Latent Structured Perceptrons for Large-Scale Learning with Hidden Information

Xu Sun, Takuya Matsuzaki, and Wenjie Li

Abstract—Many real-world data mining problems contain hidden information (e.g., unobservable latent dependencies). We propose a perceptron-style method, latent structured perceptron, for fast discriminative learning of structured classification with hidden information. We also give theoretical analysis and demonstrate good convergence properties of the proposed method. Our method extends the perceptron algorithm for the learning task with hidden information, which can be hardly captured by traditional models. It relies on Viterbi decoding over latent variables, combined with simple additive updates. We perform experiments on one synthetic data set and two real-world structured classification tasks. Compared to conventional nonlatent models (e.g., conditional random fields, structured perceptrons), our method is more accurate on real-world tasks. Compared to existing heavy probabilistic models of latent variables (e.g., latent conditional random fields), our method lowers the training cost significantly (almost one order magnitude faster) yet with comparable or even superior classification accuracy. In addition, experiments demonstrate that the proposed method has good scalability on large-scale problems.

Index Terms—Structured perceptron, latent variable, hidden information, convergence analysis, large-scale learning

1 Introduction

Many real-world data mining problems contain hidden information, which is important for successful pattern recognition in those tasks. For example, in human activity recognition problems [1], [2], [3], the training data are usually collected from different persons (because the data collected from each individual are quite limited and sparse). In this case, an important type of hidden information is the person ID, because different persons usually have very different activity patterns. If this kind of hidden information (person ID) can be successfully modeled within the learning process, it is expected to be helpful to the performance of the learning system. There are also plenty of hidden structure examples in other real-world tasks among different areas [4], [5], [6], [7], [8], [9], [10]. For example, in the syntactic parsing task for natural language processing, the hidden structures can be refined grammars that are unobservable in the supervised training data [11]. In the gesture recognition task of the computational vision area, there are also hidden structures which are crucial for successful gesture recognition [12].

In such cases, models that exploit latent variables are advantageous in learning, as presented in [4], [5], [13], [11], [6], [7], [10]. As a representative structured classification model with latent variables, the latent conditional random fields (LCRFs) have become widely used for performing a variety of tasks with hidden structures, e.g., vision recognition [5], and syntactic parsing [10], [11]. For example, Morency et al. [5] demonstrated that LCRF models can learn latent structures of vision recognition problems, and outperform several widely used conventional models, such as support vector machines (SVMs), conditional random fields (CRFs) and hidden Markov models (HMMs). Petrov and Klein [11] and Petrov [10] reported on a syntactic parsing task that LCRF models can learn more accurate grammars than conventional nonlatent techniques.

Nevertheless, the current structured latent conditional models (e.g., LCRFs) have very high computational cost. Hence, they have difficulties in dealing with large-scale problems. For example, experiments in [11] have highlighted both time and memory cost problems on training a structured latent conditional model. They used eight CPUs in parallel for three days to train the weighted grammars with latent variables. Because of the time and memory limitations, their solution fails to learn more complex grammars. Furthermore, our experiments in a text mining task confirmed that the training of LCRFs with a low Markov order is already computationally expensive. On a corpus of 20,000 sentences, our training of LCRFs took more than six days for LCRFs.

On the other hand, the perceptron models [14], [15], [16] have been shown to be competitive to the heavy learning algorithms on structured classification tasks, as presented in [17], [18]. The (structured) perceptrons have much lower computational costs. However, there was no prior work and it is unclear how to apply the perceptron algorithms for the learning task with hidden information.

With those motivations, we propose a new method, *latent structured perceptron*, for fast discriminative learning of structured classification with hidden information. Our target is threefold: First, the new method should be able to outperform conventional nonlatent models when the

- *X. Sun is with the Key Laboratory of Computational Linguistics (Peking University), Ministry of Education, Beijing, China, and the School of EECS, Peking University, No. 5 Yiheyuan Road, Haidian District, Beijing, China. E-mail: xusun@pku.edu.cn*
- *T. Matsuzaki is with the National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan. E-mail: takuya-matsuzaki@nii.ac.jp.*
- *W. Li is with the Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon 999077, Hong Kong. E-mail: cswjli@comp.polyu.edu.hk.*

Manuscript received 25 Nov. 2011; revised 2 May 2012; accepted 13 June 2012; published online 21 June 2012.
Recommended for acceptance by H. Zha.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2011-11-0723.
Digital Object Identifier no. 10.1109/TKDE.2012.129.

data have hidden information. Second, the new method should have a fundamentally faster training speed than existing structured latent conditional models, so that it can be applied to large-scale problems. Third, the new method should be theoretically sound (e.g., to have good convergence properties). The major contributions of our research are the following:

- To our knowledge, this is the first study of latent structured perceptrons. Compared with neural networks or multilayer perceptrons with hidden states, the latent structured perceptron is fundamentally different. First, the model definitions are different. The latent structured perceptron is based on the specific model definition of structured perceptron, which was first proposed by Collins [17] in 2002. Second, the functions of latent variables in latent structured perceptron are completely different than the hidden states in neural networks or multilayer perceptrons. The latent variables in latent structured perceptrons are for discriminatively modeling hidden structures, while the hidden states in neural networks or multilayer perceptrons are for feeding (forward or backward) activation functions. The latent variables in latent structured perceptrons do not use activation functions at all.
- We will make theoretical analysis on the proposed method. We will show that the latent structured perceptron has sound convergence properties.
- We will show that the proposed latent structured perceptron outperforms a variety of strong baselines, including structured perceptrons, CRFs, and LCRFs. Compared to conventional nonlatent models, our method is significantly more accurate on real-world tasks. Compared to existing heavy latent conditional models (e.g., LCRFs), our method lowers the training cost significantly (almost one order magnitude faster) yet with comparable or even superior classification accuracy. In addition, experiments demonstrate that the proposed method has good scalability on large-scale problems.

2 Related Work

We will review two well-known structured classification models, structured perceptrons and CRFs. We also introduce a representative latent conditional model for learning hidden information, LCRFs. Other related work includes SVMs with latent variables and the applications on object detection, language modeling, and so on [19], [20], [21].

Compared with the work of Yu and Joachims [21], our proposal is substantially different. The method in [21] alternates between imputing the latent variables that best explain the training sample and solving the structural SVM optimization problem while treating the latent variables as completely observed. In their setting, imputing the latent variables is task dependent. While the method in [21] requires task-specific engineering (inference algorithms) on latent information, our method has no need for such task-specific knowledge and engineering.

2.1 Structured Perceptrons

The structured perceptron is an extension of the standard nonstructured perceptron to structured classification [17]. Suppose the structured perceptron is parameterized by a weight vector $\boldsymbol{w}$, training a structured perceptron is performed in a similar way of training the binary classification perceptron. When reading through the instances of the training data, if the produced (classified) label sequence $\boldsymbol{y}_i$ for the observation sequence $\boldsymbol{x}_i$ is different from the gold-standard label sequence $\boldsymbol{y}_i^*$, the structured perceptron then updates its weight vector $\boldsymbol{w}$. Assuming a feature function that maps a pair of observation sequence $\boldsymbol{x}$ and label sequence $\boldsymbol{y}$ to a feature vector $\boldsymbol{f}$, the update is as follows:

$$\boldsymbol{w}^{i+1} = \boldsymbol{w}^i + \boldsymbol{f}(\boldsymbol{y}_i^*, \boldsymbol{x}_i) - \boldsymbol{f}(\boldsymbol{y}_i', \boldsymbol{x}_i), \tag{1}$$

where $\boldsymbol{y}_i'$ is the incorrect label sequence predicted for this observation sequence $\boldsymbol{x}_i$. To limit overfitting, *averaged perceptron* is proposed, which performs parameter averaging on structured perceptrons.

2.2 CRFs

CRFs are popular models for structured classification [22]. The probability function is defined as follows [22], [23]:

$$P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) = \frac{\exp\left[\boldsymbol{w}^\top \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{x})\right]}{\sum_{\forall \boldsymbol{y}'} \exp\left[\boldsymbol{w}^\top \boldsymbol{f}(\boldsymbol{y}', \boldsymbol{x})\right]}, \tag{2}$$

where $\boldsymbol{w}$ is a parameter vector.

Given a training set consisting of n labeled sequences, $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, for $i = 1 \ldots n$, parameter estimation is performed by maximizing the objective function,

$$\mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{n} \log P(\boldsymbol{y}_i|\boldsymbol{x}_i, \boldsymbol{w}) - R(\boldsymbol{w}). \tag{3}$$

The second term is a regularizer, typically an L_2 norm. In what follows, we denote the conditional log likelihood of each sample, $\log P(\boldsymbol{y}_i|\boldsymbol{x}_i, \boldsymbol{w})$, as $\ell(i, \boldsymbol{w})$.

2.3 LCRFs

LCRFs are defined as [5], [11]:

$$P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) \stackrel{\triangle}{=} \sum_{\boldsymbol{h}} P(\boldsymbol{y}|\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{w}) P(\boldsymbol{h}|\boldsymbol{x}, \boldsymbol{w}).$$

Each h is a latent variable in a set $\mathcal{H}(y)$, possible latent variables for the class label y, and $\mathcal{H}(y_i) \cap \mathcal{H}(y_j) = \emptyset$ if $y_i \neq y_j$. $\mathcal{H}$ is defined as the set of all possible latent variables; i.e., $\mathcal{H} = \cup_{y \in \mathcal{Y}} \mathcal{H}(y)$.

Assume that $h_{(i)}$ is the ith latent variable in the latent variable sequence $\boldsymbol{h}$. Since sequences that have any $h_{(i)} \notin \mathcal{H}(y_{(i)})$ will by definition have $P(\boldsymbol{y}|\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{w}) = 0$, the model can be simplified as:

$$P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) \stackrel{\triangle}{=} \sum_{\boldsymbol{h} \in \mathcal{H}(y_{(1)}) \times \ldots \times \mathcal{H}(y_{(m)})} P(\boldsymbol{h}|\boldsymbol{x}, \boldsymbol{w}). \tag{4}$$

$y_{(m)}$ is the mth label in the label sequence. The formula $\boldsymbol{h} \in \mathcal{H}(y_{(1)}) \times \ldots \times \mathcal{H}(y_{(m)})$ indicates that the latent labeling $\boldsymbol{h}$ "belongs" to the labeling $\boldsymbol{y}$, which can be more formally defined as follows:

$$\boldsymbol{h} \in \mathcal{H}(y_{(1)}) \times \ldots \times \mathcal{H}(y_{(m)}) \Longleftrightarrow h_{(i)} \in \mathcal{H}(y_{(i)}), i = 1 \ldots m.$$

$P(\boldsymbol{h}|\boldsymbol{x}, \boldsymbol{w})$ is defined by the usual CRF formulation, (2). Given a training set consisting of n labeled sequences,

$(\boldsymbol{x}_i, \boldsymbol{y}_i)$, for $i = 1 \ldots n$, parameter estimation is performed by optimizing the objective function,

$$L(\boldsymbol{w}) = \sum_{i=1}^{n} \log P(\boldsymbol{y}_i|\boldsymbol{x}_i, \boldsymbol{w}) - R(\boldsymbol{w}).$$

The second term is a regularizer.

3 Proposal

3.1 Problem Definition

Following the examples given in Section 1, we give a more formal definition of the problem. The task is to learn a mapping between a sequence of observations $\boldsymbol{x} = x_{(1)}, x_{(2)}, \ldots, x_{(m)}$ and a sequence of labels $\boldsymbol{y} = y_{(1)}, y_{(2)}, \ldots, y_{(m)}$. Each $y_{(i)}$ is a class label for the ith observation of an observation sequence and is a member of a label set, $\mathcal{Y}$. For each sequence, we also assume a latent variable sequence, $\boldsymbol{h} = h_{(1)}, h_{(2)}, \ldots, h_{(m)}$, which is not observable. To keep the training efficient, we restrict the model to have a disjointed set of latent variables associated with each class label; Each h_i is a member of a set, $\mathcal{H}(y_i)$, which represents the possible latent variables for the class label y_i. Note that the notation $h_{(i)}$ means the latent variable on the ith position of $\boldsymbol{h}$, while h_i means a specific latent variable in the set $\mathcal{H}$.

As we have discussed, discriminative modeling of hidden information are helpful for many real-world tasks. Nevertheless, the most recent developments of latent conditional models, LCRFs, face inefficiency problems on large-scale learning. With this motivation, we propose a novel model: *latent structured perceptron*. We will show that the latent structured perceptron is an ideal solution for large-scale learning with hidden information.

3.2 Maximized Estimation of Latent Structures

Since the hidden information (modeled by latent variables) is unobservable, a proper method should be proposed for learning the latent structures. We consider two natural schemas for learning latent structures in structured perceptron settings: maximized or randomized latent structures. Some prior work on machine translation studies also adopted the intuition of maximized estimation, and with different problem settings [24].

In the schema of maximized estimation of latent structures, the score of a gold-standard label sequence, $F(\boldsymbol{y}^*|\boldsymbol{x}, \boldsymbol{w})$, will be estimated by using the maximum score of its latent sequence:

$$F(\boldsymbol{y}^*|\boldsymbol{x}, \boldsymbol{w}) \triangleq \max_{\boldsymbol{h}:\mathrm{Proj}(\boldsymbol{h})=\boldsymbol{y}^*} F(\boldsymbol{h}|\boldsymbol{x}, \boldsymbol{w}), \tag{5}$$

where $\mathrm{Proj}(\boldsymbol{h})$ is the projection from a latent sequence $\boldsymbol{h}$ to its label sequence $\boldsymbol{y}$:

$$\mathrm{Proj}(\boldsymbol{h}) = \boldsymbol{y} \Longleftrightarrow h_{(i)} \in \mathcal{H}(y_{(i)}) \ for \ i = 1, \ldots, m, \tag{6}$$

and $F(\boldsymbol{h}|\boldsymbol{x}, \boldsymbol{w})$ is the score of a latent sequence:

$$F(\boldsymbol{h}|\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{h}, \boldsymbol{x}) = \sum_{k} [\boldsymbol{w}_k \cdot \boldsymbol{f}_k(\boldsymbol{h}, \boldsymbol{x})], \tag{7}$$

where the kth value of the global feature vector, $\boldsymbol{f}_k(\boldsymbol{h}, \boldsymbol{x})$, can be represented as a sum of the local edge features, $t(h_{(j-1)}, h_{(j)}, \boldsymbol{x}')$, or local state features, $s(h_{(j)}, \boldsymbol{x}')$, in which $\boldsymbol{x}'$ represents the context, i.e., the local observations. It is common (e.g., see [17]) for each feature to be an indicator function. For example, one such feature might be "$s(h_{(j)}, \boldsymbol{x}') = 1$ (if current context $\boldsymbol{x}'$ is the word *the*) or 0 (otherwise)." In this paper, we also use indicator features.

The latent structured perceptron attempts to minimize the difference between the score of the gold-standard latent-labeling (latent variable sequence) and the score of the (model) predicted latent labeling, with the following update formula:

$$\begin{aligned} \boldsymbol{w}^{i+1} = \boldsymbol{w}^i &+ \boldsymbol{f}\big[\arg\max_{\boldsymbol{h}} F\big(\boldsymbol{h}|\boldsymbol{y}_i^*, \boldsymbol{x}_i, \boldsymbol{w}^i\big), \boldsymbol{x}_i\big] \\ &- \boldsymbol{f}\big[\arg\max_{\boldsymbol{h}} F\big(\boldsymbol{h}|\boldsymbol{x}_i, \boldsymbol{w}^i\big), \boldsymbol{x}_i\big], \end{aligned} \tag{8}$$

where $\mathrm{argmax}_{\boldsymbol{h}} F(\boldsymbol{h}|\boldsymbol{y}_i^*, \boldsymbol{x}_i, \boldsymbol{w}^i)$ is the optimal (Viterbi) latent labeling of the gold-standard labeling $\boldsymbol{y}_i^*$; i.e., the latent sequence (conditioned on $\boldsymbol{y}^*$) with maximum score. The $\mathrm{argmax}_{\boldsymbol{h}} F(\boldsymbol{h}|\boldsymbol{x}_i, \boldsymbol{w}^i)$ is the predicted (unconstrainted) Viterbi latent labeling.

3.3 Randomized Estimation of Latent Structures

In the schema of randomized estimation of latent structures, the score of a gold-standard label sequence, $F(\boldsymbol{y}^*|\boldsymbol{x}, \boldsymbol{w})$, will be estimated by using the score of its randomized latent labeling:

$$F(\boldsymbol{y}^*|\boldsymbol{x}, \boldsymbol{w}) \triangleq F(\boldsymbol{h}'|\boldsymbol{x}, \boldsymbol{w}), \tag{9}$$

where $\boldsymbol{h}'$ is a randomly selected latent labeling from the gold-standard labeling $\boldsymbol{y}^*$. In other words, let the set $\mathcal{S} = \{\forall \boldsymbol{h}, \mathrm{Proj}(\boldsymbol{h}) = \boldsymbol{y}^*\}$, and $\boldsymbol{h}' \in \mathcal{S}$ is randomly selected from $\mathcal{S}$.

3.4 Learning Model Parameters

First, to make a comparison, we review the training of LCRFs. Fitting the LCRFs involves the normalizer, $\sum_{\forall \boldsymbol{h}} \exp \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{h}|\boldsymbol{x})$, and the summation over latent labelings for their corresponding labeling, $\sum_{\boldsymbol{h} \in \mathcal{H}(y_{(1)}) \times \ldots \times \mathcal{H}(y_{(m)})} P(\boldsymbol{h}|\boldsymbol{x}, \boldsymbol{w})$. Both of them are computationally expensive. In addition, during the training process, it requires computing the gradient of the objective function, which is also computationally expensive.

Compared to LCRFs, training the latent structured perceptron is much more efficient. It avoids computing the normalizer, the summation operation and the gradients during optimization. The major cost of the latent structured perceptron is from the computation of the Viterbi latent labelings by using the Viterbi dynamic programming algorithm.

The latent structured perceptron training algorithm is shown in Fig. 1, with a comparison to the training of the perceptron. In the algorithm, the projection from latent sequence to label sequence is to check whether the weight update is required for a sample. Since the weight update is based on the latent structures, the freedom of latent variables will not be reduced. In other words, the projection from latent sequence to label sequence is only a signal for weight updating, and the real operation of weight update is based on latent structures from maximized estimation.

Input: *Sample* $\boldsymbol{x}_i$ *with gold-standard label sequence* $\boldsymbol{y}_i^*$, *weight vector* $\boldsymbol{w}$, *and feature vector* $\boldsymbol{f}(\boldsymbol{y}, \boldsymbol{x})$
Initialization: *set parameters* $\boldsymbol{w}^1 = 0$
for $i = 1 \ldots d$ **do**
  $\boldsymbol{y}_i = \text{argmax}_{\boldsymbol{y}} F(\boldsymbol{y}|\boldsymbol{x}_i, \boldsymbol{w}^i)$
  if $\boldsymbol{y}_i \neq \boldsymbol{y}_i^*$ **then**
    $\boldsymbol{w}^{i+1} = \boldsymbol{w}^i + \boldsymbol{f}(\boldsymbol{y}_i^*, \boldsymbol{x}_i) - \boldsymbol{f}(\boldsymbol{y}_i, \boldsymbol{x}_i)$
  else
    $\boldsymbol{w}^{i+1} = \boldsymbol{w}^i$
Output: *parameter vectors* $\boldsymbol{w}^{i+1}$ *for* $i = 1 \ldots d$

Input: $\boldsymbol{x}_i$ *with* $\boldsymbol{y}_i^*$, *weight vector* $\boldsymbol{w}$, *and feature vector* $\boldsymbol{f}(\boldsymbol{y}, \boldsymbol{x})$
Initialization: *randomly initialize parameters with* $||\boldsymbol{w}^1|| \approx 0$
for $i = 1 \ldots d$ **do**
  $\boldsymbol{h}_i = \text{argmax}_{\boldsymbol{h}} F(\boldsymbol{h}|\boldsymbol{x}_i, \boldsymbol{w}^i)$
  $\boldsymbol{y}_i = \text{Proj}(\boldsymbol{h}_i)$
  if $\boldsymbol{y}_i \neq \boldsymbol{y}_i^*$ **then**
    $\boldsymbol{h}_i^* = \text{argmax}_{\boldsymbol{h}} F(\boldsymbol{h}|\boldsymbol{y}_i^*, \boldsymbol{x}_i, \boldsymbol{w}^i)$
    $\boldsymbol{w}^{i+1} = \boldsymbol{w}^i + \boldsymbol{f}(\boldsymbol{h}_i^*, \boldsymbol{x}_i) - \boldsymbol{f}(\boldsymbol{h}_i, \boldsymbol{x}_i)$
  else
    $\boldsymbol{w}^{i+1} = \boldsymbol{w}^i$
Output: *parameter vectors* $\boldsymbol{w}^{i+1}$ *for* $i = 1 \ldots d$

Fig. 1. Traditional structured perceptron algorithm (upper one), and the proposed latent structured perceptron algorithm (bottom one).

For simplicity, we only show the training based on the schema of maximized estimations, because it is straightforward to adapt this algorithm to another schema with randomized estimations. The parameters are initialized randomly. Each training sample is decoded using the current parameter settings. If the predicted latent labeling (under the current parameters) is different from the Viterbi latent labeling (conditioned on the gold-standard labeling), then the parameter vector $\boldsymbol{w}$ is updated in a simple additive fashion.

4 Theoretical Analysis

We now give theoretical analysis regarding the convergence of the latent structured perceptron. We will first analyze the case of maximized estimation of latent structures, then the case of randomized estimations. For both cases, we will show that separable data will remain separable with a bound. Moreover, after a finite number of updates, the latent structured perceptron is guaranteed to converge to the parameters with zero training error. Furthermore, as for the data which are not separable, we will derive a bound on the number of updates in the latent structured perceptron.

4.1 Maximized Estimation

To facilitate the discussion, we will refer to a problem as *separable* with the following definition:

Definition 1. *Let* $\mathcal{G}(\boldsymbol{x}_i)$ *signify all possible label sequences for an example* $\boldsymbol{x}_i$; *Let* $\overline{\mathcal{G}}(\boldsymbol{x}_i) = \mathcal{G}(\boldsymbol{x}_i) - \{\boldsymbol{y}_i^*\}$. *We will say that a training sequence* $(\boldsymbol{x}_i, \boldsymbol{y}_i^*)$ *for* $i = 1 \ldots d$ *is* separable with margin $\delta > 0$ *if there exists some vector* $\boldsymbol{U}$ *with* $\|\boldsymbol{U}\| = 1$ *such that*[1]

$$\forall i, \forall \boldsymbol{y}' \in \overline{\mathcal{G}}(\boldsymbol{x}_i), \boldsymbol{U} \cdot \boldsymbol{f}(\boldsymbol{y}_i^*, \boldsymbol{x}_i) - \boldsymbol{U} \cdot \boldsymbol{f}(\boldsymbol{y}', \boldsymbol{x}_i) \geq \delta. \tag{10}$$

For the case of maximized estimation of latent structures, a *latently separable* problem is defined as following:

Definition 2. *Let* $\mathcal{S}(\boldsymbol{x}_i)$ *signify all possible candidates of latent sequence for an example* $\boldsymbol{x}_i$; *let* $\mathcal{S}^*(\boldsymbol{x}_i) = \{\boldsymbol{h}|\text{Proj}(\boldsymbol{h}) = \boldsymbol{y}_i^*\}$, *and* $\overline{\mathcal{S}}(\boldsymbol{x}_i) = \mathcal{S}(\boldsymbol{x}_i) - \mathcal{S}^*(\boldsymbol{x}_i)$. *In other words,* $\mathcal{S}^*(\boldsymbol{x}_i)$ *represents the correct latent candidates for* $\boldsymbol{x}_i$, *and* $\overline{\mathcal{S}}(\boldsymbol{x}_i)$ *represents the incorrect ones. Then, a training sequence* $(\boldsymbol{x}_i, \boldsymbol{y}_i^*)$ *for* $i = 1 \ldots d$ *is* latently separable with margin $\overline{\delta} > 0$ *if there exists some vector* $\overline{\boldsymbol{U}}$ *with* $\|\overline{\boldsymbol{U}}\| = 1$ *such that*

$$\begin{aligned} &\forall i, \exists \boldsymbol{h} \in \mathcal{S}^*(\boldsymbol{x}_i), \forall \boldsymbol{h}' \in \overline{\mathcal{S}}(\boldsymbol{x}_i), \\ &\overline{\boldsymbol{U}} \cdot \boldsymbol{f}(\boldsymbol{h}, \boldsymbol{x}_i) - \overline{\boldsymbol{U}} \cdot \boldsymbol{f}(\boldsymbol{h}', \boldsymbol{x}_i) \geq \overline{\delta}. \end{aligned} \tag{11}$$

By using latent variables, a local state feature in the perceptron, $s(y, \boldsymbol{x}')$, will be mapped into a new feature set, $\{s(h_i, \boldsymbol{x}')\}$ for *all* possible $h_i \in \mathcal{H}(y)$, with the dimension $m_k = |\mathcal{H}(y)|$.[2] Similar mapping can be also extended to local edge features.[3] Since a global feature vector consists of local features, we use m_k to denote the *dimension augmentation* of the kth feature of the original feature vector, $\boldsymbol{f}_k(\boldsymbol{y}, \boldsymbol{x})$, then we get a vector $\boldsymbol{m} = (m_1, \ldots, m_n)$ so that a global feature vector, $\boldsymbol{f}(\boldsymbol{y}, \boldsymbol{x}_i) = (\beta_1, \ldots, \beta_n)$, will be mapped into a new feature vector with latent variables, $\boldsymbol{f}(\boldsymbol{h}, \boldsymbol{x}_i) = (\beta_1^1, \ldots, \beta_1^{m_1}, \ldots, \beta_n^1, \ldots, \beta_n^{m_n})$.

Based on the mapping, it is straightforward to prove that $\beta_k = \sum_{i=1}^{m_k} \beta_k^i$ for $k = 1 \ldots n$. Since m_k is only related to $|\mathcal{H}(y)|$ for all possible y from $\boldsymbol{f}_k(\boldsymbol{y}, \boldsymbol{x})$ and $\mathcal{H}(y)$ is fixed before the training, m_k will be an integral constant. We then define the *latent feature mapping* as the vector $\boldsymbol{m} = (m_1, \ldots, m_n)$, and we can state the following theorems:

Theorem 1. *For any sequence of training examples* $(\boldsymbol{x}_i, \boldsymbol{y}_i^*)$ *which is separable with margin* δ *by a vector* $\boldsymbol{U}$ *represented by* $(\alpha_1, \ldots, \alpha_n)$ *with* $\sum_{i=1}^n {\alpha_i}^2 = 1$, *the examples then will also be latently separable with margin* $\overline{\delta}$, *and* $\overline{\delta}$ *is bounded below by*

$$\overline{\delta} \geq \delta. \tag{12}$$

The proof is sketched in Section 9.

Theorem 2. *For any sequence of training examples* $(\boldsymbol{x}_i, \boldsymbol{y}_i^*)$ *which is separable with margin* δ, *the number of mistakes of the latent structured perceptron algorithm in Fig. 1 is bounded above by*

$$number\ of\ mistakes \leq 2M^2/\delta^2, \tag{13}$$

where M *is the maximum 2-norm of an original feature vector, i.e.,* $M = \max_{i,\boldsymbol{y}} \|\boldsymbol{f}(\boldsymbol{y}, \boldsymbol{x}_i)\|$.

The proof is sketched in Section 9.

In the case of inseparable data, we need the following definition:

1. $\|U\|$ is the 2-norm of U, i.e., $\|U\| = \sqrt{\sum_k U_k^2}$.
2. See Section 2 for $\mathcal{H}(y)$; $|set|$ means the number of elements of the set.
3. The only difference for the mapping is that an edge feature consists of more than one labels.

Definition 3. *Given a sequence $(\boldsymbol{x}_i, \boldsymbol{y}_i^*)$, for a $\overline{\boldsymbol{U}}$, $\overline{\delta}$ pair, define $g_i = \min_{\boldsymbol{h}\in\mathcal{S}^*(\boldsymbol{x}_i)} \overline{\boldsymbol{U}} \cdot \boldsymbol{f}(\boldsymbol{h}, \boldsymbol{x}_i) - \max_{\boldsymbol{h}'\in\overline{\mathcal{S}}(\boldsymbol{x}_i)} \overline{\boldsymbol{U}} \cdot \boldsymbol{f}(\boldsymbol{h}', \boldsymbol{x}_i)$. Then, define $D_{\overline{\boldsymbol{U}},\overline{\delta}}$ as the least obtainable error:*

$$D_{\overline{\boldsymbol{U}},\overline{\delta}} = \left[\sum_{i=1}^{d} \left(\max\{0, \overline{\delta} - g_i\}\right)^2\right]^{1/2}. \tag{14}$$

We can further have the following theorem:

Theorem 3. *For any training sequence $(\boldsymbol{x}_i, \boldsymbol{y}_i^*)$, the number of mistakes made by the latent structured perceptron training algorithm is bounded above by*

$$number\ of\ mistakes \le \min_{\overline{\boldsymbol{U}},\overline{\delta}} (\sqrt{2}M + D_{\overline{\boldsymbol{U}},\overline{\delta}})^2/\overline{\delta}^2, \tag{15}$$

where M is as before.

The proof can be adapted from Freund and Schapire [16] and Collins [17].

4.2 Randomized Estimation

For randomized estimation of latent structures, a *latently separable* problem is defined as following:

Definition 4. *Let $\mathcal{S}(\boldsymbol{x}_i)$ signify all possible candidates of latent sequence for an example $\boldsymbol{x}_i$, we then define $\mathcal{S}^*(\boldsymbol{x}_i) = \{\boldsymbol{h}|\mathrm{Proj}(\boldsymbol{h}) = \boldsymbol{y}_i^*\}$, and $\overline{\mathcal{S}}(\boldsymbol{x}_i) = \mathcal{S}(\boldsymbol{x}_i) - \mathcal{S}^*(\boldsymbol{x}_i)$. Then, a training sequence $(\boldsymbol{x}_i, \boldsymbol{y}_i^*)$ for $i = 1 \ldots d$ is* latently separable *with margin $\overline{\delta} > 0$ if there exists some vector $\overline{\boldsymbol{U}}$ with $\|\overline{\boldsymbol{U}}\| = 1$ such that*

$$\begin{aligned} &\forall i, \forall \boldsymbol{h} \in \mathcal{S}^*(\boldsymbol{x}_i), \forall \boldsymbol{h}' \in \overline{\mathcal{S}}(\boldsymbol{x}_i), \\ &\overline{\boldsymbol{U}} \cdot \boldsymbol{f}(\boldsymbol{h}, \boldsymbol{x}_i) - \overline{\boldsymbol{U}} \cdot \boldsymbol{f}(\boldsymbol{h}', \boldsymbol{x}_i) \ge \overline{\delta}. \end{aligned} \tag{16}$$

We can state the following theorems:

Theorem 4. *Given the latent feature mapping $\boldsymbol{m} = (m_1, \ldots, m_n)$, for any sequence of training examples $(\boldsymbol{x}_i, \boldsymbol{y}_i^*)$ which is separable with margin δ by a vector $\boldsymbol{U}$ represented by $(\alpha_1, \ldots, \alpha_n)$ with $\sum_{i=1}^n {\alpha_i}^2 = 1$, the examples then will also be latently separable with margin $\overline{\delta}$, and $\overline{\delta}$ is bounded below by*

$$\overline{\delta} \ge \delta/T, \tag{17}$$

where $T = (\sum_{i=1}^n m_i {\alpha_i}^2)^{1/2}$.

The proof is sketched in Section 9.

Theorem 5. *For any sequence of training examples $(\boldsymbol{x}_i, \boldsymbol{y}_i^*)$ which is separable with margin δ, the number of mistakes of the latent structured perceptron algorithm in Fig. 1 is bounded above by*

$$number\ of\ mistakes \le 2T^2M^2/\delta^2, \tag{18}$$

where T is as before, and M is the maximum 2-norm of an original feature vector, i.e., $M = \max_{i,\boldsymbol{y}} \|\boldsymbol{f}(\boldsymbol{y}, \boldsymbol{x}_i)\|$.

The proof is sketched in Section 9.

4.3 Interpretations and Discussions

For both maximized and randomized estimations of latent structures, we have shown that separable data will remain separable with a bound: δ and δ/T, respectively. We can see that $T = (\sum_{i=1}^n m_i {\alpha_i}^2)^{1/2} > (\sum_{i=1}^n {\alpha_i}^2)^{1/2}$. In addition, it is obvious that $(\sum_{i=1}^n {\alpha_i}^2)^{1/2} = 1$. Then,

$$T = \left(\sum_{i=1}^{n} m_i {\alpha_i}^2\right)^{1/2} > 1.$$

Hence, the randomized estimation will have a worse (smaller) lower bound. We conclude that the maximized estimation has better theoretical properties in terms of latent separability. With larger number of latent variables, m_i in the *latent feature mapping* (feature augmentation with latent variables) will have larger values. It follows that T will have a larger value. It follows that, with a larger number of latent variables, the latent separability of the maximized estimation will have a more significant advantage (compared with the randomized case).

Moreover, for both maximized/randomized cases, after a finite number of updates, the latent structured perceptron is guaranteed to converge to the parameter values with zero training error. For the maximized and randomized cases, the number of updates are bounded by $2M^2/\delta^2$ and $2T^2M^2/\delta^2$, respectively. Similar to the analysis of latent separabilities, we can see that the number of updates of the maximized case will be T times less than the one of the randomized case. With a larger number of latent variables, this advantage will be more significant.

Other than learning hidden information, it is possible that the latent variables can have other functions to make the problem more tractable. For example, one function of latent variables is to improve the separable margin of samples in practice. We have experimental observations that some originally inseparable instances (with the perceptron) became separable by adding latent variables.

5 A New Training Method

In practice, there is a refinement to the algorithm in Fig. 1, called a *parameter averaging* method, as discussed in the case of *averaged perceptrons*. Define $\boldsymbol{w}^{q,\,i}$ as the values of parameters after the ith training example has been processed in pass q over the training data. The *averaged parameters* are then defined as $\boldsymbol{\gamma}^{Q,d} = \sum_{q=1\ldots Q,\ i=1\ldots d} \boldsymbol{w}^{q,\,i}/dQ$ [16], [25].

It is a natural idea to use this averaged training method for latent structured perceptrons. However, we find this averaged training can be further improved for the latent structured perceptrons. Since the averaged parameters have better quality (for classification accuracies) than naive parameters, we have a question: Why not reinitiate the naive parameters by using averaged parameters in the training process? With this question, we consider updating the new parameters by using the averaged parameters. We propose a *parameter averaging with feedback (PAF)* algorithm for training the latent structured perceptrons: Instead of using $\boldsymbol{w}^{q,\,start} = \boldsymbol{w}^{q-1,\,end}$, we reinitiate (feeding back) the parameters of the new pass with the averaged parameters in a "good timing." We will show that a reasonable feedback schedule is the key to make the latent structured perceptron being robust and accurate.

Notes m is the number of periods when the PAF reaches the convergence;
b is the current number of period;
c is the current number of pass;
n is the number of training samples.

Procedure PAF-train
  Initialize $\boldsymbol{w}$ with random values
  $c \leftarrow 0$
  for $b \leftarrow 1$ to m
  . **for** 1 to b
  . $\boldsymbol{w} \leftarrow$ Update($\boldsymbol{w}$)
  . $c \leftarrow c + b$
  . $\boldsymbol{w} \leftarrow \overline{\boldsymbol{w}}^{iter(c)}$ in Eq. 19
  Return $\boldsymbol{w}$

Procedure Update($\boldsymbol{w}$)
  for 1 to n
  . select a sample j randomly
  . $\boldsymbol{w} \leftarrow \boldsymbol{w} + \nabla_{\boldsymbol{w}} \ell(j, \boldsymbol{w})$
  Return $\boldsymbol{w}$

Fig. 2. The PAF training. In the steps, it has the gradient term $\nabla_{\boldsymbol{w}} \ell(j, \boldsymbol{w})$, because the description is based on general stochastic learning frameworks. For latent structured perceptron, we have 0/1 loss function, and we can simply replace the gradient term with the additive update term $\boldsymbol{f}(\boldsymbol{h}_j^*, \boldsymbol{x}_j) - \boldsymbol{f}(\boldsymbol{h}_j', \boldsymbol{x}_j)$. When $\boldsymbol{h}_j^*$ equals to $\boldsymbol{h}_j'$, this additive update term is $\boldsymbol{0}$.

5.1 PAF

The naive version of parameter averaging is inspired by the averaged perceptron technique [17]. Let $\boldsymbol{w}^{iter(c),sample(d)}$ be the parameters after the dth training example has been processed in the cth pass over the training data. We define the *averaged parameters* at the end of the pass c' as:

$$\overline{\boldsymbol{w}}^{iter(c')} \triangleq \frac{\sum_{c=1...c',\, d=1...n} \boldsymbol{w}^{iter(c),sample(d)}}{nc'}. \tag{19}$$

It is noteworthy that the parameter averaging can be performed on the fly as follows:

$$\overline{\boldsymbol{w}}^t = \frac{t-1}{t}\overline{\boldsymbol{w}}^{t-1} + \frac{1}{t}\boldsymbol{w}^t,$$

where t is a count of the weight updates. Hence, there is no need to store the previous weight vectors.

As we discussed, we want to further improve the averaged training for latent structured perceptrons. A potential problem of traditional parameter averaging is that the model parameters $\boldsymbol{w}$ receive no information from the averaged parameters: The model parameters $\boldsymbol{w}$ are trained exactly the same like before (traditional parameter averaging). $\boldsymbol{w}$ could be misleading as the training goes on. To solve this problem, a natural idea is to reset $\boldsymbol{w}$ by using the averaged parameters, which are more reliable. We propose a refined version of parameter averaging by further applying a *"periodic feedback."*

We periodically reset the parameters $\boldsymbol{w}$ by using the averaged parameters $\overline{\boldsymbol{w}}$. The interval between a feedback operation and its previous operation is called a *training period* or simply a *period*. It is important to decide when to do the feedback, i.e., the length of each period should be adjusted reasonably as the training goes on. For example, at the early stage of the training, the $\boldsymbol{w}$ is highly noisy, so that the feedback operation to $\boldsymbol{w}$ should be performed more frequently. As the training goes on, less frequent feedback operation would be better to adequately optimize the parameters. In practice, we adopt a schedule of *linearly slowing-down feedback*.

Fig. 2 shows the steps of the PAF. Here, we analyze the averaged parameters produced by each period. We denote $\boldsymbol{w}^{b,c,d}$ as the model parameters after the dth sample is processed in the cth pass of the bth period. Without making any difference, we denote $\boldsymbol{w}^{b,c,d}$ more simply as $\boldsymbol{w}^{b,cn+d}$ where n is the number of samples in a training data. Similarly, we use $\boldsymbol{g}^{b,cn+d}$ to denote $\nabla_{\boldsymbol{w}} \ell(d, \boldsymbol{w})$ in the cth pass of the bth period. Let $\overline{\boldsymbol{w}}^{(b)}$ be the averaged parameters produced by the bth period. We can derive the explicit form of $\overline{\boldsymbol{w}}^{(1)}$:

$$\overline{\boldsymbol{w}}^{(1)} = \boldsymbol{w}^{1,0} + \sum_{d=1...n} \left(\frac{n-d+1}{n} \boldsymbol{g}^{1,d} \right). \tag{20}$$

When the second period ends, the parameters are again averaged over all previous model parameters, $\boldsymbol{w}^{1,0}, \ldots, \boldsymbol{w}^{1,n}, \boldsymbol{w}^{2,0}, \ldots, \boldsymbol{w}^{2,2n}$, and it has this form:

$$\begin{aligned} \overline{\boldsymbol{w}}^{(2)} = \boldsymbol{w}^{1,0} &+ \sum_{d=1...n} \left(\frac{n-d+1}{n} \boldsymbol{g}^{1,d} \right) \\ &+ \sum_{d=1...2n} \left(\frac{2n-d+1}{3n} \boldsymbol{g}^{2,d} \right). \end{aligned} \tag{21}$$

Similarly, the averaged parameters produced by the bth period can be expressed as follows:

$$\overline{\boldsymbol{w}}^{(b)} = \boldsymbol{w}^{1,0} + \sum_{i=1...b} \left[\sum_{d=1...in} \left(\frac{in-d+1}{ni(i+1)/2} \boldsymbol{g}^{i,d} \right) \right]. \tag{22}$$

The derivation of (22) is sketched in Section 9. We will show in experiments that this new training method is helpful for training latent structured perceptrons. We will also show in experiments that this new training method is convergent.

6 Experiments: Synthetic Data

In this section, we use synthetic data to demonstrate that the latent structured perceptron is able to address problems containing hidden information, which cannot be solved by conventional models.

The synthetic data sets are generated by an HMM model. The labels are $\{y_1, y_2\}$; the latent variables are $\{h_1, h_2, h_3, h_4\}$, with $\mathcal{H}(y_1) = \{h_1, h_2\}$ and $\mathcal{H}(y_2) = \{h_3, h_4\}$; the observable tokens are $\{x_1, x_2\}$. We vary the significance of the hidden information (dependencies among latent variables) to make experimental comparisons among different methods. The dependencies among latent variables are shown in Fig. 3. In the figure, p represents the transition probability $P(h_{(i+1)}|h_{(i)})$. A large value of p indicates a strong dependency between the two latent variables $h_{(i)}$ and $h_{(i+1)}$ (the remaining probability mass $1-p$ is uniformly distributed to the remaining transitions). The observations x_1 and x_2 are generated with $P(x|h)$. We set the observations to be weakly

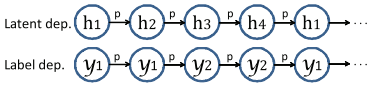


Fig. 3. Latent dependencies of the synthetic data, and the corresponding label dependencies.

dependent. The artificial label sequences are observable as oracle labelings during training but not observable during test. The latent variable sequences are unobservable in both training and test.

Modeling hidden information is important when this data set has strong hidden information (a large value of p). We illustrate this by using an example as follows: We assume the extreme case of strong hidden information that $p = 1$, and the current observations is $x_{(i)}$ and its oracle label is y_1 with the oracle latent variable h_1. Then, identifying the correct latent variable (i.e., h_1 here) is critical for the accuracy of the label sequence. If the system incorrectly decide this latent variable as h_2, then it will produce a labeling y_1, y_2, y_2, y_1 for the observations $x_{(i)}, x_{(i+1)}, x_{(i+2)}, x_{(i+3)}$, because of the dependencies learned from Fig. 3. If the system correctly decide this latent variable as h_1, then it will produce a labeling y_1, y_1, y_2, y_2, which is correct. We can see that the produced labelings can be very different if the learned latent variable on the current observation is different.

We vary the latent dependency probability p and therefore generate a variety of synthetic data with weak or strong hidden information. All the models employ the same features: node features $\{x_{(i)}\} \times \{y_{(i)}\}$ with edge features $\{y_{(i-1)}y_{(i)}\}$ for conventional models, and node features $\{x_{(i)}\} \times \{h_{(i)}\}$ with edge features $\{h_{(i-1)}h_{(i)}\}$ for latent structured perceptrons. We use the token accuracy to measure the performance.

As can be seen in Fig. 4, the latent structured perceptron outperforms conventional models like CRFs and structured perceptrons, and the significance of differences increase rapidly when the latent dependencies are becoming significant.

In this experiment, LatPerc is the latent structured perceptron with $\#LV = 2$ (i.e., setting two latent variables for each label). More latent variables (e.g., $\#LV = 4$) are not necessary in this task, because the latent dependencies are simple in this data set. The training algorithm for LatPerc is the PAF algorithm, because it achieves better performance than other training algorithms (e.g., traditional parameter averaging). In this experiment, Perc is the averaged perceptron method, because it performs better than naive perceptron methods. CRF is the CRFs with stochastic gradient descent (SGD) training. It is well known that SGD training can achieve as good accuracy as batch training methods for CRFs, SVMs and neural networks [26], [27], [28]. Moreover, the SGD training has much faster convergence speed than batch training methods.

7 Experiments: Activity Recognition

We use the ALKAN human activity recognition data set [29] for experiments. In human activity recognition problems, the training data are usually collected from different

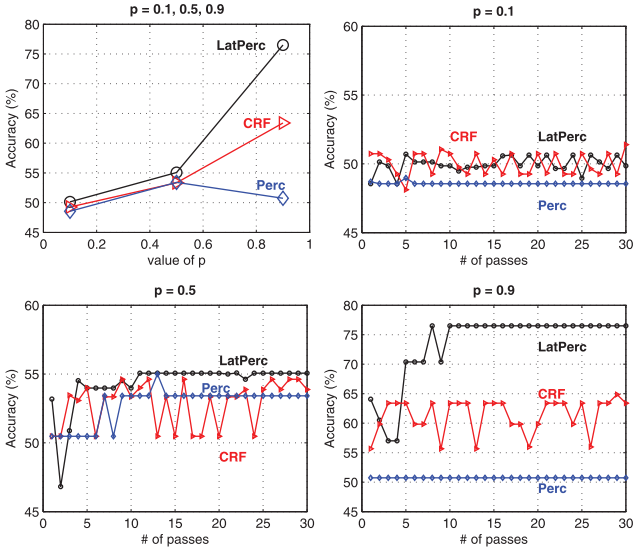


Fig. 4. Accuracy curves of the latent structured perceptron (LatPerc), structured perceptron (Perc) and CRFs on the synthetic data, when the degree of hidden information p varies.

persons (because the data collected from each individual is quite limited and sparse). In this case, an important type of hidden information is the person ID, because different persons usually have very different activity patterns. If this kind of hidden information (person ID) can be successfully modeled within the learning process, it is expected to be helpful to the performance of the learning system.

Our extracted data set contains more than 15,000 sessions, with about 500 temporal samples per session on average. The data are collected by *iPod* accelerometers with the sampling frequency of 20 Hz. A sample contains four values: time stamp and triaxial signals. For example, {539.266(s), 0.091(g), -0.145(g), -1.051(g)}.[4] There are six kinds of activity labels: `act-0` means *"walking or running,"* `act-1` means *"on an elevator or escalator,"* `act-2` means *"taking car or bus,"* `act-3` means *"taking train,"* `act-4` means *"up or down stairs,"* and `act-5` means *"standing or sitting."*

We randomly select 85 percent of samples for training, 5 percent samples for tuning hyperparameters, and the rest 10 percent samples for testing. The evaluation metric are window accuracy (the number of correctly predicted windows divided by the total number of windows). Other evaluation metrics like precision and recall are not proper in this task, because an activity segment is quite long (typically contains thousands of time windows), and small difference on the boundaries of segments can cause very different precision and recall. On the other hand, the accuracy metric is much more reliable in this scenario.

7.1 Experimental Settings

We have three traditional baselines and one latent-variable baseline to compare with the proposed method. The three traditional baselines include well-known structured classification models: perceptron (Perc-naive), averaged perceptron (Perc-avg), and CRFs. The latent-variable baseline is the powerful and heavy model: LCRFs.

4. In the example, "g" is the *acceleration rate of gravity*.

TABLE 1
Features Used in Activity Recognition

Single-axis based features:
(1) Signal strength features:
$\{s_{(i-2)},\ s_{(i-1)},\ s_{(i)},\ s_{(i+1)},\ s_{(i+2)},\ s_{(i-1)}s_{(i)},\ s_{(i)}s_{(i+1)}\}$
$\times\{y_{(i)},\ y_{(i-1)}y_{(i)}\}$
(2) Mean feature:
$m_{(i)}\ \times\{y_{(i)},\ y_{(i-1)}y_{(i)}\}$
(3) Standard deviation feature:
$d_{(i)}\ \times\{y_{(i)},\ y_{(i-1)}y_{(i)}\}$
(4) Energy feature:
$e_{(i)}\ \times\{y_{(i)},\ y_{(i-1)}y_{(i)}\}$
Multi-axis based features:
(1) Correlation features:
$\{c_{(1,2,i)}, c_{(2,3,i)}, c_{(1,3,i)}\}\ \times\{y_{(i)},\ y_{(i-1)}y_{(i)}\}$

$\mathcal{A}\times\mathcal{B}$ means a Cartesian product between two sets; i represents the window index; $h_{(i)}$ is the latent variable (for latent models) or the label (for nonlatent models).

For CRFs and LCRFs, we use SGD for their training. It is well known that SGD training can achieve as good accuracy as batch training methods [26], [27], [28], and with much faster convergence speed. Our preliminary experiments show that the batch training methods (e.g., limited-memory BFGS) require more than 200 passes in the training, but the final accuracy is almost the same like the SGD training in 50 passes. For CRFs and LCRFs, we vary the variance of the L_2 regularization from 0.1 to 100, and we found that $\sigma = 1$ worked well for both LCRFs and CRFs. For both the latent structured perceptron and LCRFs, we vary the number of latent variables per label from 2 to 5, and we will show their accuracies and efficiency (in terms of training time).

Following prior work in activity recognition [1], [2], [3], we use acceleration features, mean features, standard deviation, energy, and correlation features. The features are listed in Table 1. The relationship between single variables in observation is captured by features based on combinations of single variables. For example, the feature fragment $s_{(i-1)}s_{(i)}$ represents the combination of the previous observation variable $s_{(i-1)}$ and the current observation variable $s_{(i)}$.

Since the single-axis-based features on the three axes are extracted in the same way, for simplicity, we only describe the features on one axis. For multiaxis-based features, we use 1, 2, and 3 to index/represent the three axes. Following previous work [1], window-based features (mean, energy, deviation, etc.) are extracted. We use exactly the same feature set for all systems.

We denote the window index as i. The mean feature is simply the averaged signal strength in a window:

$$m_i = \frac{\sum_{k=1}^{|w|} s_k}{|w|},$$

where $s_1, s_2, \ldots$ are the signal magnitudes in a window. The energy feature is defined as follows:

$$e_i = \frac{\sum_{k=1}^{|w|} s_k^2}{|w|}.$$

The deviation feature is defined as follows:

$$d_i = \sqrt{\frac{\sum_{k=1}^{|w|}(s_k - m_i)^2}{|w|}},$$

TABLE 2
Results in the Activity Recognition Task

Methods	Accuracy	#passes	Time/Pass
Perc-naive	53.47 $\pm$1.77	50	3 sec
Perc-avg	56.32 $\pm$0.11	20	3 sec
CRF	55.82 $\pm$0.19	45	32 sec
LCRF	63.40 $\pm$0.94	45	121 sec
LatPerc-naive (new)	53.48 $\pm$1.43	50	14 sec
LatPerc-avg (new)	63.31 $\pm$1.81	50	15 sec
LatPerc-PAF (new)	**64.85** $\pm$1.21	30	15 sec

where the m_i is the mean value defined before. The correlation feature is defined as follows:

$$c_{1,2,i} = \frac{covariance_{1,2,i}}{d_{1,i}d_{2,i}},$$

where $d_{1,i}$ and $d_{2,i}$ are the deviation values on the ith window of axes 1 and 2, respectively. The $covariance_{1,2,i}$ is the covariance value between the ith windows of axes 1 and 2. We define correlation feature between other axis pairs in the same manner.

7.2 Results and Discussion

The major results are summarized in Table 2. In the figure, #passes is the number of training passes, which is determined by empirical convergence (for Perc-avg, CRF, LCRF, LatPerc-avg, and LatPerc-PAF). For the Perc-naive and LatPerc-naive methods, which are nonconvergent, their #passes are set as a large enough number, e.g., 50 (to make the experimental information complete, we will also show their detailed curves later).

As we can see, the proposed method, LatPerc-PAF, significantly outperforms traditional baseline methods, including structured perceptrons, averaged perceptrons, and CRFs. In addition, the proposed LatPerc method has much faster training speed than the CRF and LCRF methods. This is because the LatPerc method has very simple and efficient update schedule.

We perform experiments to compare the performance of LatPerc-Max and LatPerc-Rand in real-world applications. The results are shown in Fig. 5. As we can see, the maximized estimation performs much better than the randomized estimation in real-world applications, which confirms the theoretical superiority of the maximized estimations of hidden information.

Then, we compare the LatPerc (PAF training, maximized estimation) with traditional baseline methods with more details. The result curves are shown in Fig. 6. As we can see, the LatPerc has stable performance across different number of training passes (the curve has insignificant fluctuations). The structured perceptron has the most drastic fluctuations, because the parameter update is drastic and unstable in the naive training. The LatPerc achieves consistently higher accuracies than traditional baselines, because the activity recognition data contain strong hidden information (e.g., personal activity patterns).

We also compare the three training methods on LatPerc. The result curves are shown in Fig. 7, with latent variables varying from 2 to 5. As we can see, the proposed training method, PAF, outperforms the naive training and the averaged training. Overall, the LatPerc with $\#LV = 3, 4, 5$

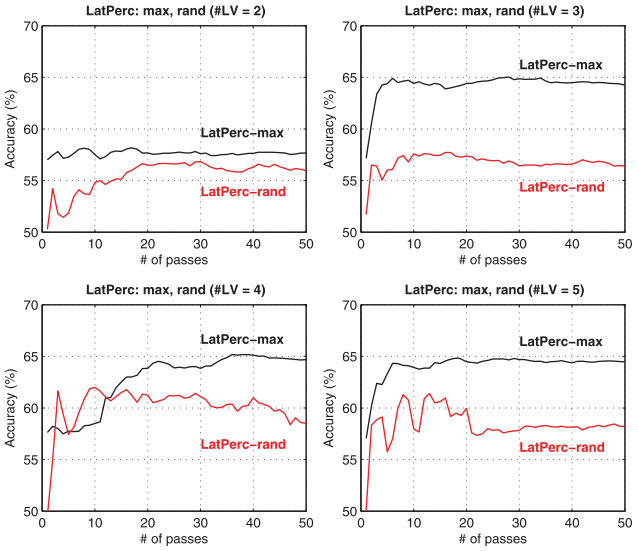


Fig. 5. Latent structured perceptron: maximized estimation (LatPerc-max) versus randomized estimation (LatPerc-rand) of hidden information. #LV is the number of latent variables for each label.

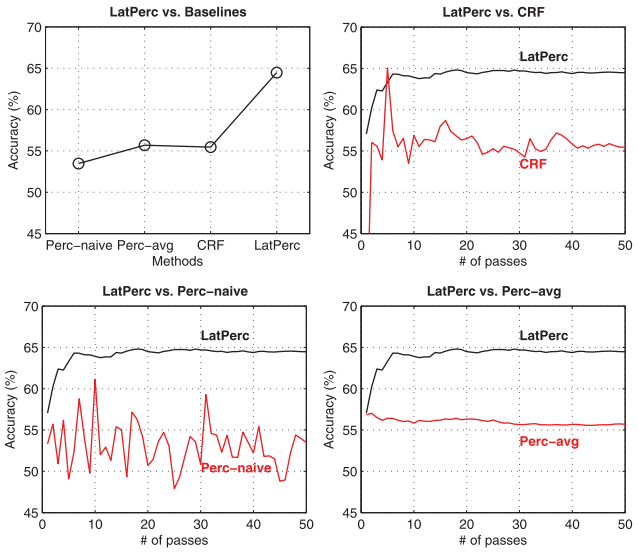


Fig. 6. The curves of the latent structured perceptron and baseline methods.

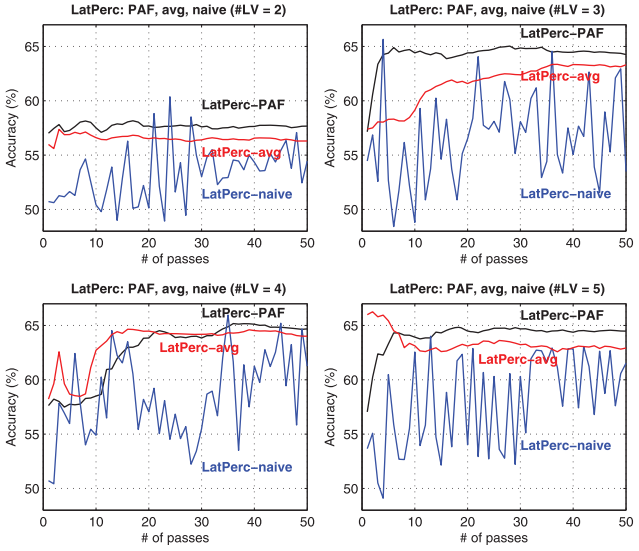


Fig. 7. Curves of the latent structured perceptron with three different training methods: naive training, averaged training (avg), and the proposed PAF training.

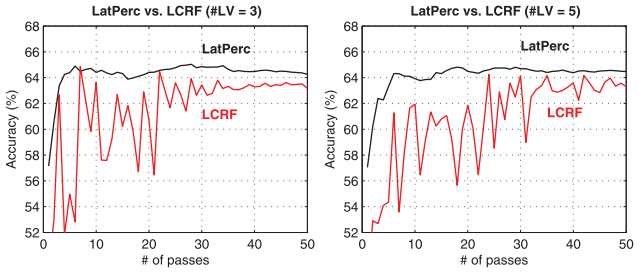


Fig. 8. The curves of the latent structured perceptron and the latent CRFs.

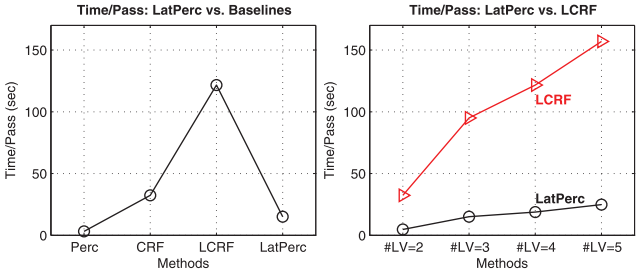


Fig. 9. Training time of different methods.

has much better performance than the LatPerc with $\#LV = 2$. This indicates that the hidden information in activity recognition can be grouped into three or more types. In this activity recognition data set, we find the naive training with latent variables had more severe fluctuations compared with the naive training without latent variables.

We also compare the LatPerc with a powerful and heavy latent variable model, LCRFs. The result curves are shown in Fig. 8. As we can see, interestingly, the LatPerc outperforms the LCRFs as well. The LatPerc is also more stable in the performance. The gap is smaller than the gaps between LatPerc and traditional nonlatent models, which is expected, because LCRFs can also effectively learn hidden information.

Finally, we show the training speed of the LatPerc and the baseline methods. The training speed is compared in terms of training time per pass. The results are shown in Fig. 9. As we can see, the LatPerc is faster than CRFs and LCRFs. Especially, the speedup of the LatPerc over the LCRFs is very significant: LatPerc is almost one order magnitude faster than LCRFs. Since both of the LatPerc and LCRFs are latent variable models, we will more focus on their comparison. The right panel in Fig. 9 shows the training speed of LatPerc and LCRFs over different number of latent variables. As we can see, the training time has a roughly linear increase on the increase of latent variables. The LatPerc has a less steep increase of training time when increase the number of latent variables. This means that the LatPerc has better scalability than LCRFs when increase the number of latent variables.

8 Experiments: Text Mining

A large amount of training data is available for the BIONLP-2004 shared task on Biomedical Named Entity Recognition (Bio-NER), which is appropriate for testing

TABLE 3
Summary of the Bio-NER Data Set

	#Abstracts	#Sentences	#Tokens
Train	2,000	20,546 (10/abs)	472,006 (23/sen)
Test	404	4,260 (11/abs)	96,780 (23/sen)

TABLE 4
Features Used in Bio-NER

Word Features:
$\{w_{(i-2)},\ w_{(i-1)},\ w_{(i)},\ w_{(i+1)},\ w_{(i+2)},\ w_{(i-1)}w_{(i)},\ w_{(i)}w_{(i+1)}\}$
$\times\{h_{(i)},\ h_{(i-1)}h_{(i)}\}$
POS Features:
$\{t_{(i-2)},\ t_{(i-1)},\ t_{(i)},\ t_{(i+1)},\ t_{(i+2)},\ t_{(i-2)}t_{(i-1)},\ t_{(i-1)}t_{(i)},\ t_{(i)}t_{(i+1)},\ t_{(i+1)}t_{(i+2)},\ t_{(i-2)}t_{(i-1)}t_{(i)},\ t_{(i-1)}t_{(i)}t_{(i+1)},\ t_{(i)}t_{(i+1)}t_{(i+2)}\}$
$\times\{h_{(i)},\ h_{(i-1)}h_{(i)}\}$
Orth. Features:
$\{o_{(i-2)},\ o_{(i-1)},\ o_{(i)},\ o_{(i+1)},\ o_{(i+2)},\ o_{(i-2)}o_{(i-1)},\ o_{(i-1)}o_{(i)},\ o_{(i)}o_{(i+1)},\ o_{(i+1)}o_{(i+2)}\}$
$\times\{h_{(i)},\ h_{(i-1)}h_{(i)}\}$

$w_{(i)}$ is the current word, $t_{(i)}$ is the POS tag, $o_{(i)}$ is the orthography mode, and $h_{(i)}$ is like before.

TABLE 5
Results in the Bio-NER Task

Methods	Accuracy	#passes	Time/Pass
Perc-naive	90.57 ±0.24	50	57 sec
Perc-avg	92.56 ±0.03	50	62 sec
CRF	92.20 ±0.04	50	191 sec
LCRF	92.06 ±0.08	40	371 sec
LatPerc-naive (new)	91.10 ±0.87	50	75 sec
LatPerc-avg (new)	92.45 ±0.10	40	80 sec
LatPerc-PAF (new)	**92.95** ±0.11	30	81 sec

the scalability of the models. The Bio-NER task is for recognizing five kinds of biomedical named-entities (DNA, RNA, Protein, etc.) on the GENIA corpus [30]. For each type of biomedical named entity, there can be subtypes of this entity. For example, there can be subtypes of DNA and they can have different patterns. Such subtypes can be treated as hidden information in this task, and we expect modeling such hidden information will be helpful to this task. Typical approach to this problem recasts it as a sequential labeling task with the `BIO` encoding.

This data set consists of 20,546 training samples (from 2,000 MEDLINE article abstracts, with 472,006 tokens) and 4,260 test samples. From the training data, we choose 2,500 samples as development data set for tuning hyperparameters. The properties of the data are summarized in Table 3. The evaluation metrics for this task [30] can be F-score or accuracy. Since in experiments we find the F-score and accuracy measures are always consistent with the same tendencies, we use the accuracy measure for simplicity and efficiency.

We use word features, POS features and orthography features (prefix, uppercase/lowercase, etc.), as listed in Table 4, and low frequency features are removed for efficiency. The experimental settings of baselines and hyperparameters are similar to those settings in the previous task of human activity recognition. For simplicity, we do not repeat the descriptions of experimental settings.

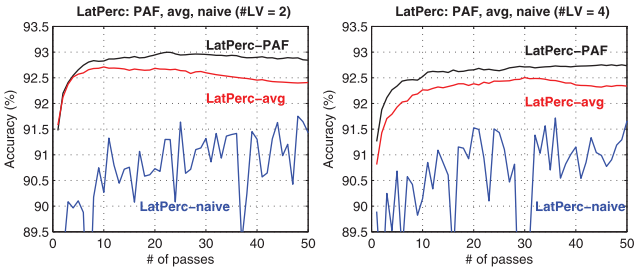


Fig. 10. Curves of the LatPerc with three different training methods in the Bio-NER task.

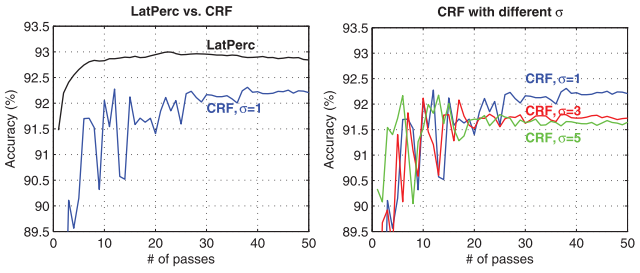


Fig. 11. LatPerc versus CRFs with different regularizers in the Bio-NER task.

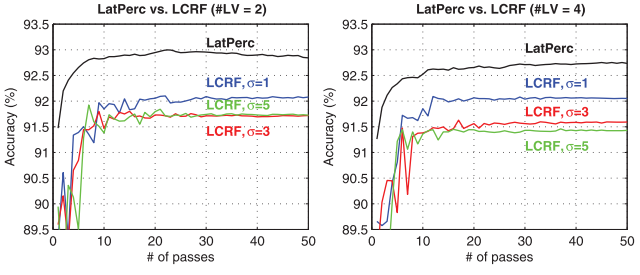


Fig. 12. LatPerc versus LCRFs with different regularizers and number of latent variables in the Bio-NER task.

8.1 Results and Discussion

The major experimental results are summarized in Table 5. As we can see, like the previous task, the proposed LatPerc method outperforms all of the baseline methods. In addition, the proposed method has much faster training speed than CRFs and LCRFs. An interesting difference compared with the activity recognition task is that the LCRF model works worse than CRFs and averaged perceptrons. The major reason could be from the nonconvexity of the objective function of the LCRFs. With a nonconvex objective function, only local optimums can be achieved by optimizing the LCRF weights. It is probable that the LCRF model did not find a good local optimum (a good local optimum means a local optimum that is close to the global optimum).

Fig. 10 shows the curves of the LatPerc with different training methods. As we can see, the proposed PAF training method works the best. Fig. 11 shows the curves of the LatPerc and the CRFs with different regularizers ($\sigma = 1, 3, 5$). We can see that the CRF model has the best performance with $\sigma = 1$.

Fig. 12 shows the curves of the LatPerc and the LCRFs with different regularizers and number of latent variables. We try $\#LV = 2, 3, 4, 5$. For simplicity, we illustrate curves with $\#LV = 2, 4$. As we can see, different from the activity recognition task, the LatPerc works well on $\#LV = 2$. Interestingly, the LatPerc works better on $\#LV = 2$ than $\#LV = 4$. This indicates that increasing the number of

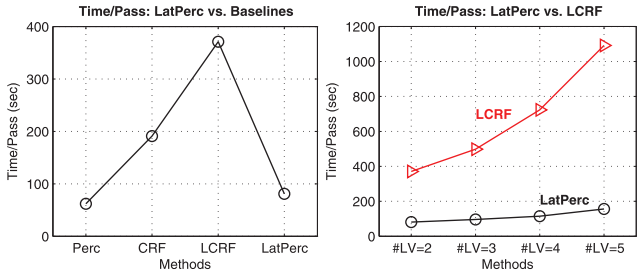


Fig. 13. Comparing the training time of different methods in the Bio-NER task.

latent variables may not always bring better performance. This is probably because increasing the number of latent variables can potentially bring higher risk of overfitting and data sparseness. Therefore, it is important to choose a proper number of latent variables for a specific data set. This can be done by using held-out data.

Finally, Fig. 13 shows the training time per pass of different methods. As we can see, the proposed LatPerc has faster training speed than CRFs and LCRFs. Especially, the speedup of the LatPerc over the LCRFs is impressive. The right panel of Fig. 13 shows the training speed of LatPerc and LCRFs over different number of latent variables. As we can see, like the previous task, the LatPerc has a less steep increase of training time when increase the number of latent variables. The flat curve of the LatPerc means that the LatPerc has good scalability. On the other hand, the LCRFs has weak scalability.

9 Proofs and Derivations

This section gives proofs of Theorem 1, 2, 4, and 5. We also give the derivation of (22) here.

Proof of Theorem 1. By the definition of seperability of δ, we know that

$$\forall i, \forall \boldsymbol{y}' \in \overline{\mathcal{G}}(\boldsymbol{x}_i), \boldsymbol{U} \cdot \boldsymbol{f}(\boldsymbol{y}_i^*, \boldsymbol{x}_i) - \boldsymbol{U} \cdot \boldsymbol{f}(\boldsymbol{y}', \boldsymbol{x}_i) \geq \delta.$$

Let $\boldsymbol{f}(\boldsymbol{y}_i^*, \boldsymbol{x}_i) = (f_1, f_2, \ldots, f_n)$ and let $\boldsymbol{f}(\boldsymbol{y}'', \boldsymbol{x}_i) = (f_1', f_2', \ldots, f_n')$, where $\boldsymbol{y}''$ is an instance of $\boldsymbol{y}' \in \overline{\mathcal{G}}(\boldsymbol{x}_i)$ so that it produces the maximum score of $\boldsymbol{U} \cdot \boldsymbol{f}(\boldsymbol{y}', \boldsymbol{x}_i)$.

Given the *latent feature mapping* $\boldsymbol{m} = (m_1, \ldots, m_n)$ and $\boldsymbol{U} \in \mathbb{R}^n$ represented by nonnegative vector $(\alpha_1, \ldots, \alpha_n)$, there exists a $\overline{\boldsymbol{U}} \in \mathbb{R}^N$ with $N = \sum_{i=1}^n m_i$ represented as follows:

$$\left(\overbrace{\alpha_1, 0, \ldots, 0}^{m_1}, \overbrace{\alpha_2, 0, \ldots, 0}^{m_2}, \ldots, \overbrace{\alpha_n, 0, \ldots, 0}^{m_n} \right).$$

Then, $\forall i, \forall \boldsymbol{h}' \in \overline{\mathcal{S}}(\boldsymbol{x}_i)$, there exists

$$\boldsymbol{h} = \left(\overbrace{f_1, 0, \ldots, 0}^{m_1}, \overbrace{f_2, 0, \ldots, 0}^{m_2}, \ldots, \overbrace{f_n, 0, \ldots, 0}^{m_n} \right)$$

and $\boldsymbol{h} \in \mathcal{S}^*(\boldsymbol{x}_i)$, so that

$$\begin{aligned} &\overline{\boldsymbol{U}} \cdot \boldsymbol{f}(\boldsymbol{h}, \boldsymbol{x}_i) - \overline{\boldsymbol{U}} \cdot \boldsymbol{f}(\boldsymbol{h}', \boldsymbol{x}_i) \\ &= \boldsymbol{U} \cdot \boldsymbol{f}(\boldsymbol{y}^*, \boldsymbol{x}_i) - \overline{\boldsymbol{U}} \cdot \boldsymbol{f}(\boldsymbol{h}', \boldsymbol{x}_i) \\ &\geq \boldsymbol{U} \cdot \boldsymbol{f}(\boldsymbol{y}^*, \boldsymbol{x}_i) - \boldsymbol{U} \cdot \boldsymbol{f}(\boldsymbol{y}'', \boldsymbol{x}_i) \\ &\geq \delta. \end{aligned} \tag{23}$$

According to the definition of latent separability in the case of maximized latent structures, the examples of $(\boldsymbol{x}_i, \boldsymbol{y}_i^*)$ are latently separable with a margin that is lower bounded by δ. □

Proof of Theorem 2. Let $\boldsymbol{w}^k$ be the parameters before the kth mistake is made, and suppose it is made at the ith example. Let $\boldsymbol{h}'$ be the output proposed at this example, $\boldsymbol{h}' = \operatorname{argmax}_{\boldsymbol{h} \in \mathcal{S}(\boldsymbol{x}_i)} \boldsymbol{f}(\boldsymbol{h}, \boldsymbol{x}_i) \cdot \boldsymbol{w}^k$. Also, let

$$\boldsymbol{h}^* = \operatorname*{argmax}_{\boldsymbol{h} \in \mathcal{S}^*(\boldsymbol{x}_i)} \boldsymbol{f}(\boldsymbol{h}, \boldsymbol{x}_i) \cdot \boldsymbol{w}^k.$$

It follows from the algorithm that $\boldsymbol{w}^{k+1} = \boldsymbol{w}^k + \boldsymbol{f}(\boldsymbol{h}^*, \boldsymbol{x}_i) - \boldsymbol{f}(\boldsymbol{h}', \boldsymbol{x}_i)$. Then, by using Theorem 1, it is clear that

$$\begin{aligned} \boldsymbol{U} \cdot \boldsymbol{w}^{k+1} &= \boldsymbol{U} \cdot \boldsymbol{w}^k + \boldsymbol{U} \cdot \boldsymbol{f}(\boldsymbol{h}^*, \boldsymbol{x}_i) - \boldsymbol{U} \cdot \boldsymbol{f}(\boldsymbol{h}', \boldsymbol{x}_i) \\ &\geq \boldsymbol{U} \cdot \boldsymbol{w}^k + \delta. \end{aligned} \tag{24}$$

It follows by induction on k that $\boldsymbol{U} \cdot \boldsymbol{w}^{k+1} \geq k\delta$ for all k (according to the initialization, $\|\boldsymbol{w}^1\| \approx 0$). In addition, because $\boldsymbol{U} \cdot \boldsymbol{w}^{k+1} \leq \|\boldsymbol{U}\| \cdot \|\boldsymbol{w}^{k+1}\|$ and $\|\boldsymbol{U}\| = 1$, it follows that

$$\|\boldsymbol{w}^{k+1}\| \geq k\delta. \tag{25}$$

On the other hand, we also derive an upper bound for $\|\boldsymbol{w}^{k+1}\|^2$. First, we have

$$\begin{aligned} \|\boldsymbol{w}^{k+1}\|^2 =& \|\boldsymbol{w}^k\|^2 + \|\boldsymbol{f}(\boldsymbol{h}^*, \boldsymbol{x}_i) - \boldsymbol{f}(\boldsymbol{h}', \boldsymbol{x}_i)\|^2 \\ &+ 2 \cdot \boldsymbol{w}^k \cdot [\boldsymbol{f}(\boldsymbol{h}^*, \boldsymbol{x}_i) - \boldsymbol{f}(\boldsymbol{h}', \boldsymbol{x}_i)]. \end{aligned} \tag{26}$$

Since $\boldsymbol{w}^k \cdot [\boldsymbol{f}(\boldsymbol{h}^*, \boldsymbol{x}_i) - \boldsymbol{f}(\boldsymbol{h}', \boldsymbol{x}_i)] < 0$ (because of the update rule) and $\boldsymbol{f}(\boldsymbol{h}^*, \boldsymbol{x}_i) \cdot \boldsymbol{f}(\boldsymbol{h}', \boldsymbol{x}_i) \geq 0$ (because of indicator features), it follows that

$$\begin{aligned} \|\boldsymbol{w}^{k+1}\|^2 &\leq \|\boldsymbol{w}^k\|^2 + \|\boldsymbol{f}(\boldsymbol{h}^*, \boldsymbol{x}_i) - \boldsymbol{f}(\boldsymbol{h}', \boldsymbol{x}_i)\|^2 \\ &\leq \|\boldsymbol{w}^k\|^2 + \|\boldsymbol{f}(\boldsymbol{h}^*, \boldsymbol{x}_i)\|^2 + \|\boldsymbol{f}(\boldsymbol{h}', \boldsymbol{x}_i)\|^2 \\ &= \|\boldsymbol{w}^k\|^2 + \sum_{j=1}^n \sum_{k=1}^{m_j} (\beta_j^{k*})^2 + \sum_{j=1}^n \sum_{k=1}^{m_j} (\beta_j^k)^2 \\ &\leq \|\boldsymbol{w}^k\|^2 + \sum_{j=1}^n \left(\sum_{k=1}^{m_j} \beta_j^{k*} \right)^2 + \sum_{j=1}^n \left(\sum_{k=1}^{m_j} \beta_j^k \right)^2 \\ &= \|\boldsymbol{w}^k\|^2 + \sum_{j=1}^n (\beta_j^*)^2 + \sum_{j=1}^n (\beta_j)^2 \\ &= \|\boldsymbol{w}^k\|^2 + \|\boldsymbol{f}(\boldsymbol{y}_i^*, \boldsymbol{x}_i)\|^2 + \|\boldsymbol{f}(\boldsymbol{y}^{\boldsymbol{h}'}, \boldsymbol{x}_i)\|^2 \\ &\leq \|\boldsymbol{w}^k\|^2 + 2M^2. \end{aligned} \tag{27}$$

Then, it follows that $\|\boldsymbol{w}^{k+1}\|^2 \leq 2kM^2$. Combining the upper bound and lower bound, we have

$$k^2\delta^2 \leq 2kM^2.$$

Then, we have $k \leq 2M^2/\delta^2$. □

Proof of Theorem 4. Given the *latent feature mapping* $\boldsymbol{m} = (m_1, \ldots, m_n)$ and $\boldsymbol{U} \in \mathbb{R}^n$ represented by $(\alpha_1, \ldots, \alpha_n)$, there exist a $\overline{\boldsymbol{U}} \in \mathbb{R}^N$ with $N = \sum_{i=1}^n m_i$ represented as follows:

$$\left(\overbrace{\alpha_1, \ldots, \alpha_1}^{m_1}, \overbrace{\alpha_2, \ldots, \alpha_2}^{m_2}, \ldots, \overbrace{\alpha_n, \ldots, \alpha_n}^{m_n} \right).$$

Then, $\forall i, \forall \boldsymbol{h} \in \mathcal{S}^*(\boldsymbol{x}_i), \forall \boldsymbol{h}' \in \overline{\mathcal{S}}(\boldsymbol{x}_i)$,

$$\begin{aligned}
&\overline{\boldsymbol{U}} \cdot \boldsymbol{f}(\boldsymbol{h}, \boldsymbol{x}_i) - \overline{\boldsymbol{U}} \cdot \boldsymbol{f}(\boldsymbol{h}', \boldsymbol{x}_i) \\
&= \overline{\boldsymbol{U}}[\boldsymbol{f}(\boldsymbol{h}, \boldsymbol{x}_i) - \boldsymbol{f}(\boldsymbol{h}', \boldsymbol{x}_i)] \\
&= \sum_{j=1}^{n} \left[\alpha_j \sum_{k=1}^{m_j} \left(\beta_j^{k*} - \beta_j^k \right) \right] \\
&= \sum_{j=1}^{n} \alpha_j \left(\sum_{k=1}^{m_j} \beta_j^{k*} - \sum_{k=1}^{m_j} \beta_j^k \right) \qquad (28) \\
&= \sum_{j=1}^{n} \alpha_j (\beta_j^* - \beta_j) \\
&= \boldsymbol{U}[\boldsymbol{f}(\boldsymbol{y}_i^*, \boldsymbol{x}_i) - \boldsymbol{f}(\boldsymbol{y}^{\boldsymbol{h}'}, \boldsymbol{x}_i)] \\
&\geq \delta,
\end{aligned}$$

where $\beta_j^{k*} \in \boldsymbol{f}(\boldsymbol{h}, \boldsymbol{x}_i)$ is the kth feature value mapped from $\beta_j^* \in \boldsymbol{f}(\boldsymbol{y}_i^*, \boldsymbol{x}_i)$ by using latent variables; similarly, $\beta_j^k \in \boldsymbol{f}(\boldsymbol{h}', \boldsymbol{x}_i)$ is mapped from $\beta_j \in \boldsymbol{f}(\boldsymbol{y}^{\boldsymbol{h}'}, \boldsymbol{x}_i)$; $\boldsymbol{y}^{\boldsymbol{h}'}$ is the label sequence projected from the latent sequence $\boldsymbol{h}'$. Let $T = \|\overline{\boldsymbol{U}}\| = (\sum_{i=1}^{n} m_i {\alpha_i}^2)^{1/2}$, then it can be seen that the vector $\overline{\boldsymbol{U}}/\|\overline{\boldsymbol{U}}\|$ latently separates the data with margin δ/T. □

Proof of Theorem 5. Let $\boldsymbol{w}^k$ be the parameters before the kth mistake is made, and suppose it is made at the ith example. Let $\boldsymbol{h}'$ be the output proposed at this example, $\boldsymbol{h}' = \operatorname{argmax}_{\boldsymbol{h} \in \mathcal{S}(\boldsymbol{x}_i)} \boldsymbol{f}(\boldsymbol{h}, \boldsymbol{x}_i) \cdot \boldsymbol{w}^k$. Also, let

$$\boldsymbol{h}^* = \operatorname*{argmax}_{\boldsymbol{h} \in \mathcal{S}^*(\boldsymbol{x}_i)} \boldsymbol{f}(\boldsymbol{h}, \boldsymbol{x}_i) \cdot \boldsymbol{w}^k.$$

Then, in a similar way like the proof of Theorem 2, we can derive that

$$\|\boldsymbol{w}^{k+1}\| \geq k\delta/T. \qquad (29)$$

Again, in a similar way like the proof of Theorem 2, we can derive an identical upper bound for $\|\boldsymbol{w}^{k+1}\|^2$:

$$\|\boldsymbol{w}^{k+1}\|^2 \leq \|\boldsymbol{w}^k\|^2 + 2M^2. \qquad (30)$$

It follows that $\|\boldsymbol{w}^{k+1}\|^2 \leq 2kM^2$. Combining the upper and lower bounds completes the proof. □

Derivation of (22). We follow the denotations of $\boldsymbol{w}^{b,cn+d}$, $\boldsymbol{g}^{b,cn+d}$, and $\overline{\boldsymbol{w}}^{(b)}$ defined in Section 5. First, based on the optimization procedure defined in Fig. 2, we analyze the resulting parameters at the end of each period. For the first period, the parameters after each update is as follows:

$$\begin{aligned}
&\boldsymbol{w}^{1,1} = \boldsymbol{w}^{1,0} + \boldsymbol{g}^{1,1}, \\
&\boldsymbol{w}^{1,2} = \boldsymbol{w}^{1,1} + \boldsymbol{g}^{1,2} = \boldsymbol{w}^{1,0} + \gamma^{(1)}\boldsymbol{g}^{1,1} + \gamma^{(1)}\boldsymbol{g}^{1,2}, \\
&\cdots \\
&\boldsymbol{w}^{1,n} = \boldsymbol{w}^{1,n-1} + \boldsymbol{g}^{1,n} \\
&\quad = \boldsymbol{w}^{1,0} + \boldsymbol{g}^{1,1} + \boldsymbol{g}^{1,2} + \cdots + \boldsymbol{g}^{1,n}.
\end{aligned}$$

At the end of the first period, the parameters are averaged as follows:

$$\begin{aligned}
\overline{\boldsymbol{w}}^{(1)} &= \frac{\sum_{d=1\ldots n} \boldsymbol{w}^{1,d}}{n} \\
&= \boldsymbol{w}^{1,0} + \sum_{d=1\ldots n} \left(\frac{n-d+1}{n} \boldsymbol{g}^{1,d} \right).
\end{aligned}$$

Then, at the beginning of the second period, we initialize the model parameters with averaged parameters: $\boldsymbol{w}^{2,0} \leftarrow \overline{\boldsymbol{w}}^{(1)}$. In the second period, the parameters are updated based on the synchronized parameters, with a new decaying rate and the passes increased to 2. When the second period ends, the parameters are again averaged over all previous model parameters, $\boldsymbol{w}^{1,0}, \ldots, \boldsymbol{w}^{1,n}, \boldsymbol{w}^{2,0}, \ldots, \boldsymbol{w}^{2,2n}$:

$$\begin{aligned}
\overline{\boldsymbol{w}}^{(2)} &= \frac{\sum_{d=1\ldots n} \boldsymbol{w}^{1,d} + \sum_{d=1\ldots 2n} \boldsymbol{w}^{2,d}}{n+2n} \\
&= \frac{n\overline{\boldsymbol{w}}^{(1)} + \sum_{d=1\ldots 2n} \boldsymbol{w}^{2,d}}{3n} \\
&= \frac{n\overline{\boldsymbol{w}}^{(1)} + [2n\overline{\boldsymbol{w}}^{(1)} + \sum_{d=1\ldots 2n}(2n-d+1)\boldsymbol{g}^{2,d}]}{3n} \\
&= \boldsymbol{w}^{1,0} + \sum_{d=1\ldots n} \left(\frac{n-d+1}{n} \boldsymbol{g}^{1,d} \right) \\
&\quad + \sum_{d=1\ldots 2n} \left(\frac{2n-d+1}{3n} \boldsymbol{g}^{2,d} \right).
\end{aligned}$$

In a similar way, it is straightforward to conclude that

$$\overline{\boldsymbol{w}}^{(b)} = \boldsymbol{w}^{1,0} + \sum_{i=1\ldots b} \left[\sum_{d=1\ldots ni} \left(\frac{ni-d+1}{n\sum_{k=1\ldots i} k} \boldsymbol{g}^{i,d} \right) \right]. \qquad (31)$$

This completes the derivation.

10 Conclusions and Future Work

We proposed a perceptron-style method, latent structured perceptron, for fast discriminative learning of structured classification with hidden information. We gave theoretical analysis and demonstrated good convergence properties of the proposed method. We performed experiments on one synthetic data set and two real-world structured classification tasks. Compared to conventional nonlatent models, our method was significantly more accurate on real-world tasks. Compared to existing heavy probabilistic models of latent variables, our method lowered the training cost significantly (almost one order magnitude faster) yet with comparable or even superior classification accuracy. In addition, experiments demonstrated that the proposed method has good scalability on large-scale problems.

Although we focused on the sequential labeling tasks in this paper, the theoretical framework and algorithm should also be applicable to a wide variety of models beyond linear-chain structures.

Acknowledgments

This work was supported by a Hong Kong RGC Project (No. PolyU 5230/08E), and National High Technology Research and Development Program of China (863 Program) (No. 2012AA011101). This work was a substantial extension of a conference paper in IJCAI 2009 [31]. The authors thank Daisuke Okanohara and Junichi Tsujii for helpful discussions, Hisashi Kashima and Naonori Ueda for the evaluation data on activity recognition, and the reviewers who gave helpful comments. X. Sun is the corresponding author.

REFERENCES

[1] L. Bao and S.S. Intille, "Activity Recognition from User-Annotated Acceleration Data," *Proc. Int'l Conf. Pervasive Computing,* pp. 1-17, 2004.

[2] N. Ravi, N. Dandekar, P. Mysore, and M.L. Littman, "Activity Recognition from Accelerometer Data," *Proc. 17th Conf. Innovative Applications of Artificial Intelligence (IAAI '05),* pp. 1541-1546, 2005.

[3] T. Huynh, M. Fritz, and B. Schiele, "Discovery of Activity Patterns Using Topic Models," *Proc. 10th Int'l Conf. Ubiquitous Computing,* pp. 10-19, 2008.

[4] A. Quattoni, M. Collins, and T. Darrell, "Conditional Random Fields for Object Recognition," *Proc. Advances in Neural Information Processing Systems (NIPS '04),* 2004.

[5] L.-P. Morency, A. Quattoni, and T. Darrell, "Latent-Dynamic Discriminative Models for Continuous Gesture Recognition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR '07),* pp. 1-8, 2007.

[6] X. Sun, L.-P. Morency, D. Okanohara, and J. Tsujii, "Modeling Latent-Dynamic in Shallow Parsing: A Latent Conditional Model with Improved Inference," *Proc. Int'l Conf. Computational Linguistics (COLING '08),* pp. 841-848, 2008.

[7] X. Sun, N. Okazaki, and J. Tsujii, "Robust Approach to Abbreviating Terms: A Discriminative Latent Variable Model with Global Information," *Proc. Ann. Meeting Assoc. Computational Linguistics (ACL '09),* pp. 905-913, Aug. 2009.

[8] D. Yu, L. Deng, and A. Acero, "Hidden Conditional Random Field with Distribution Constraints for Phone Classification," *Proc. Conf. Int'l Speech Comm. Assoc. (InterSpeech '09),* 2009.

[9] Y. Wang and G. Mori, "Max-Margin Hidden Conditional Random Fields for Human Action Recognition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR '09),* 2009.

[10] S. Petrov, "Products of Random Latent Variable Grammars," *Proc. Ann. Conf. North Am. Chapter Assoc. Computational Linguistics (NAACL '10),* 2010.

[11] S. Petrov and D. Klein, "Discriminative Log-Linear Grammars with Latent Variables," *Proc. Advances in Neural Information Processing Systems (NIPS),* pp. 1153-1160, 2008.

[12] S.B. Wang, A. Quattoni, L.-P. Morency, D. Demirdjian, and T. Darrell, "Hidden Conditional Random Fields for Gesture Recognition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR '06),* pp. 1521-1527, 2006.

[13] A. Quattoni, S. Wang, L.-P. Morency, M. Collins, and T. Darrell, "Hidden Conditional Random Fields," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 29, no. 10, pp. 1848-1852, Oct. 2007.

[14] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Rev.,* vol. 65, pp. 386-408, 1958.

[15] Y. LeCun, S. Chopra, R. Hadsell, R. Marc'Aurelio, and F.-J. Huang, "A Tutorial on Energy-Based Learning," *Predicting Structured Data,* MIT Press, 2006.

[16] Y. Freund and R. Schapire, "Large Margin Classification using the Perceptron Algorithm," *Machine Learning,* vol. 37, no. 3, pp. 277-296, 1999.

[17] M. Collins, "Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms," *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP '02),* pp. 1-8, 2002.

[18] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online Passive-Aggressive Algorithms," *J. Machine Learning Research,* vol. 7, pp. 551-585, 2006.

[19] P.F. Felzenszwalb, R.B. Girshick, D.A. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part-Based Models," *IEEE Trans. Pattern Analysis Machine Intelligence,* vol. 32, no. 9, pp. 1627-1645, Sept. 2010.

[20] C. Cherry and C. Quirk, "Discriminative, Syntactic Language Modeling through Latent SVMs," *Proc. Assoc. Machine Translation Am. (AMTA '08),* N. Calzolari, C. Cardie, and P. Isabelle, eds., 2008.

[21] C.-N.J. Yu and T. Joachims, "Learning Structural SVMs with Latent Variables," *Proc. Int'l Conf. Machine Learning (ICML '09),* p. 147, 2009.

[22] J. Lafferty, A. McCallum, and F. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," *Proc. 18th Int'l Conf. Machine Learning (ICML '01),* pp. 282-289, 2001.

[23] F. Sha and F. Pereira, "Shallow Parsing with Conditional Random Fields," *NAACL '03: Proc. Conf. North Am. Chapter Assoc. Computational Linguistics Human Language Technology,* pp. 134-141, 2003.

[24] P. Liang, A. Bouchard-Coté, D. Klein, and B. Taskar, "An End-to-End Discriminative Approach to Machine Translation," *Proc. Ann. Meeting Assoc. Computational Linguistics (ACL '06),* N. Calzolari, C. Cardie, and P. Isabelle, eds., 2006.

[25] X. Sun, H. Kashima, T. Matsuzaki, and N. Ueda, "Averaged Stochastic Gradient Descent with Feedback: An Accurate, Robust, and Fast Training Method," *Proc. Int'l Conf. Data Mining (ICDM '10),* pp. 1067-1072, 2010.

[26] S. Vishwanathan, N.N. Schraudolph, M.W. Schmidt, and K.P. Murphy, "Accelerated Training of Conditional Random Fields with Stochastic Meta-Descent," *Proc. Int'l Conf. Machine Learning (ICML '06),* pp. 969-976, 2006.

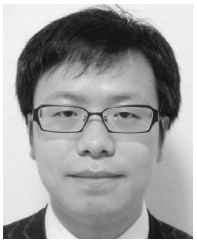
[27] L. Bottou, "Stochastic Learning," *Advanced Lectures on Machine Learning,* pp. 146-168, Springer, 2004.

[28] S. Shalev-Shwartz, Y. Singer, and N. Srebro, "Pegasos: Primal Estimated Sub-Gradient Solver for SVM," *Proc. Int'l Conf. Machine Learning (ICML '07),* 2007.

[29] Y. Hattori, M. Takemori, S. Inoue, G. Hirakawa, and O. Sudo, "Operation and Baseline Assessment of Large Scale Activity Gathering System by Mobile Device," *Proc. Multimedia, Distributed, Cooperative, and Mobile Symposium (DICOMO '10),* 2010.

[30] J.-D. Kim, T. Ohta, Y. Tsuruoka, and Y. Tateisi, "Introduction to the Bio-Entity Recognition Task at JNLPBA," *Proc. Int'l Joint Workshop NLP Biomedicine and Its Applications (BioNLP '04),* pp. 70-75, 2004.

[31] X. Sun, T. Matsuzaki, D. Okanohara, and J. Tsujii, "Latent Variable Perceptron Algorithm for Structured Classification," *Proc. 21st Int'l Joint Conf. Artificial Intelligence (IJCAI '09),* pp. 1236-1242, 2009.



Xu Sun received the BE degree from the Huazhong University of Science & Technology, the MS degree from Peking University, and the PhD degree from the University of Tokyo in 2004, 2007, and 2010, respectively. His research interests include natural language processing, data mining, and machine learning.

Takuya Matsuzaki received the MSc and PhD degrees in information science and technology from the University of Tokyo, Japan, in 2004 and 2007, respectively. He was a researcher at the University of Tokyo from 2007 to 2011, under the supervision of Prof. Junichi Tsujii. He has been an associate research professor at the National Institute of Informatics since 2012. His research interests include natural language parsing and its application.

Wenjie Li received the PhD degree from the Department of Systems Engineering and Engineering Management, Chinese University of Hong Kong, Hong Kong, in 1997. She is currently an associate professor in the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. Her main research interests include natural language processing, text mining and automatic summarization.