often improves performance because it reduces overfitting.

Large-margin perceptron training for hidden-unit CRFs can be implemented by adapting the way in which the most likely state sequence $\mathbf{y}^{**}_{1...T}$ is computed. Instead of computing $\mathbf{y}^{**}_{1...T} = \text{argmax}_{\mathbf{y}'_{1...T}} \max_{\mathbf{z}'_{1...T}} p(\mathbf{y}'_{1...T}, \mathbf{z}'_{1...T}|\mathbf{x}_{1...T})$, large-margin perceptron training computes $\mathbf{y}^{**}_{1...T} = \text{argmax}_{\mathbf{y}'_{1...T}} \max_{\mathbf{z}'_{1...T}} [p(\mathbf{y}'_{1...T}, \mathbf{z}'_{1...T}|\mathbf{x}_{1...T}) + \rho\mathcal{H}(\mathbf{y}'_{1...T}, \mathbf{y}_{1...T})]$. Herein, ρ is a parameter that determines the margin, $\mathcal{H}(\mathbf{y}'_{1...T}, \mathbf{y}_{1...T}) = \sum_{t=1}^{T} 1 - \delta(\sum_{k=1}^{K} |y'_{tk} - y_{tk}|)$ is the Hamming distance between the predicted label sequence and the true label sequence, and $\delta(\cdot)$ is the Dirac delta function.

The maximization can be performed using a variant of the Viterbi algorithm described in the previous subsection; the variant adds ρ to log-potentials that correspond to erroneous labels. The resulting algorithm penalizes sequences that are far away in terms of Hamming distance (but that may be close in terms of conditional log-likelihood). As a result, the model parameters are updated when the prediction is incorrect, *and* when the prediction is correct but lies within the margin ρ. The large-margin variant of the Viterbi algorithm is used at training time, but not at test time.

4 EXPERIMENTS

We performed experiments in which we compare the performance of hidden-unit CRFs with that of standard linear CRFs. We performed experiments on four tasks: (1) optical character recognition, (2) labeling questions and answers, (3) protein secondary structure prediction, and (4) part-of-speech tagging. Code that reproduces the results of our experiments is available from `http://cseweb.ucsd.edu/~lvdmaaten/hucrf`.

4.1 Experimental Setup

In our experiments, both the linear and the hidden-unit CRFs were initialized by sampling weights $\mathbf{W}$ (and $\mathbf{V}$ where appropriate) from a Gaussian distribution with variance 10^{-4}, and setting all biases to 0. The state transition parameters $\mathbf{A}$, the initial-state parameters $\boldsymbol{\pi}$ and the final-state parameters $\boldsymbol{\tau}$ were also initialized to 0. Unless otherwise indicated, we measure the average generalization error of the CRFs in 10-fold cross-validation experiments. Below, we describe the four data sets that were used in our experiments, as well as the parameter settings of the optimizers presented in Section 3.

Optical character recognition. The OCR data set (Taskar et al., 2004) that we used in our optical character recognition experiments contains data for 6,877 handwritten words (i.e., time series), in which each word is represented as a series of handwritten characters. The data comprises 55 unique words, and it consists of a total of 52,152 characters (i.e., frames). Each character is a binary image of size 16×8 pixels, leading to a 128-dimensional binary feature vector. The data set comprises a total of 26 unique characters (i.e., classes).

Recognizing questions and answers. The FAQ data set (McCallum et al., 2000) contains data of 48 files (i.e., time series) with questions and answers gathered from 7 UseNet multi-part FAQs. The collection of FAQs contains a total of 55,480 sentences (i.e., frames). Each sentence is described using a 24-dimensional binary vector that measures lexical characteristics of the sentence (McCallum et al. (2000) provide a description of the features). We extended the feature set with all pairwise products of the original 24 features, leading to a $24 + 24^2 = 600$-dimensional feature representation. Each sentence in the FAQ data set is labeled by one of four labels: (1) question, (2) answer, (3) header, or (4) footer. The task at hand is to predict to which class each of the sentences belongs.

Protein secondary structure prediction. The aim of protein secondary structure prediction (PSSP) is to predict a protein's secondary structure, given only its amino acid structure. Predicting the structure of a protein is fundamental to the determination of its function. In our protein secondary structure prediction experiments, we used the CB513 data set, which contains amino acid structures of 513 proteins (Cuff and Barton, 1999). The true secondary structure of these proteins was computed from atomic-resolution coordinates using DSSP (Kabsch and Sander, 1983). As is common in protein secondary structure prediction, we convert the eight-class DSSP labeling into a three-class labeling by labeling H and G as H (for helix), B and E as E (for sheet), and all other states as C (for coil). For each of the proteins, we obtain 20-dimensional position-specific score matrix (PSSM) features by querying iteratively against a *non-redundant*

protein sequences (nr) database for 5 iterations using PSI-BLAST (Altschul et al., 1997). We transform the PSSM features using the scheme described by Kim and Park (2003) and concatenate the features from the surrounding 13 frames. The resulting data set has 260 dimensions, 74,874 frames, and 3 classes.

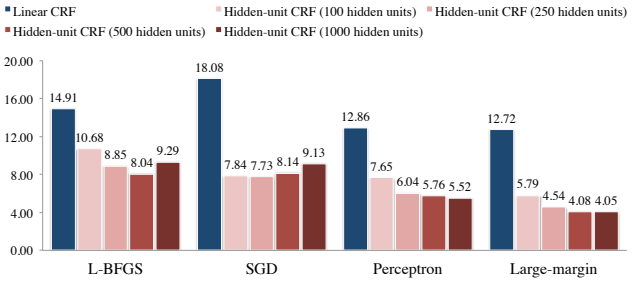
Part-of-speech tagging. The Penn Treebank corpus (Marcus et al., 2004) contains 74,029 sentences with a total of 1,637,267 words. It contains 49,115 unique words, and each word in the corpus is labeled according to its part of speech; there are a total of 43 different part-of-speech labels. We use four types of features: (1) first-order word-presence features, (2) four-character prefix presence features, (3) four-character suffix presence features, and (4) four binary lexical features that indicate the presence of, respectively, a hyphen, a capital letter, a number, and an '-ing' suffix in a word. All features are measured in a window with width 3 around the current word, which leads to a total of 212,610 features. We use a random 90% training / 10% test division for our experiments on the Penn Treebank corpus (due to the large size of the data set, we cannot perform cross-validation experiments).

We performed experiments with four optimizers, viz. (1) L-BFGS, (2) SGD, (3) perceptron training, and (4) large-margin perceptron training. The details of these four optimizers are discussed below.

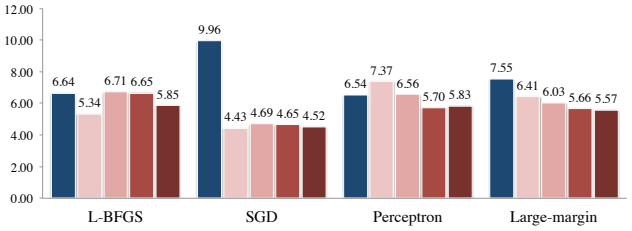
L-BFGS. In the experiments with maximum conditional likelihood using L-BFGS, we used L2-regularization. We tuned the regularization parameter based on the error on a small held-out validation set. The L-BFGS optimizer was run until the line search returned a preset minimum step size of 10^{-5}, or until the maximum number of 500 iterations was reached. Due the high dimensionality of the data, L-BFGS cannot be used on the Penn Treebank corpus (even storing parts of the Hessian is infeasible).

Stochastic gradient descent. In the experiments with maximum conditional likelihood using SGD, we did not need to use regularization. We tuned the optimal step size based on the error on a small held-out validation set. Because the time series in the Penn Treebank corpus are generally very short, we used gradients computed for mini-batches of 100 sentences in the experiments on that data set. On the other data sets, we updated the parameters using the gradient induced by a single time series. On all data sets, we ran the SGD optimizer for 100 full sweeps through the data, and we performed online averaging of the model parameters after each full sweep through the data, starting after a burn-in period of 50 iterations.

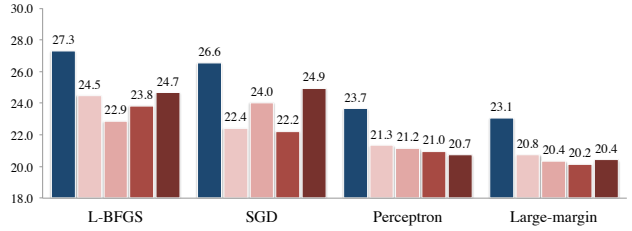
Perceptron training. In the experiments with perceptron training, we also did not need to use regularization. We performed 100 full sweeps through the data on all data sets, and we used a burn-in time of 10 iterations before starting the averaging of the model parameters. We tuned the base step size based on the error on a small held-out validation set. From the base step size, we computed parameter-specific step sizes η_θ as suggested by Gelfand et al. (2010). Test predictions are made using the adapted Viterbi algorithm described in Section 3.2.



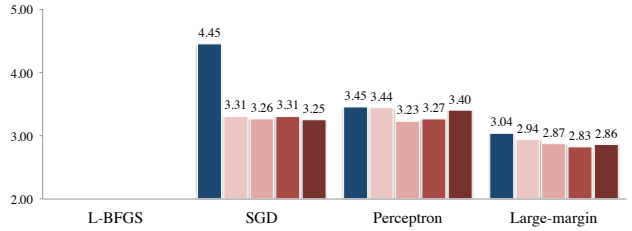
(a) Optical character recognition (OCR).



(b) Sentence labeling (FAQ).



(c) Protein secondary structure prediction (CB513).



(d) Part-of-speech tagging (Penn Treebank).

Figure 2: Generalization errors (in %) of first-order CRFs on four tasks using two different optimizers.

Large-margin training. The parameter settings for the large-margin perceptron training are identical to those of the "normal" perceptron training, except we use a margin parameter of $\rho = 0.05$. At test time, we use the adapted Viterbi algorithm from Section 3.2.

4.2 Results

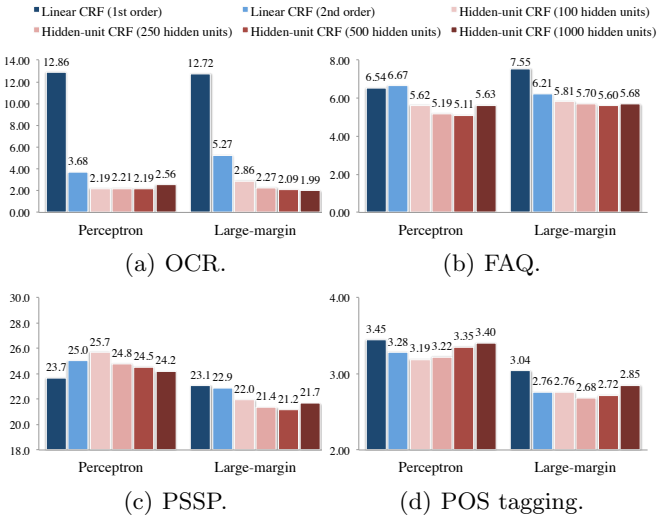
The results of our experiments on the four data sets with CRFs with first-order label chains are presented in Figure 2. The figure presents results obtained using L-BFGS, SGD, perceptron training, and large-margin training. The results in Figure 2 reveal the merits of the use of hidden units in CRFs. Hidden-unit CRFs outperform linear CRFs on all four data sets. In particular, on the OCR data set, hidden-unit CRFs obtain a decrease in generalization error of $6\% - 10\%$ compared to traditional linear CRFs, depending on which optimizer is employed. On the FAQ data set, the lowest generalization error of hidden-unit CRFs is 4.43%, compared to 6.54% for linear CRFs. On the protein secondary structure prediction task, hidden-unit CRFs achieve an performance improvement of approximately 4%. On the part-of-speech (POS) tagging task, hidden units decrease the tagging error from 3.04% to 2.83%.

From the results presented in Figure 2, we also note that perceptron-based learning algorithms outperform L-BFGS and SGD on all but the FAQ data set (on which SGD performs better). L-BFGS performs worst in almost all experiments, presumably, because its performance is very sensitive to the value of the regularization parameter. Moreover, SGD and (large-margin) perceptron training have significant computational advantages over L-BFGS, because they converge to good solutions much faster.

The results presented in Figure 2 also reveal the merits of using large-margin training: large-margin training leads to the best performance on three out of four data sets. Hidden-unit CRFs tend to benefit more from large-margin training than linear CRFs; the large margin prevents hidden-unit CRFs from overfitting.

Next to our experiments with first-order CRFs, we also performed experiments using CRFs with second-order label chains. The results of these experiments are presented in Figure 3 for perceptron and large-margin training (results obtained using L-BFGS and SGD are omitted because of space limitations). The results presented in the figure show that the use of higher-order CRFs may improve the performance of hidden-unit CRFs. In particular, hidden-unit CRFs with second-order label chains achieve a generalization error of 1.99% on the OCR data set, compared to 4.05% error using first-order CRFs. On the Penn Treebank data set, the use of second-order label chains reduces the tagging error from 2.83% to 2.68%.

In Table 1, we compare the performance of hidden-unit CRFs with the performance of competing models on three of the four tasks. The table reveals that hidden-unit CRFs outperform a range of state-of-the-art models and systems on all three tasks.



(a) OCR. (b) FAQ.

(c) PSSP. (d) POS tagging.

Figure 3: Generalization errors (in %) of second-order CRFs on four tasks using two different optimizers.

5 DISCUSSION

Below, we discuss the computational complexity of hidden-unit CRFs, and we discuss the similarities and differences of hidden-unit CRFs and related models.

Computational complexity. The additional representative power of hidden-unit CRFs comes at a computational cost. When performing inference, the main computational difference between linear CRFs and hidden-unit CRFs lies in the computation of the potentials $\sum_{\mathbf{z}_t} \Psi_t(\mathbf{x}_t, \mathbf{z}_t, \mathbf{y}_{t-1}, \mathbf{y}_t)$. For linear CRFs, the potentials can be computed in $\mathcal{O}(DK + K^2)$, whereas it requires $\mathcal{O}((D + K)H + K^2)$ to compute these potentials hidden-unit CRFs (where D is the data dimensionality, K is the number of classes, and H is the number of hidden units). Having said that, parallelization of the computations in hidden-unit CRFs is straightforward, and the overall computational cost still scales linearly in the number of training instances.

Related models. Hidden-unit CRFs may be considered as a natural extension of discriminative RBMs (Larochelle and Bengio, 2008) to time series data. This relation suggests that the data-dependent parameters of hidden-unit CRFs can be pre-trained using discriminative RBMs. As a computationally less expensive alternative, pre-training of the data-dependent parameters may also be performed by training an RBM on $p(\mathbf{x}, \mathbf{y})$ using contrastive divergence (Hinton, 2002).

Hidden-unit CRFs can also be considered as the probabilistic counterpart of conditional neural fields (CNFs; LeCun et al. (1998); Peng et al. (2010)), which are CRFs equipped with deterministic hidden units. As a result, hidden-unit CRFs have a different activation

Table 1: Performance of various techniques on three data sets compared to that of hidden-unit CRFs. The best performance on each data set is boldfaced.

Optical character recognition	
Linear-chain CRF (Do and Artieres, 2010)	14.20%
Max-margin Markov net (Do and Artieres, 2010)	13.46%
SEARN (Daumé III et al., 2009)	9.09%
SVM + CRF (Hoefel and Elkan, 2008)	5.76%
Deep learning (Do and Artieres, 2010)	4.44%
Cond. graphical models (Pérez-Cruz et al., 2007)	2.70%
Hidden-unit CRF	**1.99%**
Protein secondary structure prediction	
PSIRED (Jones, 1999)	24.0%
SVM (Kim and Park, 2003)	23.4%
SPINE (Dor and Zhou, 2007)	23.2%
YASSP (Karypis, 2006)	22.2%
Conditional neural field (Peng et al., 2010)	**19.5%**
Hidden-unit CRF	20.2%
Part-of-speech tagging	
Linear-chain CRF (Lafferty et al., 2001)	4.27%
Second-order HMM (Brants, 2000)	3.30%
Maximum-entropy (Ratnaparkhi, 1996)	3.14%
Second-order discr. HMM (Collins, 2002)	2.93%
Hidden-unit CRF	**2.68%**

function than CNFs. A potential advantage of the hidden-unit CRF's activation function is that it does not allow hidden units to "veto" a decision. Moreover, hidden-unit CRFs have the advantage of being fully probabilistic models, which allows for natural extensions to semi-supervised learning settings.

Hidden-unit CRFs also bear some resemblance to approaches that train a deep network (often in an unsupervised manner), and train a linear CRF or Viterbi decoder on the output of the resulting network (Do and Artieres, 2010; Lee et al., 2010; Mohamed et al., 2009; Prabhavalkar and Fosler-Lussier, 2010). Such approaches differ from hidden-unit CRFs in that (1) they often do not use label information to train the weights between the data and the hidden units and/or (2) they do not train all state-transition and data-dependent parameters jointly. As a result, the hidden units in these models may discover latent distributed representations that are suboptimal for classification.

Other CRF models that are related to hidden-unit CRFs are models that use latent variables, which are assignment variables of a Gaussian mixture (Cheng et al., 2009; Sha and Saul, 2009). Such approaches were developed specifically for speech recognition, and are generally not applicable to high-dimensional data because, unlike the hidden-unit CRF, their memory complexity is quadratic in the data dimensionality.

Perceptron training of hidden-unit CRFs can be viewed as conditional herding (Gelfand et al., 2010) for time series. The only difference between the two is that perceptron training averages over parameters during training, whereas conditional herding averages over test predictions. We performed preliminary experiments with conditional herding in hidden-unit CRFs, and found that it performs roughly on par with perceptron training. However, in models where the maximization over $(\mathbf{z}_{1,\ldots,T}, \mathbf{y}_{1,\ldots,T})$ cannot be performed exactly (e.g., when the label field has a loopy structure), the sequence of model parameters that results from the perceptron updates becomes much more chaotic, as a result of which parameter averaging may result in poor models. Hence, conditional herding is likely to work much better when the label graph is loopy.

6 CONCLUSIONS

In the paper, we introduced hidden-unit CRFs; a powerful new model for the classification and segmentation of time series that learns a latent distributed representation of the data that improves classification. We showed that the conditional independence properties of hidden-unit CRF can be exploited to perform inference efficiently. We developed and investigated various training approaches for hidden-unit CRFs, and showed that training of hidden-unit CRFs is still practical for very large models (our largest model has over 200 million parameters). Finally, our experimental evaluations revealed that the performance of hidden-unit CRFs is very competitive on a wide range of tasks.

In future work, we aim to investigate more complex variants of hidden-unit CRFs such as models with three-way interactions (Memisevic et al., 2011). We also aim to investigate hidden-unit CRFs in which the label units form a more complex network (e.g., a planar graph), and we aim to investigate segmental variants of hidden-unit CRFs (Sarawagi and Cohen, 2005). Other directions for future work entail developing pre-training approaches, extending the model to semi-supervised learning settings by combining its learning signal with that of its generative counterpart, and investigating training approaches based on expectation maximization (EM).

Acknowledgements

We thank Antal van den Bosch for helpful discussions, and Jian Peng for assistance in computing the protein sequence profiles. This work was performed while Laurens van der Maaten was at University of California, San Diego. This work was supported by NWO award number 680.50.0908, by EU-FP7 SSPNet, and by NSF award numbers 0812576, 0447903, and 1018433.

References

S.F. Altschul, T.L. Madden, A.A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, 1997.

C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, NY, 2006.

T. Brants. TnT – A statistical part-of-speech tagger. In *Proceedings of the* 6^{th} *Applied Natural Language Processing Conference*, 2000.

C.-C. Cheng, F. Sha, and L.K. Saul. Matrix updates for perceptron training of continuous-density Hidden Markov Models. In *Proceedings of the* 26^{th} *International Conference on Machine Learning*, pages 153–160, 2009.

M. Collins. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 Conference on Empirical methods in Natural Language Processing*, volume 10, pages 1–8, 2002.

M. Collins. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4): 589–637, 2003.

J.A. Cuff and G.J. Barton. Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. *Proteins: Structure, Function, and Genetics*, 34:508–519, 1999.

H. Daumé III, J. Langford, and D. Marcu. Search-based structured prediction. *Machine Learning Journal*, 75(3):297–325, 2009.

T.-M.-T. Do and T. Artieres. Neural conditional random fields. In *Proceedings of the* 13^{th} *International Conference on Artificial Intelligence and Statistics (AI-STATS)*, pages 177–184, 2010.

O. Dor and Y. Zhou. Achieving 80% ten-fold cross-validated accuracy for secondary structure prediction by large-scale training. *Proteins: Structure, Function, and Bioinformatics*, 66(3), 2007.

A. Gelfand, L. van der Maaten, Y. Chen, and M. Welling. On herding and the perceptron cycling theorem. In *Advances in Neural Information Processing*, page (in press), 2010.

X. He, R.S. Zemel, and M.Á. Carreira-Perpiñán. Multiscale Conditional Random Fields for image labeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 695–702, 2004.

G.E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14 (8):1771–1800, 2002.

G. Hoefel and C. Elkan. Learning a two-stage SVM/CRF sequence classifier. In *Proceedings of the ACM Conference on Information and Knowledge Management*, pages 271–278, 2008.

D.T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, 292(2):195–202, 1999.

W. Kabsch and C. Sander. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22 (12):2577–2637, 1983.

G. Karypis. YASSPP: Better kernels and coding schemes lead to improvements in protein secondary structure prediction. *Proteins: Structure, Function, and Bioinformatics*, 64(3):575–586, 2006.

H. Kim and H. Park. Protein secondary structure prediction based on an improved support vector machines approach. *Protein Engineering*, 16, 2003.

S. Kumar and M. Hebert. Discriminative fields for modeling spatial dependencies in natural images. In *Advances in Neural Information Processing Systems*, volume 16, 2003.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the* 18^{th} *International Conference on Machine Learning*, pages 282–289, 2001.

J. Lafferty, X. Zhu, and Y. Liu. Kernel Conditional Random Fields: Representation and clique selection. In *Proceedings of the* 21^{st} *International Conference on Machine Learning*, pages 64–71, 2004.

H. Larochelle and Y. Bengio. Classification using discriminative Restricted Boltzmann Machines. In *Proceedings of the* 25^{th} *International Conference on Machine Learning*, pages 536–543, 2008.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

H. Lee, Y. Largman, P. Pham, and A.Y. Ng. Unsupervised feature learning for audioclassification using convolutional deep belief networks. In *Advances in Neural Information Processing Systems*, volume 22, pages 1096–1104, 2010.

Y. Liu, J. Carbonell, P. Weigele, and V. Gopalakrishnan. Segmentation conditional random fields (SCRFs): A new approach for protein fold recognition. In *ACM International Conference on Research in Computational Molecular Biology*, pages 408–422, 2005.

M. Marcus, B. Santorini, and M. Marcinkiewicz. Building a large annotated corpus of English: the

Penn Treebank. In G. Sampson and D. McCarthy, editors, *Corpus Linguistics: Readings in a Widening Discipline*. Continuum, 2004.

A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the International Conference on Machine Learning*, pages 591–598, 2000.

R. Memisevic, C. Zach, G. Hinton, and M. Pollefeys. Gated softmax classification. In *Advances in Neural Information Processing Systems*, 2011.

A.R. Mohamed, G. Dahl, and G.E. Hinton. Deep belief networks for phone recognition. In *NIPS Workshop on Deep Learning for Speech Recognition*, 2009.

J. Peng, L. Bo, and J. Xu. Conditional neural fields. In *Advances in Neural Information Processing Systems*, volume 22, pages 1419–1427, 2010.

F. Pérez-Cruz, Z. Ghahramani, and M. Pontil. Conditional graphical models. In G.H. Bakir, T. Hofmann, B. Schölkopf, A.J. Smola, B. Taskar, and S.V.N. Vishwanathan, editors, *Predicting Structured Data*, pages 265–282. MIT Press, 2007.

R. Prabhavalkar and E. Fosler-Lussier. Backpropagation training for multilayer conditional random field based phone recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2010.

A. Quattoni, S. Wang, L.-P. Morency, M. Collins, and T. Darrell. Hidden conditional random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(10), 2010.

A. Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, New Brunswick, NJ, 1996. Association for Computational Linguistics.

N. Le Roux and Y. Bengio. Representational power of Restricted Boltzmann Machines and Deep Belief Networks. *Neural Computation*, 20(6):1631–1649, 2008.

R.R. Salakhutdinov, A. Mnih, and G.E. Hinton. Restricted Boltzmann Machines for collaborative filtering. In *Proceedings of the* 24^{th} *International Conference on Machine Learning*, pages 791–798, 2007.

S. Sarawagi and W.W. Cohen. Semi-Markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems*, volume 17, 2005.

T. Schmah, G.E. Hinton, R. Zemel, S.L. Small, and S. Strother. Generative versus discriminative RBM models for classification of fMRI images. In *Advances in Neural Information Processing Systems*, volume 21, pages 1409–1416, 2009.

F. Sha and F. Pereira. Shallow parsing with Conditional Random Fields. In *Proceedings of Human Language Technology – NAACL 2003*, pages 213–220, 2003.

F. Sha and L.K. Saul. Large margin training of continuous-density hidden Markov models. In J. Keshet and S. Bengio, editors, *Automatic Speech and Speaker Recognition: Large Margin and Kernel Methods*, pages 101–114. Wiley & Sons, 2009.

C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*, 2007.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing*, volume 16, 2004.