Latent Variable Perceptron Algorithm for Structured Classification

Xu Sun[†] **Takuya Matsuzaki**[†] **Daisuke Okanohara**[†] **Jun'ichi Tsujii**[†‡§]
[†]Department of Computer Science, University of Tokyo,
Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033, Japan
[‡]School of Computer Science, University of Manchester, UK
[§]National Centre for Text Mining, UK
{sunxu, matuzaki, hillbig, tsujii}@is.s.u-tokyo.ac.jp

Abstract

We propose a perceptron-style algorithm for fast discriminative training of structured latent variable model, and analyzed its convergence properties. Our method extends the perceptron algorithm for the learning task with latent dependencies, which may not be captured by traditional models. It relies on Viterbi decoding over latent variables, combined with simple additive updates. Compared to existing probabilistic models of latent variables, our method lowers the training cost significantly yet with comparable or even superior classification accuracy.

1 Introduction

Real-world problems may contain latent dependencies (i.e., hidden sub-structures) that are difficult to be captured by conventional models for structured classification, such as conditional random fields (CRFs). In such cases, models that exploit latent variables are advantageous in learning [Petrov and Klein, 2008]. As a result, recently discriminative probabilistic latent variable models (DPLVMs) have become popular for performing a variety of tasks with latent dependencies, e.g., vision recognition [Morency *et al.*, 2007], syntactic parsing [Petrov and Klein, 2008], and syntactic chunking [Sun *et al.*, 2008].

Morency et al. [2007] demonstrated that DPLVM models can learn latent dependencies of natural problems efficiently, and outperform several widely-used conventional models, such as support vector machines (SVMs), conditional random fields and hidden Markov models (HMMs). Petrov and Klein [2008] reported on a syntactic parsing task that DPLVM models can learn more compact and accurate grammars than that of the conventional techniques without latent variables.

However, experimental results in Petrov et al. [2008] have highlighted both time and memory cost problems on training. They used 8 CPUs in parallel for 3 days to train the weighted grammars with latent variables. And because of the memory limitations, their solution fails to learn more complex grammars. Furthermore, our experiments on the named entity recognition (NER) task confirmed that the training of DPLVMs with a low Markov order is already computationally expensive on large scale problems. On a corpus of 10K sentences, the training took more than a week for a linear chain model with latent variables.

On the other hand, the perceptron algorithms (including voted or averaged version) [Rosenblatt, 1958; LeCun *et al.*, 2006] have been shown to be competitive to the modern learning algorithms on structured classification tasks, while their computational costs are much lower [Freund and Schapire, 1999; Collins, 2002]. However, it is still not clear how to apply the perceptron algorithms for the learning task with latent dependencies.

In this paper we propose a novel latent variable perceptron algorithm (*latent perceptron* below) as an alternative to the DPLVMs. We will show that the latent perceptron gives comparable or even superior performance than the DPLVMs, while it is significantly faster, making it amenable to large scale problems.

2 Problem Definition and Background

In practice, given a limited training data, the relationship between specific observations and their contexts may be best modeled at a level finer than explicit class labels but coarser than direct token observations.

For example, in the noun phrase (NP) chunking task, suppose that there are two lexical sequences, "*He is her* –" and "*He gave her* –". They would both be conventionally labeled as 'BOB', where B signifies the '*beginning NP*', and O '*outside NP*'. However, this labeling is too general to encapsulate their respective syntactic dependencies, which is crucial for labeling the next word. For "*He is her* –", the NP started by '*her*' is still incomplete, so the label for '–' is likely to be I, which signifies the continuation of the phrase, e.g., "*[He] is [her brother]*". In contrast, for "*He gave her* –", the phrase started by '*her*' is normally self-complete and makes the next label more likely to be B, e.g., "*[He] gave [her] [flowers]*". We expect better performance if the model is able to represent the 'latent' fact, that in the first case, '*her*' is most likely the possesive pronoun, while in the second case, it is probably the indirect object pronoun.

Following this example, we give a more formal definition of the problem. The task is to learn a mapping between a sequence of observations $\mathbf{x} = x_1, x_2, \ldots, x_m$ and a sequence

of labels $\mathbf{y} = y_1, y_2, \ldots, y_m$. Each y_j is a class label for the j'th token of a observation sequence and is a member of a label set, Y. For each sequence, we also assume a vector of latent variables $\mathbf{h} = h_1, h_2, \ldots, h_m$, 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_j is a member of a set, $\mathbf{H}_{y_j}$, which represents the possible latent variables for the class label y_j.

Following the problem definition described above, DPLVMs are defined as [Morency *et al.*, 2007; Petrov and Klein, 2008]:

$$P(\mathbf{y}|\mathbf{x}, \mathbf{\Theta}) = \sum_{\mathbf{h}} P(\mathbf{y}|\mathbf{h}, \mathbf{x}, \mathbf{\Theta}) P(\mathbf{h}|\mathbf{x}, \mathbf{\Theta}), \quad (1)$$

where $\mathbf{\Theta}$ represents the parameters of the model. Since sequences with $\mathbf{h}_j \notin \mathbf{H}_{y_j}$ will have $P(\mathbf{y}|\mathbf{x}, \mathbf{\Theta}) = 0$ by the restriction of disjointed association, the model can be further defined as [Morency *et al.*, 2007; Petrov and Klein, 2008]:

$$P(\mathbf{y}|\mathbf{x}, \mathbf{\Theta}) = \sum_{\mathbf{h} \in \mathbf{H}_{y_1} \times \ldots \times \mathbf{H}_{y_m}} P(\mathbf{h}|\mathbf{x}, \mathbf{\Theta}), \quad (2)$$

where $P(\mathbf{h}|\mathbf{x}, \mathbf{\Theta})$ is defined using the usual conditional random field formulation:

$$P(\mathbf{h}|\mathbf{x}, \mathbf{\Theta}) = \frac{\exp \mathbf{\Theta} \cdot \mathbf{f}(\mathbf{h}, \mathbf{x})}{\sum_{\forall \mathbf{h}} \exp \mathbf{\Theta} \cdot \mathbf{f}(\mathbf{h}, \mathbf{x})}, \quad (3)$$

in which $\mathbf{f}(\mathbf{h}, \mathbf{x})$ is the global feature vector. Given training sequences $(\mathbf{x}_i, \mathbf{y}_i^*)$ for $i = 1 \ldots d$, training is performed by optimizing the likelihood of the training data, usually with a regularization of $R(\mathbf{\Theta})$:

$$L(\mathbf{\Theta}) = \sum_{i=1}^{d} \log P(\mathbf{y}_i^*|\mathbf{x}_i, \mathbf{\Theta}) - R(\mathbf{\Theta}). \quad (4)$$

3 Latent Variable Perceptron

Based on the above problem definition, now we present an alternative model for DPLVMs. We define the score of a label sequence $F(\mathbf{y}|\mathbf{x}, \mathbf{\Theta})$ as the maximum score among its latent sequence:

$$F(\mathbf{y}|\mathbf{x}, \mathbf{\Theta}) = \max_{\mathbf{h}:\mathrm{Proj}(\mathbf{h})=\mathbf{y}} F(\mathbf{h}|\mathbf{x}, \mathbf{\Theta}), \quad (5)$$

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

$$\mathrm{Proj}(\mathbf{h}) = \mathbf{y} \iff h_j \in \mathbf{H}_{y_j} \ for \ j = 1, \ldots, m, \quad (6)$$

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

$$F(\mathbf{h}|\mathbf{x}, \mathbf{\Theta}) = \mathbf{\Theta} \cdot \mathbf{f}(\mathbf{h}, \mathbf{x}) = \sum_k \mathbf{\Theta}_k \cdot \mathbf{f}_k(\mathbf{h}, \mathbf{x}), \quad (7)$$

where the k'th value of the global feature vector, $\mathbf{f}_k(\mathbf{h}, \mathbf{x})$, can be represented as a sum of the local edge features, $t(h_{j-1}, h_j, \mathbf{x}')$, or local state features, $s(h_j, \mathbf{x}')$, in which $\mathbf{x}'$ reprents the context, i.e., the local observations. It is common (e.g., see [Collins, 2002]) for each feature to be an indicator function. For example, one such feature might be "$s(h_j, \mathbf{x}') = 1$ (if current context $\mathbf{x}'$ is the word *the*) or 0 (otherwise)". In this paper, we also use indicator features.

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

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

Figure 1: The training algorithms of the perceptron (upper one) and the latent perceptron (lower one).

The latent perceptron attempts to minimize the difference between the score of the goal feature vector for a training instance $(\mathbf{x}_i, \mathbf{y}_i^*)$ and the score of current feature vector of the best latent sequence, with the following update formula:

$$\begin{aligned} \mathbf{\Theta}^{i+1} = \mathbf{\Theta}^i &+ \mathbf{f}[\mathrm{argmax}_{\mathbf{h}} F(\mathbf{h}|\mathbf{y}_i^*, \mathbf{x}_i, \mathbf{\Theta}^i), \mathbf{x}_i] \\ &- \mathbf{f}[\mathrm{argmax}_{\mathbf{h}} F(\mathbf{h}|\mathbf{x}_i, \mathbf{\Theta}^i), \mathbf{x}_i], \end{aligned} \quad (8)$$

where $\mathrm{argmax}_{\mathbf{h}} F(\mathbf{h}|\mathbf{y}_i^*, \mathbf{x}_i, \mathbf{\Theta}^i)$ is the Viterbi path of the latent sequences under the constraint of gold standard label sequence, and $\mathrm{argmax}_{\mathbf{h}} F(\mathbf{h}|\mathbf{x}_i, \mathbf{\Theta}^i)$ is the current best latent sequence.

3.1 Learning Model Parameters

First, to make a comparison, we review the training of DPLVMs. Fitting the DPLVMs involves the normalizer, $\sum_{\forall \mathbf{h}} \exp \mathbf{\Theta} \cdot \mathbf{f}(\mathbf{h}|\mathbf{x})$, and the marginalization over latent sequences for corresponding label sequence, $\sum_{\mathbf{h} \in \mathbf{H}_{y_1} \times \ldots \times \mathbf{H}_{y_m}} P(\mathbf{h}|\mathbf{x}, \mathbf{\Theta})$. Both of them are computationally expensive. In addition, if to adopt a Quasi-Newton optimizer as most DPLVMs did, it requires computing the derivatives $\partial L(\mathbf{\Theta}) / \partial \mathbf{\Theta}$, which is also computationally expensive.

Compared to DPLVMs, training the latent perceptron is much more efficient. It avoids computing the normalizer, the marginalization operation and the derivatives during optimization. The major cost of the latent perceptron is from

the computation of the best latent path by using the Viterbi algorithm.

The latent perceptron training algorithm is shown in Figure 1, with a comparison to the training of the perceptron. The parameters are initialized randomly. Each sentence in turn is decoded using the current parameter settings. If the latent path with the highest score under the current model is incorrect, then the parameter vector $\mathbf{\Theta}$ is updated in a simple additive fashion.

A Modified Estimation Method for Latent Perceptron

There is a refinement to the algorithm in Figure 1 , called the 'averaged parameters' method. Define $\mathbf{\Theta}^{q,\,i}$ as the values of parameters after the i'th training example has been processed in pass q over the training data. The *averaged parameters* are then defined as $\gamma^{Q,d} = \sum_{q=1\ldots Q,\ i=1\ldots d} \mathbf{\Theta}^{q,\,i}/dQ$ [Freund and Schapire, 1999].

However, we found this refinement is not adequate for the latent perceptron. Naturally, one possible way to further control the overfitting is to 'decay' the new parameters by using the averaged parameters. Following this idea, we propose a *modified averaged parameters* algorithm for training the latent perceptron: instead of using $\mathbf{\Theta}^{q,\,start} = \mathbf{\Theta}^{q-1,\,end}$, we re-initiate the parameters of the new iteration with the averaged parameters in each k iteration, i.e., $\mathbf{\Theta}^{q,\,start} = \gamma^{q-1,\,end}$ when $q\%k = 0$. Here, $\%$ represents the mod operation. Selecting a proper value of k is important for the performance, and this can be done by using a development data set. We normally choose the value of k within the range of [2,8], depending on the specific task.

Although this modification does not improve the performance of the perceptron, it reduces the overfitting of the latent perceptron and improves its performance. Therefore, we use this modified estimation method for training the latent perceptron.

3.2 Convergence Analysis

We now give theorems regarding the convergence of the latent perceptron. We will prove that a separable data will remain separable with a bound. Moreover, after a finite number of updates, the latent perceptron is guaranteed to converge to the parameter values with zero training error. Furthermore, as for the data which is not separable, we will derive a bound on the number of updates in the latent perceptron.

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

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

$$\forall i, \forall \mathbf{z} \in \overline{\mathbf{G}}(\mathbf{x}_i), \mathbf{U} \cdot \mathbf{f}(\mathbf{y}_i^*, \mathbf{x}_i) - \mathbf{U} \cdot \mathbf{f}(\mathbf{z}, \mathbf{x}_i) \geq \delta. \quad (9)$$

Also, a *latently separable* problem is defined as following:

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

$$\begin{aligned} &\forall i, \forall \mathbf{h} \in \mathbf{GH}^*(\mathbf{x}_i), \forall \mathbf{z} \in \overline{\mathbf{GH}}(\mathbf{x}_i), \\ &\overline{\mathbf{U}} \cdot \mathbf{f}(\mathbf{h}, \mathbf{x}_i) - \overline{\mathbf{U}} \cdot \mathbf{f}(\mathbf{z}, \mathbf{x}_i) \geq \overline{\delta}. \end{aligned} \quad (10)$$

By using latent variables, a local state feature in the perceptron, $s(y, \mathbf{x}')$, will be mapped into a new feature set, $\{s(h_i, \mathbf{x}')\}$ for *all* possible $h_i \in \mathbf{H}_y$, with the dimension $m_k = |\mathbf{H}_y|$.[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 k'th feature of the original feature vector, $\mathbf{f}_k(\mathbf{y}, \mathbf{x})$, then we get a vector $\mathbf{m} = (m_1, \ldots, m_n)$ so that a global feature vector, $\mathbf{f}(\mathbf{y}, \mathbf{x}_i) = (\beta_1, \ldots, \beta_n)$, will be mapped into a new feature vector with latent variables, $\mathbf{f}(\mathbf{h}, \mathbf{x}_i) = (\beta_1^1, \ldots, \beta_1^{m_1}, \ldots, \beta_n^1, \ldots, \beta_n^{m_n})$.

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 $|\mathbf{H}_y|$ for all possible y from $\mathbf{f}_k(\mathbf{y}, \mathbf{x})$ and $\mathbf{H}_y$ is fixed before the training, m_k will be an integral constant. We then define the *latent feature mapping* as the vector $\mathbf{m} = (m_1, \ldots, m_n)$, and we can state the following theorems:

Theorem 1. *Given the latent feature mapping $\mathbf{m} = (m_1, \ldots, m_n)$, for any sequence of training examples $(\mathbf{x}_i, \mathbf{y}_i^*)$ which is separable with margin δ by a vector $\mathbf{U}$ 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*

$$\overline{\delta} \geq \delta / T, \quad (11)$$

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

The proof is sketched in Appendix.

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

$$number\ of\ mistakes \leq 2T^2M^2/\delta^2, \quad (12)$$

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

The proof is sketched in Appendix.

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

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

$$D_{\overline{\mathbf{U}}, \overline{\delta}} = [\sum_{i=1}^{d} (\max\{0, \overline{\delta} - g_i\})^2]^{1/2}. \quad (13)$$

[1] $||\mathbf{U}||$ is the 2-norm of $\mathbf{U}$, i.e., $||\mathbf{U}|| = \sqrt{\sum_k \mathbf{U}_k^2}$.

[2] See Section 2 for $\mathbf{H}_y$; $|set|$ 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.

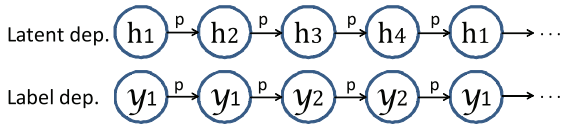

Figure 2: Latent dependencies of the synthetic data, and the corresponding label dependencies.

Hence we can further have the following theorem:

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

$$number\ of\ mistakes \leq \min_{\mathbf{U},\delta}(\sqrt{2}M + D_{\mathbf{U},\delta})^2/\delta^2, \quad (14)$$

where M *is as before.*

The proof can be adapted from Freund and Shapire [1999] and Collins [2002].

4 Experiments on Synthetic Data

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

The synthetic datasets are generated by the HMM model. The labels are $\{y_1, y_2\}$; the latent variables are $\{h_1, h_2, h_3, h_4\}$, with the association $\mathbf{H}_{y_1} = \{h_1, h_2\}$ and $\mathbf{H}_{y_2} = \{h_3, h_4\}$; the observable tokens are $\{x_1, x_2\}$. We vary the significance of the latent dependencies to make comparisons. The latent dependencies are illustrated in Figure 2. P represents the transition probability $P(h_j|h_i)$. A large value of P indicates a strong dependency (the remaining prob. mass '$1-P$' is uniformly distributed to the remaining transitions). x_1 and x_2 are generated with $P(x_1|h_i)$ and $P(x_2|h_i)$, but since this is not the focus, we set them to be very weakly dependent. Therefore, one artificial label sequence could be like '$y_1, y_1, y_2, y_2, y_1, \ldots$', while the observable tokens will appear like a random sequence. The artificial label sequences are observable as gold standards during training but not observable during test. The latent variable sequences are unobservable in both training and test.

We varied the probability P and therefore generated a variety of synthetic data with latent dependencies. All the models employ similar feature templates: 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 models; therefore only the 1st order dependencies among labels or latent variables are considered. We use the label accuracy to measure the performance.

As can be seen in Figure 3, the latent perceptron and DPLVMs[4] outperform conventional models like CRFs, and the gap increases rapidly when the latent dependencies are significant. When P is 0.9, the accuracy of the latent perceptron and DPLVMs are both 84.9%, in contrast to CRFs' 63.4% and the perceptron's 57.3%. In most cases, latent perceptron performs as well as DPLVMs, as can be seen from their almost identical curves. We also found that, by using more latent variables than necessary (e.g., 4 latent variables for each label), both the latent perceptron and DPLVMs kept roughly the same accuracies.

[4]Since the batch-training performs better than stochastic-training for DPLVMs (see next section), we adopts the batch-training here.

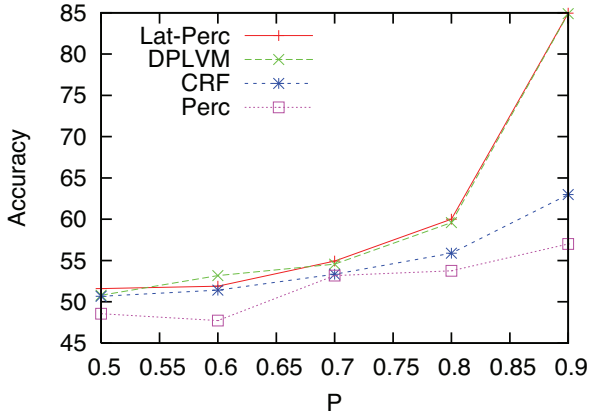

Figure 3: The curves of the latent perceptron (Lat-Perc), DPLVMs, perceptron (Perc) and CRFs, when P varies.

Note that, in the previous section, we have proved the convergence in the worst case, by using a very strict definition of *latent separability*. Clearly there are many 'separable' cases under the latent perceptron that do not satisfy this strict definition.

Here, we relax the 'separation' under the latent perceptron as follows: $\max_{\mathbf{h}\in\mathbf{GH}^*(\mathbf{x}_i)} \boldsymbol{\Theta}' \cdot \mathbf{f}(\mathbf{h}, \mathbf{x}_i) > \max_{\mathbf{z}\in\overline{\mathbf{GH}}(\mathbf{x}_i)} \boldsymbol{\Theta}' \cdot \mathbf{f}(\mathbf{z}, \mathbf{x}_i)$ for $i = 1 \ldots d$. Then, there are many cases that an originally inseparable instance (under the perceptron) become 'separable' by involving latent variables. For example, with $P = 0.9$, among the 3,000 training instances of this data, we found that less than 10% instances were separable when use the perceptron (trained with enough iterations), i.e., $\boldsymbol{\Theta} \cdot \mathbf{f}(\mathbf{y}_i^*, \mathbf{x}_i) > \boldsymbol{\Theta} \cdot \max_{\mathbf{z}\in\overline{\mathbf{G}}(\mathbf{x}_i)} \mathbf{f}(\mathbf{z}, \mathbf{x}_i)$, while the separable instances increased to more than 40% when use the latent perceptron, i.e., $\max_{\mathbf{h}\in\mathbf{GH}^*(\mathbf{x}_i)} \boldsymbol{\Theta}' \cdot \mathbf{f}(\mathbf{h}, \mathbf{x}_i) > \max_{\mathbf{z}\in\overline{\mathbf{GH}}(\mathbf{x}_i)} \boldsymbol{\Theta}' \cdot \mathbf{f}(\mathbf{z}, \mathbf{x}_i)$.

5 Experiments on Real-World Tasks

In this section we perform experiments on two real-world tasks, biomedical named entity recognition and noun phrase chunking. As the baseline systems, we implemented DPLVMs and CRFs by extending the HCRF library developed by Morency et al. [2007]. We added a popular batch-training optimizer, the Limited-Memory BFGS algorithm (L-BFGS) [Nocedal and Wright, 1999]. For higher efficiency, we also implemented the stochastic gradient descent algorithm (SGD) [Bottou, 1998]. To reduce overfitting, we employed a Gaussian prior. We varied the the variance of the Gaussian prior (with values 10^k, k from -3 to 3), and we found that $\sigma^2 = 1.0$ was optimal for both DPLVMs and CRFs on the development data and used it throughout the experi-

Word Features:
$\{w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}, w_{i-1}w_i, w_iw_{i+1}\}$
$\times\{h_i,\ h_{i-1}h_i,\ h_{i-2}h_{i-1}h_i\}$
POS Features:
$\{t_{i-2},\ t_{i-1},\ t_i,\ t_{i+1},\ t_{i+2},\ t_{i-2}t_{i-1},\ t_{i-1}t_i,\ t_it_{i+1},$ $t_{i+1}t_{i+2}, t_{i-2}t_{i-1}t_i, t_{i-1}t_it_{i+1}, t_it_{i+1}t_{i+2}\}$
$\times\{h_i,\ h_{i-1}h_i,\ h_{i-2}h_{i-1}h_i\}$
Orth. Features:
$\{o_{i-2},\ o_{i-1},\ o_i,\ o_{i+1},\ o_{i+2},\ o_{i-2}o_{i-1},\ o_{i-1}o_i,$ $o_io_{i+1}, o_{i+1}o_{i+2}\}$
$\times\{h_i,\ h_{i-1}h_i,\ h_{i-2}h_{i-1}h_i\}$

Table 1: Feature templates used in the Bio-NER task. w_i is the current word, t_i is the POS tag, o_i is the orthography mode, and h_i is the latent variable (for latent models) or the label (for conventional models).

Models	F	#iters	Train-time
Lat-Perc, cc=2, odr=2	**70.51**	20	≈30 hr
DPLVM-SGD, cc=2, odr=2	N/A	N/A	>240 hr
Perc, cc=2, odr=2	70.07	24	6 hr
Lat-Perc, cc=2, odr=1	**70.13**	24	3 hr
DPLVM-SGD, cc=2, odr=1	68.74	100	≈120 hr
Perc, cc=2, odr=1	68.93	20	1 hr
Lat-Perc, cc=5, odr=1	**68.49**	60	3 hr
DPLVM-BFGS, cc=5, odr=1	N/A	N/A	>240 hr
DPLVM-SGD, cc=5, odr=1	66.01	90	≈60 hr
Perc, cc=5, odr=1	65.61	22	<1 hr
CRF-BFGS, cc=5, odr=1	66.12	200	≈40 hr
Lat-Perc, cc=15, odr=1	67.33	45	1 hr
DPLVM-BFGS, cc=15, odr=1	**67.41**	300	≈160 hr
DPLVM-SGD, cc=15, odr=1	65.96	100	≈30 hr
Perc,cc=15, odr=1	65.83	20	<1 hr
CRF-BFGS, cc=15, odr=1	64.17	300	≈20 hr

Table 2: Comparisons among models on the Bio-NER task. Odr=2 means it is a 2nd-order model. #iter is the number of training iterations, which is determined by using the development data.

ments in this section.

5.1 BioNLP/NLPBA-2004 Shared Task (Bio-NER)

A large amount of training data is available for the Bio-NER task, which are appropriate to be used for testing the scalability of the models. Named entity recognition aims to identify and classify technical terms in a given domain. The bio-NER task is for recognizing 5 kinds of biomedical named-entities on the GENIA corpus [Kim *et al.*, 2004]. The typical approach to this problem recast it as a sequential labeling task with the `BIO` encoding. The data consists of 1,800 abstracts (from MEDLINE) for training, 200 abstracts for the development data, and 404 abstracts for testing. The standard evaluation metrics [Kim *et al.*, 2004] are precision p, recall r, and the F-measure given by $F = 2pr/(p + r)$.

We use word features, POS features and orthography features (prefix, uppercase/lowercase, etc.), as listed in Table 1. For both the latent perceptron and DPLVMs, we varied the number of latent variables per label from 2 to 8 on our preliminary experiments, and we finally use 4. The latent perceptron adopts the *modified averaged parameter* training (see comparisons in next subsection), and the perceptron adopts the original one.

As can be seen in Table 2, we performed 4 sets of experiments by scaling the models from simple to complex[5]. Cc=n means only features occurring n times or more are included. When cc=15, the training speed[6] of the latent perceptron is 130 times faster than that of the DPLVMs with batch-training (L-BFGS). The latent perceptron performs almost as well as DPLVM-BFGS, and outperforms DPLVM-SGD and other models. When more features were added by setting cc=5, DPLVMs with batch-training became intractable, while the latent perceptron still worked efficiently and achieved much better performance than the DPLVM-SGD, by reducing over 7.0% errors. It also outperformed the CRFs and the perceptron with statistically significant differences (McNemar's significance test).

When cc=2, CRFs also became intractable. When the 2nd order dependencies were further added, even the stochastic-training became intractable for DPLVMs. On the other hand, the latent perceptron worked efficiently, with further error reductions.

Overall, with the same feature set, the latent perceptron demonstrates consistent superiority over the DPLVM-SGD model and the perceptron. Compared to a straightforward high order perceptron, one advantage of the latent perceptron comes from its use of latent variables to adaptively learn long range dependencies. With the F-measure of 70.51%, the latent perceptron achieved a comparable performance to the state-of-the-art systems, and it would be further improved when enhanced with external features or resources, as is done in other systems. The state-of-the-art systems is Okanohara et al. [2006] (71.4%), who used a semi-Markov CRF model with specifically designed global features. Whereas in our latent perceptron model, we did not use such global features. Finkel et al. [2004] (70.0%) used a maximum entropy Markov model with external resources like web-querying. Settles [2004] (69.8%) used a CRF model with additional semantic knowledge.

5.2 Noun Phrase Chunking Task (NP-Chunking)

We choose the NP-chunking task [Sang and Buchholz, 2000] because it is a popular benchmark for testing a structured classification model. In this task, the non-recursive cores of noun phrases called base NPs are identified. The training set consists of 8,936 sentences, and the test set consists of 2,012 sentences. We use features that depend on words and POS tags (see Table 3), and our preliminary experiments in this task suggested the use of 5 latent variables for each label.

In this experiment, our main purpose is to compare the traditional parameter averaging technique with our modified version. However, we also showed the performance of other models for reference.

[5]The major feature templates are similar.

[6]On Intel Dual-Core Xeon 5160/3GHz CPU, excluding time for feature generation and data input/output.

Word Features:
$\{w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}, w_{i-1}w_i, w_iw_{i+1}\}$
$\times\{h_i,\ h_{i-1}h_i,\ h_{i-2}h_{i-1}h_i\}$
POS Features:
$\{t_{i-1},\ t_i,\ t_{i+1},\ t_{i-2}t_{i-1},\ t_{i-1}t_i,\ t_it_{i+1},\ t_{i+1}t_{i+2},$
$t_{i-2}t_{i-1}t_i, t_{i-1}t_it_{i+1}, t_it_{i+1}t_{i+2}\}$
$\times\{h_i,\ h_{i-1}h_i,\ h_{i-2}h_{i-1}h_i\}$

Table 3: Feature templates used in the NP-chunking experiments. w_i, t_i and h_i are as before.

Models	F	#iters	Train-time
Lat-Perc, mavg	**94.37**	80	2 hr
Lat-Perc, avg	94.13	70	2 hr
Lat-Perc	92.06	25	0.2 hr
DPLVM-BFGS	94.29	300	≈120 hr
DPLVM-SGD	94.03	100	≈20 hr
Perc, mavg	94.25	65	<1 hr
Perc, avg	94.27	68	<1 hr
Perc	91.33	12	0.1 hr
CRF-BFGS	94.25	280	≈30 hr

Table 4: Comparisons on the NP-chunking task. All the models set cc=0 and odr=2, i.e., all features are included and the 2nd order dependency is used.

As can be seen in Table 4, the latent perceptron with *modified averaged parameter* training (*mavg*) achieves the best performance, which is significantly better than the same model trained by the traditional parameter averaging (*avg*). Compared with the *avg*, we also observed that the *mavg* training (on the latent perceptron) reached an empirically more stable convergent plateau. For example, further training with the *mavg* for 200 iterations following the beginning of the plateau, the fluctuation of its F-measure was bounded by $\pm 0.1\%$, while for the *avg* training, it reached $\pm 0.8\%$. The latent perceptron with *mavg* training also outperformed the DPLVM model with stochastic training. As for the perceptron, we did not observe significant difference between the training of *mavg* and *avg*. In this simple task, based on a 2nd-order dependency and a rich feature set, we observed that the superiority of the latent perceptron over other models is not very significant.

Finally, with the F-measure of 94.37%, the performance of the latent perceptron is one of the best reports on this data. The state-of-the-art systems include Sha & Pereira [2003] (94.38% by a 2nd order CRF model) and Kudo & Matsumoto [2001] (94.22% by the combination of multiple SVM models).

6 Conclusions

We proposed the latent variable perceptron algorithm to learn latent dependencies, and analyzed its convergence properties. We showed that simply using averaged parameter training on latent perceptron is not good enough. To address this problem, we proposed the modified averaged parameter training. Compared to existing probabilistic models of latent variables, our method lowers the training cost significantly yet with comparable or even superior classification accuracy. Although we only 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 structure.

Acknowledgments

We thank Xia Zhou and Galen Andrew for good suggestions on improving the paper-writing. We also thank the anonymous reviewers who gave very helpful comments.

Appendix: Proof of Convergence[7]

Proof of theorem 1. Given the *latent feature mapping* $\mathbf{m} = (m_1, \ldots, m_n)$ and $\mathbf{U} \in \mathbb{R}^n$ represented by $(\alpha_1, \ldots, \alpha_n)$, there exist a $\overline{\mathbf{U}} \in \mathbb{R}^N$ with $N = \sum_{i=1}^n m_i$ represented as follows: $(\overbrace{\alpha_1, \ldots, \alpha_1}^{m_1}, \overbrace{\alpha_2, \ldots, \alpha_2}^{m_2}, \ldots, \overbrace{\alpha_n, \ldots, \alpha_n}^{m_n})$. Then $\forall i, \forall \mathbf{h} \in \mathbf{GH}^*(\mathbf{x}_i), \forall \mathbf{z} \in \overline{\mathbf{GH}}(\mathbf{x}_i)$,

$$
\begin{aligned}
&\overline{\mathbf{U}} \cdot \mathbf{f}(\mathbf{h}, \mathbf{x}_i) - \overline{\mathbf{U}} \cdot \mathbf{f}(\mathbf{z}, \mathbf{x}_i) \\
&= \overline{\mathbf{U}}[\mathbf{f}(\mathbf{h}, \mathbf{x}_i) - \mathbf{f}(\mathbf{z}, \mathbf{x}_i)] \\
&= \sum_{j=1}^{n} [\alpha_j \sum_{k=1}^{m_j} (\beta_j^{k*} - \beta_j^k)] \\
&= \sum_{j=1}^{n} \alpha_j (\sum_{k=1}^{m_j} \beta_j^{k*} - \sum_{k=1}^{m_j} \beta_j^k) \\
&= \sum_{j=1}^{n} \alpha_j (\beta_j^* - \beta_j) \\
&= \mathbf{U}[\mathbf{f}(\mathbf{y}_i^*, \mathbf{x}_i) - \mathbf{f}(\mathbf{y}^{\mathbf{z}}, \mathbf{x}_i)] \\
&\geq \delta,
\end{aligned}
\tag{15}
$$

where $\beta_j^{k*} \in \mathbf{f}(\mathbf{h}, \mathbf{x}_i)$ is the k'th feature value mapped from $\beta_j^* \in \mathbf{f}(\mathbf{y}_i^*, \mathbf{x}_i)$ by using latent variables; similarly, $\beta_j^k \in \mathbf{f}(\mathbf{z}, \mathbf{x}_i)$ is mapped from $\beta_j \in \mathbf{f}(\mathbf{y}^{\mathbf{z}}, \mathbf{x}_i)$; $\mathbf{y}^{\mathbf{z}}$ is the label sequence projected from the latent sequence $\mathbf{z}$.

Let $T = ||\overline{\mathbf{U}}|| = (\sum_{i=1}^n m_i {\alpha_i}^2)^{1/2}$, then it can be seen that the vector $\overline{\mathbf{U}}/||\overline{\mathbf{U}}||$ latently separates the data with margin δ/T. □

Proof of theorem 2. Let $\mathbf{\Theta}^k$ be the parameters before the k'th mistake is made, and suppose it is made at the i'th example. Let $\mathbf{z}$ be the output proposed at this example, $\mathbf{z} = \mathbf{argmax}_{\mathbf{h} \in \mathbf{GH}(\mathbf{x}_i)} \mathbf{f}(\mathbf{h}, \mathbf{x}_i) \cdot \mathbf{\Theta}^k$. Also, let $\mathbf{h}_i^* = \mathbf{argmax}_{\mathbf{h} \in \mathbf{GH}^*(\mathbf{x}_i)} \mathbf{f}(\mathbf{h}, \mathbf{x}_i) \cdot \mathbf{\Theta}^k$. Then, by using *theorem 1* and adapting Collins [2002] for the lower bound of $||\mathbf{\Theta}^{k+1}||$, we can derive that

$$
||\mathbf{\Theta}^{k+1}|| \geq k\delta/T. \tag{16}
$$

We also derive an upper bound for $||\mathbf{\Theta}^{k+1}||^2$. First, we have

[7] For a more detailed version of the proof, see Sun [2009].

$$||\mathbf{\Theta}^{k+1}||^2 = ||\mathbf{\Theta}^k||^2 + ||\mathbf{f}(\mathbf{h}_i^*, \mathbf{x}_i) - \mathbf{f}(\mathbf{z}, \mathbf{x}_i)||^2 + 2 \cdot \mathbf{\Theta}^k \cdot [\mathbf{f}(\mathbf{h}_i^*, \mathbf{x}_i) - \mathbf{f}(\mathbf{z}, \mathbf{x}_i)]. \quad (17)$$

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

$$\begin{aligned}
||\mathbf{\Theta}^{k+1}||^2 &\leq ||\mathbf{\Theta}^k||^2 + ||\mathbf{f}(\mathbf{h}_i^*, \mathbf{x}_i) - \mathbf{f}(\mathbf{z}, \mathbf{x}_i)||^2 \\
&\leq ||\mathbf{\Theta}^k||^2 + ||\mathbf{f}(\mathbf{h}_i^*, \mathbf{x}_i)||^2 + ||\mathbf{f}(\mathbf{z}, \mathbf{x}_i)||^2 \\
&= ||\mathbf{\Theta}^k||^2 + \sum_{j=1}^{n}\sum_{k=1}^{m_j}(\beta_j^{k*})^2 + \sum_{j=1}^{n}\sum_{k=1}^{m_j}(\beta_j^k)^2 \\
&\leq ||\mathbf{\Theta}^k||^2 + \sum_{j=1}^{n}(\sum_{k=1}^{m_j}\beta_j^{k*})^2 + \sum_{j=1}^{n}(\sum_{k=1}^{m_j}\beta_j^k)^2 \quad (18) \\
&= ||\mathbf{\Theta}^k||^2 + \sum_{j=1}^{n}(\beta_j^*)^2 + \sum_{j=1}^{n}(\beta_j)^2 \\
&= ||\mathbf{\Theta}^k||^2 + ||\mathbf{f}(\mathbf{y}_i^*, \mathbf{x}_i)||^2 + ||\mathbf{f}(\mathbf{y}^{\mathbf{z}}, \mathbf{x}_i)||^2 \\
&\leq ||\mathbf{\Theta}^k||^2 + 2M^2.
\end{aligned}$$

Then, it follows that $||\mathbf{\Theta}^{k+1}||^2 \leq 2kM^2$.

Combining the upper bound and lower bound completes the proof. □

References

[Bottou, 1998] Léon Bottou. Online algorithms and stochastic approximations. *Online Learning and Neural Networks. Saad, David. Cambridge University Press*, 1998.

[Collins, 2002] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. *Proc. of EMNLP'02*, 2002.

[Finkel *et al.*, 2004] Jenny Finkel, Shipra Dingare, Huy Nguyen, Malvina Nissim, Christopher Manning, and Gail Sinclair. Exploiting context for biomedical entity recognition: From syntax to the web. *Proc. of JNLPBA'04*, 2004.

[Freund and Schapire, 1999] Yoav Freund and Robert Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

[Kim *et al.*, 2004] Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, and Yuka Tateisi. Introduction to the bio-entity recognition task at JNLPBA. *Proceedings of JNLPBA'04*, pages 70–75, 2004.

[Kudo and Matsumoto, 2001] Taku Kudo and Yuji Matsumoto. Chunking with support vector machines. *Proc. of NAACL'01*, 2001.

[LeCun *et al.*, 2006] Yann LeCun, Sumit Chopra, Raia Hadsell, Ranzato Marc'Aurelio, and Fu-Jie Huang. A tutorial on energy-based learning. *Predicting Structured Data. MIT Press*, 2006.

[Morency *et al.*, 2007] Louis-Philippe Morency, Ariadna Quattoni, and Trevor Darrell. Latent-dynamic discriminative models for continuous gesture recognition. *Proceedings of CVPR'07*, pages 1–8, 2007.

[Nocedal and Wright, 1999] Jorge Nocedal and Stephen J. Wright. Numerical optimization. *Springer*, 1999.

[Okanohara *et al.*, 2006] Daisuke Okanohara, Yusuke Miyao, Yoshimasa Tsuruoka, and Jun'chi Tsujii. Improving the scalability of semi-markov conditional random fields for named entity recognition. *Proc. of ACL'06*, 2006.

[Petrov and Klein, 2008] Slav Petrov and Dan Klein. Discriminative log-linear grammars with latent variables. *Proc. of NIPS'08*, 2008.

[Rosenblatt, 1958] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[Sang and Buchholz, 2000] E. T. K. Sang and S. Buchholz. Introduction to the CoNLL-2000 shared task: Chunking. *Proc. of CoNLL'00*, pages 127–132, 2000.

[Settles, 2004] Burr Settles. Biomedical named entity recognition using conditional random fields and rich feature sets. *Proc. of JNLPBA'04*, 2004.

[Sha and Pereira, 2003] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. *Proc. of HLT/NAACL'03*, 2003.

[Sun *et al.*, 2008] Xu Sun, Louis-Philippe Morency, Daisuke Okanohara, and Jun'ichi Tsujii. Modeling latent-dynamic in shallow parsing: A latent conditional model with improved inference. *Proceedings of the 22nd International Conference on Computational Linguistics (COLING'08)*, pages 841–848, 2008.

[Sun, 2009] Xu Sun. Latent variable perceptron algorithm: Proof of convergence. *Technical Report (TR-ML-2009-3), Tsujii Laboratory, University of Tokyo*, 2009.