Large-Scale Hierarchical Classification via Stochastic Perceptron

Dehua Liu and **Bojun Tu** and **Hui Qian**[*] and **Zhihua Zhang**
College of Computer Science and Technology
Zhejiang University, Hangzhou 310007, China
{dehua, tubojun, qianhui, zhzhang}@zju.edu.cn

Abstract

Hierarchical classification (HC) plays an significant role in machine learning and data mining. However, most of the state-of-the-art HC algorithms suffer from high computational costs. To improve the performance of solving, we propose a stochastic perceptron (SP) algorithm in the large margin framework. In particular, a stochastic choice procedure is devised to decide the direction of next iteration. We prove that after finite iterations the SP algorithm yields a sub-optimal solution with high probability if the input instances are separable. For large-scale and high-dimensional data sets, we reform SP to the kernel version (KSP), which dramatically reduces the memory space needed. The KSP algorithm has the merit of low space complexity as well as low time complexity. The experimental results show that our KSP approach achieves almost the same accuracy as the contemporary algorithms on the real-world data sets, but with much less CPU running time.

1 Introduction

In the practical application, we often meet hierarchical classification problems where outputs are structured. For example, in document classification at the web site, it is often preferable to categorize a given text document into hierarchical classes. By convention, the collection of text document is organized as a tree; that is, the output is encoded as a tree and the hierarchical structure is prespecified.

Existing flat classification approaches, which predict only classes at leaf nodes, are not able to capture the mutual relationship between the nodes of the output tree in question, so they can not be directly applied to the hierarchical classification problem. Thus, it is inherently challenging to solve the hierarchical classification problem.

Typically, there are two approaches for handling this problem. The first one is a local approach (Koller and Sahami 1997; Silla and Freitas 2010). The key idea is to construct local classifiers from the top of tree to the bottom. For example, one constructs classifiers at each level of the category tree by invoking multi-classification algorithms (Clare and King 2003), or constructs classifier at each node or each parent node (Dumais and Chen 2000; Wu, Zhang, and Honavar 2005; Cesa-Bianchi, Gentile, and Zaniboni 2006; Zhou, Xiao, and Wu 2011). Dłez, del Coz, and Bahamonde(2010) devised a bottom-up learning strategy and each node classifier is built by taking into account other node classifiers.

The other is a global approach which builds a single classification model based on the training set, considering the class hierarchy as a whole. Compared with the local approach, the total size of the global classification model is typically considerably smaller. Additionally, the interdependence between different classes can be taken into account in a straightforward way by the global classifier. The global approach includes large margin methods (Taskar, Guestrin, and Koller 2003; Cai and Hofmann 2004; Dekel, Keshet, and Singer 2004; Taskar, Chatalbashev, and Guestrin 2005; Tsochantaridis et al. 2005; Rousu et al. 2006; Sarawagi and Gupta 2008), conditional random fields (Lafferty, McCallum, and Pereira 2001), Bayesian models (Gyftodimos and Flach 2003; Barutcuoglu, Schapire, and Troyanskaya 2006; Gopal et al. 2012), etc.

The large margin methods and conditional random fields can be formed as a SVM optimization, transforming it to the dual form and invoking quadratic programming (QP) routine. There are totally $n(|\mathcal{Y}|-1)$ variables in QP where n is the size of the training data set and $|\mathcal{Y}|$ is the total number of vectors in output space $\mathcal{Y}$. In many cases, both n and $|\mathcal{Y}|$ may be extremely large. Thus, these algorithms are time costly, even after the number of variables is reduced by some techniques.

In this paper we introduce perceptron algorithm to develop a global approach for the HC problem. The perceptron can be viewed as a routine to find a feasible point of a set of linear inequalities (Blum and Dunagan 2002). Here we can also regard it as a procedure to obtain a relaxed maximum of margin (refer to section 3.3 for details), leading to a promising way to handle the large margin model. The perceptron or stochastic gradient descent algorithms which are similar to ours have been widely used in the domain of binary, multi-class task or structured prediction (Freund and Schapire 1999; Collins 2002; Dekel, Keshet, and Singer 2004; Crammer et al. 2006; Ratliff, Bagnell, and Zinkevich 2007; Shalev-Shwartz, Singer, and Srebro 2007; Wang, Crammer, and Vucetic 2010). All these algorithms

[*] Corresponding author.

are especially suitable for online setting in which weights updating direction are determined by current coming instance. They are also guaranteed by the theoretical upper bounds on the corresponding loss functions or number of predictive mistakes.

In order to reveal the close relation to the large margin model, we design different perceptron algorithms in which the next updating direction is determined by the whole set of samples. Thus it is suitable for batch learning. We first devise a perceptron algorithm in which the weight vector is updated by adding the most violate feature gap (see Algorithm 1). However, Algorithm 1 is not our main concern, and it is used to introduce Algorithm 2, the stochastic version. In Algorithm 2, multiplicative weights (MW) method (Littlestone 1987) is used to construct a stochastic procedure to decide the next updating direction, which largely improves the prediction accuracy compared to Algorithm 1. Additionally, in order to consider the application in high dimensional data, we suggest a kernel stochastic perceptron algorithm (Algorithm 3), which can significantly save the memory space. This kernel algorithm has the advantage of much lower computational complexity in comparison with the current state-of-the-art algorithms, i.e., $O(n(dx+dy)\log(n)/\epsilon^2)$ with dx the dimension of features $\mathbf{x}$ and dy the dimension of output $\mathbf{y}$.

The paper is organized as follows: In Section 2, we formulate the problem more concretely and review the popular large margin models closely related to our approach. In Section 3, we propose a so-called hard perceptron algorithm for HC, and then derive a stochastic perceptron algorithm, also extend to the kernel version. We also analyze their convergence and computational complexity in this section. In Section 4, we use Algorithm 3 to conduct the numerical experiments, showing that our algorithm achieves nearly the same accuracy as state-of-the-art algorithms. We also report the CPU times on different benchmark data sets and the performance of different kernels used in the algorithm.

2 Problem Formulation

We are given a training data set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ where $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^{dx}$ is the input vector and $\mathbf{y}_i \in \mathcal{Y} \subset \{1, -1\}^{dy}$ is the corresponding output, the output is hierarchically organized. In particular, $\mathbf{y}$ is encoded as a tree, each node of the tree corresponds to a component of $\mathbf{y}$, see Figure 1 for an illustration. We set the component to be 1 if the category (represented as paths in the tree) of $\mathbf{x}$ contains the corresponding node, and -1 otherwise. The current purpose is to learn a map $\mathbf{y} = h(\mathbf{x})$ from the data set $\mathcal{D}$.

Suppose that the learning map $h(\mathbf{x})$ is of the following parametric form:

$$h_{\mathbf{w}}(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{w}, \mathbf{x}, \mathbf{y}),$$

where $f(\mathbf{w}, \mathbf{x}, \mathbf{y})$ is a function $f : \mathcal{W}\times\mathcal{X}\times\mathcal{Y} \longmapsto \mathbb{R}$ and $\mathcal{W}$ is the parameter space. f is called a scoring function measuring the confidence of output $\mathbf{y}$ given the parameter $\mathbf{w}$ and input $\mathbf{x}$. Simply, we consider f belonging to a linear family, i.e., $f(\mathbf{w}, \mathbf{x}, \mathbf{y}) = \mathbf{w}^\top\Phi(\mathbf{x}, \mathbf{y})$. We refer to $\Phi(\mathbf{x}, \mathbf{y})$ as the joint feature map of input and output. In the binary case where $y \in \{1, -1\}$, $\Phi(\mathbf{x}, y)$ is usually defined as $y\phi(\mathbf{x})$.

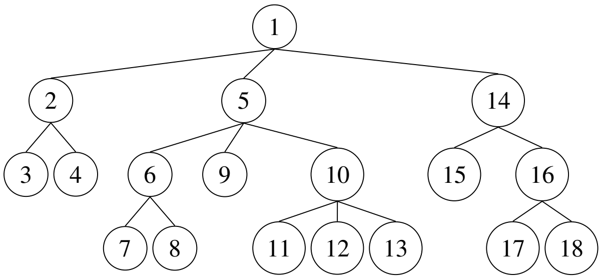

Figure 1: Structure of the output vector

However, when the output is complicated, the definition of $\Phi(\mathbf{x}, \mathbf{y})$ is not straightforward. We here follow the setting of Cai and Hofmann (2004), who defined $\Phi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) \otimes \psi(\mathbf{y})$. with $\phi(\mathbf{x}) \in \mathbb{R}^d$ and $\psi(\mathbf{y}) \in \mathbb{R}^k$. precisely,

$$[\Phi(\mathbf{x}, \mathbf{y})]_{i+(j-1)d} = [\phi(\mathbf{x})]_i[\psi(\mathbf{y})]_j.$$

Furthermore, $\Phi(\mathbf{x}, \mathbf{y})$ (or $\phi(\mathbf{x})$ and $\psi(\mathbf{y})$) is not necessarily explicitly available. In this case, we resort to a kernel trick (Shawe-Taylor and Cristianini 2004). Particularly,

$$\begin{aligned} K((\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2)) &\triangleq \langle\Phi(\mathbf{x}_1, \mathbf{y}_1), \Phi(\mathbf{x}_2, \mathbf{y}_2)\rangle \\ &= \langle\phi(\mathbf{x}_1), \phi(\mathbf{x}_2)\rangle\langle\psi(\mathbf{y}_1), \psi(\mathbf{y}_2)\rangle \\ &= K_I(\mathbf{x}_1, \mathbf{x}_2)K_O(\mathbf{y}_1, \mathbf{y}_2), \end{aligned}$$

where $K_I(\cdot,\cdot)$ and $K_O(\cdot,\cdot)$ are the reproducing kernel functions defined on the input space $\mathcal{X} \times \mathcal{X}$ and output space $\mathcal{Y} \times \mathcal{Y}$, respectively.

Letting $\Delta\Phi(\mathbf{x}, \mathbf{z}) = \Phi(\mathbf{x}, \mathbf{y}) - \Phi(\mathbf{x}, \mathbf{z})$ denote feature gap, where $\mathbf{y}$ is the corresponding output of $\mathbf{x}$, we give the definition of separability of data as follows.

Definition 1 *The data set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ is said to be separable, if*

$$\max_{\mathbf{w}\in B} \min_{i,\mathbf{z}\neq\mathbf{y}_i} \mathbf{w}^\top\Delta\Phi(\mathbf{x}_i, \mathbf{z}) = \delta > 0$$

where $B = \{\mathbf{w}; \|\mathbf{w}\|_2 \leq 1\}$.

Additionally, $\min_{i,\mathbf{z}\neq\mathbf{y}_i} \mathbf{w}^\top\Delta\Phi(\mathbf{x}_i, \mathbf{z})$ is called the margin of the training data set $\mathcal{D}$. Suppose $\mathbf{w}_* \in B$ satisfies $\min_{i,\mathbf{z}\neq\mathbf{y}_i} \mathbf{w}_*^\top\Delta\Phi(\mathbf{x}_i, \mathbf{z}) = \delta$. Our task is to learn the vector $\mathbf{w}_*$ based on the training data set.

2.1 The Large Margin Method

The most popular works related to ours are categorized to large margin framework which has been studied in (Taskar, Guestrin, and Koller 2003; Cai and Hofmann 2004; Dekel, Keshet, and Singer 2004; Taskar, Chatalbashev, and Guestrin 2005; Tsochantaridis et al. 2005; Rousu et al. 2006; Sarawagi and Gupta 2008). The large margin method for the HC problem (LMM-HC) is based on the following optimization problem

$$\tfrac{1}{2}\|\mathbf{w}\|_2^2 \quad (1)$$

$$\text{s.t.}\mathbf{w}^\top\Delta\Phi(\mathbf{x}_i, \mathbf{z}) \geq \gamma,\ \forall 1 \leq i \leq n, \forall\mathbf{z} \in \mathcal{Y}, \mathbf{z} \neq \mathbf{y}_i. \quad (2)$$

LMM-HC can be further extended by adding slack variables to deal with non-separable case, or replacing γ with a more general loss function. Here we do not give the details.

Unlike the conventional SVM for the binary classification problem, the number of constraints in LMM-HC grows exponentially with respect to the dimension of output vector, and the dual QP problem has exponentially corresponding variables. Thus, a lot of work has been focused on how to reduce the number of variables to polynomial size. For example, Tsochantaridis et al. (2005) devised an algorithm that creates a nested sequence of successively tighter relaxation of the original problem using a cutting plane method. Then the active set of constraints reduced to polynomial size. Rousu et al. (2006) devised an efficient algorithm by marginalizing the dual problem.

However, due to the high computational complexity of QP, it is still infeasible to solve the large scale hierarchical problem. In fact, the algorithm of Tsochantaridis et al. (2005) has complexity $O(K(n+m)^3+Kn(dx+dy))$ where m is the largest size of active sets and K is the number of iteration. The algorithm in (Rousu et al. 2006) have complexity $O(|E|^3n^3)$ where $|E|$ represents the number of edges in the tree.

This encourages us to develop a more efficient algorithm to solve the HC problem. In the next section we propose perceptron algorithms, without being troubled by the exponential size of constraints.

3 Perceptron Algorithms for HC

Note that the constraints (2) can be written compactly as $\min_{i,\mathbf{z}\neq\mathbf{y}_i} \mathbf{w}^\top\Delta\Phi(\mathbf{x}_i,\mathbf{z}) \geq \gamma$. The optimization problem (1,2) is equivalent to

$$\max_{\mathbf{w}\in B} \min_{i,\mathbf{z}\neq\mathbf{y}_i} \mathbf{w}^\top\Delta\Phi(\mathbf{x}_i,\mathbf{z}) \tag{3}$$

The idea is similar to the work of binary classification given by Clarkson, Hazan, and Woodruff(2010). However, it is hard to rewrite the min step of (3) as minimization over a simplex as in the case of binary classification.

3.1 Hard Perceptron Algorithm

We first devise a so-called *hard perceptron* (HP) to the HC problem, which is given in Algorithm 1. The weight $\mathbf{w}$ is updated by adding the most violate feature gap (its index i_t minimize $\mathbf{w}_t^\top\Delta\Phi(\mathbf{x}_i,\mathbf{z}_i)$).

This algorithm can be viewed as an alternative process to obtain the minimax of the function $\mathbf{w}^\top\Delta\Phi(\mathbf{x}_i,\mathbf{z})$. At iteration t, we fix $\mathbf{w}_t$ to optimize w.r.t. (i, $\mathbf{z}$) with the constraint $\mathbf{z}\neq\mathbf{y}_i$, then we fix $(i_t,\mathbf{z}_{i_t})$ to optimize w.r.t. $\mathbf{w}\in B$.

Note that the perceptron algorithm does not guarantee to increase the margin after every iteration. However, after finite steps, it is proved to obtain the sub-optimal solution (see Theorem 1). In fact, we will see that Theorem 1 reveals a relationship between the maximum margin and perceptron algorithm.

3.2 The Stochastic Perceptron Algorithm

HP algorithm is deterministic. The main problem that HP algorithm encounter is that the noisy data may lead to wrong updating direction since at every iteration HP chooses the most violate feature gap under the current estimate w_t. It reslut in that the performance of HP is a little less assuming than the state-of-the-art algorithm.

Algorithm 1 Hard Perceptron for HC (HP)

```
Initialization $\mathbf{w}_1$;
for $t = 1$ to $T$ do
    for $i = 1$ to $n$ do
        $\mathbf{z}_i = \arg\min_{\mathbf{z}\neq\mathbf{y}_i} \mathbf{w}_t^\top\Delta\Phi(\mathbf{x}_i,\mathbf{z})$;
        denote $\mathbf{v}_t(i) = \mathbf{w}_t^\top\Delta\Phi(\mathbf{x}_i,\mathbf{z}_i)$;
    end for
    $i_t = \arg\min_i \mathbf{v}_t(i)$;
    return $\mathbf{w}_t$, if $\mathbf{v}_t(i_t) > 0$;
    $\bar{\mathbf{w}}_{t+1} = \bar{\mathbf{w}}_t + \frac{1}{\sqrt{T}}\Delta\Phi(\mathbf{x}_{i_t},\mathbf{z}_{i_t})$;
    $\mathbf{w}_{t+1} = \frac{\bar{\mathbf{w}}_{t+1}}{\max(1,\|\bar{\mathbf{w}}_{t+1}\|_2)}$;
end for
return $\bar{\mathbf{w}} = \frac{1}{T}\sum_t \mathbf{w}_t$.
For given input $\mathbf{x}$, the output is
$h(\mathbf{x}) = \arg\max_{\mathbf{z}} \bar{\mathbf{w}}^\top\Phi(\mathbf{x},\mathbf{z})$.
```

Algorithm 2 Stochastic Perceptron Algorithm for HC

```
Initialization $\bar{\mathbf{w}}_1, \mathbf{u}_1 = \mathbf{1}_n$;
for $t = 1$ to $T$ do
    $\mathbf{p}_t = \frac{\mathbf{u}_t}{\|\mathbf{u}_t\|_1}$;
    $\mathbf{w}_t = \frac{\bar{\mathbf{w}}_t}{\max(1,\|\bar{\mathbf{w}}_t\|_2)}$;
    for $i = 1$ to $n$ do
        $\mathbf{z}_i = \arg\min_{\mathbf{z}\neq\mathbf{y}_i} \mathbf{w}_t^\top\Delta\Phi(\mathbf{x}_i,\mathbf{z})$;
        $\mathbf{v}_t(i) = \mathbf{w}_t^\top\Delta\Phi(\mathbf{x}_i,\mathbf{z}_i)$;
        $\mathbf{u}_{t+1}(i) = \mathbf{u}_t(i)(1-\eta\mathbf{v}_t(i)+\eta^2\mathbf{v}_t(i)^2)$;
    end for
    sample $i_t$ from distribution $\mathbf{p}_t$;
    $\bar{\mathbf{w}}_{t+1} = \bar{\mathbf{w}}_t + \frac{1}{\sqrt{T}}\Delta\Phi(\mathbf{x}_{i_t},\mathbf{z}_{i_t})$;
end for
return $\bar{\mathbf{w}} = \frac{1}{T}\sum_t \mathbf{w}_t$.
For given input $\mathbf{x}$, the output is
$h(\mathbf{x}) = \arg\max_{\mathbf{z}} \bar{\mathbf{w}}^\top\Phi(\mathbf{x},\mathbf{z})$.
```

Therefore, we employ a stochastic procedure to chose i_t from the set $\{1,...,n\}$. The algorithm is based on the multiplicative weights (MW) method (Littlestone 1987; Clarkson, Hazan, and Woodruff 2010).

Definition 2 *(MW algorithm). Let $\mathbf{v}_1,...,\mathbf{v}_T \in \mathbb{R}^n$ be a sequence of vectors, the Multiplicative Weights (MW) algorithm is the following: Start from $\mathbf{u}_1 = \mathbf{1}_n$, and for $t \geq 1, \eta > 0$,*

$$\mathbf{p}_t = \mathbf{u}_t/\|\mathbf{u}_t\|_1$$
$$\mathbf{u}_{t+1}(i) = \mathbf{u}_t(i)(1-\eta\mathbf{v}_t(i)+\eta^2\mathbf{v}_t(i)^2).$$

where $\mathbf{u}_t(i)$ is the i-th component of vector $\mathbf{u}$.

Based on the MW algorithm, we summarize our *stochastic perceptron* (SP) in Algorithm 2. Although the algorithm is only guaranteed to be convergent for separable data sets, we find in the experiments that it also performs well on real-world data sets which are non-separable.

3.3 Convergence Analysis

In this section we analyze the convergence property of both Algorithms 1 and 2.

The following lemma is due to Zinkevich(2003), a building block of our results.

Lemma 1 *Consider a sequence of vectors* $\phi_1, ..., \phi_T \in \mathbb{R}^d$ *satisfying* $\|\phi_i\|_2 \leq R$. *Suppose* $\mathbf{w}_1 = \mathbf{0}$, *and* $\bar{\mathbf{w}}_{t+1} = \bar{\mathbf{w}}_t + \frac{1}{\sqrt{T}}\phi_t$, $\mathbf{w}_{t+1} = \frac{\bar{\mathbf{w}}_{t+1}}{\max(1, \|\bar{\mathbf{w}}_{t+1}\|_2)}$. *Then*

$$\max_{\mathbf{w} \in B} \sum_{t=1}^{T} \mathbf{w}^\top \phi_t - \sum_{t=1}^{T} \mathbf{w}_t^\top \phi_t \leq 2R\sqrt{T}.$$

Based on the above lemma, we have the following convergence theorem of Algorithm 1.

Theorem 1 *Suppose* $\|\Phi(\mathbf{x}, \mathbf{y})\|_2 < R$ *and set the iteration number* $T \geq (2R/\epsilon)^2$. *If Algorithm 1 stops at iteration* $t < T$, *the returned vector* $\mathbf{w}_t$ *successfully separates the sample dataset* $\mathcal{D}$, *i.e.,* $\min_i\{\mathbf{w}_t^\top \Delta\Phi(\mathbf{x}_i, \mathbf{z}_i)\} > 0$. *If the algorithm stops at* $t = T$, *it returns* $\bar{\mathbf{w}} = \frac{1}{T}\sum_t \mathbf{w}_t$ *which is an* ϵ *sub-optimal solution, i.e.,*

$$\min_{i, \mathbf{z} \neq \mathbf{y}_i} \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}_t^\top \Delta\Phi(\mathbf{x}_i, \mathbf{z}) \geq \delta - \epsilon.$$

Proof: If Algorithm 1 stops at $t < T$, this implies $\mathbf{v}_t(i_t) > 0$, i.e., $\mathbf{v}_t(i) = \mathbf{w}_t^\top \Delta\Phi(\mathbf{x}_i, \mathbf{z}_i) > 0 \ \forall i \in \{1, ..., n\}$. If the algorithm stops at $t = T$, we have

$$\begin{aligned} & \min_{i, \mathbf{z} \neq \mathbf{y}_i} \sum_{t=1}^{T} \mathbf{w}_t^\top \Delta\Phi(\mathbf{x}_i, \mathbf{z}) \geq \sum_t \mathbf{w}_t^\top \Delta\Phi(\mathbf{x}_{i_t}, \mathbf{z}_{i_t}) \\ & \geq \max_{\mathbf{w}: \|\mathbf{w}\|_2 \leq 1} \sum_t \mathbf{w}^\top \Delta\Phi(\mathbf{x}_{i_t}, \mathbf{z}_{i_t}) - 2R\sqrt{T} \\ & \geq T\delta - 2R\sqrt{T} \\ & \geq T(\delta - \epsilon) \end{aligned}$$

The first inequality is implied from Algorithm 1, the second comes from Lemma 1, and the third from Definition 1. ■

To capture the difference between expectation of $\mathbf{v}_t(i_t)$ and the minimal of $\mathbf{v}_t(i)$ among $i \in \{1, ..., n\}$, we introduce the following lemma.

Lemma 2 *The MW algorithm satisfies*

$$\begin{aligned} \sum_t \mathbf{p}_t^\top \mathbf{v}_t \quad \leq \quad & \min_{1 \leq i \leq n} \sum_{t=1}^{T} \max(-1/\eta, \mathbf{v}_t(i)) \\ & + \frac{\log n}{\eta} + \eta \sum_{t=1}^{T} \mathbf{p}_t^\top \mathbf{v}_t^2, \end{aligned}$$

where $\mathbf{v}_t^2 = (\mathbf{v}_t(1)^2, \ldots, \mathbf{v}_t(n)^2)^\top$.

The next lemma states the difference between $\mathbf{w}_t^\top \Delta\Phi(\mathbf{x}_{i_t}, \mathbf{z}_{i_t})$ and its expectation.

Lemma 3 *If* η *and* T *satisfy* $9\eta^2 T - 8(\eta + 4)\log n > 0$, *then with probability at least* $1 - O(1/n)$, *we have*

$$|\sum_t \mathbf{w}_t^\top \Delta\Phi(\mathbf{x}_{i_t}, \mathbf{z}_{i_t}) - \sum_t \mathbf{p}_t^\top \mathbf{v}_t| \leq 3R\eta T$$

Finally, we present the theoretical guarantee for Algorithm 2.

Theorem 2 *Suppose* $\|\Phi(\mathbf{x}, \mathbf{y})\|_2 < R$, *and set the iteration number* $T \geq \left[\frac{(8R^2+6R+1)\sqrt{\log n}+2R}{\epsilon}\right]^2 = O((\log n)/\epsilon^2)$, *and* $\eta = \min\{c\sqrt{\frac{\log n}{T}}, \frac{1}{2R}\}$. *Then with probability 1-*$O(\frac{1}{n})$ *Algorithm 2 returns* $\bar{\mathbf{w}} = \frac{1}{T}\sum_t \mathbf{w}_t$ *which is an* ϵ *optimal solution, i.e.,*

$$\min_{i, \mathbf{z} \neq \mathbf{y}_i} \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}^\top \Delta\Phi(\mathbf{x}_i, \mathbf{z}) \geq \delta - \epsilon.$$

Proof: Consider that

$$\begin{aligned} & \min_{i, \mathbf{z} \neq \mathbf{y}_i} \sum_t \mathbf{w}_t^\top \Delta\Phi(\mathbf{x}_i, \mathbf{z}) \\ \geq \quad & \min_i \sum_t \min_{\mathbf{z} \neq \mathbf{y}_i} \mathbf{w}_t^\top \Delta\Phi(\mathbf{x}_i, \mathbf{z}) \\ = \quad & \min_i \sum_t \mathbf{v}_t(i) \geq \sum_t \mathbf{p}_t^\top \mathbf{v}_t - \frac{\log n}{\eta} - \eta \mathbf{p}_t^\top \mathbf{v}_t^2 \\ \geq \quad & \sum_t \mathbf{w}_t^\top \Delta\Phi(\mathbf{x}_{i_t}, \mathbf{z}_{i_t}) - 3R\eta T - \frac{\log n}{\eta} 4R^2\eta T \\ \geq \quad & T\delta - 2R\sqrt{T} - 3R\eta T - 4R^2\eta T - \frac{\log n}{\eta} \\ \geq \quad & T(\delta - \epsilon). \end{aligned}$$

The third line is implied by Algorithm 2, the fourth line comes from Lemma 2, the fifth line from Lemma 3 and the fact $\sum_t \mathbf{p}_t^\top \mathbf{v}_t^2 \leq 4R^2 T$, and the sixth line is due to Definition 1. ■

Keep in mind that $\min_{i, \mathbf{z} \neq \mathbf{y}_i} \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}_t^\top \Delta\Phi(\mathbf{x}_i, \mathbf{z}) \leq \delta$. Theorem 1 and 2 suggest that after finite iterations, HP and SP algorithms are able to achieve nearly maximum of the margin (with a small ϵ gap).

3.4 Kernel Stochastic Perceptron

Recall that in SP algorithm the dimension of both $\mathbf{w}$ and $\Phi(\mathbf{x}, \mathbf{y})$ is $dim(\phi(\mathbf{x})) \times dim(\psi(\mathbf{y}))$, which may be very high in practice and beyond the capacity of memory storage of conventional computer architecture. This problem can be avoid by kernelize the SP. Thus, we devise a kernel version of SP, which is given in Algorithm 3. We call the resulting procedure a *kernel stochastic perceptron* (KSP).

The KSP algorithm not only holds nonlinear modeling ability, but also reduces the space complexity. Typically, we assume the kernel on space $\mathcal{Y} \times \mathcal{Y}$ is linear, i.e., $K_O(\mathbf{y}, \mathbf{z}) = \mathbf{y}^\top \mathbf{z}$.

Theorem 3 *The KSP algorithm takes iteration* $T = O(\frac{\log n}{\epsilon^2})$ *and returns* ϵ *sub-optimal solution with probability* $1 - O(\frac{1}{n})$, *with total running time* $O(n(dx + dy)(\log n)/\epsilon^2)$.

3.5 Maximizing Linear Function on Output Space

In Algorithms 1, 2 and 3, we need to solve $\mathbf{z} = \text{argmax}_{\mathbf{u}} \, \mathbf{a}^\top \mathbf{u}$ in the prediction stage. Since the output vector $\mathbf{u} \in \mathcal{Y}$ is represented as a tree structure, every component of $\mathbf{u}$ corresponds to a node in the tree. Thus, we can

Algorithm 3 Kernel Stochastic Perceptron Algorithm (KSP) for HC

```
Input $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, $dy$(the dimension of $\mathbf{y}$);
Initialization $\mathbf{u}_1 = \mathbf{1}_n, \alpha_1 = 0$;
for $t = 1$ to $T$ do
  $\mathbf{p}_t = \frac{\mathbf{u}_t}{\|\mathbf{u}_t\|_1}$;
  for $i = 1$ to $n$ do
    if $t = 1$, $\mathbf{a}_{t,i} = \mathbf{0}_{dy}$
    else $\mathbf{a}_{t,i} = \mathbf{a}_{t-1,i} + K_I(\mathbf{x}_{i_{t-1}}, \mathbf{x}_i)\mathbf{c}_{t-1}$
    $\mathbf{z}_i = \arg\max_{\mathbf{z} \neq \mathbf{y}_i} \mathbf{a}_{t,i}^\top \mathbf{z}$;
    $\mathbf{v}_t(i) = \frac{1}{\beta_t} \mathbf{a}_{t,i}^\top (\mathbf{y}_i - \mathbf{z}_i)$;
    $\mathbf{u}_{t+1}(i) = \mathbf{u}_t(i)(1 - \eta \mathbf{v}_t(i) + \eta^2 \mathbf{v}_t(i)^2)$;
  end for
  sample $i_t$ from distribution $\mathbf{p}_t$;
  $\mathbf{c}_t = \mathbf{y}_{i_t} - \mathbf{z}_{i_t}$;
  $\alpha_{t+1} = \alpha_t + \frac{K_I(\mathbf{x}_{i_t}, \mathbf{x}_{i_t})\mathbf{c}_t^\top \mathbf{c}_t}{T} + 2\frac{\max(1, \sqrt{\alpha_t})}{\sqrt{T}} \mathbf{v}_t(i_t)$;
  $\beta_{t+1} = \sqrt{T} \max(1, \sqrt{\alpha_{t+1}})$;
end for
For given input $\mathbf{x}$, the output is
$h(\mathbf{x}) = \arg\max_{\mathbf{z}} \sum_{t=1}^T K_I(\mathbf{x}_{i_t}, \mathbf{x})(\mathbf{y}_{i_t} - \mathbf{z}_{i_t})^\top \mathbf{z}$
```

use dynamic programming to compute $\mathbf{z} = \text{argmax}_{\mathbf{u}}\, \mathbf{a}^\top \mathbf{u}$. Specifically, we derive the recursion as follows. Let $C(i)$ denote the set of children of node i and $f_+(i)$ denote the largest value of $\mathbf{a}^\top \mathbf{z}$ restricted on the subtree rooted at node i. $f_-(i)$ is the value restricted on the subtree rooted at node i with node i takeing value -1. Then

$$\begin{aligned} f_-(i) &= -a_i + \sum_{k \in C(i)} f_-(k); \\ f_+(i) &= \max\{f_-(i), a_i + \sum_{k \in C(i)} f_+(k)\}. \end{aligned}$$

In the training stage, in order to compute $\mathbf{z} = \text{argmax}_{\mathbf{u} \neq \mathbf{y}}\, \mathbf{a}^\top \mathbf{u}$, we need to obtain the top two maximum points $(\mathbf{z}_1, \mathbf{z}_2)$ and to compare the maximum point $\mathbf{z}_1$ with $\mathbf{y}$. If $\mathbf{z}_1 = \mathbf{y}$, the solution is $\mathbf{z} = \mathbf{z}_2$; otherwise, $\mathbf{z} = \mathbf{z}_1$. Let $f_{2+}(i)$ denote the second largest value with restriction on the subtree rooted at node i. We have the recursion as follows.

$$\begin{aligned} f_-(i) &= -a_i + \sum_{k \in C(i)} f_-(k); \\ [f_+(i), f_{2+}(i)] &= \max\big\{f_-(i), \\ &\qquad \{a_i + f_{2+}(j) + \sum_{k \in C(i), k \neq j} f_+(k)\}_{j \in C(i)}\big\}. \end{aligned}$$

3.6 Accelerate Techniques

Our SP and KSP algorithms can be accelerated. Note that we need to traverse from 1 to n for very large n, thus time costly. We divide the dataset $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$ into k subsets with almost the same size. Instead of traversing from 1 to n, we successively do SP (or KSP) on the sets $\{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$, and use previous result as hot start.

Another trick is followed by observing that the inner loop traversing from 1 to n can run in parallel, so we can take advantage of multiple CPU kernels. Similarly as above, we divide the dataset $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$ into k subsets with nearly the same size, and run the inner loop on k CPUs to accelerate the algorithms.

4 Experiments

In this section we conduct experimental analysis on the popular benchmark datasets: CCAT, ECAT, GCAT, MCAT, NEWS20[1], and WIPO[2]. The first four datasets CCAT, ECAT, GCAT and MCAT can be found from (Lewis et al. 2004). We summary the basic information of these data sets in Table 1.

Table 1: Summary of The Datasets

Data	♯ features	♯ training set	♯ test set	♯ labels	depth
CCAT	47236	10786	5000	34	3
ECAT	47236	3449	5000	26	2
GCAT	47236	6970	5000	33	2
MCAT	47236	5882	5000	10	2
NEWS20	62061	15935	3993	27	2
WIPO	74435	1352	358	189	4

The classification accuracy is evaluated by standard information retrieval statistics: precision (P), recall (R) and $F1$ with $F1 = \frac{2PR}{P+R}$. The precision and recall are computed over all micro-label predictions in the test set.

We give some implementation details. Although the convergence analysis in section 3.3 need to set initial weight $\mathbf{w}_1 = \mathbf{0}$, we find it is better to set $\mathbf{w}_1$ to be nonzero in practice. The R which is a factor of iteration number T can be estimated as $R = \max_{1 \leq i \leq n} \sqrt{K_I(\mathbf{x}_i, \mathbf{x}_i) K_O(\mathbf{y}_i, \mathbf{y}_i)}$. In our implementation of the algorithm, we only need to return the last iteration vector $\mathbf{w}_T$, because when T is large enough, $1/T \sum_i \mathbf{w}_i$ is almost the same as $\mathbf{w}_T$. Parameter ϵ indicating the training error are set to be 0.1 throughout the paper. And $\eta = \min\{1/(2R), c\sqrt{(\log n)/T}\}$ is provided in Theorem 2 with the small constant number c estimated by cross-validation.

We first compare our algorithm with the methods of Rousu et al.(2006) and Tsochantaridis et al.(2005), which are respectively denoted by LM-1 and LM-2 for simplicity. We choose the linear kernel in our KSP algorithm. We also report the result of the HP algorithm.

The results are summarized in Table 2. From this table, we see that although the KSP algorithm looks much simple, it still achieves high accuracy in comparison with the other algorithms.

Next, we report the CPU times of the three algorithms: KSP, LM-1 and LM-2, which are implemented in Matlab on the same machine. For comparison fair, we do not use the accelerate technique described in Section 3.6. The running times are listed in Table 3. The results agree with our theoretical analysis for KSP in Section 3.3 and 3.4. That is, the

[1] http://people.csail.mit.edu/jrennie/20Newsgroups/

[2] http://www.wipo.int/classifications /ipc/en/support/

Table 2: Classification accuracy with different algorithms

Data	HP			KSP (linear kernel)			LM-1			LM-2		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
CCAT	0.93	0.60	0.73	0.89	0.78	0.83	0.92	0.78	0.85	0.93	0.78	0.85
ECAT	0.95	0.62	0.75	0.93	0.84	0.88	0.93	0.84	0.88	0.95	0.84	0.89
GCAT	0.93	0.66	0.77	0.94	0.75	0.84	0.93	0.78	0.85	0.94	0.78	0.85
MCAT	0.92	0.81	0.86	0.98	0.95	0.96	0.97	0.95	0.96	0.98	0.95	0.96
NEWS20	0.93	0.39	0.56	0.87	0.83	0.85	0.92	0.85	0.88	0.92	0.86	0.89
WIPO	0.92	0.72	0.81	0.92	0.73	0.81	0.93	0.72	0.81	0.95	0.67	0.78

KSP algorithm has very low computational complexity, so it is very efficient.

Table 3: CPU time (s) with KSP, LM-1 and LM-2

Data	Algorithms		
	KSP (linear)	LM-1	LM-2
CCAT	347	4579	18693
ECAT	97	686	3974
GCAT	251	2723	9831
MCAT	100	754	1691
NEWS20	559	4406	39305
WIPO	241	531	13892

Finally, we compare the use of different kernels in Algorithm 3, illustrating influence of choice of kernels. In particular, we employ three popular kernels: linear, polynomial, and Gaussian kernel. The polynomial kernel is defined as $K_I(\mathbf{x}_i, \mathbf{x}_j) = (a + \mathbf{x}_i^\top \mathbf{x}_j)^d$ where $a > 0$ is a constant and d is the polynomial order. As for the Gaussian kernel, it takes the form $K_I(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\gamma))$. The hyper parameters a, d and variance γ can be learned via cross-validation.

We give the classification results on the test data sets in Table 4. It can be seen from Table 4 that KSP is less sensitive to the kernels. However, the kernel form provide the opportunity to use more complex learning model. What is more, the non-linear kernels may perform better on some other data sets rather than listed in Table 1.

5 Conclusion

In this paper we have proposed a stochastic perceptron (SP) based large margin method to solve the hierarchical classification problem. We have also derived kernel stochastic perceptron via the kernel trick. We have proved that our algorithms obtain the ϵ sub-optimal solution with high probability if the data set is separable. We have conducted experimental analysis. The experimental results on the real-world benchmark data sets have demonstrated that our algorithms hold the same performance of the existing state-of-the-art algorithms in prediction accuracy but with much lower computational complexity.

Although our algorithms have been developed for the hierarchical classification problem, they can also be used for other structured prediction problems with slight modifications, such as image segmentation, part-of-speech-tagging, label sequence learning, protein structure prediction, etc. Therefore our algorithms are potentially useful for large-scale and high-dimensional structured data set problems.

Table 4: Comparison with different kernels for KSP

Data	Kernel	P	R	F1
CCAT	Linear	0.89	0.78	0.83
	Polynomial	0.88	0.78	0.83
	Gaussian	0.90	0.74	0.81
ECAT	Linear	0.93	0.84	0.88
	Polynomial	0.93	0.84	0.88
	Gaussian	0.94	0.78	0.85
GCAT	Linear	0.94	0.75	0.84
	Polynomial	0.93	0.76	0.84
	Gaussian	0.96	0.70	0.81
MCAT	Linear	0.98	0.95	0.96
	Polynomial	0.97	0.95	0.96
	Gaussian	0.97	0.92	0.95
NEWS20	Linear	0.87	0.83	0.85
	Polynomial	0.88	0.81	0.84
	Gaussian	0.94	0.74	0.83
WIPO	Linear	0.92	0.73	0.81
	Polynomial	0.91	0.72	0.81
	Gaussian	0.94	0.56	0.70

Acknowledgments

Dehua Liu and Zhihua Zhang acknowledge support from the Natural Science Foundations of China (No. 61070239). Hui Qian acknowledge support from the Natural Science Foundations of China (No. 90820306).

References

Barutcuoglu, Z.; Schapire, R.; and Troyanskaya, O. 2006. Hierarchical multi-label prediction of gene function. *Bioinformatics* 22(7):830–836.

Blum, A., and Dunagan, J. 2002. Smoothed analysis of the perceptron algorithm for linear programming. In *Proc. of the 13th Annual ACM-SIAM Symp. on Discrete Algorithms*, 905–914.

Cai, L., and Hofmann, T. 2004. Hierarchical document categorization with support vector machines. In *Proceedings of the ACM Thirteenth Conference on Information and Knowledge Management*.

Cesa-Bianchi, N.; Gentile, C.; and Zaniboni, L. 2006. Incremental algorithms for hierarchical classification. *Journal of Machine Learning Research* 7:31–54.

Clare, A., and King, R. D. 2003. Learning quickly when irrelevant attributes abound: A new linearthreshold algorithm. *Bioinformatics* 19:ii42–ii49.

Clarkson, K. L.; Hazan, E.; and Woodruff, D. P. 2010. Sublinear optimization for machine learning. In *IEEE 51st Annual Symposium on Foundations of Computer Science*.

Collins, M. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*.

Crammer, K.; Dekel, O.; Keshet, J.; Shalev-Shwartz, S.; and Singer, Y. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research* 7:551–585.

Dekel, O.; Keshet, J.; and Singer, Y. 2004. Large margin hierarchical classification. In *ICML*.

Dumais, S., and Chen, H. 2000. Hierarchical classification of web content. In *Research and Development in Information Retrieval*.

Dłez, J.; del Coz, J. J.; and Bahamonde, A. 2010. A semi-dependent decomposition approach to learn hierarchical classifiers. *Pattern Recognition* 43(11):3795–3804.

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

Gopal, S.; Yang, Y.; Bai, B.; and Niculescu-Mizil, A. 2012. Bayesian models for large-scale hierarchical classification. In *NIPS*.

Gyftodimos, G., and Flach, P. 2003. Hierarchical bayesian networks: an approach to classification and learning for structured data. In *Proceedings of the ECML/PKDD 2003 Workshop on Probablistic Graphical Models for Classification*.

Koller, D., and Sahami, M. 1997. Hierarchically classifying documents using very few words. In *ICML*.

Lafferty, J.; McCallum, A.; and Pereira, F. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Lewis, D. D.; Yang, Y.; Rose, T. G.; and Li, F. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research* 5:361–397.

Littlestone, N. 1987. Learning quickly when irrelevant attributes abound: A new linear threshold algorithm. *Machine Learning* 2(4):285–318.

Ratliff, N. D.; Bagnell, J. A.; and Zinkevich, M. A. 2007. (online) subgradient methods for structured prediction. In *AISTATS*.

Rousu, J.; Saunders, C.; Szedmak, S.; and Shawe-Taylor, J. 2006. Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research* 7:1601–1626.

Sarawagi, S., and Gupta, R. 2008. Accurate max-margin training for structured output spaces. In *ICML*.

Shalev-Shwartz, S.; Singer, Y.; and Srebro, N. 2007. Pegasos: Primal estimated sub-gradient solver for svm. In *ICML*.

Shawe-Taylor, J., and Cristianini, N. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Silla, C. N., and Freitas, A. A. 2010. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery* 22(1-2):31–72.

Taskar, B.; Chatalbashev, V.; and Guestrin, C. 2005. Learning structured prediction models: A large margin approach. In *ICML*.

Taskar, B.; Guestrin, C.; and Koller, D. 2003. Max margin markov networks. In *NIPS*.

Tsochantaridis, I.; Joachims, T.; Hofmann, T.; and Altun, Y. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research* 6:1453–1484.

Wang, Z.; Crammer, K.; and Vucetic, S. 2010. Multi-class pegasos on a budget. In *ICML*.

Wu, F.; Zhang, J.; and Honavar, V. 2005. Learning classifiers using hierarchically structured class taxonomies. In *Proceedings of symposium on abstraction reformulation, and approximation*.

Zhou, D.; Xiao, L.; and Wu, M. 2011. Hierarchical classification via orthogonal transfer. In *ICML*.

Zinkevich, M. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, 928–936.