# Learning Flexible Features for Conditional Random Fields

Liam Stewart, Xuming He, Richard S. Zemel

***Abstract*— Extending traditional models for discriminative labeling of structured data to include higher-order structure in the labels results in an undesirable exponential increase in model complexity. In this paper, we present a model that is capable of learning such structures using a random field of parameterized features. These features can be functions of arbitrary combinations of observations, labels and auxiliary hidden variables. We also present a simple induction scheme to learn these features, which can automatically determine the complexity needed for a given data set. We apply the model to two real-world tasks, information extraction and image labeling, and compare our results to several other methods for discriminative labeling.**

***Index Terms*— machine learning, statistical models, induction, text analysis, pixel classification, markov random fields.**

## I. Introduction

MANY domains, such as computational biology and natural language processing, contain data that can naturally be grouped into sequences. For many tasks, it is essential that these sequences be treated as a whole, rather than as a collection of independent observations, in order to exploit inherent structures. Sequence labeling, in which each observation $x$ of a sequence $X$ is associated with a label $y$ from some finite set of labels $\mathcal{Y}$, is a common operation. For example, a computational biologist may be interested in predicting protein secondary structures given a sequence of amino acids, or a speech recognition system may need to infer phonemes given a recording of a human speaking.

It is widely recognized that generative models are not appropriate for labeling sequences for two reasons. First, conditional independence assumptions must be made about the observations to maintain tractability, and second, the learning criteria is generally quite different from how the model will be used in practice [1]. In contrast to generative models, discriminative models directly model the conditional probability of the labels given the observations. There exist many methods to perform classification when the dimensionality of the data is fixed and when the number of labels is finite. Most, however, assume that data items are independent and identically distributed (IID), and while they can be applied to data where the IID assumption no longer holds, the results may be poor because the components of the data are treated separately.

The current state-of-the-art model for sequence labeling is the conditional random field (CRF) which constructs a distribution over the labels, conditioned on the observations, as a log-linear combination of features, simple functions of the observations and labels [2], [3]. The most common CRF architecture, the linear chain, uses features that depend on the observations and on pairs of adjacent labels. The probability of a set of labels $Y$ for a sequence $X$ under a linear chain CRF with $n$ features $f_1, \ldots, f_n$ is

$$P(Y|X) = \frac{1}{Z(X)} \prod_{t=1}^{T} \exp\left(\sum_n \lambda_n f_n(y_{t-1}, y_t|X)\right) \quad (1)$$

where $Z(X)$ is the normalization constant.

In a standard linear chain CRF with discrete observations and labels, the features are defined so that there is a single binary feature function corresponding to each combination of observation and label components. For example, a model applied to identifying addresses in a document may include a feature that is one if and only if $y_{t-1}$ is a city, $y_t$ is a state, and $x_t$, the $t^{th}$ word in the document, begins with a capital letter.

Extending CRFs to incorporate higher order features has proven to be quite difficult as they tend to use a large number of features; the complexity of the model is exponential in the order of the features. To avoid overfitting, ever increasing amounts of training data are needed. Most of the methods that have been proposed for extending CRFs to handle larger scale structure have done so in ways that either are ad hoc or constrain the type of problems that can be modeled.

In this paper, we propose a method of incorporating larger scale structures in sequence labeling without unduly increasing model complexity. Our method involves adding latent variables to a CRF. This development is analogous to that provided by Boltzmann machines, which extended Markov random fields (MRFs) to include latent variables. Here the latent variables essentially allow the model to maintain internal state: the values of the latent variables, based on the observations, directly influence the choice of labels. The model, which we call the conditional field of experts (CFOE), uses parameterized features and is capable of learning larger scale structures conditioned on observations.

The outline of the paper is as follows. In Section II, we formally present the CFOE model and discuss inference and parameter learning. We also present a feature induction scheme that is capable of learning not only the feature parameters, but also the structure of the model; that is, what features are in the model. As our model is fairly general, we discuss a number of issues that must be addressed when the model is applied to specific tasks. Section III is a discussion of how the CFOE relates to other models for labeling structured data. In Section

L. Stewart, X. He, and R. Zemel are with the Department of Computer Science at the University of Toronto, 10 King's College Road, Toronto, Ontario, Canada M5S 3G4. E-mail: {liam,hexm,zemel}@cs.toronto.edu

IV, we examine the performance of CFOEs and compare them to chain-structured latent state models on two tasks: a synthetic problem that contains large scale structure, and an information extraction task. We then explore how CFOEs may be extended beyond one-dimensional sequences to learn structure in higher dimensional data, and present results for a two-dimensional image labeling task.

## II. The CFOE Model

The features used in CRF models are typically constructed by hand, and since it is not always known what kinds of features will be needed, models can end up with thousands of features to cover the many different kinds of structures which may (or may not) be present in the data. If we know what kinds of structures are present we could pre-specify the appropriate features. In some cases, direct searches may be feasible, but this is inefficient for anything but low-order models. Ideally, we would like to learn what structures are present in data and discover robust and efficient feature functions.

Disregarding observations, we can consider standard CRF feature functions as providing a particular basis set for representing label patterns. A CRF feature function on a group of $M$ labels can be represented as a matrix, or *template*, with one element for every possible label (row) of each of the $M$ components (columns). The templates are binary, with one and only one non-zero element per column corresponding to the label to match in that component. Evaluating the feature function can be thought of as matching its template against the $M$ labels.

Motivated by our desire to have robust features that can learn structure, we propose using a collection of parameterized features, each based on a matrix with non-binary elements. The matrices are soft versions of CRF templates which essentially incorporate multiple templates at varying strengths. Training a model using such features involves learning appropriate parameters in the matrices of the feature functions.

If the columns of a feature are softmaxed (exponentiated and normalized), such a feature can be thought of as inducing a distribution over each label variable. A high entropy distribution indicates that the feature does not care what the label is since the probabilities for each value are roughly equal. As the entropy decreases, the feature becomes more specific in what it predicts. Considering all $M$ labels together, the feature can be seen as carving out a region in the space of configurations that it matches; a collection of such features performs dimensionality reduction with respect to a fully-enumerated $M^{th}$-order CRF.

### A. Basic Formulation

Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ be the observations for a particular structure. We assume that all observations $\mathbf{x}_i$ are vectors in $\mathbb{R}^d$. Categorical observations can be represented using an indicator vector where the $j$th element of $\mathbf{x}_i$ is one if and only if $x_i$ is the $j$th possible value. Let $Y = \{y_1, \ldots, y_T\}$ be a set of label variables associated with $X$. The $y_i$ are assumed to be discrete with $y_i \in \mathcal{Y}, |\mathcal{Y}| = R$; we also represent these using an indicator vector.

A CFOE model consists of $N$ parameterized feature functions $f_1, \ldots, f_N$. Associated with each feature are a particular number of labels, which, for simplicity, we assume are contiguous: feature $f_n$ encodes a pattern on $K_n$ labels starting at location $t_n$ using a parameter matrix $U_n = [\mathbf{u}_{n,1}, \ldots, \mathbf{u}_{n,K_n}]$. The feature can also encode a pattern on $L_n$ observations (which are also assumed to be contiguous) starting at location $s_n$ using a parameter matrix $W_n = [\mathbf{w}_{n,1}, \ldots, \mathbf{w}_{n,L_n}]$. $f_n$ also has a bias parameter $b_n$. Associated with each feature is a latent random variable $h_n$. It is not strictly necessary to include the latent variables in the parameterization; however, as we will see below, including them generalizes the form of feature functions and allows for non-linear interactions between labels.

The specific form of the feature function is

$$f_n(Y, h_n|X) = h_n \left[ b_n + \sum_{k=1}^{K_n} \mathbf{u}_{n,k} \cdot y_{t_n+k-1} + \sum_{l=1}^{L_n} \mathbf{w}_{n,l} \cdot \mathbf{x}_{s_n+l-1} \right]. \quad (2)$$

For convenience, we will let

$$c_n(Y|X) = b_n + \sum_{k=1}^{K_n} \mathbf{u}_{n,k} \cdot y_{t_n+k-1} + \sum_{l=1}^{L_n} \mathbf{w}_{n,l} \cdot \mathbf{x}_{s_n+l-1}. \quad (3)$$

A CFOE defines the joint probability of label and latent variables of a structure using a Boltzmann distribution in which the energy function $E(Y, H|X)$ is given by an additive model:

$$\begin{aligned} P(Y, H|X) &= \frac{1}{Z(X)} \exp\left(-E(Y, H|X)\right) \\ &= \frac{1}{Z(X)} \exp\left(\sum_n f_n(Y, h_n|X)\right). \end{aligned} \quad (4)$$

The probability of a structure is obtained by integrating out the latent variables:

$$\begin{aligned} P(Y|X) &= \sum_H P(Y, H|X) \\ &\propto \prod_n \sum_{h_n} \exp\left(f_n(Y, h_n|X)\right). \end{aligned} \quad (5)$$

The energy function is now comprised of a collection of non-linear terms, the exact form of which will depend on the values that the latent variables can take on.

A CFOE model can include a baseline local classifier that provides local predictions about the labels given the observations. If one is included, the CFOE features will typically be defined only on the labels and will account for correlations in the labels that are not captured by the baseline classifier. The resulting model can be expressed as a product model. Assuming that one local classifier is used to label each element and writing $P_l(y|\mathbf{x})$ for the distribution produced by the local classifier, the model is

$$P(Y|X) \propto \prod_n \sum_{h_n} \exp\left(f_n(Y, h_n|X)\right) \prod_{t=1}^{T} P_l(y_t|\mathbf{x}_t). \quad (6)$$

Logistic regression (LR) and multi-layed perceptrons (MLP) are the two baseline classifiers that we explore in the experiments (see Section IV).

*Example:* To illustrate the idea of parameterized features and to contrast them with CRF features, consider a sequential structure consisting of six elements where the labels of the elements are taken from the set $\{A, B, C, D\}$. We will consider only features on the labels for simplicity, although it is possible to have features on both the labels and the observations.

Suppose that one commonly occurring pattern can be represented by the pattern of labels `(B|D)A*D`, where `|` means 'or' and `*` represents any label. That is, the pattern consists of a B or a D, followed by an A, then any label, and finally a D.

A standard CRF requires eight features of order four to model this structure: one to match `BAAD`, another for `BABD`, and so on. If the pattern was not known ahead of time, the model would need to include all 256 possible fourth-order features. For a particular subsequence, each feature would be matched to determine whether or not that structure occurs.

In contrast, we can use a parameterized feature which effectively incorporates all eight CRF features. Intuitively, this feature should assign high probability to label B or D in position 1, high probability to label A in position 2, high probability to any label in position 3, and high probability to label D in position 4. The process of matching, which we formalize in Equation 2, is done by element-wise multiplication of the label and matrix elements followed by the summation of all intermediate values to produce a final value. A high value indicates a good match.

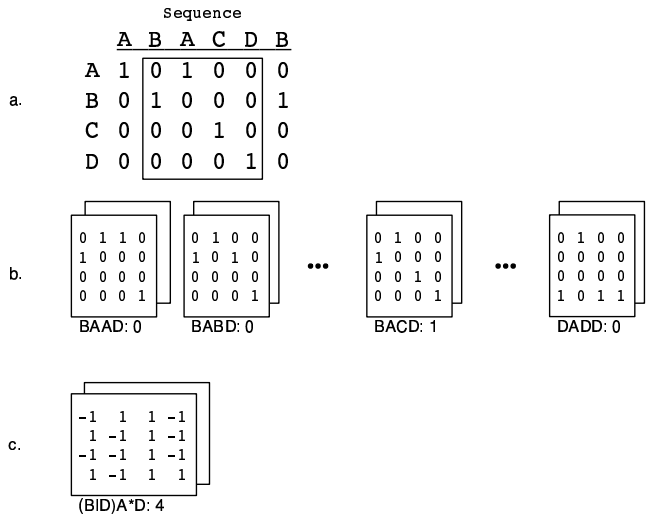


Fig. 1. An example contrasting the application of a collection of CRF features with the application of a single parameterized feature to a subsequence of length four. All features are 4 by 4 matrices. (a) The labels are first represented by columns of indicator vectors. (b) Each 4x4 block represents a CRF feature, which is a template. A CRF matches the collection of templates against the subsequence, where a match equals one if and only if the template exactly matches the subsequence. (c) In contrast, a single CFOE feature applied to the subsequence produces a high value.

The process of applying a set of CRF features and a single CFOE feature is illustrated in Figure 1 for the sequence `ABACDA`. The subsequence of interest starts at an offset of one.

To give a concrete illustration of a CFOE for the example sequence, we might use three features which encode patterns on four observations as well as four labels. Therefore, $K_i = L_i = 4, 1 \leq i \leq 3$ and $t_1 = s_1 = 1, t_2 = s_2 = 2$, and $t_3 = s_3 = 3$. The graphical model corresponding to this CFOE is shown in Figure 2.

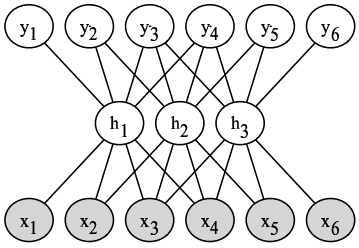


Fig. 2. The graphical model corresponding to the application of a CFOE to a sequence. The bottom row of shaded nodes corresponds to the observations, which are always observed. The top row are the labels, which are observed during training but latent during testing. The middle row are the variables corresponding to the features; these are always latent. In this CFOE, $N = 3$.

### B. Inference and Learning

The goal of inference is to obtain the marginal distribution of the labels $Y$ and latent variables $H$ given an observed sequence $X$. In general, it is expensive to compute the exact marginals of the latent variables and the labels because the maximum clique size can be very large. However, it is easy to compute the approximate marginals. In particular, we can use Gibbs sampling, a Markov chain Monte Carlo (MCMC) method which repeatedly samples all variables. Each step in a cycle consists of taking a sample from the posterior distribution of a variable given the values of all others [4]. The bipartite nature of the CFOE model (see Figure 2) leads to an efficient implementation of Gibbs sampling. The label variables can be sampled in parallel because they are conditionally independent given the latent variables. Similarly, the latent variables are conditionally independent given the label variables and can be sampled in parallel. The posterior probability of a label variable given the latent variables $\{h_n\}$ is

$$P(y_j = v|H, X) \propto \exp\left(\sum_{(n,k)=j} u_{n,k,v} h_n\right) \cdot P_l(y_j = v|\mathbf{x}_t). \tag{7}$$

where the summation $\sum_{(n,k)=j}$ is over features $n$ that reference $y_j$ at index $k$. The posterior probability of a latent variable given labels depends on the possible values of the latent node. In the case of binary latent variables, we have the following posterior distribution:

$$P(h_n = s|Y, X) \propto \exp\left(s \cdot c_n(Y|X)\right). \tag{8}$$

Other methods of inference, such as variational techniques or faster sampling techniques like Swendsen-Wang [5], could

also be used to compute marginals. Inference is used during training, in which the parameters of the model are adapted to a set of data. Standard training techniques, like maximum likelihood, are not easy to use because of the presence of the normalization term $Z(X)$ in Equation 4. One possible approach is to approximate the expectations by Markov chain Monte Carlo (MCMC) sampling, but this requires extensive computation and the estimated gradients tend to be very noisy.

We apply the contrastive divergence (CD) algorithm [6], a learning method that overcomes the difficulty of computing expectations under the model distribution. The key benefit of applying CD to learning parameters in a random field is that rather than requiring convergence to equilibrium, one only needs to take a few steps in the Markov chain to approximate the gradients. This can result in huge computational savings as the gradients must be updated repeatedly. Learning is further simplified in a CFOE due to the efficient implementation of the Gibbs sampler, as mentioned previously.

The original contrastive divergence algorithm was developed for unsupervised learning where the goal is to learn a representation for data. The algorithm optimizes the parameters of a model by maximizing the approximate likelihood of the data. We extend it here to apply to a supervised situation: we are given both observations and labels at training, but at testing we are given just the observation. Hence in this paper the objective of the algorithm is to maximize conditional likelihood.

To be specific, let $\theta$ be the model parameters, the maximum conditional likelihood criterion can be written as

$$\theta^* = \arg\max_{\theta} \sum_{j \in D} \log P(Y^j | X^j; \theta). \tag{9}$$

where $D$ is the training set. We begin by expressing the probability of the label in the following form, after marginalizing out hidden variables:

$$P(Y|X) \propto \exp(\sum_n \log \sum_{h_n} \exp(f_n(Y, h_n|X))). \tag{10}$$

Let

$$g_n(Y|X, \theta_n) = \log \sum_{h_n} \exp(f_n(Y, h_n|X)) \tag{11}$$

where $\theta_n$ are the parameters in the function $g_n$. The supervised CD algorithm uses the following gradient information to maximize the conditional log likelihood by iterative gradient ascent:

$$\Delta\theta_n \propto \left\langle \frac{\partial g_n}{\partial \theta_n} \right\rangle_{P_0(Y|X)} - \left\langle \frac{\partial g_n}{\partial \theta_n} \right\rangle_{P_m(Y|X)} \tag{12}$$

where $P_0(Y|X)$ is the data distribution defined by $D$, and $P_m(Y|X)$ is the reconstructed data distribution after $m$ steps of Gibbs sampling that starts from the ground-truth data. The partial derivatives in Equation 12 can be expanded as follows:

$$\begin{aligned} \frac{\partial g_n}{\partial \mathbf{u}_{n,k}} &= \frac{\sum_{h_n} \exp(f_n(Y, h_n|X)) y_{t_n+k-1} h_n}{\sum_{h_n} \exp(f_n(Y, h_n|X))} \\ &= \sum_{h_n} P(h_n|Y, X) y_{t_n+k-1} h_n \end{aligned} \tag{13}$$

$$\begin{aligned} \frac{\partial g_n}{\partial \mathbf{w}_{n,l}} &= \frac{\sum_{h_n} \exp(f_n(Y, h_n|X)) \mathbf{x}_{s_n+l-1} h_n}{\sum_{h_n} \exp(f_n(Y, h_n|X))} \\ &= \sum_{h_n} P(h_n|Y, X) \mathbf{x}_{s_n+l-1} h_n \end{aligned} \tag{14}$$

If a baseline classifier is included in the CFOE model, it can may be trained at the same time as the CFOE features or it may be trained first and then fixed. In the former case, the local classifier integrates as another component in the CD algorithm (see [6]). In the latter case, a discount factor may be applied at both CFOE training time and test time in order to compensate for over-confident predictions of the local classifier.

At test time, we are given just the observations. The goal is to infer the values of the labels. Ideally, the labels would be chosen use maximum a posteriori estimation (Viterbi decoding). However, this is difficult to perform in CFOE models due to the loopy nature and large size of the graph. Instead, we use the Gibbs sampling procedure described above to choose labels based on maximum posterior marginals:

$$y_i^* = \arg\max_{y_i} P(y_i|X). \tag{15}$$

### C. Feature Induction

The complexity of CFOE models depends on the size of feature patterns and the number of features in the model. While the feature size can be viewed as a 'knob' providing flexibility to model designer, it is desirable to determine automatically how many features should be included during learning. In general, without much domain knowledge, methods such as (cross-)validation are used to aid this decision, but they are computationally expensive.

We perform a simple type of model selection by inducing the features in a forward step-wise fashion. Our approach takes advantage of the fact that the exponential part of the marginal distribution (Equation 10) is an additive function of $g_n$:

$$F(Y|X, \theta) = \sum_n g_n(Y|X, \theta_n). \tag{16}$$

Hence the log likelihood can be viewed as a functional $\mathcal{L}$ of the additive function $F(Y|X, \theta)$. We adopt a functional gradient ascent method, adding a new $g$ into $F$ at each step to maximize the log likelihood:

$$\hat{g} = \max_g \mathcal{L}(F + g). \tag{17}$$

We impose two constraints on $g$: first, $g$ has the same functional form as the family of $g_n$, which is a nonlinear function with respect to the parameters in our case; second, the parameter $\theta$ in $g$ has a bounded norm, i.e., $||\theta|| \leq C$, so that adding each $g$ will change $F$ only slightly. Under these assumptions, the cost function in Equation 17 can be approximated by its first order expansion:

$$\mathcal{L}(F + g) \approx \mathcal{L}(F) + \langle \nabla \mathcal{L}(F), g \rangle \tag{18}$$

and the optimal $\hat{g}$ at step $k$ can be written as $g(Y|X, \hat{\theta}_k)$, where

$$\hat{\theta}_k = \arg\max_{\theta_k} \langle \nabla \mathcal{L}(F), g(Y|X, \theta_k) \rangle \tag{19}$$

For simplicity, if we consider a single data point, then $\mathcal{L} = \log P(Y|X)$. The functional gradient can be written as

$$\langle \nabla \mathcal{L}(F), g(Y|X,\theta_k) \rangle = g(Y|X,\theta_k) - \langle g(Y|X,\theta_k) \rangle_{P_F(Y|X)} \quad (20)$$

where $P_F(Y|X)$ is the model probability with $F$ as its exponential part. The functional gradient is a nonlinear function of the parameter $\theta_k$ only, so that we can use a gradient-based method to optimize $\hat{\theta}_k$ in Equation 19.

Notice that each induction step in the standard functional gradient method, as in [7], [8], requires first searching the direction in feature space that maximizes data likelihood, followed by a line search to determine the stepsize in that direction. Usually, both steps in the induction involve expensive Markov Chain Monte Carlo (MCMC) sampling of the random field [8], [9]. In [8], the first step is approximated by the Contrastive Divergence (CD) algorithm, and a re-weighting scheme in the second, which requires careful monitoring of the effective sample size and an approximation of the feature functions.

In our functional gradient approach, we also use the CD algorithm to compute the new cost in Equation 19 approximately, but we use a fixed stepsize. Therefore, our approach can be viewed as a simpler and faster unweighted version of the induction procedure of [8], in which each induced expert is optimized directly given a bound on the norm of its weights, and always a one-unit stepsize. As no line search is involved, the approximation of feature functions in facilitating the MCMC sampling and the re-weighting scheme are avoided. The modified procedure is equivalent to boosting with a fixed step size in functional gradient ascent. Adopting a fixed step size avoids the line search, but it does not allow the contribution of each feature function to be weighted. However, one would expect the fact that re-sampling after each round of induction and using small search steps can mitigate any advantage of weighting the features.

This feature induction approach can be viewed as learning a second form of structure in CRFs. Whereas the parametrized features allow the system to determine an appropriate basis in which to represent regularities in the label/observation patterns, induction allows the system to find an appropriate number of bases for the given dataset.

### *D. Instantiating the Model*

We have presented the CFOE model in a general way. However, a variety of details need to be filled in before the model can be applied. In this section, we discuss some of the design decisions that must be made when using CFOEs. These decisions allow CFOEs to run efficiently, avoid approximations that are required in their most general form, and provide some flexibility with respect to the learned features.

*1) Latent Variables:* We have not yet specified the values that the latent variables of the features can take on. There are a variety of possibilities, including $h_n \in \{0, 1\}$, $h_n \in \{\pm 1\}$, and $h_n \in \{-1, 0, 1\}$. A value of zero turns off a feature, making it contribute nothing to the energy, a value of 1 specifies a "like" opinion and lowers the energy, and a value of $-1$ raises the energy (cf. Equation 4).

In this paper we consider binary latent variables, taking on values in $\{\pm 1\}$. Following Equation 5, the conditional distribution $P(Y|X)$ can be written as

$$P(Y|X) \propto \prod_n \cosh c_n(Y|X) \prod_{t=1}^{T} P_l(y_t|\mathbf{x}_t). \quad (21)$$

The posterior probability of a latent variable given all of the labels is

$$P(h_n = 1|Y, X) = \sigma\left(2c_n(Y|X)\right) \quad (22)$$

where $\sigma(\cdot)$ is the logistic function.

*2) Parameter Sharing in Features:* We have explored two forms of parameter sharing in CFOEs. In one, features with compatible indices (based on connectivity) share the same parameters. Referring back to our example CFOE from Section II-A, the three features could share their parameter matrices since they all have the same connectivity. This is the standard parameter-sharing used in sequential models such as linear chain CRFs and time-delay neural networks in speech recognition, as well as object recognition networks like LeNet [10]. One can think of this strategy as treating the shared parameter matrix as a sliding window that is matched and shifted across the entire sequence. This results in a model with a degree of translation invariance.

Note that it is possible to allow partial matches to avoid edge effects. If a feature refers to invalid label nodes (eg. they are at $t < 1$ or $t > T$ in a sequence), the links to the invalid nodes are dropped and only those that are valid are included in the final model.

A second form of parameter sharing involves sharing columns of parameters within a parameter matrix. Columns with the same parameters capture the idea that the labels that they match should be the same. We can thus obtain coarse-grained features that match more global characteristics of a structure, and by incorporating features with different levels of parameter sharing, a CFOE can operate at multiple levels of granularity. For example, a CFOE could include fine features to match small-scale structure as well as coarser features to capture larger-scale structures.

A particularly useful form of feature is a global feature which looks at all labels and captures global regularities. For sequences of a fixed size, global features are easy to implement. However, for sequences whose size can vary, they are more difficult to define. We approach this by creating a feature of a fixed size, and then scaling that feature to cover the entire item, using parameter tying as required. This scheme makes the assumption that global structure is scale invariant, which may not always be appropriate. The feature induction scheme can select which features should be induced at each step based on their applicability.

With respect to the gradients required during training, when parameters are tied, the gradient for each tied parameter vector is the sum over the individual gradients for each instance in the instantiation of the model on the data where the parameter vector is used.

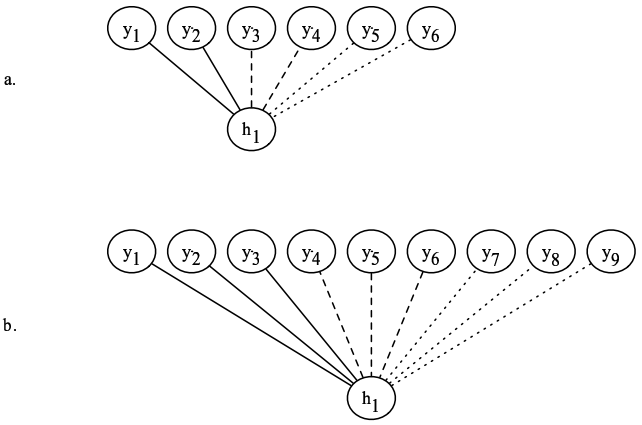


Fig. 3. An example of parameter sharing and global features. Edges with the same line style share the same parameters. The feature in (a) is a coarse feature on six labels. It expresses the constraint that the labels with shared parameters should have the same value. In (b), the feature has been expanded over a sequence of length 9.

Returning to our example CFOE from Section II-A, we could imagine having a coarse feature with three columns in its parameter matrix and having each column apply to two labels. This is illustrated in Figure 3 (a). If we needed to observe sequences of different lengths, we could make the feature into a global feature by allowing it to expand and contract (to a minimum sequence length of 3) as required. An expansion to a sequence of length 9 is shown in Figure 3 (b).

## III. Related Work

While both CRFs and CFOEs model the conditional distribution $P(Y|X)$ directly, there are a number of differences between the two types of models. The most notable difference is the use of parameterized features in CFOEs. It may appear that in allowing parameterized features the increase in the number of model parameters is prohibitive. However, this is not the case. Consider CRF and CFOE models where the features are defined on groups of $M$ labels, each label taking on one of $R$ possible values. A CRF requires $R^M$ features, and since each feature has a single parameter, the model has $R^M$ parameters. In contrast, a single CFOE feature requires $R \cdot M$ parameters so a CFOE model with $N$ features has $N \cdot R \cdot M$ parameters, a number that is linear, rather than exponential, in $M$. This represents a significant reduction in model complexity especially since $N$ will likely be much smaller than $R^M$.

There are also some differences between CRFs and CFOEs with respect to the view of features as templates. The process of matching a template to a group of labels involves matching each label and then combining the results. That is, each column of the matrix is matched, producing either zero (the label did not match the template) or a non-zero value (the label matched the template). For CRFs, a match always results in a value of one. A CRF integrates the per-label results by multiplying them together. Thus, features in CRFs are and-like features. In contrast, a CFOE feature is or-like as the results are added together. While multiplicative features are powerful, the coupling of parameters in a CFOE would make learning more difficult. The latent variables in CFOE models provide some flexibility. Given the values of the latent variables, the labels are independent. However, if the values of the latent variables are not known, the labels are coupled together, albeit in a way different from a straight multiplicative combination.

Kernel-based CRFs [11]–[13] also address the same representation issue by implicitly using linear combinations of a potentially infinite feature sets. It can be shown, by the representation theorem, that the potential functions have the form of a weighted combination of kernel functions evaluated at training data points, as well as all the configurations of cliques in the random field. As such, they can be viewed as CRFs with a nonparametric feature representation. Our model employs a parametric feature representation, of which the parameters can be easily interpreted as label/input patterns. Some kernel-based CRFs train the model parameters based on a maximum-margin criterion [11], [13], to maximize the generalization performance. However, due to the complexity of the criteria, approximations must be made. Also, typically a greedy approach is employed by kernel-based CRFs to choose a subset of active features from a large feature candidate set and the corresponding linear parameters. The search can be quite difficult when the clique size or dataset is big.

In addition to the aforementioned work on CRFs and its variants, the CFOE bears a close relationship to many other probabilistic models. A primary related formulation is the product of experts (POE) which constructs a density model of an observation as a product of parametrized filter outputs, or experts [6]. The exponential-family harmonium (EFH) is a POE with the same bipartite setup as the CFOE, using latent variables with distributions drawn from the exponential family [14]. A different variant of POEs is the field of experts (FOE) model which replicates the experts in an overlapping fashion across an entire image to generate a density of a high-dimensional input from lower-dimensional features [15].

The CFOE model can be seen as a conditional version of these POE models, and particularly FOEs. Like a FOE, a CFOE can use multiple features replicated across a sequence or an image and combined multiplicatively. The task, however, is to model labellings rather than observations. Another important difference between CFOEs and FOEs is the inclusion of the latent variables associated with the features, or experts, which links CFOEs to EFHs. These latent variables enable the system to form a nonlinear basis for the label patterns, and provide a rich vocabulary for weighting the feature functions in the log-linear combination.

There have been several attempts to extend traditional CRFs to include models of larger scale structures in a tractable manner. Skip-chain CRFs model data-dependent long-term structure by assuming that similar observations have similar labels [16]. Starting with a standard linear chain CRF, an edge is added between pairs of label nodes whose observations are similar, where the similarity measure is defined by the modeler. Other proposals make strong limiting assumptions. For example, the semi-Markov CRF groups together adjacent label nodes with the same value into a segment, so a semi-Markov CRF of order $M$ is a linear chain CRF of order at most $M$ with the constraint that all $M$ labels must be the same [17].

Neither of these models are very general; they are only useful when the data exhibits some fairly strong characteristics.

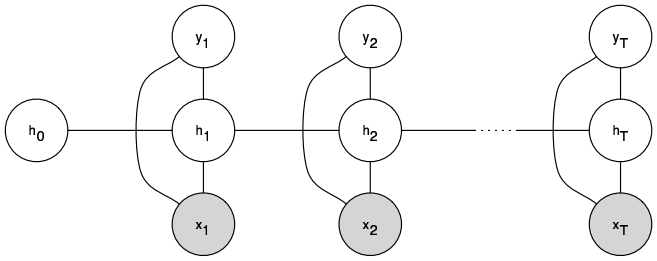


Fig. 4. Graphical model for the HRF.

An alternative approach to learning sequential structure is embodied in latent state models such as the input-output hidden Markov model (IOHMM) [18] and the hidden random field (HRF) [19]. The IOHMM is a directed graphical model that posits the existence of a latent finite state machine; it is similar to a hidden Markov model except that the transition distributions are conditional on the observations. The HRF is the undirected version of the IOHMM. Its graphical model, shown in Figure 4, is very similar to that of the CFOE (Figure 2), except that the connectivity between the label and the latent nodes is sparser and that there are links between the hidden nodes in the intermediate level. The graphical model for the HRF also has links between the labels and the observations; adding local classifiers to a CFOE would add similar links in its graphical model. Both IOHMMs and HRFs can perform well when the number of latent states is known. However, training can take a long time and, more critically, it can be very difficult to choose a good number of latent states, especially if the data is complex and noisy. In our experience, it is considerably easier to choose CFOE architectures than it is to choose the number of latent states for IOHMM and HRF models. It is also not clear how to generalize these two models to higher-dimensional data.

## IV. Experiments

We investigated and evaluated the CFOE on several data sets including a simple synthetic problem, an information extraction task, and an image labeling task.

In our experiments, we chose to make the latent variables $h$ binary, taking on values in $\{\pm 1\}$. We also chose to restrict set of feature functions to depend only on the labels and not on the observations. The baseline classifiers that we use are logistic regression (for the synthetic problem and the information extraction task) and a multi-layer perceptron (for the image labeling task). MAP estimation is used to pick labels for all models other than the CFOEs, for which we use MPM.

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

### A. Synthetic Problem: Ambiguous Input

To investigate the type of features CFOE models can learn and to evaluate their efficacy, we designed a simple synthetic problem that incorporates ambiguous input and long-term dependencies.

TABLE I
Observation-to-label mapping for the synthetic problem.

| Observation | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Label | A | B | C | Q | D, E |

There are five observations $\{1, 2, 3, 4, 5\}$ and six labels $\{A, B, C, D, E, Q\}$. The mapping from observations to labels, shown in Table I, is deterministic with the exception of an ambiguity with observation 5, which can map either to D or to E. The ambiguity can be resolved by using knowledge of the label patterns: QABCDQ and QBCEQ, which are separated by sections of continuous Qs. D and E only appear after a BC so the ambiguity can be resolved when it is known whether or not an A precedes BC.

*1) Setup:* Both the training set and the test set consisted of 100 sequences of length 50. The observations used at time $t$ are just the raw observations (1, 2, 3, 4, or 5) in a one-hot encoding. Given a sufficiently large window over the observations, any method can resolve the ambiguity present in the data; however, it may not be possible to allow large windows with real, high-dimensional input data so we limited the window to be the current observation to simulate such conditions.

We trained a logistic regression model, a linear chain CRF model of order 2, an IOHMM, an HRF, and several CFOE models with varying numbers of features. Both the IOHMM and the HRF had three latent states (the minimum required to model the data). We trained several CFOE models with features that looked at groups of 6 contiguous labels, including models with 1, 2, and 6 features and one that used feature induction (adding up to 25 features). All CFOE models used the pre-trained logistic regression model to get initial noisy classifications and parameter sharing to apply features over sequences. We also allowed partial applications of features at the edges of sequences.

For the logistic regression, IOHMM, and HRF models, training was stopped when the relative change in the log likelihood was less than $1 \times 10^{-6}$. For the IOHMM and the HRF at most 100 iterations of optimization were done during the M step, and the total number of iterations of EM was limited to be at most 100. The CFOE models that did not use feature induction were trained for exactly 100 iterations as the computation of the likelihood was expensive. Three reconstruction steps were done in the reconstruction phase of CD and 50 iterations of Gibbs sampling were done to compute the approximate marginals at test time; none of the samples were discarded as burn-in.

All training runs were restarted five times with different initial parameter settings in order to estimate training times. For models with latent variables, the restarts helped deal with the problem of local optima. For each model, the run that had the best performance on the training data was selected for testing.

*2) Results:* All models correctly labeled the As, Bs, Cs and Qs. As expected, the logistic regression and CRF models

were not able to distinguish between D and E. Because of a slight imbalance between the D and E labels in the training data (there were 128 Ds and 145 Es), both classified input 4 as an E. The IOHMM and HRF were able to model the data perfectly.

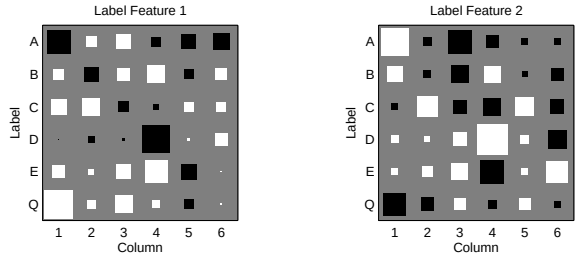


Fig. 5. Parameter matrices learned by a CFOE with 2 features. White is positive, black is negative, and blot size describes the magnitude.

All CFOE variants modeled the data perfectly. To illustrate the kinds of features that a CFOE can learn, we show the parameter matrices learned by a CFOE with 2 features in Figure 5. These parameter matrices include some interesting patterns: the first feature prefers the pattern Q**E and dislikes A**D. These two patterns essentially encode the two possible label patterns, which can be used to disambiguate an input of 5. The second label feature learns the opposite patterns, preferring A**D over Q**E. However, it also includes parameters similar to the first feature, albeit at different offsets, illustrating that a feature can match multiple configurations. Similar effects are seen in the parameters learned by the other CFOE models.

As mentioned earlier, it is possible for CRFs to perform well on the problem. However, the features must either include more observations ($\mathbf{x}_t$ to $\mathbf{x}_{t-3}$) or more labels (fourth-order). In both cases, the number of parameters in the models is very large compared with the number of parameters required by CFOEs with a small number of features. A second-order CRF with a window of 4 has 22,500 parameters. A fourth-order CRF with a window of 1 has 6480, an order of magnitude less, but still a significant number. By comparison, a CFOE with two features of width 6 and a logistic regression classifier has only 104 parameters. The IOHMM and the HRF models have 180 parameters.

TABLE II
TRAINING TIMES OF THE MODELS AVERAGED OVER FIVE RUNS.

| Method | Mean Time (s) | Standard Deviation |
|---|---|---|
| LR | $3.0 \times 10^1$ | $8.6 \times 10^0$ |
| CRF | $3.9 \times 10^2$ | $1.6 \times 10^2$ |
| IOHMM | $2.2 \times 10^3$ | $2.2 \times 10^2$ |
| HRF | $3.2 \times 10^3$ | $4.2 \times 10^3$ |
| CFOE | $8.2 \times 10^2$ | $1.7 \times 10^1$ |

The mean training time of the models is shown in Table II. The results for the CFOE are for the model with two features; the time given does not include the time needed to train the logistic regression classifier, which was negligible. The variance for the HRF is large because the model seems to be very susceptible to local optima. During training, it was observed that some runs were very short, only three to five iterations, and other were much longer.

### *B. Cora: Reference Paper Citations*

The Cora citations dataset[1] consists of 500 bibliography entries from academic papers. There are 13 possible labels for each token in each entry: author, book title, date, editor, institution, journal, location, note, pages, publisher, tech, title, and volume. While bibliography entries are generally short and follow some conventions, they are interesting because they can display large variations in total length and the length of each section within an entry. In addition, each entry does not always include all possible sections and there can be differences in the ordering of sections. Three example sequences are shown in Figure 6.

1) A. Cau, R. Kuiper, and W.-P. de Roever. Formalising Dijkstra's development strategy within Stark's formalism. In C. B. Jones, R. C. Shaw, and T. Denvir, editors , Proc. 5th. BCS-FACS Refinement Workshop, 1992.
2) M. Kitsuregawa, H. Tanaka, and T. Moto-oka. Application of hash to data base machine and its architecture. New Generation Computing, 1(1), 1983.
3) W. Cohen. Learning from textbook knowledge: A case study. In AAAI-90, 1990.

Fig. 6. Three example items from the Cora citations dataset.

*1) Setup:* We divided the data set into a training set of 350 citations chosen at random and a test set containing the remainder. The observation features that we used included 23 regular expression features and list-based features (such as person and place names) as well as 1374 vocabulary features created by extracting whole words from the training data. The regular expression and list-based features that we used are shown in Table III. All features are binary, and, except for the "EndsIn" features, ignore trailing commas, periods, colons, and semi-colons. The observation feature set is fairly simple and could be developed further.

We trained several models including a logistic regression model and an IOHMM with six latent states. The observation vectors were augmented to include observations from a window of 3 around the current observation: $\mathbf{x}'_t = [\mathbf{x}_{t-1}; \mathbf{x}_t; \mathbf{x}_{t+1}]$. The dimensionality of the augmented observation vectors was 4191.

We attempted to use cross-validation to choose the number of latent states for the IOHMM model, but this was aborted as it would have taken over three weeks to complete, even with no random restarts and with running the cross-validation procedure for each state on a separate CPU. The number of iterations of EM was limited to 100 and the number of iterations of optimization in the M step was limited to 100.

We also trained CFOE models that used two types of features: local features which looked at groups of 10 contiguous labels and global features of size 6 that were scaled to fit each sequence. The first model, referred to as CFOE(lr),

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

TABLE III
OBSERVATION FEATURES USED WITH THE CORA DATA

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

combined 10 features of each type with the basic logistic regression model. CFOE(ind-1) combined 10 local features with a logistic regression classifier built with a simple feature induction scheme based on the method proposed by [20] for CRFs that produced a set of 1605 observations features chosen from a window of width 9 centered on each observation. CFOE(ind-2) added 10 global features to CFOE(ind-1).

*2) Results:* The models were compared using two metrics: the average per-sequence prediction accuracy and the average F1 score. Per-sequence accuracy is defined to be the number of correctly labeled elements divided by the total number of elements:

$$accuracy(Y, Y') = \frac{\sum_{t=1}^{T} [\![ y_t = y'_t ]\!]}{T}. \tag{23}$$

The F1 score, or F measure, is defined to be the harmonic mean of the precision and the recall. With respect to a specific label $l$, let $A$ be the number of true positives, $B$ be the number of false negatives, $C$ be the number of false positives, and $D$ be the number of true negatives. Precision is the fraction of tokens identified as $l$ that really are $l$.

$$P = \frac{A}{A + C}. \tag{24}$$

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

$$R = \frac{A}{A + B}. \tag{25}$$

The F1 score is thus

$$\begin{aligned} F1 &= \frac{2PR}{P + R} \\ &= \frac{2A}{2A + B + C}. \end{aligned} \tag{26}$$

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

TABLE IV
RESULTS FOR THE INFORMATION EXTRACTION TASK. ALL RESULTS ARE IN PERCENT.

| Model | Accuracy | F1 |
|---|---|---|
| LR | 85.1 | 76.9 |
| IOHMM | 86.3 | 74.0 |
| CFOE(lr) | 88.9 | 79.9 |
| CFOE(ind-1) | 93.1 | 85.2 |
| CFOE(ind-2) | 93.1 | 85.0 |

The results are shown in Table IV. The IOHMM performed quite poorly relative to the other methods. This is because it is very difficult to choose how many latent states are needed and training the model takes such a long time that cross-validation is impractical. The CFOE models were able to improve upon the results of the local classifiers upon which they were based. The capabilities for modeling larger scale and global structure provided by CFOE(lr) helped that model improve upon the basic logistic regression model. By using improved local features, both CFOE(ind-1) and CFOE(ind-2) were able to improve the results even further. It is not clear that the addition of global features in CFOE(ind-2) is useful. However, the global features that it did learn are interesting. Figure 7 shows the parameters of one global feature from CFOE(ind-2) that captures the global structure of a bibliographic reference. Author names are followed by a title, which may then be followed by some other types of words. It strongly prefers location, date, or page words as the last words in an entry.

The best CFOE results are shy of those reported by [21] (accuracy: 95.37, F1: 91.50), who developed a CRF specially tailored for the task with a large and complicated feature library that included features of varying order and a large set of observation features. The feature induction scheme presented by [20] was used to select the best features to use. It is difficult to speculate about why the CFOE models obtained lower F1 scores than their CRF without knowing the details of the model and feature set, although we expect that expanding our set of observation features will help improve our results even further.

We suspect that higher-order CRFs may not do much better than regular CRFs as the size of the sections in a citation are, in general, quite large when compared to the limited orders that are practical with higher-order CRFs. It might be necessary to construct a CRF with a very high order in order to capture knowledge about how several sequences are structured. However, the exponential increase in model complexity is prohibitive as most of the features in such a model would be unused and training would much more difficult due to the high number of parameters and the possibility of overfitting. The CFOE model does not have these drawbacks; we believe that further improvements to the application of the CFOE to this dataset would be more fruitful than exploring higher-order CRFs.

The time it took to train each of our models is shown in Table V. The time shown for the CFOE does not include the time taken to train the logistic regression model with feature

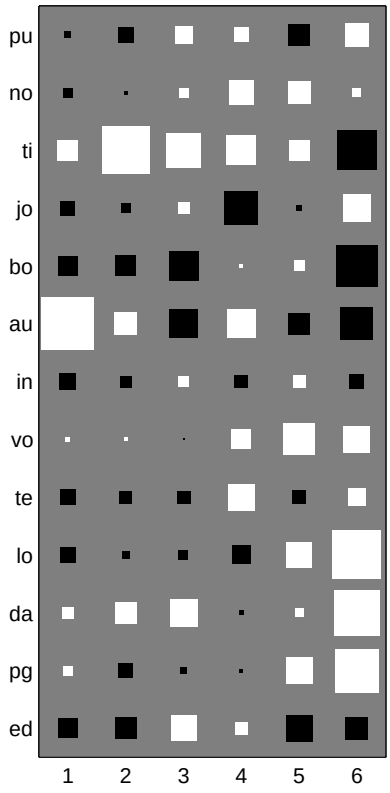


Fig. 7. Parameters for one global feature learned by CFOE(ind).

TABLE V
TRAINING TIMES OF THE MODELS.

| Method | Time (s) |
|---|---|
| LR | $8.4 \times 10^4$ |
| IOHMM | $8.1 \times 10^5$ |
| CFOE(ind-2) | $2.4 \times 10^4$ |

induction (roughly 7.5 hours). In general, all models required considerable time to train, although the IOHMM was the most costly to train. The longer training time of IOHMM models (longer still when cross-validation is used) is certainly not justified given the disappointing results on the Cora references data set.

### C. Image Labeling

CFOEs can be extended fairly easily to apply to structured data beyond simple one-dimensional sequences. In this section we consider extending the CFOE to apply to a two-dimensional structure learning problem, involving labeling pixels in images.

We apply CFOEs to a database of labeled images[2], a 100-image subset of the Corel image database, consisting of African and Arctic wildlife natural scenes. Each image is $180 \times 120$ pixels, and each pixel was manually labeled as one of 7 classes: 'rhino/hippo', 'polar bear', 'vegetation', 'sky', 'water', 'snow' and 'ground'. For comparison purposes, we incorporated the same MLP classifier used in this earlier work[2] into our CFOE; we also split the training and testing data in the same way.

The CFOE model applied to this dataset has two types of features with different sizes. Small features are defined on $6 \times 6$ regions and overlap by 3 horizontally and 3 vertically. The overlap is chosen to achieve a trade-off between coverage of the label field and the model complexity. Large features are defined on the whole label field which is divided into non-overlapped $6 \times 6$ patches of size $20 \times 30$ pixel units. All the connections of each large feature within each patch share the same parameters. During training, we alternatively induced the small and large features for 16 epochs for a total of 32 features.

The effectiveness of the feature induction is shown in Figure 8, in which we evaluated the accuracy rate of the model with first $k$ features on the test data for different $k$ values. The performance increases as features are induced, and it asymptotes after several iterations. The whole training process required 13 hours with our system setup, which is slightly less than half the time required to train a full model with the same size, in which the features are trained in parallel as opposed to the sequential feature induction scheme employed here.

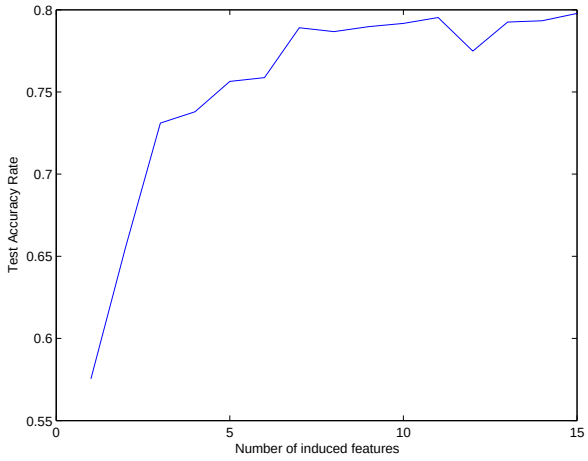


Fig. 8. Test performance with different number of features.

We also trained two additional models with different feature sizes. One had $4 \times 4$ features with an overlap of 2 in each direction; the other used $12 \times 12$ features overlapping by 6 in each direction. Table VI summarizes the models' testing accuracy averaged over 5 runs, which shows that the CFOE performance is not very sensitive to different feature configurations.

TABLE VI
IMAGE LABELING ACCURACY RATES FOR THE MODELS WITH DIFFERENT FEATURE SIZES.

| | $4 \times 4$ | $6 \times 6$ | $12 \times 12$ |
|---|---|---|---|
| Accuracy | $78.3 \pm 0.5$ | $79.8 \pm 0.5$ | $79.0 \pm 0.6$ |

We evaluate the performance of our model by comparing with a simple CRF and the MLP classifier. The simple CRF is also built upon the MLP classifier, but includes only pairwise interactions between each site and its 4 nearest neighbors

[2]Available at `http://www.cs.toronto.edu/~hexm`.

on the lattice of pixels. We used a set of input-independent and homogeneous feature functions as in [2]. The correct classification rates on the test sets are shown in Table VII.

The CFOE results are also compared to the results of the multiscale Conditional Random Field (mCRF) model proposed by [22] and the TextonBoost model in [23]. The label feature representation in the CFOE is based on this earlier model, but the training is quite different. The mCRF model structure is tuned manually, and all the features are learned simultaneously. In the CFOE, we only specify the maximum size of label/input pattern, leaving the incremental learning procedure to decide on the complexity of the feature set, based on the dataset. The TextonBoost model used a different set of bottom-up shape cues, and included pairwise interactions only.

The outputs of selected models are shown in several images in Figure 9. We can see that the induced CFOE model has better performance than the MLP classifier and the CRF-like model. Furthermore, it provides almost identical performance to the mCRF model reported in [22]. Note that our induced model has a considerably simpler structure than the mCRF model, with only 32 features in total. Compared to the batch training in that model, the CFOE feature induction procedure reduces the computational requirements since we only search the parameter space of a single feature function at each stage.

TABLE VII
IMAGE LABELING ACCURACY RATES FOR THE DIFFERENT MODELS. ALL RESULTS ARE IN PERCENT.

| Model | Accuracy |
|---|---|
| MLP | 66.9 |
| MLP+MRF | 70.6 |
| CFOE | $79.8 \pm 0.5$ |
| mCRF | 80.0 |
| TextonBoost | 74.6 |

## V. CONCLUSIONS AND FUTURE WORK

We have presented the CFOE, a generalization of traditional CRF models that allows for the inclusion of larger scale features without an exponential increase in model complexity. By defining features as parametrized templates rather than delta functions, CFOE models can learn structures present in the model and can be seen as doing dimensionality reduction on the space of label configurations. CFOEs allow for a rich vocabulary of features, can incorporate observations in several ways, and are amenable to feature induction schemes to automatically learn a complex model by adding features in an incremental fashion. We have shown that CFOEs can learn larger scale structures and are competitive with other models for discriminative labeling in a variety of domains.

The flexibility and expressiveness of the CFOE models still requires some design decisions as the modeler must experiment with and make decisions about the types of label structures that should be present in a model. However, the modeler need not pre-specify the exact form of such structures, only the family of possible structures. Each feature can tailor its parameters to focus on structures within the set of labels and observations on which it depends. The parametrized features of CFOEs allows for a second form of structure learning, as feature induction determines appropriate structures for the given problem.

Several avenues based on this formulation merit further exploration. The current implementation of the CFOE incorprates the input information solely through site-wise classifiers while its higher order features do not depend on the observations. This limits the representation power of the CFOE model. As suggested in our formulation, it is straightforward to add the observations into our higher order features as bias terms. When the input dimensionality is large, a better regularization method is needed for training those augmented features. In addition, an earlier instantiation of this method in the domain of image labeling [22] employed global features of the data; while these learned some interesting features, more work is needed to determine if they are useful in sequence labeling and other structured output tasks.

For sequential data, it may be interesting to use one or more latent-state temporal models, such as IOHMMs or HRFs, rather than a simple local classifier. These models would combine the flexibility of state-based models for variable memory with the power of the CFOE for recognizing larger structures. It is also possible to incorporate a CRF model, but the direct label-to-label edges complicate sampling as the label nodes can no longer be considered to be conditionally independent given the latent variables. However, efficient sampling schemes based on belief propagation can be formulated.

Although we have only examined CFOEs in two domains we believe that their modeling flexibility and their ability to model larger scale structures can be useful in other domains, such as bioinformatics, where they may be useful for performing protein secondary structure prediction, and language processing.

## ACKNOWLEDGEMENTS

We would like to acknowledge useful discussions with Miguel Carreira-Perpiñán, Max Welling, Sam Roweis, and Geoff Hinton. Funding by grants from Communications and Information Technology Ontario (CITO) and the Natural Sciences and Engineering Research Council of Canada (NSERC).

## REFERENCES

[1] A. McCallum, D. Freitag, and F. Pereira, "Maximum entropy markov models for information extraction and segmentation," in *International Conference on Machine Learning (ICML) 18*, 2000.

[2] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *International Conference on Machine Learning (ICML) 18*, 2001.

[3] S. Kumar and M. Hebert, "Discriminative random fields: A discriminative framework for contextual interaction in classification," in *ICCV2003*, 2003.

[4] D. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge: Cambridge University Press, 2003.

[5] R. H. Swendsen and J. S. Wang, "Nonuniversal critical dynamics in monte carlo simulations," *Physical Review Letters*, vol. 58, no. 2, pp. 86–88, 1987.

[6] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, vol. 14, pp. 1771–1800, 2002.

[7] S. D. Pietra, V. J. D. Pietra, and J. D. Lafferty, "Inducing features of random fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 380–393, 1997.

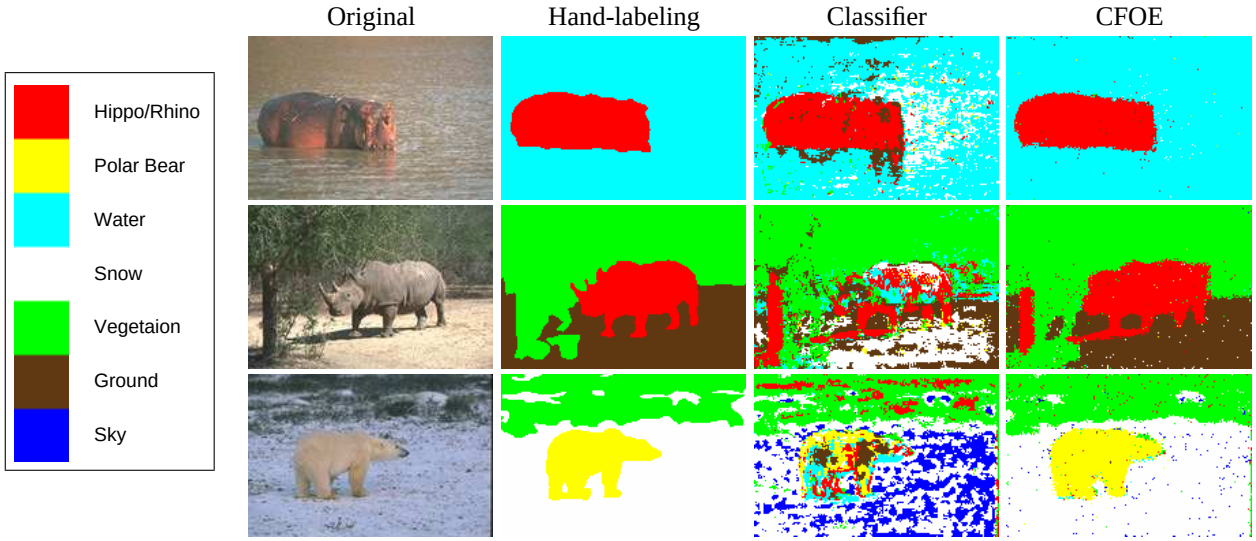


Fig. 9. Some labeling results for the Corel dataset using the pixel-wise classifier and the CFOE model.


[8] M. Welling, R. S. Zemel, and G. E. Hinton, "Self Supervised Boosting," in *Neural Information Processing Systems (NIPS) 15*, 2003.

[9] S. Zhu, Y. Wu, and D. Mumford, "Minimax entropy principle and its application to texture modeling," *Neural Computation*, vol. 9, no. 8, pp. 1627–1660, 1997.

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998.

[11] Y. Altun, I. Tsochantaridis, and T. Hofmann, "Hidden markov support vector machines," in *International Conference on Machine Learning (ICML) 20*, 2003.

[12] J. Lafferty, X. Zhu, and Y. Liu, "Kernel conditional random fields: Representation and clique selection," in *International Conference on Machine Learning (ICML) 21*, 2004.

[13] B. Taskar, C. Guestrin, and D. Koller, "Max-margin markov networks," in *Neural Information Processing Systems Conference (NIPS03)*, 2003.

[14] M. Welling, M. Rosen-Zvi, and G. Hinton, "Exponential family harmoniums with an application to information retrieval," in *Neural Information Processing Systems (NIPS) 17*, 2005.

[15] S. Roth and M. J. Black, "Fields of experts: A framework for learning image priors," in *Computer Vision and Pattern Recognition (CVPR)*, 2005.

[16] C. Sutton and A. McCallum, "Collective segmentation and labeling of distant entities in information extraction," in *ICML Workshop on Statistical Relational Learning*, 2004.

[17] S. Sarawagi and W. W. Cohen, "Semi-markov conditional random fields for information extraction," in *Neural Information Processing Systems (NIPS) 17*, 2005.

[18] Y. Bengio and P. Frasconi, "An input output HMM architecture," in *Neural Information Processing Systems (NIPS) 7*, 1995.

[19] S. Kakade, Y. W. Teh, and S. Roweis, "An alternate objective function for markovian fields," in *International Conference on Machine Learning (ICML) 19*, 2002.

[20] A. McCallum, "Efficiently Inducing Features of Conditional Random Fields," in *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2003.

[21] F. Peng and A. McCallum, "Accurate information extraction from research papers using conditional random fields," in *Human Language Technology conference - North American chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2004.

[22] X. He, R. S. Zemel, and M. A. Carreira-Perpiñán, "Multiscale conditional random fields for image labeling," in *Computer Vision and Pattern Recognition (CVPR)*, 2004.

[23] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation," in *European Conference on Computer Vision (ECCV)*, 2006.


**Liam Stewart** received a B.Sc. (Honors) degree in computing science from the University of Alberta in 2003 and a M.Sc. in computer science from the University of Toronto in 2005. His research interests include probabilistic models for machine learning and Bayesian methods and their application to real-world problems. He is currently a software engineer with Google, Inc. in Mountain View, California.

**Xuming He** received the B.Sc. and M.Sc. degrees in electronics engineering from Shanghai Jiao Tong University, Shanghai, China, in 1998 and 2001, respectively. He is a Ph.D. student in computer science at the University of Toronto, Toronto, Canada. His research interests include image segmentation and labeling, vision-based navigation, and undirected graphical models.

**Richard S. Zemel** received a B.A. (Honors) degree from Harvard University in 1984, and M.Sc. (1989) and Ph.D. (1993) degrees from the University of Toronto. He held postdoctoral positions at the Salk Institute and at Carnegie Mellon University before becoming a faculty member at the University of Arizona. He is currently an Associate Professor in the Computer Science Department, University of Toronto. He is also an Adjunct Member of the Center for Vision Research at York University, and a Program Member of the Canadian Institute for Advanced Research. His research interests include include statistical learning, representations of visual motion, and probabilistic models of neural representations.