Exploiting Known Taxonomies in Learning Overlapping Concepts

Lijuan Cai and **Thomas Hofmann**
Department of Computer Science
Brown University, Providence, RI, USA
{ljcai, th}@cs.brown.edu

Abstract

Many real-world classification problems involve large numbers of overlapping categories that are arranged in a hierarchy or taxonomy. We propose to incorporate prior knowledge on category taxonomy directly into the learning architecture. We present two concrete multi-label classification methods, a generalized version of Perceptron and a hierarchical multi-label SVM learning. Our method works with arbitrary, not necessarily singly connected taxonomies, and can be applied more generally in settings where categories are characterized by attributes and relations that are not necessarily induced by a taxonomy. Experimental results on WIPO-alpha collection show that our hierarchical methods bring significant performance improvement.

1 Introduction

Many real-world classification tasks involve large numbers of overlapping categories. Prominent examples include the International Patent Classification scheme (approx. 69,000 patent groups), the Open Directory project (approx. 590,000 categories for Web pages), and the Gene Ontology (approx. 17,000 terms to describe gene products). In most cases, instances are assigned to more than one category, since categories are rarely mutually exclusive. This leads to large scale multi-label classification problems. The categories are typically organized in *hierarchies* or *taxonomies*, most commonly by introducing superordinate concepts and by relating categories via 'is-a' relationships. Multiply connected taxonomies are not uncommon in this context.

We believe that taxonomies encode valuable domain knowledge which learning methods should be able to capitalize on, in particular since the number of training examples for individual classes may be very small when dealing with tens of thousands or more classes. The potential loss of valuable information by ignoring class hierarchies has been pointed out before and has led to a number of approaches that employ different ways to exploit hierarchies [McCallum *et al.*, 1998; Wang *et al.*, 1999; Dumais and Chen, 2000].

In this paper, we present an approach for systematically incorporating domain knowledge about the relationships between categories into the Perceptron learning and SVM classification architecture. The rest of the paper is organized as follows. Section 2 introduces two ways of representing taxonomy knowledge. First, derive pairwise similarities between categories based on their relative locations in the taxonomy in order to tie learning across categories. Second, adapt the standard 0/1 loss function to weigh misclassification errors in accordance with the taxonomy structure. In Section 3 we formulate the hierarchical learning problem in terms of a joint large margin problem, for which we derive an efficient training algorithm. In Section 4 we propose a hierarchical Perceptron algorithm that exploits taxonomies in a similar fashion. Section 5 examines related work. Section 6 presents experimental results which show the two new hierarchical algorithms bring significant improvements on all metrics. Conclusions and future work are discussed in Section 7.

2 Utilizing Known Taxonomies

2.1 Problem Setting

We assume the patterns such as documents are represented as vectors, $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$, which can be mapped to a higher dimensional feature space as $\phi(\mathbf{x})$. We denote the set of categories by $\mathcal{Y} = \{1, \ldots, q\}$, a category by $y \in \mathcal{Y}$, and a label set by $Y \in \mathrm{P}(\mathcal{Y})$ where $\mathrm{P}(\mathcal{Y})$ is the power set of $\mathcal{Y}$. A *taxonomy* is a directed acyclic graph $(\mathcal{V}, E)$ with nodes $\mathcal{V} \supseteq \mathcal{Y}$ such that the set of terminal nodes equals $\mathcal{Y}$, formally $\mathcal{Y} = \{y \in \mathcal{V} : \nexists v \in \mathcal{V}, (y, v) \in E\}$. Note that we do not assume that a taxonomy is singly connected (tree or forest), but allow for converging nodes. In some cases one wants to express that items belong to a super-category, but to none of the terminal categories in $\mathcal{Y}$, we suggest to model this by formally adding one terminal node to each inner node, representing a "miscellaneous" category; this avoids the problem of partial paths.

In multi-label learning, we aim at finding a mapping $f : \mathcal{X} \to \mathrm{P}(\mathcal{Y})$, based on a sample of training pairs $\{(x_i, Y_i),\ i = 1, \ldots, n\} \subseteq \mathcal{X} \times \mathrm{P}(\mathcal{Y})$. A popular approach as suggested, for instance by [Schapire and Singer, 2000], is to actually learn a ranking function over the categories for each pattern, $g : \mathcal{X} \to S_q$, where S_q is the set of permutations of ranks 1 to q. In order to get a unique subset of labels, one then needs address the additional question on how to select the number of categories a pattern should be assigned to.

It is common to define the ranking function g implicitly via a scoring function $F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, such that $g(\mathbf{x})(y) < g(\mathbf{x})(y')$ whenever $F(\mathbf{x}, y) > F(\mathbf{x}, y')$, i.e. categories with higher F-values appear earlier in the ranking (for ease of presentation we ignore ties). Notation $g(y)$ is used when clear from context.

2.2 Class Attributes

Following [Cai and Hofmann, 2004] we suggest to use scoring functions F that are linear in some joint feature representation $\mathbf{\Phi}$ of inputs and categories, namely $F(\mathbf{x}, y; \mathbf{w}) \equiv \langle \mathbf{w}, \mathbf{\Phi}(\mathbf{x}, y) \rangle$, where $\mathbf{w}$ is a weight vector. Following [Tsochantardis *et al.*, 2004; Cai and Hofmann, 2004] $\mathbf{\Phi}$ will be chosen to be of the form $\mathbf{\Lambda}(y) \otimes \phi(\mathbf{x})$, where $\mathbf{\Lambda}(y) = (\lambda_1(y), \ldots, \lambda_s(y))^T \in \mathbb{R}^s$ refers to an attribute vector representing categories and $\otimes$ is the Kronecker product. One can interpret $\mathbf{w}$ in terms of a stacked vector of individual weight vectors, i.e. $\mathbf{w} = (\mathbf{w}_1^T, \ldots, \mathbf{w}_s^T)^T$, leading to the additive decomposition $F(\mathbf{x}, y; \mathbf{w}) = \sum_{r=1}^{s} \lambda_r(y) \langle \mathbf{w}_r, \mathbf{x} \rangle$. The general idea is that the notion of class attributes will allow generalization to take place across (similar) categories and not just across training examples belonging to the same category. In the absence of such information, one can set $s = q$ and define $\lambda_r(y) = \delta_{ry}$ which leads to $F(\mathbf{x}, y) = \langle \mathbf{w}_y, \mathbf{x} \rangle$.

How are we going to translate the taxonomy information into attributes for categories? The idea is to treat the nodes in the taxonomy as properties. Formally, we define

$$\lambda_v(y) = \begin{cases} t_v, & \text{if } v \in \text{anc}(y) \\ 0, & \text{otherwise}, \end{cases} \tag{1}$$

where $t_v \geq 0$ is the attribute value for node v. In the simplest case, t_v can be set to a constant, like 1. We denote by $\text{anc}(y)$ the set of ancestor nodes of y in the taxonomy including y itself (for notational convenience). This leads to an intuitive decomposition of the scoring function F into contributions from all nodes along the paths from a root node to a specific terminal node.

2.3 Loss Functions

A standard loss function for the multi-label case is to use the *symmetric difference* between the predicted and the actual label set, i.e. to count the number of correct categories missed plus the number of incorrect categories that have been assigned, $\triangle_0(Y, \hat{Y}) \equiv |Y \ominus \hat{Y}|$.

Yet, in many applications, the actual loss of a predicted label set relative to the true set of category labels will depend on the relationship between the categories. As a motivation we consider the generic setting of routing items based on their membership at nodes in the taxonomy. For instance, in a news routing setting, readers may sign-up for specific topics by selecting an appropriate node, which can either be a terminal node in the taxonomy (e.g. the category "Soccer") or an inner node (e.g. the super-category "Sports"). Note that while we assume that all items can be assigned to terminal nodes of the taxonomy only, customers may prefer to sign-up for many categories *en bloc* by selecting an appropriate super-category.

We assume that there is some relative sign-up volume $s_v \geq 0$ for each node as well as costs c^- of missing a relevant item and c^+ for assigning an irrelevant item. For any label set Y, define $\text{anc}(Y) \equiv \{v \in \mathcal{V} : \exists y \in Y, v \in \text{anc}(y)\}$. Now we can quantify the loss in the following manner:

$$\triangle(Y, \hat{Y}) = c^- \sum_{v \in \text{anc}(Y)} s_v + c^+ \sum_{v \in \text{anc}(\hat{Y})} s_v - (c^- + c^+) \sum_{\substack{v \in \text{anc}(Y) \\ \cap \text{anc}(\hat{Y})}} s_v \tag{2}$$

Note that only nodes in the symmetric difference $\text{anc}(Y) \ominus \text{anc}(\hat{Y})$ contribute to the loss. In the following we will simplify the presentation by assuming that $c^- = c^+ = 1$. Then, by further setting $s_v = 1$ ($\forall v \in \mathcal{V}$) one gets $\triangle(Y, \hat{Y}) = |\text{anc}(Y) \ominus \text{anc}(\hat{Y})|$. Intuitively, this means that one colors all nodes that are on a path to a node in Y with one color, say blue, and all nodes on paths to nodes in $\hat{Y}$ with another color, say yellow. Nodes that have both colors (blue+yellow=green) are correct, blue nodes are the ones that have been missed and yellow nodes are the ones that have been incorrectly selected; both types of mistakes contribute to the loss proportional to their volume.

During training, this loss function is difficult to deal with directly, since it involves sets of labels. Rather, we would like to work with pairwise contributions, e.g. involving terms

$$\triangle(y, y') = |\text{anc}(y) \ominus \text{anc}(y')| \,. \tag{3}$$

In singly connected taxonomies Eq. (3) is equivalent to the length of the (undirected) shortest path connecting the nodes y and y', suggested by [Wang *et al.*, 1999]. In order to relate the two, we state the following proposition.

Proposition 1. *For any $Y, \hat{Y} \subseteq \mathcal{Y}$ satisfying $Y \not\subseteq \hat{Y}$ and $\hat{Y} \not\subseteq Y$,*

$$|anc(Y) \ominus anc(\hat{Y})| \leq \sum_{\substack{y \in Y - \hat{Y} \\ \hat{y} \in \hat{Y} - Y}} |anc(y) \ominus anc(\hat{y})|$$

For learning with ranking functions g, we translate this into

$$\triangle(Y, g) = \sum_{\substack{y \in Y, \hat{y} \in \mathcal{Y} - Y \\ g(y) > g(\hat{y})}} |\text{anc}(y) \ominus \text{anc}(\hat{y})| \,. \tag{4}$$

We look at every pair of categories where an incorrect category comes before a correct category in the order defined by g and count the symmetric difference of the respective ancestor sets as the corresponding loss.

3 Hierarchical Support Vector Machines

3.1 Multi-label Classification

We generalize the multiclass SVM formulation of [Crammer and Singer, 2001] to a multi-label formulation similar to that in [Elisseff and Weston, 2001]. For a given set of correct categories Y_i we denote the complement by $\bar{Y}_i = \mathcal{Y} - Y_i$. Then following [Elisseff and Weston, 2001] we approximate the separation margin of $\mathbf{w}$ with respect to the i-th example as

$$\gamma_i(\mathbf{w}) \equiv \min_{y \in Y_i, \bar{y} \in \bar{Y}_i} \langle \Phi(\mathbf{x}_i, y) - \Phi(\mathbf{x}_i, \bar{y}), \mathbf{w} \rangle \,. \tag{5}$$

Our formulation aims at maximizing the margin over the whole training set, i.e. $\max_{\mathbf{w}: \|\mathbf{w}\|=1} \min_i \gamma_i(\mathbf{w})$. This is

equivalent to minimizing the norm of the weight vector $\mathbf{w}$ while constraining all (functional) margins to be greater than or equal to 1. A generalized soft-margin SVM formulation can be obtained by introducing slack variables ξ_i's. The penalty is scaled proportional to the loss associated with the violation of the respective category ordering, a mechanism suggested before (cf. [Tsochantardis *et al.*, 2004; Cai and Hofmann, 2004]). Putting these ideas together yields the convex quadratic program (QP)

$$\min_{\mathbf{w},\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \tag{6}$$

$$\text{s.t. } \langle \mathbf{w}, \delta\Phi_i(y,\bar{y})\rangle \geq 1 - \frac{\xi_i}{\triangle(y,\bar{y})}\,, \quad (\forall i, y \in Y_i, \bar{y} \in \bar{Y}_i)$$

$$\xi_i \geq 0, \quad (\forall i)\,.$$

where $\delta\Phi_i(y,\bar{y}) \equiv \Phi(\mathbf{x}_i, y) - \Phi(\mathbf{x}_i, \bar{y})$.

This formulation is similar to the one used in [Schapire and Singer, 2000] and generalizes the ranking-based multi-label SVM formulation of [Elisseff and Weston, 2001]. Following [Crammer and Singer, 2001] we have not included bias terms for categories. The efforts are put into correctly ordering each pair of positive/negative labels. We can use a size prediction mechanism such as the one in [Elisseff and Weston, 2001] to convert the category ranking into an actual multi-label classification.

The dual QP of (6) becomes

$$\max_{\boldsymbol{\alpha}} \Theta(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\sum_{\substack{y\in Y_i\\ \bar{y}\in\bar{Y}_i}} \alpha_{iy\bar{y}} \tag{7}$$

$$- \frac{1}{2}\sum_{i,j}\sum_{\substack{y\in Y_i\\ \bar{y}\in\bar{Y}_i}}\sum_{\substack{r\in Y_j\\ \bar{r}\in\bar{Y}_j}} \alpha_{iy\bar{y}}\alpha_{jr\bar{r}}\langle\delta\Phi_i(y,\bar{y}), \delta\Phi_j(r,\bar{r})\rangle,$$

$$\text{s.t. } \alpha_{iy\bar{y}} \geq 0\ (\forall i, y \in Y_i, \bar{y} \in \bar{Y}_i)\,,\ \sum_{\substack{y\in Y_i\\ \bar{y}\in\bar{Y}_i}} \frac{\alpha_{iy\bar{y}}}{\triangle(y,\bar{y})} \leq C\ (\forall i)\,.$$

Note that

$$\begin{aligned}&\langle\delta\Phi_i(y,\bar{y}), \delta\Phi_j(r,\bar{r})\rangle\\ =&\langle\boldsymbol{\Lambda}(y) - \boldsymbol{\Lambda}(\bar{y}), \boldsymbol{\Lambda}(r) - \boldsymbol{\Lambda}(\bar{r})\rangle\langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle\,.\end{aligned} \tag{8}$$

Herein one can simply replace the inner products by corresponding kernel functions. It is straightforward to observe that $\frac{1}{n}\sum_i \xi_i$ yields an upper bound on the training loss of the resulting classifier measured by $\triangle$ in the following sense. Define the *maximum loss* as

$$\triangle_x(Y, g) \equiv \max_{y\in Y, \bar{y}\in\bar{Y}: g(y)\geq g(\bar{y})} \triangle(y,\bar{y})\,. \tag{9}$$

The *maximal loss* over a set of examples is defined as

$$\triangle_x(F, \{\mathbf{x}_i, Y_i\}_{i=1}^n) \equiv \frac{1}{n}\sum_{i=1}^{n}\triangle_x(Y_i, g(\mathbf{x}_i; F)) \tag{10}$$

Proposition 2. *Denote by $(\hat{\mathbf{w}}, \hat{\boldsymbol{\xi}})$ a feasible solution of the QP in* (6)*. Then $\frac{1}{n}\sum_{i=1}^{n}\hat{\xi}_i$ is an upper bound on the empirical maximal loss $\triangle_x(F(;\hat{\mathbf{w}}), \{\mathbf{x}_i, Y_i\}_{i=1}^n)$.*

Algorithm 1 Hierarchical Multilabel SVM

1: inputs: training data $\{\mathbf{x}_i, Y_i\}_{i=1}^n$, tolerance $\epsilon \geq 0$
2: initialize $S_i = \emptyset$, $\alpha_{iy\bar{y}} = 0, \forall\, i, y \in Y_i, \bar{y} \in \bar{Y}_i$.
3: **repeat**
4: select $\hat{\imath} = \operatorname{argmax}_{i=1}^n \psi_i$
5: select $(\hat{y}, \hat{\bar{y}}) = \operatorname{argmax}_{y\in Y_{\hat{\imath}}, \bar{y}\in\bar{Y}_{\hat{\imath}}} G_{\hat{\imath}y\bar{y}}$
6: expand working set: $S_{\hat{\imath}} = S_{\hat{\imath}} \cup \{(\hat{y}, \hat{\bar{y}})\}$
7: solve QP over subspace $\{\alpha_{\hat{\imath}y\bar{y}} : (y,\bar{y}) \in S_{\hat{\imath}}\}$
8: reduce working set: $S_{\hat{\imath}} = S_{\hat{\imath}} - \{(y,\bar{y}) : \alpha_{\hat{\imath}y\bar{y}} = 0\}$
9: **until** $\psi_{\hat{\imath}} \leq \epsilon$

Note that to minimize (an upper bound on) the loss in Eq. (4), we could simply assign one slack variable $\xi_{iy\bar{y}}$ for every triplet of instance, positive label, and negative label. This leads to a dual program similar to Eq. (7) except the second set of constraints become $\frac{\alpha_{iy\bar{y}}}{\triangle(y,\bar{y})} \leq C\,\forall i, y \in Y, \bar{y} \in \bar{Y}$. We have yet to explore this direction.

3.2 Optimization Algorithm

The derived QP can be very large. We therefore employ an efficient optimization algorithm that is inspired by the SMO algorithm [Platt, 1999] and that performs a sequence of subspace ascents on the dual, using the smallest possible subsets of variables that are not coupled with the remaining variables through constraints. Our algorithm successively optimizes over subspaces spanned by $\{\alpha_{iy\bar{y}} : y \in Y_i, \bar{y} \in \bar{Y}_i\}$ for some selected instance i. Moreover an additional variable selection is performed within each subspace. This strategy is also known as column generation [Demiriz *et al.*, 2002]. Define

$$\mathbf{w}(\boldsymbol{\alpha}) \equiv \sum_{i=1}^{n}\sum_{y\in Y_i, \bar{y}\in\bar{Y}_i} \alpha_{iy\bar{y}}\delta\Phi_i(y,\bar{y}) \tag{11}$$

$$G_{iy\bar{y}} \equiv \triangle(y,\bar{y})\,(1 - \langle\delta\Phi_i(y,\bar{y}), \mathbf{w}(\boldsymbol{\alpha})\rangle) \tag{12}$$

$$l_i \equiv \max\left(\max_{i, y\in Y_i, \bar{y}\in\bar{Y}_i}\{G_{iy\bar{y}}\}\quad,\quad 0\right) \tag{13}$$

$$u_i \equiv \begin{cases}\min_{\substack{i, y\in Y_i, \bar{y}\in\bar{Y}_i:\\ \alpha_{iy\bar{y}}>0}} G_{iy\bar{y}} & \text{if } \zeta_i = 0\\ \min\left(\min_{\substack{i, y\in Y_i, \bar{y}\in\bar{Y}_i:\\ \alpha_{iy\bar{y}}>0}} G_{iy\bar{y}}\ ,\ 0\right) & \text{if } \zeta_i > 0\,,\end{cases} \tag{14}$$

where $\zeta_i = C - \sum_{y\in Y_i, \bar{y}\in\bar{Y}_i}\frac{\alpha_{iy\bar{y}}}{\triangle(y,\bar{y})}$. Define $\psi_i \equiv l_i - u_i$. By derivation similar to that in [Cai and Hofmann, 2004], it can be shown that $\psi_i = 0\,(\forall i)$ is the necessary and sufficient condition for a feasible solution to be optimal. Hence the score ψ_i is used for selecting subspaces. $G_{iy\bar{y}}$ is also used to select new variables to expand the active set of each subspace. The resulting algorithm is depicted in Algorithm 1.

More details on convergence and sparseness of a more general class of algorithms can be found in [Tsochantardis *et al.*, 2004].

4 Hierarchical Perceptron

Although SVM is competitive in generating high-quality classifiers, it can be computationally expensive. The Perceptron algorithm [Rosenblatt, 1958] is known for its simplicity

Algorithm 2 Hierarchical Minover Perceptron algorithm

1: inputs: training data $\{\mathbf{x}_i, Y_i\}_{i=1}^n$, desired margin c.
2: Initialize $\alpha_{iy\bar{y}} = 0, \forall\, i, y \in Y_i, \bar{y} \in \bar{Y}_i$
3: **repeat**
4: $(\hat{i}, \hat{y}, \hat{\bar{y}}) = \text{argmin}_{i, y\in Y_i, \bar{y}\in\bar{Y}_i} \langle \mathbf{w}(\boldsymbol{\alpha}), \delta\Phi_i(y, \bar{y})\rangle$
5: **if** $\langle \mathbf{w}(\boldsymbol{\alpha}), \delta\Phi_{\hat{i}}(\hat{y}, \hat{\bar{y}})\rangle > c$ **then**
6: terminate with a satisfactory solution
7: **else**
8: $\alpha_{\hat{i}\hat{y}\hat{\bar{y}}} \leftarrow \alpha_{\hat{i}\hat{y}\hat{\bar{y}}} + \triangle(\hat{y}, \hat{\bar{y}})$
9: **end if**
10: **until** maximal number of iterations are performed

and speed. In this section, we propose a hierarchical Perceptron algorithm 2 using the minimum-overlap (Minover) learning rule [Krauth and Mézard, 1987].

The Minover Perceptron uses the instance that violates the desired margin the worst to update the separating hyperplane. We can also deal with the instances sequentially, as in a truly online fashion. Using the minimum overlap selection rule effectively speeds up the convergence and yields sparser solutions.

If using taxonomy-based class attribute scheme in Eq. (1) with $t_v = 1$, the simple update rule in step 8 of Algorithm 2 can be decomposed into

$$\mathbf{w}_v \leftarrow \mathbf{w}_v + \triangle(y, \bar{y})\phi(\mathbf{x}_i) \quad \forall v : v \in \text{anc}(y) - \text{anc}(\bar{y})$$
$$\mathbf{w}_v \leftarrow \mathbf{w}_v - \triangle(y, \bar{y})\phi(\mathbf{x}_i) \quad \forall v : v \in \text{anc}(\bar{y}) - \text{anc}(y)$$

Only the weight vectors of those nodes that are predecessors of y or $\bar{y}$ but not both will be updated. Other nodes are left intact. This strategy is also used in [Dekel *et al.*, 2004] for online multiclass classification. The more severe the loss is incurred, the more dramatic the update will be. Moreover step 8 not only updates the scoring functions of the two classes in question, but also spread the impact to other classes sharing affected ancestors with them.

5 Related Work

Many approaches for hierarchical classification use a decision tree like architecture, associating with each inner node of the taxonomy a classifier that learns to discriminate between the children [Dumais and Chen, 2000; Cesa-Bianchi *et al.*, 2005]. While this offers advantages in terms of modularity, the local optimization of (partial) classifiers at every inner node is unable to reflect a more global objective.

[Cesa-Bianchi *et al.*, 2005] introduces a loss function, called the H-loss, specifically designed to deal with the case of partial and overlapping paths in tree-structured taxonomies. [Cesa-Bianchi *et al.*, 2006] has proposed B-SVM, which also uses the H-loss and uses a decoding scheme that explicitly computes the Bayes-optimal label assignment based on the H-loss and certain conditional independence assumptions about label paths. The loss function we proposed in Eq. (2) exploits the taxonomy in a different way from H-loss, partly because we always convert partial path categories to complete path ones. Our loss function is inspired by real applications like routing and subscription to a taxonomy. So misclassifications are penalized along all ancestors that miss relevant patterns or include irrelevant ones. In H-loss, however, if punishment already occurs to a node, its descendents are not penalized again. In addition, our loss function works with arbitrary taxonomy, not just trees.

[Rousu *et al.*, 2005] applies the Maximum-Margin Markov Networks [Taskar *et al.*, 2003] to hierarchical classification where the taxonomy is regarded as Markov Networks. They propose a simplified version of H-loss that decomposes into contributions of edges so as to marginalize the exponential-sized problem into a polynomial one. In our methods, learning occurs on taxonomy nodes instead of edges. We view the taxonomy as a dependency graph of "is-a" relation.

[Cai and Hofmann, 2004] proposes a hierarchical SVM that decomposes discriminant functions into contributions from different levels of the hierarchy, the same way as this work. Compared to [Cai and Hofmann, 2004], which was restricted to multiclass classification, however, we deal with the additional challenge posed by overlapping categories, i.e. the multi-label problem, for which we employ the category ranking approach proposed in [Schapire and Singer, 2000].

In summary, our major contributions are: 1) Formulate multilabel classification as a global joint learning problem that can take taxonomy information into account. 2) Exploit taxonomy by directly encoding structure in the scoring function used to rank categories 3) Propose a novel taxonomy-based loss function between overlapping categories that is motivated by real applications. 4) Derive a sparse optimization algorithm to efficiently solve the joint SVM formulation. Compared to multiclass classification, sparseness is even more important now that there are more constraints and hence more dual variables. 5) Present a hierarchical Perceptron algorithm that takes advantage of the proposed methods of encoding known taxonomies.

6 Experiments

6.1 Experimental Setup

In this section we compare our hierarchical approaches against their flat counterparts on WIPO-alpha, a data set comprising patent documents.

Taxonomy-derived attributes are employed in the hierarchical approaches. For comparison purpose, t_v in Eq. (1) is set to $1/\sqrt{\text{depth}}$ where depth $\equiv \max_y |\{anc(y)\}|$, so that $\max_y \|\mathbf{\Lambda}(y)\| = 1$ for either the flat or the hierarchical models. In the experiments, hierarchical loss equals half the value in Eq. (3) for historical reason. The hierarchical learning employs this hierarchical loss while the flat one employs the $0-1$ loss. We used a linear kernel and set $C = 1$. Each instance is normalized to have 2-norm of 1. In most experiments, the test performance is evaluated by cross-validation and then macro-averaging across folds.

The measures we used include *one-accuracy*, *average precision*, *ranking loss*, *maximal loss*, and *parent one-accuracy*. The first three are standard metrics for multilabel classification problem [Schapire and Singer, 2000; Elisseff and Weston, 2001]. *One-accuracy* (acc) measures the empirical probability of the top-ranked label being relevant to the document. *Average precision* (prec) measures the quality of la-

section	#cat	#doc	#cat /doc	acc (%)		prec (%)		$\triangle_x$-loss		rloss (%)		pacc (%)	
				flat	hier	flat	hier	flat	hier	flat	hier	flat	hier
A	846	15662	1.44	53.8	**54.2**	56.0	**57.3**	1.66	**1.34**	9.09	**3.85**	70.3	**73.5**
B	1514	17626	1.48	37.3	**37.8**	40.8	**42.5**	2.08	**1.76**	12.6	**5.38**	59.9	**65.0**
C	1152	14841	2.11	45.3	**45.4**	45.5	**45.9**	2.10	**1.68**	10.1	**4.95**	67.8	**73.3**
D	237	2194	1.53	47.3	**48.6**	52.7	**55.0**	1.82	**1.45**	11.7	**7.35**	67.7	**71.6**
E	267	3586	1.34	38.7	**38.7**	44.9	**46.5**	1.99	**1.63**	12.7	**7.44**	63.7	**66.2**
F	862	8011	1.45	36.6	**37.6**	41.3	**43.4**	2.07	**1.69**	11.6	**5.13**	59.7	**65.0**
G	565	12944	1.29	**47.2**	47.2	52.3	**52.8**	1.73	**1.50**	10.5	**5.46**	64.9	**67.0**
H	462	13178	1.35	48.7	**49.2**	55.1	**56.2**	1.63	**1.34**	8.25	**4.15**	66.5	**69.6**

Table 1: SVM experiments on WIPO-alpha corpus. Each row is on categories under the specified top level node (i.e. section). The results are from random 3-fold cross-validation. Better performance is marked in bold face. "#cat/doc" refers to the average number of categories per document, "flat" the flat SVM and "hier" the hierarchical SVM.

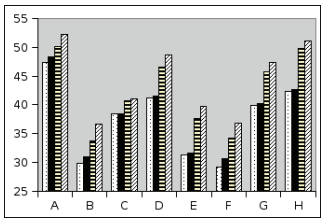

Figure 1: The four columns, from left to right, depict one accuracy for flat and hierarchical Perceptron, and average precision for flat and hierarchical Perceptron.

section	acc (%)		prec (%)		$\triangle_x$-loss		pacc (%)	
	flat	hier	flat	hier	flat	hier	flat	hier
A	14.8	**16.3**	16.8	**20.5**	2.71	**2.28**	39.3	**47.3**
B	12.4	**14.0**	14.0	**17.9**	2.78	**2.39**	40.1	**48.6**
C	14.8	**16.3**	15.6	**18.2**	2.79	**2.23**	46.6	**58.1**
D	19.4	**21.3**	24.0	**27.3**	2.58	**2.06**	49.3	**56.3**
E	14.4	**15.2**	19.9	**22.4**	2.66	**2.19**	45.2	**50.4**
F	13.4	**14.9**	16.6	**20.2**	2.74	**2.25**	41.5	**51.1**
G	11.4	**12.4**	14.9	**18.4**	2.75	**2.40**	35.1	**41.9**
H	15.1	**16.2**	19.2	**22.6**	2.67	**2.12**	40.2	**48.4**

Table 2: SVM experiments on WIPO-alpha corpus with subsampling. Three documents or less are sampled for each category.

bel rankings. Precision is calculated at each position where a positive label occurred, as if all the labels ranked higher than it including itself are predicted as relevant. These precision values are then averaged to obtain average precision. *Ranking loss* (rloss) measures the average fraction of positive label and negative label pairs that are misordered. These metrics are described in details in [Schapire and Singer, 2000].

Maximal loss, denoted by $\triangle_x$, was introduced in Eq. (10). It is measured by the hierarchical loss function. We also evaluate *parent one-accuracy* (pacc), which measures the one-accuracy at the category's parent nodes level.

6.2 Experiments on WIPO-alpha Collection

WIPO-alpha collection comprises patent documents released by the World Intellectual Property Organization (WIPO) [1]. which are classified into IPC categories. IPC is a 4-level hierarchy consisting of sections, classes, subclasses, and groups. The categories in our experiments refer to main groups which are all leaves at the same depth in the hierarchy. Each document is labeled with one primary category as well as any number of secondary categories. Both types of categories are used to form a multi-label corpus. We have performed independent experiments on taxonomies under the 8 top-level sections.

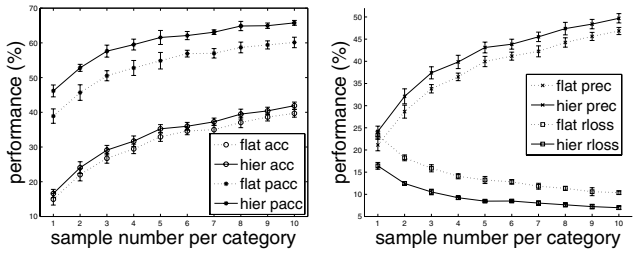

Figure 2: Flat and hierarchical SVM on section D data, with varying training set size. A small number of documents are sampled from each category for training purpose. The learned classifiers are tested on all remaining documents. This is repeated 10 times for each sampling number. The bars depict sample standard deviation.

Document parsing was performed with the Lemur toolkit [2]. Stop words are removed. Stemming is not performed. Word counts from title and claim fields are used as document features. Table 1 summarizes the SVM performance across the sections. The hierarchical SVM significantly outperforms the flat SVM in terms of $\triangle_x$-loss, ranking loss and parent accuracy in each individual setting. This can be attributed not only to the fact that the hierarchical approach explicitly optimizes an upper bound on the $\triangle_x$-loss, but also to the specific hierarchical form of the discriminant function. Moreover, hierarchical SVM often produces higher classification accuracy and average precision with gains being more moderate. To see if the improvement is statistically significant, we conducted 10-fold cross-validation on section E and then paired permutation test. The achieved level of significance is less than 0.08 for one accuracy, and less than 0.005 for the other four measures.

Figure 1 depicts the performance of Perceptron algorithm with the same setting. We allow Perceptron to run until convergence. It takes significantly less time than SVM but reaches lower performance. We observe the hierarchical Perceptron performs better in all cases.

In addition we randomly sampled 3 documents from each

[1] www.wipo.int/ibis/datasets

[2] www.lemurproject.org

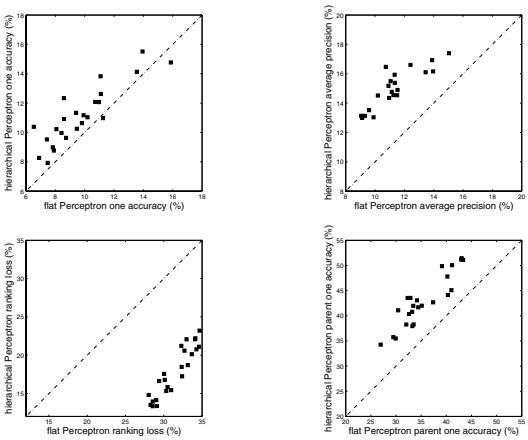

Figure 3: Flat and hierarchical Perceptron on subsampled data.

category to simulate the situation where data are only available in small quantities. The results in Table 2 show that the hierarchical SVM outperforms the flat SVM in all cases. The relative gains are somewhat higher than for the complete training set. Fig 2 demonstrates how the performance gains vary with the size of training data. We observe that hierarchical SVM excels in all runs. The gains appear to be slightly larger when the training set is sparser, except for one accuracy.

Figure 3 compares flat and hierarchical Perceptron with the same subsampling setting as above. Each 3-fold cross validation on a random subset of documents under one section constitutes one sample in the figure, with each section contributing 3 samples. We observe the hierarchical approach helps with one accuracy most times. It always significantly improves average precision, ranking loss and parent accuracy.

7 Conclusions

In this paper a hierarchical loss function has been derived from real applications. We have proposed a large margin architecture for hierarchical multilabel categorization. It extends the strengths of Support Vector Machine classification to take advantage of information about class relationships encoded in a taxonomy. The parameters of the model are fitted by optimizing a joint objective. A variable selection algorithm has been presented to efficiently deal with the resulting quadratic program. We have also proposed a hierarchical Perceptron algorithm that couples the discriminant functions according to the given hierarchy and employs the hierarchical loss in its updating rule. Our experiments show the proposed hierarchical methods significantly outperform the flat methods. Although they aim at reducing the hierarchical loss, the taxonomy-based approaches improve other measures such as one accuracy and average precision. Future work includes more directly working with the hierarchical loss we proposed and comparing our methods with other hierarchical methods.

References

[Cai and Hofmann, 2004] Lijuan Cai and Thomas Hofmann. Hierarchical document categorization with support vector machines. In *Proceedings of the Conference on Information and Knowledge Management (CIKM)*, 2004.

[Cesa-Bianchi *et al.*, 2005] N. Cesa-Bianchi, C. Gentile, A. Tironi, and L. Zaniboni. Incremental algorithms for hierarchical classification. In *Advances in Neural Information Processing Systems*, 2005.

[Cesa-Bianchi *et al.*, 2006] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Hierarchical classification: Combining bayes with svm. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

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

[Dekel *et al.*, 2004] Ofer Dekel, Joseph Keshet, and Yoram Singer. Large margin hierarchical classification. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, 2004.

[Demiriz *et al.*, 2002] Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1–3):225–254, 2002.

[Dumais and Chen, 2000] S. T. Dumais and H. Chen. Hierarchical classification of Web content. In *SIGIR*, pages 256–263, 2000.

[Elisseff and Weston, 2001] André Elisseff and Jason Weston. A kernel method for multi-labelled classification. In *Proceedings of the Neural Information Processing Systems conference (NIPS)*, pages 681–687, 2001.

[Krauth and Mézard, 1987] Werner Krauth and Marc Mézard. Learning algorithms with optimal stability in neural networks. *Journal of Physics A.*, 20(745):L745–L752, 1987.

[McCallum *et al.*, 1998] A. McCallum, R. Rosenfeld, T. Mitchell, and A. Ng. Improving text clasification by shrinkage in a hierarchy of classes. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 359–367, 1998.

[Platt, 1999] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press, 1999.

[Rosenblatt, 1958] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[Rousu *et al.*, 2005] Juho Rousu, Craig Saunders, Sandor Szedmak, and John Shawe-Taylor. Learning hierarchical multi-category text classification models. In *22nd International Conference on Machine Learning (ICML)*, 2005.

[Schapire and Singer, 2000] R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[Taskar *et al.*, 2003] Benjamin Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In *Proceedings of Neural Information Processing Systems conference (NIPS)*, 2003.

[Tsochantardis *et al.*, 2004] I. Tsochantardis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the 21st International Conference on Machine Learning (ICML'04)*, 2004.

[Wang *et al.*, 1999] K. Wang, S. Zhou, and S. C. Liew. Building hierarchical classifiers using class proximity. In *Proceedings of VLDB-99, 25th International Conference on Very Large Data Bases*, pages 363–374, 1999.