Hierarchical Classification via Orthogonal Transfer

Dengyong Zhou DENZHO@MICROSOFT.COM
Lin Xiao LIN.XIAO@MICROSOFT.COM
Mingrui Wu MINGRUIW@MICROSOFT.COM
Microsoft Research, 1 Microsoft Way, Redmond, WA 98052, USA

Abstract

We consider multiclass classification problems where the set of labels are organized hierarchically as a category tree. We associate each node in the tree with a classifier and classify the examples recursively from the root to the leaves. We propose a hierarchical Support Vector Machine (SVM) that encourages the classifier at each node to be different from the classifiers at its ancestors. More specifically, we introduce regularizations that force the normal vector of the classifying hyperplane at each node to be orthogonal to those at its ancestors as much as possible. We establish conditions under which training such a hierarchical SVM is a convex optimization problem, and develop an efficient dual-averaging method for solving it.

1. Introduction

In many multiclass classification problems, such as document and web categorization, the set of possible labels are often organized in a hierarchical structure, i.e., a category tree or a more general taxonomy. While we can approach such problems using a generic multiclass classifier such as the multiclass Support Vector Machine (SVM) (Weston & Watkins, 1999; Crammer & Singer, 2001), a challenging question is how we can improve the classification accuracy by using the hierarchical structure as side information.

One straightforward way for exploiting the hierarchical structure is to decouple the problem into a set of independent classification problems, each defined for an internal node in the hierarchy, for classification between its immediate subclasses (Koller & Sahami, 1997; Weigend et al., 1999; Dumais & Chen, 2000). To better exploit the semantic relationship embedded in the hierarchy, some researchers imposed statistical similarity constraints between the probabilistic models for adjacent nodes in the hierarchy (e.g., McCallum et al., 1998). Similarly, several work on multi-task and transfer learning employed hierarchy-induced regularizations that try to make the classifiers at adjacent nodes as close as possible (Cai & Hofmann, 2004; Dekel et al., 2004; Evgeniou et al., 2005).

Another popular method for hierarchical classification is to use tree-induced loss functions (Cai & Hofmann, 2004; Dekel et al., 2004; Cesa-Bianchi et al., 2006). Roughly speaking, a tree-induced loss for misclassifying two classes is proportional to the length of the undirected path connecting these two classes (their graph distance). A closely related approach is to embed the category tree into a Euclidean space such that two classes connected by a shorter path stay closer in the embedding space (Weinberger & Chapelle, 2008). Using tree-induced losses captures the idea that misclassifications between similar classes with shorter graph distances are less severe thus should receive less penalty. On the other hand, it is often the case that classification between classes with shorter graph distances are much harder than classifying classes with longer graph distances. In a sense, this approach does not deal with the hardness of classifying similar classes at lower levels in the hierarchy, but rather downplays the difficulty by assigning them small penalties.

In this paper, we develop a hierarchical classification method that directly tackles the difficulty of classifying very similar classes at lower levels of the hierarchy. In particular, our formulation encourages the classifier at each node be different from the classifiers at its ancestors. The key observation is that the semantic relationships among the categories in a hierarchical structure are usually of the type of generalization-specialization. In other words, the lower level categories are supposed to have the same general properties as the higher level categories plus additional more specific properties (a similar observation was made

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011. Copyright 2011 by the author(s)/owner(s).

in Koller & Sahami, 1997). For example, for classification between documents on `sports` and `computer science`, the frequency of the word `computer` is a very indicative feature. However, between the two subclasses `compiler` and `operating system` in `computer science`, the word `parsing` can be much more indicative than `computer`. In general, classifications at different levels of the hierarchy may rely on different features, or different combinations of the same features.

In the context of hierarchical SVM, we formalize the above observation by introducing regularizations that encourage the normal vector of the classifying hyperplane at each node to be orthogonal to those at its ancestors as much as possible. We establish necessary and sufficient conditions under which training such a hierarchical SVM is a convex optimization problem. We also develop a variant of the dual-averaging method (Nesterov, 2009; Xiao, 2010) which is very efficient for solving such problems. We evaluate the method on a number of real-world text categorization tasks and obtain state-of-the-art performance.

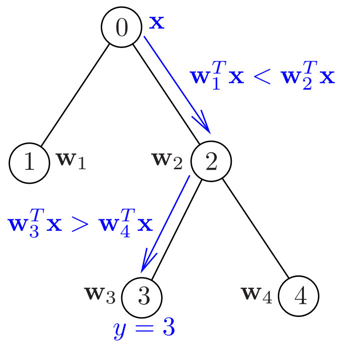
2. Problem Setting

Let $\mathcal{X} \subset \mathcal{R}^n$ be the instance domain and let $\mathcal{Y}$ be the set of labels. Without loss of generality, let $\mathcal{Y} = \{1, \ldots, m\}$. In the context of hierarchical classification, the labels in $\mathcal{Y}$ are identified as nodes in a category tree. We assume that the root of the tree does *not* belong to $\mathcal{Y}$, since it can label all possible instances and does not give meaningful classification. For convenience, let 0 denote the root and let $\overline{\mathcal{Y}} = \mathcal{Y} \cup \{0\}$. For each node $i \in \overline{\mathcal{Y}}$, denote by $\mathcal{C}(i)$ the set of children of i, and $\mathcal{S}(i)$ the set of siblings of i. In addition, let $\mathcal{A}(i)$ be the set of ancestors of i, excluding 0 and itself; and let $\mathcal{D}(i)$ be the set of descendants of i, excluding itself. Finally, let $\mathcal{A}^+(i) = \mathcal{A}(i) \cup \{i\}$ and $\mathcal{D}^+(i) = \mathcal{D}(i) \cup \{i\}$.

Let $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)\}$ be a set of training examples, where each $\mathbf{x}_k \in \mathcal{X}$ and each $y_k \in \mathcal{Y}$. Our goal is to learn a classification function $f : \mathcal{X} \to \mathcal{Y}$ that attains a small classification error. In this paper, we associate each node $i \in \mathcal{Y}$ with a vector $\mathbf{w}_i \in \mathcal{R}^n$, and focus on classifiers $f(\mathbf{x})$ that are parameterized by $\mathbf{w}_1, \ldots, \mathbf{w}_m$ through the following recursive procedure:

$$f(\mathbf{x}) = \left\{ \begin{array}{l} \textbf{initialize } i := 0 \\ \textbf{while } \mathcal{C}(i) \text{ is not empty} \\ \quad i := \underset{j \in \mathcal{C}(i)}{\text{argmax}} \ \mathbf{w}_j^T \mathbf{x} \\ \textbf{return } i \end{array} \right\}. \qquad (1)$$

In other words, an instance is labeled sequentially by choosing the category of which the associated vector outputs the largest score among its siblings, until a leaf node is reached. An example of this recursive procedure is shown in Figure 1. This classifier always return a leaf node. For a testing example $(\mathbf{x}, y)$ where the label y is not a leaf node, a classification error is declared if and only if $y \notin \mathcal{A}^+(f(\mathbf{x}))$.



0
$\mathbf{x}$
$\mathbf{w}_1^T\mathbf{x} < \mathbf{w}_2^T\mathbf{x}$
1
$\mathbf{w}_1$
$\mathbf{w}_2$
2
$\mathbf{w}_3^T\mathbf{x} > \mathbf{w}_4^T\mathbf{x}$
$\mathbf{w}_3$
3
$\mathbf{w}_4$
4
$y = 3$

Figure 1. An example with $\mathcal{Y} = \{1, 2, 3, 4\}$. The instance $\mathbf{x}$ is classified recursively until it reaches the leaf node $y = 3$.

Similar recursive classifiers have been used before, e.g., in Koller & Sahami (1997) and Dumais & Chen (2000). However, the training problem was usually decomposed as training a separate classifier at each non-leaf node in the category tree. In our work, we introduce regularizations that couple the classifiers at different levels, and consider a joint training problem over the whole tree (see Section 3).

Non-recursive classifiers have also been used in hierarchical classification. One popular choice is

$$f(\mathbf{x}) = \underset{i \in \mathcal{Y}}{\text{argmax}} \ \mathbf{w}_i^T \mathbf{x}. \qquad (2)$$

In order to account for the hierarchical structure, the vectors $\mathbf{w}_i$ are further parametrized as

$$\mathbf{w}_i = \textstyle\sum_{j \in \mathcal{A}^+(i)} \mathbf{u}_j, \qquad (3)$$

i.e., $\mathbf{u}_i = \mathbf{w}_i - \mathbf{w}_{p(i)}$, where $p(i)$ denotes the parent node of i. In similarity-based transfer learning, the norms of the $\mathbf{u}_i$'s are used as regularizations, which encourages $\mathbf{w}_i$ to be close to $\mathbf{w}_{p(i)}$ (e.g, Cai & Hofmann, 2004; Dekel et al., 2004).

The recursive classifier (1) allows both training and testing at each internal node to focus on specific features that are most pertinent to classification among its immediate children, which usually is a very small subset of all categories. This can lead to much higher accuracy in classifying the local sibling classes than using the flat classifier (2), which is trained by always considering all categories in the tree. One might argue that mistakes made at higher levels in a recursive classifier cannot be corrected at lower levels. However, this is compensated by higher accuracies at each level.

Another advantage of the recursive classifier (1) is its computational efficiency. For example, to classify $\mathbf{x}$ on a complete d-ary tree with height h, the number of inner products $\mathbf{w}_i^T\mathbf{x}$ required for a recursive classifier is $O(hd)$, while the flat classifier (2) requires $O(d^h)$. Dumais & Chen (2000) also reported large gains in computational efficiency by using recursive classifiers.

3. Hierarchical SVM with Orthogonal Transfer

In this section, we describe a hierarchical SVM for training classifiers of the form (1). It is clear that the task of learning $f(\mathbf{x})$ is reduced to learning the set of vectors $\{\mathbf{w}_i \,|\, i \in \mathcal{Y}\}$, which correspond to the normal vectors of the classification hyperplanes.

As explained in the introduction, our method is motivated by the observation that accurate classification at different levels of the hierarchy may rely on different features, or different combinations of the same features. In order to capture such effects, we introduce the regularization terms $|\mathbf{w}_i^T\mathbf{w}_j|$ to add to the classical hinge loss in an objective function that is to be minimized. These regularization terms encourage each normal vector $\mathbf{w}_i$ to be orthogonal to those at its ancestors. To be precise, given a set of training examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, we propose to solve the following optimization problem

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\sum_{i,j=1}^{m} K_{ij}\left|\mathbf{w}_i^T\mathbf{w}_j\right| + \frac{C}{N}\sum_{k=1}^{N}\xi_k \qquad (4) \\ \text{subject to} \quad & \mathbf{w}_i^T\mathbf{x}_k - \mathbf{w}_j^T\mathbf{x}_k \ge 1 - \xi_k,\ \forall\, j \in \mathcal{S}(i), \\ & \qquad \forall\, i \in \mathcal{A}^+(y_k),\ \forall\, k \in \{1, \ldots, N\}, \\ & \xi_k \ge 0,\ \forall\, k \in \{1, \ldots, N\}. \end{aligned}$$

Here the optimization variables are the normal vectors $\mathbf{w}_1, \ldots, \mathbf{w}_m$ and the slack variables $\xi_1, \ldots, \xi_N$. The parameter C controls the relative weights between the regularization terms and the average hinge loss. We assume the parameters $K_{ij} \ge 0$ for $i, j = 1, \ldots, m$, and they can be considered as entries of a nonnegative matrix $\mathbf{K} \in \mathcal{R}^{m\times m}$. Without loss of generality, let $\mathbf{K}$ be symmetric, i.e., $K_{ij} = K_{ji}$ for all $i, j = 1, \ldots, m$.

We have the following remarks on (4):

- To reflect the hierarchical structure embedded in a tree, we always set $K_{ij} = 0$ whenever node i is neither an ancestor nor a descendent of node j. Thus we can rewrite the regularization terms $\frac{1}{2}\sum_{i,j=1}^{m} K_{ij}\left|\mathbf{w}_i^T\mathbf{w}_j\right|$ more explicitly as
$$\frac{1}{2}\sum_{i\in\mathcal{Y}} K_{ii}\|\mathbf{w}_i\|^2 + \sum_{i\in\mathcal{Y}}\sum_{j\in\mathcal{A}(i)} K_{ij}\left|\mathbf{w}_i^T\mathbf{w}_j\right|.$$
The first term contains the squared L_2 norms of $\mathbf{w}_i$, which is common for most variants of SVM. The second term penalizes the absolute values of the inner products between the vectors $\mathbf{w}_i$ and $\mathbf{w}_j$ whenever $j \in \mathcal{A}(i)$. This effectively encourages orthogonality among the normal vectors of the classification hyperplanes.
- In the constraints, each example $(\mathbf{x}_k, y_k)$ is used for discriminating its category y_k and all its ancestors from their own siblings. Classifier pairs that are not siblings do not appear together in a constraint. This reflects the recursive nature of the classifier (1), and also leads to higher computational efficiency. Again consider the example of a complete d-ary tree with height h. The number of constraints in our formulation is $O(Ndh)$, which is much smaller than $O(Nd^h)$ in many other variants of the multiclass SVM (see Section 5).
- Following Crammer & Singer (2001), we use the same slack variable ξ_k for all the discriminative constraints associated with the example $(\mathbf{x}_k, y_k)$. Nevertheless, the classification margins at different levels in the hierarchy can be effectively differentiated by setting the diagonal coefficients K_{ii}.

3.1. Convexity

We are interested in finding conditions on the parameters K_{ij} (i.e., the matrix $\mathbf{K}$) such that the problem (4) is a convex optimization problem. Since the constraints and the hinge loss only involve linear functions, it suffices to establish convexity of the function

$$\Omega(\mathbf{w}) = \frac{1}{2}\sum_{i,j=1}^{m} K_{ij}\left|\mathbf{w}_i^T\mathbf{w}_j\right|. \qquad (5)$$

Here $\mathbf{w} \in \mathcal{R}^{mn}$ denotes the concatenation of the vectors $\mathbf{w}_1, \ldots, \mathbf{w}_m$. To present our results, we first define the *comparison matrix* $\overline{\mathbf{K}}$ of $\mathbf{K}$, whose entries are given as

$$\overline{K}_{ij} = \begin{cases} |K_{ii}| & \text{if } i = j, \\ -|K_{ij}| & \text{otherwis.} \end{cases}$$

Theorem 1. *The function $\Omega(\mathbf{w})$ is convex if the matrix $\mathbf{K}$ is nonnegative (entry-wise) and its comparison matrix $\overline{\mathbf{K}}$ is positive semidefinite. If $n \ge m - 1$, then these conditions are also necessary.*

We note that in most applications the number of features n is much larger than the number of labels m, thus the conditions in Theorem 1 are necessary and sufficient for convexity for such applications.

Due to space limit, all proofs for results in this paper are given in the technical report (Zhou et al., 2011).

In Section 4.2, we will develop an efficient algorithm for solving problem (4), which relies on the concept of *strong convexity*. The function Ω is strongly convex if there exists $\sigma > 0$ such that $\forall \alpha \in [0,1]$ and $\forall \mathbf{u}, \mathbf{v} \in \mathcal{R}^{mn}$,

$$\Omega(\alpha\mathbf{u}+(1-\alpha)\mathbf{v}) \leq \alpha\Omega(\mathbf{u})+(1-\alpha)\Omega(\mathbf{v})-\frac{\sigma}{2}\alpha(1-\alpha)\|\mathbf{u}-\mathbf{v}\|^2.$$

The constant σ is called the *convexity parameter* of Ω (see, e.g., Nesterov, 2004, § 2.1.3). A slightly stronger condition establishes strong convexity for Ω:

Corollary 2. *If the matrix* $\mathbf{K}$ *is nonnegative and its comparison matrix* $\overline{\mathbf{K}}$ *is positive definite, then* $\Omega(\mathbf{w})$ *is strongly convex with a convexity parameter* $\lambda_{\min}(\overline{\mathbf{K}})$, *which is the smallest eigenvalue of* $\overline{\mathbf{K}}$.

It appears that there are still lots of freedom in choosing the parameters K_{ij}. While further work are needed to develop more principled methods for choosing these parameters, we found some simple heuristics that work reasonably well in practice. One simple choice is to set

$$K_{ij} = K_{ji} = \begin{cases} |\mathcal{D}^+(i)| & \text{if } i = j, \\ \alpha & \text{if } i \in \mathcal{A}(j), \\ 0 & \text{else,} \end{cases} \tag{6}$$

where $\alpha > 0$ is a parameter. For problems with relatively small tree hierarchy, setting $\alpha = 1$ often gives a positive definite $\overline{\mathbf{K}}$. Otherwise, we can always reduce the value of α, or increase the diagonal values K_{ii}, to make $\overline{\mathbf{K}}$ positive definite.

It is worth noticing that the choice of $\mathbf{K}$ in (6) implies $K_{ii} > K_{jj}$ whenever $i \in \mathcal{A}(j)$. In SVM training, this effectively encourages $\|\mathbf{w}_i\| < \|\mathbf{w}_j\|$ for $i \in \mathcal{A}(j)$. As shown in the classical theory on SVM (Vapnik, 1998), a small norm $\|\mathbf{w}\|$ corresponds to a large classification margin. Hence, the choice of $\mathbf{K}$ in (6) tends to give a larger classification margin at a higher-level classification. This is in accordance with the intuition that the classification tasks are relatively easier at higher or more general levels, and become more difficult from the top to the bottom of the tree.

3.2. Representer Theorem

Theorem 3. *If the matrix* $\mathbf{K}$ *is nonnegative and its comparison matrix* $\overline{\mathbf{K}}$ *is positive definite, then the solution to the optimization problem (4) admits a representation of the form* $\mathbf{w}_i = \sum_{k=1}^{N} c_{ik}\mathbf{x}_k$, *for all* $i \in \mathcal{Y}$, *where the* c_{ik}*'s are scalar coefficients.*

The representer theorem implies that our method can be considered in a more general reproducing kernel Hilbert space (RKHS) by choosing a problem specific reproducing kernel. Therefore more expressive nonlinear classifiers can also be established (see, e.g., Schölkopf & Smola, 2001).

4. Optimization Algorithm

Establishing convexity of an optimization problem does not always mean that it can be readily solved by existing algorithms and available software. This is certainly the case for the problem (4). In particular, it cannot be easily transformed into any standard conic optimization form that lends itself to efficient interior-point methods. Neither does it fit any available general or special-purpose SVM solver.

We propose to transform the problem (4) into an unconstrained optimization problem, and solve it using a subgradient-based algorithm. More specifically, by eliminating the slack variables $\xi_1, \ldots, \xi_N$ in (4), we arrive at the following equivalent problem

$$\underset{\mathbf{w}\in\mathcal{R}^{mn}}{\text{minimize}} \quad J(\mathbf{w}) \triangleq \Omega(\mathbf{w}) + H(\mathbf{w}), \tag{7}$$

where $H(\mathbf{w})$ is the hinge loss, given by

$$H(\mathbf{w}) = \frac{C}{N}\sum_{k=1}^{N} \max\left\{0, \max_{\substack{j\in\mathcal{S}(i),\\ i\in\mathcal{A}^+(y_k)}}\left\{1 - \mathbf{w}_i^T\mathbf{x}_k + \mathbf{w}_j^T\mathbf{x}_k\right\}\right\}.$$

The hinge-loss function $H(\mathbf{w})$ is convex, since it is the sum of N piece-wise linear functions, each being a pointwise maximum of several linear functions (see, e.g., Boyd & Vandenberghe, 2004, §3.2). If $\mathbf{K}$ satisfies the conditions in Theorem 1, then $\Omega(\mathbf{w})$ is also a convex function, and so is their sum $J(\mathbf{w})$. On the other hand, both $\Omega(\mathbf{w})$ and $H(\mathbf{w})$ are nondifferentiable. So we resort to subgradient-based methods for solving (7).

Surprisingly, even computing a subgradient of $J(\mathbf{w})$ can be very subtle.

4.1. Computing the Subgradients

The subtlety lies in computing a subgradient for $\Omega(\mathbf{w})$. Given its expression in (5), it is tempting to assembling a subgradient of $\Omega(\mathbf{w})$ by first obtaining a subgradient for each summand $K_{ij}\left|\mathbf{w}_i^T\mathbf{w}_j\right|$ and then add them together. However, this rule of subdifferential calculus is valid only if all the summands are convex (see, e.g., Nesterov, 2004, Chapter 3). Although $\Omega(\mathbf{w})$ is convex (assuming conditions in Theorem 1 are satisfied), the summands $K_{ij}\left|\mathbf{w}_i^T\mathbf{w}_j\right|$, when $i \neq j$, are not convex. Indeed, subgradient does not exist for them almost everywhere. Therefore, the subdifferential calculus for sum of convex functions does not apply here.

In general, computing subgradients of a convex function like $\Omega(\mathbf{w})$ (which is the sum of both convex and nonconvex functions) requires the theory and heavy machinery of *lexicographic differentiation* (Nesterov, 2005). Luckily for us, the particular form of $\Omega(\mathbf{w})$ admits a simple expression for its subgradient:

Theorem 4. *Assume* $\mathbf{K}$ *is nonnegative and* $\overline{\mathbf{K}}$ *is positive semidefinite. Then a subgradient of* $\Omega(\mathbf{w})$ *is given by* $\mathbf{g}^{\Omega} = (\mathbf{g}^{\Omega}_1, \ldots, \mathbf{g}^{\Omega}_m)$, *where*

$$\mathbf{g}^{\Omega}_i = K_{ii}\mathbf{w}_i + \sum_{j\neq i} \operatorname{sign}(\mathbf{w}_i^T\mathbf{w}_j)K_{ij}\mathbf{w}_j,$$

and $\operatorname{sign}(\alpha)$ *equals* 1 *if* $\alpha>0$, -1 *if* $\alpha<0$, *and* 0 *if* $\alpha=0$.

The form of $\mathbf{g}^{\Omega}$ above indeed looks as if it were computed by following the subdifferential calculus for sum of convex functions. Nevertheless, justifying it requires nontrivial proof (Zhou et al., 2011).

In contrast, a subgradient of $H(\mathbf{w})$ can be computed by following the subdifferential calculus for sum of convex functions. For each example $(\mathbf{x}_k, y_k)$, let

$$\big(i(k), j(k)\big) = \underset{j\in\mathcal{S}(i),\ i\in\mathcal{A}^+(y_k)}{\operatorname{argmax}} \left\{1 - \mathbf{w}_i^T\mathbf{x}_k + \mathbf{w}_j^T\mathbf{x}_k\right\},$$

and compute a vector $\mathbf{h}^k = (\mathbf{h}^k_1, \ldots, \mathbf{h}^k_m)$ as follows:

- If $1 - \mathbf{w}^T_{i(k)}\mathbf{x}_k + \mathbf{w}^T_{j(k)}\mathbf{x}_k \leq 0$, then let $\mathbf{h}^k = 0$;
- Otherwise, let $\mathbf{h}^k_{i(k)} = -\mathbf{x}_k$, $\mathbf{h}^k_{j(k)} = \mathbf{x}_k$, and let $\mathbf{h}^k_i = 0$ for all $i \in \mathcal{Y} \setminus \{i(k), j(k)\}$.

Then a subgradient of $H(\mathbf{w})$ is given by the sum $\mathbf{g}^H = \frac{C}{N}\sum_{k=1}^{N}\mathbf{h}^k$. Finally, using again the subdifferential calculus for sum of convex functions, a subgradient of $J(\mathbf{w})$ is given by $\mathbf{g}^J = \mathbf{g}^{\Omega} + \mathbf{g}^H$.

The classical subgradient method for general nonsmooth convex optimization has a convergence rate $O(1/\sqrt{t})$ (see, e.g., Nesterov, 2004, Chapter 3). In this paper, we present a more efficient algorithm for solving the problem (7) when $J(\mathbf{w})$ is strongly convex (by choosing $\mathbf{K}$ appropriately). Next, we first describe the algorithm in a more general context, then discuss how to apply it to the problem (7).

4.2. A Regularized Dual Averaging Method

Consider convex optimization problems of the form

$$\underset{\mathbf{w}\in\mathcal{W}}{\text{minimize}} \quad J(\mathbf{w}) \triangleq \phi(\mathbf{w}) + \Psi(\mathbf{w}), \tag{8}$$

where $\mathcal{W}$ is a closed convex subset of $\mathcal{R}^{mn}$. The objective function $J(\mathbf{w})$ is decomposed as the sum of two parts. We assume that $\phi(\mathbf{w})$ is convex, and $\Psi(\mathbf{w})$ is strongly convex with convexity parameter $\sigma > 0$.

Algorithm 1 is a new variant of the regularized dual averaging (RDA) method (Nesterov, 2009; Xiao, 2010) that is well suitable for solving (8). In particular, Algorithm 1 without steps 3 and 4 corresponds to a special case of the RDA method in Xiao (2010). This algorithm enjoys an $O(\ln(t)/\sigma t)$ convergence rate. To

Algorithm 1 RDA Method with Optimality Bounds

```
input: training examples $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)\}$,
    constant $C > 0$, and accuracy $\epsilon > 0$
initialization: $\mathbf{w}(1) = 0$, $\overline{\mathbf{g}}(0) = 0$, $\delta(0) = 0$,
    upper bound $\overline{J}(1) = C$, lower bound $\underline{J}(1) = 0$
repeat for $t = 1, 2, 3, \ldots$
  1. Compute a subgradient $\mathbf{g}(t) \in \partial\phi(\mathbf{w}(t))$, and
       $\overline{\mathbf{g}}(t) = \frac{t-1}{t}\overline{\mathbf{g}}(t-1) + \frac{1}{t}\mathbf{g}(t)$
  2. Compute the next weight vector:
       $\mathbf{w}(t+1) = \underset{\mathbf{w}\in\mathcal{W}}{\operatorname{argmin}}\Big\{\overline{\mathbf{g}}(t)^T\mathbf{w} + \Psi(\mathbf{w})\Big\}$   (9)
  3. Update the upper bound $\overline{J}$:
       $\overline{J}(t+1) = \min\Big\{\overline{J}(t),\ J(\mathbf{w}(t+1))\Big\}$
  4. Update the lower bound $\underline{J}$:
       $\delta(t) = \frac{t-1}{t}\delta(t-1) + \frac{1}{t}\big(\phi(\mathbf{w}(t)) - \mathbf{g}(t)^T\mathbf{w}(t)\big)$
       $\underline{J}(t+1) = \delta(t) + \overline{\mathbf{g}}(t)^T\mathbf{w}(t+1) + \Psi(\mathbf{w}(t+1))$
       $\underline{J}(t+1) := \max\big\{\underline{J}(t),\ J(t+1)\big\}$
until $\overline{J}(t+1) - \underline{J}(t+1) \leq \epsilon$
```

be more specific, let $J^{\star}$ denote the optimal objective value, and let $\underline{\mathbf{w}}(t)$ denote the solution among $\mathbf{w}(1), \ldots, \mathbf{w}(t)$ that has the smallest objective value (which can be easily recorded), then

$$J\left(\underline{\mathbf{w}}(t)\right) - J^{\star} \leq O\left(\frac{\ln(t)}{\sigma t}\right). \tag{10}$$

However, this theoretical result does not provide a practical stopping criterion that guarantees a solution of specified precision. The difficulty is that the hidden constants in the $O(\cdot)$ notation is problem-dependent and hard to estimate.

The extra steps 3 and 4 in each iteration of Algorithm 1 provide both an upper bound and a lower bound on the objective value. Therefore we have an effective stopping criterion to produce an ϵ-approximate solution, without the need to estimate any problem-dependent constant. This type of guarantee is very rare for simple subgradient-based methods. Updating the upper bounds $\overline{J}(t)$ in Algorithm 1 is straightforward, so we only need to explain the lower bounds $\underline{J}(t)$. Since $\mathbf{g}(t)$ is a subgradient of ϕ at $\mathbf{w}(t)$, we have

$$\phi(\mathbf{w}) \geq \phi(\mathbf{w}(t)) + \mathbf{g}(t)^T\big(\mathbf{w} - \mathbf{w}(t)\big), \quad \forall\, \mathbf{w} \in \mathcal{W}.$$

Therefore, for all $\mathbf{w} \in \mathcal{W}$,

$$J(\mathbf{w}) \geq \frac{1}{t}\sum_{\tau=1}^{t}\Big(\phi(\mathbf{w}(\tau)) + \mathbf{g}(\tau)^{\top}(\mathbf{w} - \mathbf{w}(\tau))\Big) + \Psi(\mathbf{w}).$$

Taking the minimum on both sides, we have

$$\begin{aligned}
&\min_{\mathbf{w}\in\mathcal{W}} J(\mathbf{w}) \\
&\geq \min_{\mathbf{w}\in\mathcal{W}} \left\{ \frac{1}{t}\sum_{\tau=1}^{t}\Big(\phi(\mathbf{w}(\tau)) + \mathbf{g}(\tau)^T(\mathbf{w}-\mathbf{w}(\tau))\Big) + \Psi(\mathbf{w})\right\} \\
&= \underbrace{\frac{1}{t}\sum_{\tau=1}^{t}\Big(\phi(\mathbf{w}(\tau)) - \mathbf{g}(\tau)^T\mathbf{w}(\tau)\Big)}_{} + \min_{\mathbf{w}\in\mathcal{W}}\left\{\overline{\mathbf{g}}(t)^T\mathbf{w} + \Psi(\mathbf{w})\right\} \\
&= \qquad \delta(t) \quad + \overline{\mathbf{g}}(t)^T\mathbf{w}(t+1) + \Psi(\mathbf{w}(t+1)).
\end{aligned}$$

The last line above is precisely what is used to compute the new lower bound in step 4.

The complexity in (10) is given in terms of number of iterations (i.e., number of subgradient queries). For Algorithm 1 to be practically efficient, The minimization problem in (9) needs to be easy to solve. For example, in the special case of $\Psi(\mathbf{w}) = (\sigma/2)\|\mathbf{w}\|^2$ and $\mathcal{W} = \mathcal{R}^{mn}$, it has a closed form solution

$$\mathbf{w}(t+1) = -\frac{1}{\sigma}\overline{\mathbf{g}}(t), \tag{11}$$

and the computational cost per iteration is $O(mn)$.

4.3. Splitting the Objective

Algorithm 1 can only be applied to solve the problem (7) when $J(\mathbf{w})$ is strongly convex. This can be guaranteed by choosing the matrix $\mathbf{K}$ properly. Assuming that $\mathbf{K}$ is nonnegative and its comparison matrix $\overline{\mathbf{K}}$ is positive definite, then the function $\Omega(\mathbf{w})$ is strongly convex (Corollary 2). Since $J(\mathbf{w})$ is the sum of $\Omega(\mathbf{w})$ and a convex function $H(\mathbf{w})$, it is also strongly convex with the same convexity parameter (see, e.g. Nesterov, 2004, Chapter 2). Let $\lambda_{\min} > 0$ be the smallest eigenvalue of $\overline{\mathbf{K}}$, then both $\Omega(\mathbf{w})$ and $J(\mathbf{w})$ are strongly convex with convexity parameter $\sigma = \lambda_{\min}$.

To apply Algorithm 1, we also need to split $J(\mathbf{w})$ into the form of (8) appropriately. In particular, the obvious splitting by assigning $\phi(\mathbf{w}) = H(\mathbf{w})$ and $\Psi(\mathbf{w}) = \Omega(\mathbf{w})$ does not work, because in this case the minimization problem in (9) does not admit a simple solution. In our implementation, we use

$$\begin{aligned}
\phi(\mathbf{w}) &= \Omega(\mathbf{w}) - \frac{\lambda_{\min}}{2}\|\mathbf{w}\|^2 + H(\mathbf{w}), \\
\Psi(\mathbf{w}) &= \frac{\lambda_{\min}}{2}\|\mathbf{w}\|^2.
\end{aligned}$$

With this splitting, the function $\Psi(\mathbf{w})$ is strongly convex with convexity parameter $\sigma = \lambda_{\min}$. The function $\phi(\mathbf{w})$ is convex, since subtracting $\lambda_{\min}$ from the diagonals of $\mathbf{K}$ still leaves the corresponding comparison matrix positive semidefinite. With this splitting, we can calculate subgradients of $\phi(\mathbf{w})$ following Section 4.1, and the equation (9) can be replaced by (11).

5. Related Work

We had a general discussion of related work in the introduction. Here we give the exact formulations of some of them that we will compare with in Section 6.

- FlatMult. This is the flat multiclass SVM of Crammer & Singer (2001). It uses the flat classifier (2) and trains the classifier by solving

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\sum_{i\in\mathcal{Y}}\|\mathbf{w}_i\|^2 + \frac{C}{N}\sum_{k=1}^{N}\xi_k \\
\text{subject to} \quad & \mathbf{w}_{y_k}^T\mathbf{x}_k - \mathbf{w}_i^T\mathbf{x}_k \geq 1 - \xi_k, \\
& \quad \forall\, i\in\mathcal{Y}\backslash\{y_k\},\ \forall\, k\in\{1,\ldots,N\}, \\
& \xi_k \geq 0,\ \forall\, k\in\{1,\ldots,N\}.
\end{aligned} \tag{12}$$

  Since this formulation ignores the hierarchical structure, we apply it only to the leaf labels in the tree, i.e., we replace $\mathcal{Y}$ by the set of leaves $\mathcal{L}$.

- HierMult. This is the hierarchical multiclass SVM suggested by Dumais & Chen (2000). It uses the recursive classifier (1), and solves a separate flat multiclass SVM at each non-leaf node. That is, for each $j \in \overline{\mathcal{Y}} \setminus \mathcal{L}$, we solve a problem like (12) by replacing $\mathcal{Y}$ with $\mathcal{C}(j)$.

- Transfer. This approach uses the recursive classifier (1), and employs a regularization that encourages the classifiers to be close to its ancestors:

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\sum_{i\in\mathcal{Y}}\Big(\|\mathbf{w}_i\|^2 + \sum_{j\in\mathcal{C}(i)}\|\mathbf{w}_i - \mathbf{w}_j\|^2\Big) + \frac{C}{N}\sum_{k=1}^{N}\xi_k \\
\text{subject to} \quad & \mathbf{w}_i^T x_k - \mathbf{w}_j^T x_k \geq 1 - \xi_k,\ \forall\, j\in\mathcal{S}(i), \\
& \quad \forall\, i\in\mathcal{A}^+(y_k), \forall\, k\in\{1,\ldots,N\}, \\
& \xi_k \geq 0, \forall\, k\in\{1,\ldots,N\}.
\end{aligned}$$

  In light of the parametrization (3), we have the relationship $\|\mathbf{w}_i - \mathbf{w}_j\| = \|\mathbf{u}_j\|$ for $j \in \mathcal{C}(i)$.

- TreeLoss. This is a form of the hierarchical SVM of Cai & Hofmann (2004). It uses the flat classifier (2) and solves a training problem in terms of the incremental vectors $\mathbf{u}_i$ that appeared in the parametrization (3). More specifically,

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\sum_{i\in\mathcal{Y}}\|\mathbf{u}_i\|^2 + \frac{C}{N}\sum_{k=1}^{N}\xi_k \\
\text{subject to} \quad & \sum_{i\in\mathcal{A}^+(y_k)}\mathbf{u}_i^T\mathbf{x}_k - \sum_{j\in\mathcal{A}^+(y)}\mathbf{u}_j^T\mathbf{x}_k \geq 1 - \frac{\xi_k}{\Delta(y_k,y)}, \\
& \qquad \forall\, y\in\mathcal{Y}\setminus\{y_k\}, \\
& \xi_k \geq 0,\ \forall\, k\in\{1,\ldots,N\}.
\end{aligned}$$

  The path-loss coefficient $\Delta(y_k, y)$ is set to be half of the graph distance (length of shortest path) between y_k and y in the tree structure.

Table 1. Some statistics of RCV1-v2/LYRL2004

	$\lvert\mathcal{Y}\rvert$	$\lvert\mathcal{L}\rvert$	Total	Train	Test
CCAT	31	26	209,133	5,810	203,323
MCAT	9	7	189,211	5,438	183,773
ECAT	23	18	71,356	2,196	69,160

Table 2. Testing performance on CCAT (%)

	0/1 Loss	Tree Loss	Micro F1
FlatMult	26.93(±0.28)	37.46(±0.47)	73.07(±0.28)
HierMult	26.19(±0.28)	35.80(±0.45)	73.81(±0.28)
Transfer	26.21(±0.26)	35.60(±0.42)	73.79(±0.26)
TreeLoss	37.57(±0.35)	50.45(±0.49)	62.43(±0.35)
Orthognl	**23.62(±0.43)**	**33.12(±0.58)**	**76.38(±0.43)**

Table 3. Testing performance on MCAT (%)

	0/1 Loss	Tree Loss	Micro F1
FlatMult	7.03(±0.20)	**8.96(±0.27)**	92.97(±0.20)
HierMult	7.36(±0.21)	9.32(±0.27)	92.64(±0.21)
Transfer	8.20(±0.22)	10.72(±0.31)	91.80(±0.22)
TreeLoss	26.16(±0.34)	34.28(±0.45)	73.84(±0.34)
Orthognl	**6.57(±0.19)**	**8.74(±0.27)**	**93.43(±0.19)**

Table 4. Testing performance on ECAT (%)

	0/1 Loss	Tree Loss	Micro F1
FlatMult	**16.36(±0.30)**	23.71(±0.40)	**83.64(±0.30)**
HierMult	**15.99(±0.31)**	**22.77(±0.40)**	**84.01(±0.31)**
Transfer	17.04(±0.36)	24.33(±0.48)	82.96(±0.36)
TreeLoss	26.71(±0.51)	36.49(±0.74)	73.29(±0.51)
Orthognl	**16.18(±0.31)**	24.21(±0.45)	**83.82(±0.31)**

Table 5. Some statistics of the hierarchical subset

	$\lvert\mathcal{Y}\rvert$	$\lvert\mathcal{L}\rvert$	Total	Train	Test
CCAT	16	12	138,125	3,676	134,449
MCAT	7	5	123,343	3,682	119,661
ECAT	18	13	57,582	1,734	55,848

Table 6. Testing performance on CCAT.Hierarchical (%)

	0/1 Loss	Tree Loss	Micro F1
FlatMult	21.39(±0.29)	32.66(±0.56)	78.61(±0.29)
HierMult	21.41(±0.29)	32.23(±0.53)	78.59(±0.29)
Transfer	21.91(±0.31)	33.45(±0.56)	78.09(±0.31)
TreeLoss	26.32(±0.39)	40.95(±0.66)	73.68(±0.39)
Orthognl	**17.46(±0.74)**	**28.97(±0.95)**	**82.54(±0.74)**

Table 7. Testing performance on MCAT.Hierarchical (%)

	0/1 Loss	Tree Loss	Micro F1
FlatMult	5.23(±0.21)	5.90(±0.23)	94.77(±0.21)
HierMult	4.84(±0.20)	5.41(±0.21)	95.16(±0.20)
Transfer	4.70(±0.20)	5.27(±0.22)	95.30(±0.20)
TreeLoss	13.34(±0.71)	16.31(±1.00)	86.66(±0.71)
Orthognl	**3.00(±0.14)**	**3.56(±0.15)**	**97.00(±0.14)**

Table 8. Testing performance on ECAT.Hierarchical (%)

	0/1 Loss	Tree Loss	Micro F1
FlatMult	13.57(±0.26)	**20.76(±0.39)**	86.43(±0.26)
HierMult	13.57(±0.27)	**20.53(±0.40)**	86.43(±0.27)
Transfer	13.67(±0.28)	**20.88(±0.41)**	86.33(±0.28)
TreeLoss	17.13(±0.37)	26.20(±0.63)	82.87(±0.37)
Orthognl	**12.47(±0.30)**	**20.89(±0.57)**	**87.53(±0.30)**

6. Preliminary Experiments

We evaluated our method on the widely used text categorization benchmark called RCV1-v2/LYRL2004 (Lewis et al., 2004). The documents have been tokenized, stopworded and stemmed to 47,236 unique tokens (features) and represented as L2-normalized log TF-IDF vectors. The top categories MCAT, CCAT and ECAT were used to form three classification tasks. For each task, we excluded documents with multiple labels to stick with a tree structure. The other top category GCAT was not considered because its structure is flat rather than organized hierarchically. Some statistics of the dataset are summarized in Table 1, where $|\mathcal{Y}|$ is the number of all categories and $|\mathcal{L}|$ is the number of leaf categories.

We compared our method with the four methods described in Section 5. We converted all of them into unconstrained optimization problems by eliminating the slack variables, similar to (7). Then we solved them using Algorithm 1, as they all fit into the structure of (8). We used the parameter $C = 1$ in all formulations, which is common in text classification (e.g., Joachims, 1998; Lewis et al., 2004; Cai & Hofmann, 2004). We also tried different values of C varying from 1 to 100, but didn't observe significant differences in the results. Our method is listed as Orthognl. We chose the matrix $\mathbf{K}$ according to (6). For this dataset, setting $\alpha = 1$ makes the comparison matrices positive definite.

The evaluation metrics include 0/1 loss (error rate), tree-induced loss (half of the graph distance between two categories), and micro-average F1 score (see, e.g., Lewis et al., 2004). The results are summarized in Tables 2-4, all shown in percentage. Each entry in the tables shows the average metric, as well as standard deviation, computed over 50 rounds of random samplings. The training/testing split ratio for each round is the same as shown in Table 1. The numbers in bold fonts are best results judged by t-test with a significance level of 0.01. We see that our method outperformed other approaches on CCAT and MCAT, but is slightly worse than HierMult on ECAT, mainly in terms of tree-induced loss.

We also did experiments on a particular subset of RCV1-v2/LYRL2004. This subset was built simply by removing the first-level categories that have no descendants. In other words, we only kept those branches in the tree that have at least two levels. This subset emphasizes the effects of hierarchical structure. The statistics of this subset are summarized in Table 5. The classification performances of different methods are summarized in Tables 6-8. We see that on this subset of data, our orthogonal transfer method performed even better compared with other approaches.

7. Conclusions

We proposed a novel method called orthogonal transfer for hierarchical classification, which specifically tackles the difficulty of classifying similar classes in the lower levels of the category hierarchy. We presented a convex optimization formulation for the problem and devised an efficient dual averaging method for solving it. Preliminary experiments show that our method can effectively exploit the hierarchical structure and is able to produce improved classification accuracy.

Several results of this paper can be of independent interests: the necessary and sufficient conditions for $\Omega(\mathbf{w})$ to be convex, the derivation of its subgradient, and the general RDA method with optimality bounds.

As a future work, we are very interested in analyzing orthogonal transfer from a learning theory perspective. We also hope to investigate more principled methods for choosing or learning $\mathbf{K}$ for better performance.

Acknowledgments

We thank Professor Yurii Nesterov for comments on the convexity conditions and pointers to the subject of lexicographic differentiation.

References

Boyd, S. and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, 2004.

Cai, L. and Hofmann, T. Hierarchical document categorization with support vector machines. In *Proc. 13th ACM International Conference on Information and Knowledge Management*, pp. 78–87, 2004.

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

Crammer, K. and Singer, Y. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2: 265–292, 2001.

Dekel, O., Keshet, J., and Singer, Y. Large margin hierarchical classification. In *Proc. 21st International Conference on Machine Learning*, pp. 27–34, 2004.

Dumais, S. T. and Chen, H. Hierarchical classification of web content. In *Proceedings of SIGIR'00*, pp. 256–263, 2000.

Evgeniou, T., Micchelli, C. A., and Pontil, M. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

Joachims, T. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, pp. 137–142, 1998.

Koller, D. and Sahami, M. Hierarchically classifying docuemnts using very few words. In *Proc. 14th Intl. Conf. Machine Learning*, pp. 171–178, 1997.

Lewis, D. D., Yang, Y., Rose, T., and Li, F. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5: 361–397, 2004.

McCallum, A. K., Rosenfeld, R., Mitchell, T. M., and Ng, A. Y. Improving text classification by shrinkage in a hierarchy of classes. In *Proc. 15th Intl. Conf. on Machine Learning*, pp. 359–367, 1998.

Nesterov, Yu. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer, Boston, 2004.

Nesterov, Yu. Lexicographic differentiation of nonsmooth functions. *Mathematical Programming*, 104: 669–700, 2005.

Nesterov, Yu. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120: 221–259, 2009.

Schölkopf, B. and Smola, A. J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

Vapnik, V. N. *Statistical learning theory*. John Wiley & Sons, New York, 1998.

Weigend, A. S., Wiener, E. D., and Pedersen, J. O. Exploiting hierarchy in text categorization. *Information Retrieval*, 1:193–216, 1999.

Weinberger, K. and Chapelle, O. Large margin taxonomy embedding with an application to document categorization. In *Advances in Neural Information Processing Systems 21*, pp. 1737–1744, 2008.

Weston, J. and Watkins, C. Support vector machines for multi-class pattern recognition. In *Proceedings of the 6th European Symposium on Artificial Neural Networks (ESANN)*, pp. 219–224, 1999.

Xiao, L. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11:2543–2596, 2010.

Zhou, D., Xiao, L., and Wu, M. Hierarchical classification via orthogonal transfer. Technical Report MSR-TR-2011-54, Microsoft Research, 2011.