Training Conditional Random Fields using Virtual Evidence Boosting

Lin Liao **Tanzeem Choudhury**[†] **Dieter Fox** **Henry Kautz**

University of Washington
Department of Computer Science & Engineering
Seattle, WA 98195

[†] Intel Research
1100 NE 45th St.
Seattle, WA 98105

Abstract

While conditional random fields (CRFs) have been applied successfully in a variety of domains, their training remains a challenging task. In this paper, we introduce a novel training method for CRFs, called virtual evidence boosting, which simultaneously performs feature selection and parameter estimation. To achieve this, we extend standard boosting to handle virtual evidence, where an observation can be specified as a distribution rather than a single number. This extension allows us to develop a unified framework for learning both local and compatibility features in CRFs. In experiments on synthetic data as well as real activity classification problems, our new training algorithm outperforms other training approaches including maximum likelihood, maximum pseudo-likelihood, and the most recent boosted random fields.

1 Introduction

Conditional random fields (CRFs) are undirected graphical models that were developed for labeling relational data [Lafferty *et al.*, 2001]. By directly modeling the conditional distribution over hidden states given the observations, CRFs make no assumptions on the dependency structure between observations. CRFs are thus especially suitable for classification tasks with complex and overlapped observations. However, training CRFs with very large numbers of features, especially continuous features, is still very challenging. Standard training algorithms based on maximum likelihood (ML) require running inference at each iteration of the optimization, which can be very expensive. Moreover, since exact inference can easily become intractable in large and dense networks, people often have to resort to approximate inference techniques such as loopy belief propagation and Markov Chain Monte Carlo. As a consequence, the ML learning procedure could converge to suboptimal results or even diverge.

An alternative is to maximize the pseudo-likelihood of the training data (MPL) [Besag, 1975]. The essential idea of MPL is to convert a CRF into a set of *independent patches*; each patch consists of a hidden node and the *true* values of its direct neighbors. By applying ML on this simplified model, MPL is usually very efficient and has been successful in several domains. However, no general guidance has been given on when MPL can be safely used, and indeed MPL has been observed to over-estimate the dependency parameters in some experiments [Geyer and Thompson, 1992].

In addition, neither ML nor MPL performs *feature selection* explicitly, and neither of them is able to adequately handle continuous observations. These limitations make them unsuitable for some tasks, such as activity recognition based on real sensor data and identifying the set of features that are most useful for classification. Alternatively, boosting has been successfully used for feature selection in the context of classification problems [Viola and Jones, 2002]. However, its application to *relational* data remains an unsolved problem since it assumes the independence of hidden labels.

In this paper, we show how to seamlessly integrate boosting and CRF training, thereby combining the capabilities of both paradigms. The integration is achieved by cutting a CRF into individual patches, as done in MPL, and using these patches as training instances for boosting. The key difference to MPL, however, is that in our framework the neighbor labels are not treated as observed, but as *virtual evidences* or *beliefs*. Therefore, our approach can be seen as a "soft" version of MPL, and is able to avoid over-estimating the neighborhood dependencies, as often happens in MPL.

This paper has three main contributions. First, we extend the standard boosting algorithm to handle input features that are either virtual evidences in the form of likelihood values or deterministic quantities. Second, based on the extended boosting algorithm, we present a general approach to training CRFs. This approach is able to

- perform feature selection and parameter estimation in a *unified* and *efficient* manner,
- select compatibility features and thus learn *dependency structures* in the relational data, and
- handle both discrete and continuous observations.

Third, we perform experimental validation of our algorithm on real world activity classification tasks as well as synthetic data using CRFs with different degrees of complexity. In the comparison, our approach consistently outperforms other training techniques.

The rest of this paper is organized as follows. We discuss related work in the next section. In Section 3, we extend boosting with virtual evidence. Then, in Section 4, we apply

this extension to training CRFs. Finally, we show experimental results and conclude.

2 Related Work

Several techniques have been presented that perform feature selection during CRF training.

For discrete features, McCallum [2003] suggested an efficient method of feature induction by iteratively increasing conditional log-likelihood. Dietterich *et al.* [2004] applied gradient tree boosting to select features for CRFs; their algorithm combines boosting with parameter estimation for 1D linear-chain models.

The work closest to ours is boosted random fields (BRFs) [Torralba *et al.*, 2004], developed specifically for vision tasks in which graphs are often very densely connected. Both BRFs and our approach combine boosting and belief propagation, and both can select compatibility dependencies as well as local features. However, there are a few key differences. First, for learning compatibility dependencies, BRFs use linear combinations of *beliefs* as weak learners instead of combinations of CRF features. So their models are not compatible with standard inference techniques. Second, BRFs assume densely-connected graphs with weak pairwise connections. As we will show in our experiments, when the dependencies are strong, BRFs could perform poorly. Third, since BRFs formulate the inductions of local and compatibility features differently, they have to specify the number of each type of features separately. In contrast, our approach treats both types of features in a consistent manner and thus the algorithm can determine which type is better at a given iteration. In our experiments, our algorithm *automatically* picks reliable local features at the beginning and starts selecting compatibility features after accumulating enough local evidence.

3 Boosting with Virtual Evidence

Boosting is a general approach for supervised learning [Freund and Schapire, 1997]. Given the training data $(\mathbf{x}_1, y_1)$, ..., $(\mathbf{x}_N, y_N)$, where $\mathbf{x}_i$ is a feature vector and y_i is the label, boosting works by sequentially learning a set of weak classifiers and combining them for final decisions. While traditional boosting algorithms assume feature values be deterministic, in this section we extend them with *virtual evidence* [Pearl, 1988], *i.e.*, a feature could be a distribution over its domain rather than a single, observed value. Specifically, we generalize the LogitBoost algorithm [Friedman *et al.*, 2000], which directly handles probabilities and is closely related to random field models. For simplicity, we will explain LogitBoost and our extension only for the binary classification case, *i.e.*, $y_i \in \{0, 1\}$, but both can be easily extended to multi-class problems [Friedman *et al.*, 2000].

3.1 LogitBoost Algorithm

LogitBoost minimizes the negative log-likelihood

$$-\log p(y_1, \ldots, y_N \mid \mathbf{x}_1, \ldots, \mathbf{x}_N) = -\sum_{i=1}^{N} \log p(y_i \mid \mathbf{x}_i)\,, \quad (1)$$

where the right term follows from the independence between labels. Using a logistic regression model, $p(y_i \mid \mathbf{x}_i)$ can be computed as

$$p(y_i \mid \mathbf{x}_i) = \begin{cases} \frac{e^{F(\mathbf{x}_i)}}{e^{F(\mathbf{x}_i)}+e^{-F(\mathbf{x}_i)}} & \text{if } y_i = 1; \\ \frac{e^{-F(\mathbf{x}_i)}}{e^{F(\mathbf{x}_i)}+e^{-F(\mathbf{x}_i)}} & \text{if } y_i = 0. \end{cases} \quad (2)$$

where $e^{F(\mathbf{x}_i)}$ and $e^{-F(\mathbf{x}_i)}$ represent the *potentials* for $y_i = 1$ and 0, respectively, and $F = \sum_{m=1}^{M} f_m(\mathbf{x}_i)$ refers to the sum of feature functions or *ensemble* of weak learners. LogitBoost minimizes the objective function (1) with respect to F in a stepwise way. Specifically, given the current ensemble F, LogitBoost selects the next weak learner $f_m(\mathbf{x})$ using a *Newton step* and adds $f_m(\mathbf{x})$ to the ensemble. It has been shown that computing the Newton step is equivalent to solving the following weighted least-square-error (WLSE) problem:

$$f_m(\mathbf{x}) \quad = \quad \operatorname*{argmin}_f \sum_{i=1}^{N} w_i \left(f(\mathbf{x}_i) - z_i\right)^2\,, \quad (3)$$

where $w_i = p(y_i \mid \mathbf{x}_i)(1 - p(y_i \mid \mathbf{x}_i))$ and $z_i = \frac{y_i - 0.5}{p(y_i|\mathbf{x}_i)}$ are the *weight* and *working response* for sample i.

3.2 Extension with Virtual Evidence

In this section we extend LogitBoost to handle virtual evidences or beliefs. That is, for each feature in the feature vector $\mathbf{x}_i$, the input to boosting could be a probabilistic distribution over that feature's domain, as opposed to a single, observed value [1]. We denote a distribution over the cross-product of feature values as $\mathbf{ve}(\mathbf{x}_i)$ with domain $\{1, \ldots, X_i\}$. We again aim to minimize the negative log-likelihood, $-\sum_{i=1}^{N} \log p(y_i|\mathbf{ve}(\mathbf{x}_i))$, where $p(y_i|\mathbf{ve}(\mathbf{x}_i))$ represents the posterior probability of a true label *conditioned* on its virtual evidence. We can compute this term as follows:

$$p(y_i|\mathbf{ve}(\mathbf{x}_i)) = \begin{cases} \frac{\sum_{\mathbf{x}_i=1}^{X_i} \mathbf{ve}(\mathbf{x}_i) e^{F(\mathbf{x}_i)}}{\sum_{\mathbf{x}_i=1}^{X_i} \mathbf{ve}(\mathbf{x}_i)(e^{F(\mathbf{x}_i)}+e^{-F(\mathbf{x}_i)})} & \text{if } y_i = 1; \\ \frac{\sum_{\mathbf{x}_i=1}^{X_i} \mathbf{ve}(\mathbf{x}_i) e^{-F(\mathbf{x}_i)}}{\sum_{\mathbf{x}_i=1}^{X_i} \mathbf{ve}(\mathbf{x}_i)(e^{F(\mathbf{x}_i)}+e^{-F(\mathbf{x}_i)})} & \text{if } y_i = 0. \end{cases} \quad (4)$$

Here $\sum_{\mathbf{x}_i=1}^{X_i} \mathbf{ve}(\mathbf{x}_i) e^{\pm F(\mathbf{x}_i)}$ computes the *expected potentials*, which replace the potentials in (2). It is also helpful to think of $\mathbf{ve}(\mathbf{x}_i)$ as a message in random field models; thus (4) is consistent with the belief update in belief propagation.

To determine the next weak learner $f_m(\mathbf{x})$, we modify the LogitBoost error criterion (3) by taking the expectation w.r.t. the virtual evidence:

$$\begin{aligned} f_m(\mathbf{x}) \quad &= \quad \operatorname*{argmin}_f \sum_{i=1}^{N} w_i E\left(f(\mathbf{x}_i) - z_i\right)^2 \\ &= \quad \operatorname*{argmin}_f \sum_{i=1}^{N} \sum_{\mathbf{x}_i=1}^{X_i} w_i\, \mathbf{ve}(\mathbf{x}_i) \left(f(\mathbf{x}_i) - z_i\right)^2\,, \quad (5) \end{aligned}$$

where w_i and z_i can be computed as in LogitBoost, using $p(y_i|\mathbf{ve}(\mathbf{x}_i))$ obtained from (4).

The algorithm is described in Alg. 1, which constructs F in M iterations. Within each iteration, the algorithm first formulates the WLSE problem (line 2 to 6), and then solves it to

[1] In this paper virtual evidence is always a discrete distribution, which is enough for training CRFs with discrete hidden states.

inputs: training data $(\mathbf{ve}(\mathbf{x}_i), y_i)$, with $y_i \in \{0, 1\}$, $1 \leq i \leq N$, and $F = 0$
output: F that approximately minimizes Eq. (1)

1 **for** $m = 1, 2, \cdots, M$ **do**
2 **for** $i = 1, 2, \cdots, N$ **do**
3 Compute likelihood $p(y_i|\mathbf{ve}(\mathbf{x}_i))$ using Eq. (4);
4 Compute weight $w_i = p(y_i|\mathbf{ve}(\mathbf{x}_i))(1 - p(y_i|\mathbf{ve}(\mathbf{x}_i))$;
5 Compute working response $z_i = \frac{y_i - 0.5}{p(y_i|\mathbf{ve}(\mathbf{x}_i))}$;
6 **end**
7 Obtain "best" $f_m(\mathbf{x})$ by solving Eq. (5);
8 Update $F(\mathbf{x}) = F(\mathbf{x}) + f_m(\mathbf{x})$;
9 **end**

Algorithm 1: Extending LogitBoost with virtual evidence

obtain the next weak learner (line 7). When $\mathbf{ve}(\mathbf{x}_i)$ is a deterministic value, Eq. (4) becomes (2) and Eq. (5) becomes (3); thus we get exactly the original LogitBoost algorithm. So our extension is a generalization of the original LogitBoost and is able to handle deterministic evidence as well. It can be shown to have the same property as standard LogitBoost, *i.e.*, it minimizes the objective function using Newton steps,

4 Virtual Evidence Boosting for Training Conditional Random Fields

The boosting algorithm in previous section assumes *independence* between training examples. How can they be applied to CRFs in which the labels are dependent? Similar to the MPL algorithm, we first convert a CRF into a set of *individual patches*; each patch consists of a hidden node, its direct neighbors, and the observations. The key difference with MPL is that, instead of using the true labels of neighbors, we use the messages from the neighbors as virtual evidences. Then we apply extended boosting to these independent patches for feature selection and parameter estimation. Based on these ideas, we develop a new training algorithm for CRFs, called *virtual evidence boosting* (VEB). VEB is a very general technique: It can handle both continuous and discrete observations, and can be used in CRFs with arbitrary structures. We will explain VEB in the context of binary classification; the algorithm can be readily extended to multi-class labeling, and we have done that for our experiments.

4.1 The VEB Algorithm

The VEB algorithm is described in Alg. 2. The crucial difference between the VEB and extended LogitBoost algorithm lies in the way virtual evidences are handled. The VEB considers two types of evidences for a node y_i. The first type is the *hard* evidence given as input to the algorithm, which corresponds to the observations $\mathbf{x}_i$ in the training data. The second type is the *soft* evidence, corresponding to the messages from neighboring nodes $n(y_i)$; these messages are obtained by running belief propagation with the current ensemble (*i.e.*, linear combination of feature functions) F. Our extension to boosting allows us to treat both types as virtual evidence, denoted as $\mathbf{ve}(\mathbf{x}_i, n(y_i))$, so that we can learn the CRF's local features and compatibility features in a unified framework.

inputs: structure of CRF and training data $(\mathbf{x}_i, y_i)$, with $y_i \in \{0, 1\}$, $1 \leq i \leq N$, and $F = 0$
output: Learned F

1 **for** $m = 1, 2, \cdots, M$ **do**
2 Run BP using F to get virtual evidences $\mathbf{ve}(\mathbf{x}_i, n(y_i))$;
3 **for** $i = 1, 2, \cdots, N$ **do**
4 Compute likelihood $p(y_i|\mathbf{ve}(\mathbf{x}_i, n(y_i)))$ using Eq. (8);
5 Compute weight
 $w_i = p(y_i|\mathbf{ve}(\mathbf{x}_i, n(y_i)))(1 - p(y_i|\mathbf{ve}(\mathbf{x}_i, n(y_i)))$;
6 Compute working response $z_i = \frac{y_i - 0.5}{p(y_i|\mathbf{ve}(\mathbf{x}_i, n(y_i)))}$;
7 **end**
8 Obtain "best" f_m by solving Eq. (5);
9 Update $F = F + f_m$;
10 **end**

Algorithm 2: Training CRFs using VEB

Note that while virtual evidences are provided as inputs to Alg. 1, VEB must update $\mathbf{ve}(\mathbf{x}_i, n(y_i))$ at each iteration, because the messages from $n(y_i)$ keep changing as VEB updates F. Specifically, to compute the virtual evidence of y_i from a neighbor y_k, $\mathbf{ve}_i(y_k)$, we distinguish messages before and after multiplying the compatibility potentials $e^{F(y_k, y_i)}$. We denote these messages as $\lambda_{k \to i}(y_k)$ and $\mu_{k \to i}(y_i)$, respectively. These messages are computed iteratively during belief propagation [Pearl, 1988]:

$$\lambda_{k \to i}(y_k) = \alpha e^{F(y_k, \mathbf{x}_k)} \prod_{j \in n(y_k), j \neq i} \mu_{j \to k}(y_k) \quad (6)$$

and

$$\mu_{k \to i}(y_i) = \beta \sum_{y_k} e^{F(y_k, y_i)} \lambda_{k \to i}(y_k), \quad (7)$$

where α and β are used for normalization. As can be seen, λ–messages contain information about the distribution of the sending node y_k, and μ–messages contain information about which values the recipient node y_i should prefer. While the μ–messages correspond exactly to the messages sent in regular belief propagation, we use each $\lambda_{k \to i}(y_k)$ message as virtual evidence $\mathbf{ve}_i(y_k)$. At the end of belief propagation, we generate the combined virtual evidence $\mathbf{ve}(\mathbf{x}_i, n(y_i))$ for node y_i by "stacking" the observations, $\mathbf{x}_i$, and the received virtual evidence messages $\mathbf{ve}_i(y_k)$. The combined virtual evidence can then be used to compute the posterior distribution $p(y_i|\mathbf{ve}(\mathbf{x}_i, n(y_i)))$ using (4). Equivalently, from belief propagation we have

$$p(y_i|\mathbf{ve}(\mathbf{x}_i, n(y_i))) = \gamma e^{F(y_i, \mathbf{x}_i)} \prod_{k \in n(y_i)} \mu_{k \to i}(y_i) \quad (8)$$

where γ is used for normalization.

To summarize, at each iteration, VEB updates the virtual evidences via belief propagation using only those local features and compatibility potentials already contained in F. Initially, when $F = 0$, all posterior distributions $p(y_i|\mathbf{ve}(\mathbf{x}_i, n(y_i)))$ are uniform. It then proceeds exactly as the extended LogitBoost algorithm in order to add one more weak learner f_m. In our experiments we found it sufficient to run belief propagation for only one iteration per learning cycle, which greatly increases the efficiency of the approach.

4.2 Feature Selection in VEB

A key step in Alg. 2 is step 8, which finds the "best" weak learner f_m by solving the WLSE problem. Note that since a weak learner in CRFs is a certain kind of combination of features, the algorithm is essentially performing feature selection and parameter estimation. In this paper we only consider weak learners that are linear combinations of features and involve one single type of local attribute or neighbor [2]. In a nutshell, to determine the "best" weak learner, our approach enumerates all types of local attributes and neighbor relations. For each type, it computes the optimal parameters of the weak learner and the least square error. Then it picks the one with the overall least square error as well as its optimal parameters. In this section, we discuss how to formulate weak learners for CRFs with binary labels and how to estimate the optimal parameters efficiently. Specifically, we consider three different cases: when the weak learner involves a continuous attribute, a discrete attribute, or a neighbor relationship. While the first two cases can be treated just like in regular LogitBoost, we apply extended boosting for the neighbor relationships to handle virtual evidences, with the evidences provided by belief propagation.

First, for a **continuous attribute** $\mathbf{x}^{(k)}$, the weak learner is a linear combination of decision stumps:

$$f(\mathbf{x}^{(k)}) \quad = \quad \alpha_1 \delta(\mathbf{x}^{(k)} \geq h) + \alpha_2 \delta(\mathbf{x}^{(k)} < h),$$

where h is the threshold, and α_1 and α_2 are the feature weights. We get their (approximately) optimal values by solving the WLSE problem in (3). Specifically, h is determined using some heuristic, *e.g.*, to maximize the information gain. Then we compute the optimal α_1 and α_2 analytically by setting the first-order partial derivative of the square error equal to zero. Thus we get

$$\alpha_1 = \frac{\sum_{i=1}^N w_i z_i \delta(\mathbf{x}_i^{(k)} \geq h)}{\sum_{i=1}^N w_i \delta(\mathbf{x}_i^{(k)} \geq h)} \quad \text{and} \quad \alpha_2 = \frac{\sum_{i=1}^N w_i z_i \delta(\mathbf{x}_i^{(k)} < h)}{\sum_{i=1}^N w_i \delta(\mathbf{x}_i^{(k)} < h)}$$

Second, given a **discrete attribute** $\mathbf{x}^{(k)} \in \{1, \cdots, D\}$, the weak learner is expressed as

$$f(\mathbf{x}^{(k)}) \quad = \quad \sum_{d=1}^{D} \alpha_d \delta(\mathbf{x}^{(k)} = d)\,,$$

where α_d is the weight for feature $\delta(\mathbf{x}^{(k)} = d)$, an indicator function which is 1 if $\mathbf{x}^{(k)} = d$ and 0 otherwise. The optimal weights can be calculated similarly as:

$$\alpha_d \quad = \quad \frac{\sum_{i=1}^N w_i z_i \delta(\mathbf{x}_i^{(k)} = d)}{\sum_{i=1}^N w_i \delta(\mathbf{x}_i^{(k)} = d)}$$

Third, given a certain type of **neighbor** and corresponding virtual evidence $\mathbf{ve}_i(y_k)$, the weak learner is the weighted sum of two indicator functions (compatibility features):

$$f(y_k) \quad = \quad \sum_{d=0}^{1} \alpha_d \delta(y_k = d).$$

Solving the WLSE problem with virtual evidence, as in (5), we get the optimal weights:

$$\alpha_d \quad = \quad \frac{\sum_{i=1}^N w_i z_i \mathbf{ve}_i(y_k = d)}{\sum_{i=1}^N w_i \mathbf{ve}_i(y_k = d)}$$

[2] Complex weak learners, such as decision trees involving different attributes, can also be learned in similar ways.

We can unify the three cases for computing optimal feature weights as follows:

$$\alpha_d \quad = \quad \frac{\sum_{i=1}^N w_i z_i c_{di}}{\sum_{i=1}^N w_i c_{di}} \tag{9}$$

Here c_{di} is the count of feature d in data instance i (assume we have cut CRFs into individual patches). c_{di} can be 0 and 1 for local features, or a real number between 0 and 1 for compatibility features. It can also be greater than 1 if we allow parameter sharing within an instance, for example, when a node is connected with more than one neighbor of the same type. Thus in our approach parameter estimation is solved by simply performing *feature counting*, which makes the whole algorithm very efficient.

It is important to notice that the algorithm typically first picks reliable local attributes since messages from neighbors are close to uniform at the beginning. Then, after some iterations, it starts picking compatibility features as those messages provide more information.

5 Experiments

We evaluate the performance of VEB and compare it with other alternatives: boosted random fields (BRF), maximum likelihood (ML), and maximum pseudo-likelihood (MPL). We perform experiments using both synthetic data and two different activity recognition datasets. In all these experiments, we run VEB as well as BRF for 50 iterations, and in each iteration we run one iteration of belief propagation. In ML and MPL learning, we use a shrinkage prior with zero mean and unit variance. ML and MPL optimization are implemented using a quasi-Newton procedure. For ML we evaluate the likelihood using the Bethe method [Yedidia *et al.*, 2005] and its gradient using belief propagation. All the learned models are tested using the MAP belief propagation, except that we have to use a specific inference algorithm for BRF, as described in [Torralba *et al.*, 2004]. All accuracies are calculated based on the MAP labels. All experiments were run on a standard desktop PC with 3.4GHz CPU and 1GB of memory.

5.1 Synthetic Data

VEB versus BRF

VEB and BRF are similar. However, BRF assumes the graphs are densely connected and thereby each individual message is not very informative, while VEB does not make any assumption about the connectivity structure. This difference is significant because although their assumption is often true for the vision applications discussed in [Torralba *et al.*, 2004], it may be invalid for many other applications. In this experiment, we examine the performance of VEB and BRF as the dependencies between nodes get stronger.

The synthetic data is generated using a first-order Markov chain with binary labels. To emphasize the difference on learning compatibility features, we intentionally use weak observation models: each label is connected to 50 binary observations and the conditional probabilities in the observation models are uniformly sampled from the range $[0.45, 0.55]$. We adjust the transition probabilities (from label 0 to 0 and

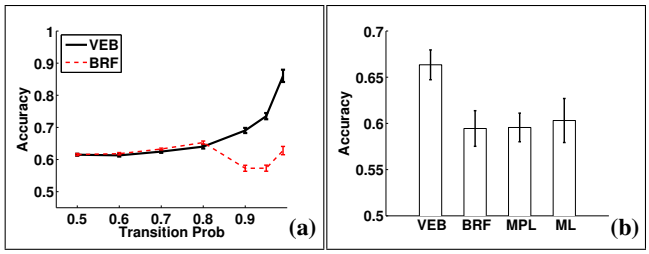

Figure 1: Classification accuracies in experiments using synthetic data, where the error bars indicate 95% confidence intervals. (a) VEB vs. BRF when the transition probabilities (pairwise dependencies) turn from weak to strong. (b) Comparison of different learning algorithms for feature selection.

from label 1 to 1) from 0.5 to 0.99. For each given transition and observation model, we generate ten 2,000-labels chains and perform leave-one-out cross-validation using a linear-chain CRF. We additionally run the experiments several times by randomly generating different observation models.

The running durations of both algorithms are very similar, so we only compare the accuracies. The average accuracies using VEB and BRF and their confidence intervals are shown in Fig. 1(a). It is clear that when the compatibility dependencies are not strong (transition probabilities range from 0.5 to 0.8), both methods give very similar accuracies. However, as the dependencies get stronger (from 0.9 to 0.99), VEB dramatically outperforms BRF, mainly because the weak interaction assumption underlying BRF does not hold any more.

Feature Selection in Complex Models

Many real sequential estimation problems have long-range dependencies, which can be modeled using high-order Markov models. However in practice it is often impossible to know the exact order, so people may have to use Markov models that have longer dependencies than the actual data. In this experiment, we simulate this scenario by generating synthetic data using a high-order Markov model, whose transition probability $p(y_n \mid y_{1:n-1}) = p(y_n \mid y_{n-k})$, where k is a constant (the observation model is similar as the one in the previous experiment). That is, a label y_n only depends on one past label y_{n-k}, but the value of k is unknown to the CRF model. Specifically, we pick k from 1 to 5, and we set the transition probability $p(y_n \mid y_{n-k})$ as 0.9 if $y_n = y_{n-k}$ and 0.1 otherwise. For a given k, we generate ten 2,000-long chains and perform leave-one-out cross-validation. We repeat the experiment for different k's and compute the average.

Since the exact value of k is unknown to the CRF model, we generate a densely-connected CRF that has connections between each pair of nodes whose distance is less than or equal to 5; then the CRF was trained using different algorithms. In our experiments, VEB can reliably identify the correct values of k, *i.e.*, picking only pairwise features whose distance is k or multiples of k. Although BRF also performs feature selection and structure learning, it does not perform as well as VEB. The average classification accuracies are shown in Fig. 1(b). Because VEB can robustly extract the sparse structures, it significantly outperforms other approaches. As to the running time, VEB, BRF, and MPL are all quite efficient; each training takes only tens of seconds. In contrast, the training using ML takes about 20 minutes.

Training algorithm	Average accuracy
VEB	94.1%
BRF	88.0%
ML + all observations	87.7%
ML + boosting	88.5%
MPL + all observations	87.9%
MPL + boosting	88.5%

Table 1: Average accuracy for indoor activities

5.2 Real Activity Recognition Data

CRFs are well-suited for the tasks of activity recognition using real sensor data, because of the overlaps between measurements and the strong relationships between activities. In this section, we compare different training approaches on such CRFs. Although VEB and BRF can handle continuous sensor measurements directly, doing that in ML and MPL is not straightforward. The performance of ML and MPL is terrible if we simply use the continuous measurements as feature values. This is due to the fact that such features correspond to a zero-mean Gaussian assumption, which could be far from the truth. We try two tricks to circumvent this difficulty. One is to learn decision stumps for all observations, using the heuristics as in VEB and BRF. The other is to use boosting (*e.g.*, LogitBoost in our experiment) to select a set of decision stump features and these decision stumps are then fed into ML and MPL for weight estimation (as done in [Friedman *et al.*, 2007]).

Indoor Activity Recognition

In the first experiment, one person collected audio, acceleration, and light sensor data as he stayed indoors using a small wearable device. The total length of the data set is about 1,100 minutes, recorded over a period of 12 days. The goal is to recognize the person's major indoor activities including computer usage, meal, meeting, TV watching and sleeping. We segmented the data into one-minute chunks and manually labeled the activity at each minute for the purpose of supervised learning and testing. For each chunk of data, we computed 315 feature values, which included energy in various frequency bands (log and linear) of the signal, autocorrelation, different entropy measures, *etc*. These features are fed into the CRF as observations, and one linear chain CRF is created per day. We evaluate our algorithm using leave-one-out cross-validation. Because the person performs different activities in different days, the accuracies can vary significantly from day to day. Some activities, such as meals, are hard to recognize, while other activities, such as sleeping, are relatively easy to recognize. As a result, in the leave-one-day-out cross-validation, the accuracies for different days vary significantly and the standard deviation is rather large. The overall average accuracies using the different methods are shown in Table 1, in which VEB is about 5% to 6% better than ML and MPL, no matter how they incorporate the continuous observations. Even though the error bars of the different techniques overlap, our approach is significantly (0.05 level) better than its competitors when we evaluate the combination of the different experiments reported in the paper.

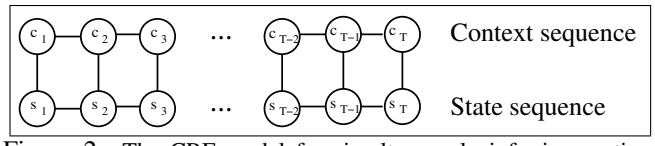

Figure 2: The CRF model for simultaneously inferring motion states and spatial contexts (observations are omitted for simplicity).

Training algorithm	Average overall accuracy
VEB	$88.8 \pm 4.4\%$
MPL + all observations	$72.1 \pm 3.9\%$
MPL + boosting	$70.9 \pm 6.5\%$
HMM + AdaBoost	$85.8 \pm 2.7\%$

Table 2: Accuracy for inferring states and contexts

Recognizing Motions and Spatial Contexts
Subramanya *et al.* [2006] proposed a model for simultaneously recognizing motion states (*e.g.*, stationary, walking, running, driving, and going up/down stairs) and spatial contexts (*e.g.*, indoors, outdoors, vehicle) from wearable sensors. They train local features using AdaBoost and incorporate the boosted classifiers as observation into an HMM that infers jointly the states and contexts. Their data set consists of 12 episodes and about 100,000 labels. In our experiment, we perform the same task with the same data set, but using a CRF instead of an HMM. As shown in Fig. 2, the CRF captures the pairwise relations between states and contexts. We perform leave-one-out cross-validation using different learning approaches. In such large and loopy CRFs, ML becomes completely intractable and does not finish in two days. MPL and VEB take about 2 hours for training [3]. The overall average accuracies and confidence intervals are shown in Table 2. Our VEB clearly outperforms MPL, as well as the result in the original paper.

6 Conclusions and Future Work

We presented a novel and unified training algorithm for CRFs, called virtual evidence boosting, which can simultaneously select informative features (both discrete and/or continuous) and estimate optimal parameters. As part of this training algorithm, we generalized the LogitBoost algorithm to handle virtual evidence. By treating neighborhood relations as features, our approach also learns the connectivity structure of CRFs.

Our experimental results demonstrate that virtual evidence boosting significantly outperforms other training approaches in both synthetic data and real world applications such as activity recognition. In future work, we want to compare our approach with max-margin techniques [Taskar *et al.*, 2004] and explore the possibility of learning feature conjunctions.

Acknowledgments

We thank Benson Limketkai for the help on preparing the last dataset. This work was supported by the NSF under grant numbers IIS 0433637 and IIS-0093406, and the UW CSE Educator's Fellowship. It has also been supported by DARPA's ASSIST and CALO Programs (contract numbers: NBCH-C-05-0137, SRI subcontract 27-000968).

[3] We did not implement BRF for this task because extending BRF to different types of hidden nodes is not straightforward.

References

[Besag, 1975] J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24, 1975.

[Dietterich *et al.*, 2004] T. Dietterich, A. Ashenfelter, and Y. Bulatov. Training conditional random fields via gradient tree boosting. In *Proc. of the International Conference on Machine Learning (ICML)*, 2004.

[Freund and Schapire, 1997] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[Friedman *et al.*, 2000] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 38(2):337–374, 2000.

[Friedman *et al.*, 2007] S. Friedman, D. Fox, and H. Pasula. Voronoi random fields: Extracting the topological structure of indoor environments via place labeling. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[Geyer and Thompson, 1992] C. J. Geyer and E. A. Thompson. Constrained Monte Carlo Maximum Likelihood for dependent data. *Journal of Royal Statistical Society*, 1992.

[Lafferty *et al.*, 2001] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of the International Conference on Machine Learning (ICML)*, 2001.

[McCallum, 2003] Andrew McCallum. Efficiently inducing features or conditional random fields. In *Proc. of the Conference on Uncertainty in Artificial Intelligence*, 2003.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[Subramanya *et al.*, 2006] A. Subramanya, A. Raj, J. Bilmes, and D. Fox. Recognizing activities and spatial context using wearable sensors. In *Proc. of the Conference on Uncertainty in Artificial Intelligence*, 2006.

[Taskar *et al.*, 2004] B. Taskar, C. Guestrin, V. Chatalbashev, and D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2004.

[Torralba *et al.*, 2004] A. Torralba, K. P. Murphy, and W. T. Freeman. Contextual models for object detection using boosted random fields. In *Advances in Neural Information Processing Systems (NIPS)*, 2004.

[Viola and Jones, 2002] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 2002.

[Yedidia *et al.*, 2005] J.S. Yedidia, W.T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.