Large Margin Hierarchical Classification

Ofer Dekel OFERD@CS.HUJI.AC.IL
Joseph Keshet JKESHET@CS.HUJI.AC.IL
Yoram Singer SINGER@CS.HUJI.AC.IL
School of Computer Science and Engineering, The Hebrew University, Jerusalem, 91904, Israel

Abstract

We present an algorithmic framework for supervised classification learning where the set of labels is organized in a predefined hierarchical structure. This structure is encoded by a rooted tree which induces a metric over the label set. Our approach combines ideas from large margin kernel methods and Bayesian analysis. Following the large margin principle, we associate a prototype with each label in the tree and formulate the learning task as an optimization problem with varying margin constraints. In the spirit of Bayesian methods, we impose similarity requirements between the prototypes corresponding to adjacent labels in the hierarchy. We describe new online and batch algorithms for solving the constrained optimization problem. We derive a worst case loss-bound for the online algorithm and provide generalization analysis for its batch counterpart. We demonstrate the merits of our approach with a series of experiments on synthetic, text and speech data.

1. Introduction

Multiclass categorization problems are concerned with the task of assigning labels to instances where the possible labels come from a predefined set. It is typically assumed that the set of labels has no underlying structure and therefore different classification mistakes are of the same severity. However, in many natural machine learning problems the set of labels is structured and different types of misclassifications should be treated differently. In this paper we focus on a particular type of structure over labels, namely a hierarchical one. In hierarchical classification the set of labels is arranged in a predefined hierarchy which takes the form of a rooted tree. For instance, both the Yahoo! hierarchy and the open directory project (ODP) classify Internet web-sites to categories which reside in a tree. The labels in this tree entertain a special semantic, namely, if an internal vertex in the tree represents some topic then its children will correspond to refinements of this topic. As an example, the vertex in the Yahoo! hierarchy representing the topic *Sports*, points to numerous sub-topics such as *Cricket*, *Curling*, and *Canoeing*. Following the link to the sub-topic *Curling* we find that it points to numerous sub-sub-topics such as *Equipment* and *Tournaments*. Each of these topics, as well as most of the topics in the Yahoo! hierarchy, is associated with a set of webpages. A second notable example is speech phoneme classification. Short speech utterances are typically divided into phonetic classes. Phonetic theory of spoken speech embed the set of phonemes of western languages in a phonetic hierarchy where the phonemes are leaves of the tree and broad phonetic groups, such as vowels and consonants, are internal vertices. In this paper we conduct experiments with these two hierarchical problems.

The problem of hierarchical classification, in particular hierarchical document classification, has been tackled by numerous researchers (see for instance (Koller & Sahami, 1997; McCallum et al., 1998; Weigend et al., 1999; Dumais & Chen, 2000)). Most previous work on hierarchical classification decouples the problem into independent classification problems by assigning and training a classifier for each internal vertex in the hierarchy. To accommodate the semantics imposed by the hierarchical structure, some researchers have imposed statistical similarity constraints between the probabilistic models for adjacent vertices in the hierarchy (e.g. (McCallum et al., 1998)). In probabilistic settings, statistical similarities can be enforced using techniques such as back-off estimates (Katz, 1987) and shrinkage (McCallum et al., 1998).

A significant amount of recent work on classification problems, both binary and multiclass, has been devoted to the theory and application of large margin classifiers. See for instance the book of Vapnik (1998) and the references therein. In this paper, we describe, analyze, and apply a large margin approach to hierarchical classification which

Appearing in *Proceedings of the 21^{st} International Conference on Machine Learning*, Banff, Canada, 2004.

is in the spirit of statistical approaches. As in large margin methods, we associate a vector in a high dimensional space with each label in the hierarchy. We call this vector the *prototype* of the label, and classify instances according to their similarity to the various prototypes. We relax the requirements of correct classification to large margin constraints and attempt to find prototypes that comply with these constraints. In the spirit of Bayesian methods, we impose similarity requirements between the prototypes corresponding to adjacent labels in the hierarchy. The result is an algorithmic solution that may tolerate minor mistakes, such as predicting a sibling of the correct label, but avoids gross errors, such as predicting a vertex in a completely different part of the tree.

Many hierarchical datasets contains a very large number of examples. For instance, the file containing just the ODP hierarchy itself, without the documents, is 250Mb long. To cope with large amounts of data we devise an online algorithm that is both memory efficient and simple to implement. Our algorithmic solution builds on the pioneering work of Warmuth and colleagues. In particular, we generalize and fuse ideas from (Crammer et al., 2003; Herbster, 2001; Kivinen & Warmuth, 1997). These papers discuss online learning of large-margin classifiers. On each round, the online hypothesis is updated such that it complies with margin constraints imposed by the example observed on this round. Along with the margin constraints, the update is required to keep the new classifier fairly close to the previous one. We show that this idea can also be exploited in our setting, resulting in a simple online update which can be used in conjunction with kernel functions. Furthermore, using methods for converting online to batch learning (e.g. (Cesa-Bianchi et al., 2004)), we show that the online algorithm can be used to devise a batch algorithm with theoretical guarantees and good empirical performance.

The paper is organized as follows. In Sec. 2 we formally describe the hierarchical classification problem and establish our notation. Sec. 3 constitutes the algorithmic core of the paper. In this section we describe an online algorithm, called *online Hieron*, for hierarchical classification and prove a worst case bound on its performance. In Sec. 4 we describe a conversion of the online Hieron into a well performing batch algorithm, the *batch Hieron*. In Sec. 5 we conclude the paper with a series of experiments on synthetic data, a text corpus, and speech data.

2. Problem Setting

Let $\mathcal{X} \subseteq \mathbb{R}^n$ be an instance domain and let $\mathcal{Y}$ be a set of labels. In the hierarchical classification setting $\mathcal{Y}$ plays a double role: first, as in traditional multiclass problems, it encompasses the set of labels, namely each instance in $\mathcal{X}$ is associated with a label $v \in \mathcal{Y}$. Second, $\mathcal{Y}$ defines a set of vertices arranged in a rooted tree $\mathcal{T}$. We denote $k = |\mathcal{Y}|$, for concreteness we assume that $\mathcal{Y} = \{0, \ldots, k-1\}$ and let 0 be the root of $\mathcal{T}$.

For any pair of labels $u, v \in \mathcal{Y}$, let $\gamma(u, v)$ denote their distance in the tree. That is, $\gamma(u, v)$ is defined to be the number of edges along the (unique) path from u to v in $\mathcal{T}$. The distance function $\gamma(\cdot, \cdot)$ is in fact a metric over $\mathcal{Y}$ since it is a non-negative function, $\gamma(v, v) = 0$, $\gamma(u, v) = \gamma(v, u)$ and the triangle inequality always holds with equality. As stated above, different classification errors incur different levels of penalty, and in our model this penalty is defined by the tree distance $\gamma(u, v)$. We therefore say that the *tree induced error* incurred by predicting the label v when the correct label is u is $\gamma(u, v)$.

We receive a training set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ of instance-label pairs, where each $\mathbf{x}_i \in \mathcal{X}$ and each $y_i \in \mathcal{Y}$. Our goal is to learn a classification function $f : \mathcal{X} \rightarrow \mathcal{Y}$ which attains a small tree induced error. We focus on classifiers that are of the following form: each label $v \in \mathcal{Y}$ has a matching prototype $\mathbf{W}^v \in \mathbb{R}^n$, where $\mathbf{W}^0$ is fixed to be the zero vector and every other prototype can be any vector in $\mathbb{R}^n$. The classifier f makes its predictions according to the following rule,

$$f(\mathbf{x}) \;=\; \operatorname*{argmax}_{v \in \mathcal{Y}} \mathbf{W}^v \cdot \mathbf{x} \;\; . \tag{1}$$

The task of learning f is reduced to learning $\mathbf{W}^1, \ldots, \mathbf{W}^{k-1}$.

For every label other than the tree root $v \in \{\mathcal{Y} \setminus 0\}$, we denote by $\mathcal{A}(v)$ the parent of v in the tree. Put another way, $\mathcal{A}(v)$ is the vertex adjacent to v which is closer to the tree root 0. We also define $\mathcal{A}^{(i)}(v)$ to be the ith ancestor of v (if such an ancestor exists). Formally, $\mathcal{A}^{(i)}(v)$ is defined recursively as follows,

$$\mathcal{A}^{(0)}(v) = v \quad \text{and} \quad \mathcal{A}^{(i)}(v) \;=\; \mathcal{A}(\mathcal{A}^{(i-1)}(v)) \;\; .$$

For each label $v \in \mathcal{Y}$, define $\mathcal{P}(v)$ to be the set of labels along the path from 0 (the tree root) to v,

$$\mathcal{P}(v) \;=\; \left\{ u \in \mathcal{Y} \;:\; \exists i \;\; u = \mathcal{A}^{(i)}(v) \right\} \;\; .$$

For technical reasons discussed shortly, we prefer not to deal directly with the set of prototypes $\mathbf{W}^0, \ldots, \mathbf{W}^{k-1}$ but rather with the difference between each prototype and the prototype of its parent. Formally, define $\mathbf{w}^0$ to be the zero vector in $\mathbb{R}^n$ and for each label $v \in \mathcal{Y} \setminus 0$, let $\mathbf{w}^v = \mathbf{W}^v - \mathbf{W}^{\mathcal{A}(v)}$. Each prototype now decomposes to the sum

$$\mathbf{W}^v \;=\; \sum_{u \in \mathcal{P}(v)} \mathbf{w}^u \;\; . \tag{2}$$

The classifier f can be defined in two equivalent ways: by setting $\{\mathbf{W}^v\}_{v \in \mathcal{Y}}$ and using Eq. (1), or by setting

$\{\mathbf{w}^v\}_{v\in\mathcal{Y}}$ and using Eq. (2) in conjunction with Eq. (1). Throughout this paper, we often use $\{\mathbf{w}^v\}_{v\in\mathcal{Y}}$ as a synonym for the classification function f. As a design choice, our algorithms require that adjacent vertices in the label tree have similar prototypes. The benefit of representing each prototype $\{\mathbf{W}^v\}_{v\in\mathcal{Y}}$ as a sum of vectors from $\{\mathbf{w}^v\}_{v\in\mathcal{Y}}$ is that adjacent prototypes $\mathbf{W}^v$ and $\mathbf{W}^{\mathcal{A}(v)}$ can be kept close by simply keeping $\mathbf{w}^v = \mathbf{W}^v - \mathbf{W}^{\mathcal{A}(v)}$ small. Sec. 3 and Sec. 4 address the task of learning the set $\{\mathbf{w}^v\}_{v\in\mathcal{Y}}$ from labeled data.

3. An Online Algorithm

In this section we derive and analyze an efficient online learning algorithm named *online Hieron* for the hierarchical classification problem. In online settings, learning takes place in rounds. On round i, an instance, denoted $\mathbf{x}_i$, is presented to the learning algorithm. Hieron maintains a set of prototypes which is constantly updated in accordance with the quality of its predictions. We denote the set of prototypes used to extend the prediction on round i by $\{\mathbf{w}_i^v\}_{v\in\mathcal{Y}}$. Therefore, the prediction of Hieron for $\mathbf{x}_i$ is,

$$\hat{y}_i = \operatorname*{argmax}_{v\in\mathcal{Y}} \mathbf{W}_i^v \cdot \mathbf{x}_i = \operatorname*{argmax}_{v\in\mathcal{Y}} \sum_{u\in\mathcal{P}(v)} \mathbf{w}_i^u \cdot \mathbf{x}_i . \quad (3)$$

Then, the correct label y_i is revealed and the algorithm suffers an instantaneous error. The error that we employ in this paper is the tree induced error. Using the notation above, the error on round i equals $\gamma(y_i, \hat{y}_i)$.

Our analysis, as well as the motivation for the online update that we derive below, assumes that there exists a set of prototypes $\{\boldsymbol{\omega}^v\}_{v\in\mathcal{Y}}$ such that for every instance-label pair $(\mathbf{x}_i, y_i)$ and every $r \neq y_i$ it holds that,

$$\sum_{v\in\mathcal{P}(y_i)} \boldsymbol{\omega}^v \cdot \mathbf{x}_i - \sum_{u\in\mathcal{P}(r)} \boldsymbol{\omega}^u \cdot \mathbf{x}_i \geq \sqrt{\gamma(y_i, r)} . \quad (4)$$

The above difference between the projection onto the prototype corresponding to the correct label and any other prototype is a generalization of the notion of margin employed by multiclass problems (Weston & Watkins, 1999). Put informally, we require that the margin between the correct and each of the incorrect labels be at least the square-root of the tree-based distance between them. The goal of the Hieron algorithm is to find a set of prototypes which fulfills the margin requirement of Eq. (4) while incurring a minimal tree-induced error until such a set is found. However, the tree-induced error is a combinatorial quantity and is thus difficult to minimize directly. We instead use a construction commonly used in large margin classifiers and employ the the convex hinge-loss function

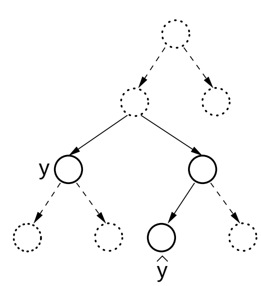

Figure 1. An illustration of the update: only the vertices depicted using solid lines are updated.

$$\ell(\{\mathbf{w}^v\}, \mathbf{x}, y) = \left[\sum_{v\in\mathcal{P}(\hat{y})} \mathbf{w}^v \cdot \mathbf{x} - \sum_{v\in\mathcal{P}(y)} \mathbf{w}^v \cdot \mathbf{x} + \sqrt{\gamma(y,\hat{y})}\right]_+ , \quad (5)$$

where $[z]_+ = \max\{z, 0\}$. In the sequel we show that $\ell^2(\{\mathbf{w}^v\}, \mathbf{x}, y)$ upper bounds $\gamma(y,\hat{y})$ and use this fact to attain a bound on $\sum_{i=1}^m \gamma(y_i, \hat{y}_i)$.

The online Hieron algorithm belongs to the family of *conservative* online algorithms, which update their classification rules only on rounds on which prediction mistakes are made. Let us therefore assume that there was a prediction mistake on round i. We would like to modify the set of vectors $\{\mathbf{w}_i^v\}$ so as to satisfy the margin constraints imposed by the ith example. One possible approach is to simply find a set of vectors that solves the constraints in Eq. (4) (Such a set must exist since we assume that there exists a set $\{\boldsymbol{\omega}_i^v\}$ which satisfies the margin requirements for *all* of the examples.) There are however two caveats in such a greedy approach. The first is that by setting the new set of prototypes to be an arbitrary solution to the constraints imposed by the most recent example we are in danger of forgetting what has been learned thus far. The second, rather technical, complicating factor is that there is no simple analytical solution to Eq. (4). We therefore introduce a simple constrained optimization problem. The objective function of this optimization problem ensures that the new set $\{\mathbf{w}_{i+1}^v\}$ is kept close to the current set while the constraints ensure that the margin requirement for the pair $(y_i, \hat{y}_i)$ is fulfilled by the new vectors. Formally, the new set of vectors is the solution to the following problem,

$$\min_{\{\mathbf{w}^v\}} \frac{1}{2} \sum_{v\in\mathcal{Y}} \|\mathbf{w}^v - \mathbf{w}_i^v\|^2 \quad (6)$$
$$\text{s.t.} \sum_{v\in\mathcal{P}(y_i)} \mathbf{w}^v \cdot \mathbf{x}_i - \sum_{u\in\mathcal{P}(\hat{y}_i)} \mathbf{w}^u \cdot \mathbf{x}_i \geq \sqrt{\gamma(y_i, \hat{y}_i)} .$$

First, note that any vector $\mathbf{w}^v$ corresponding to a vertex v that does not belong to neither $\mathcal{P}(y_i)$ nor $\mathcal{P}(\hat{y}_i)$ does not change due to the objective function in Eq. (6), hence, $\mathbf{w}_{i+1}^v = \mathbf{w}_i^v$. Second, note that if $v \in \mathcal{P}(y_i) \cap \mathcal{P}(\hat{y}_i)$ then the contribution of the $\mathbf{w}^v$ cancels out. Thus, for this case as well we get that $\mathbf{w}_{i+1}^v = \mathbf{w}_i^v$. In summary, the vectors that we need to actually update correspond to the vertices in the set $\mathcal{P}(y_i) \Delta \mathcal{P}(\hat{y}_i)$ where Δ designates the symmetric difference of sets (see also Fig. 1).

To find the solution to Eq. (6) we introduce a Lagrange multiplier α_i, and formulate the optimization problem in

Algorithm 1 Online Hieron

Initialize: $\forall v \in \mathcal{Y} : \mathbf{w}_v^1 = \mathbf{0}$

for $i = 1, 2, \ldots m$

- Receive an instance $\mathbf{x}_i$
- Predict: $\hat{y}_i = \arg\max_{v \in \mathcal{Y}} \sum_{u \in \mathcal{P}(v)} \mathbf{w}_i^u \cdot \mathbf{x}_i$
- Receive the correct label y_i
- Suffer loss: $\ell\left(\{\mathbf{w}_i^v\}, \mathbf{x}_i, y_i\right)$ [see Eq. (5)]
- Update:

$$\begin{aligned} \mathbf{w}_{i+1}^v &= \mathbf{w}_i^v + \alpha_i \mathbf{x}_i \qquad v \in \mathcal{P}(y_i) \backslash \mathcal{P}(\hat{y}_i) \\ \mathbf{w}_{i+1}^v &= \mathbf{w}_i^v - \alpha_i \mathbf{x}_i \qquad v \in \mathcal{P}(\hat{y}_i) \backslash \mathcal{P}(y_i) \end{aligned}$$

where
$$\alpha_i = \frac{\ell\left(\{\mathbf{w}_i^v\}, \mathbf{x}_i, y_i\right)}{\gamma(y_i, \hat{y}_i)\, \|\mathbf{x}_i\|^2}$$

the form of a Lagrangian. We set the derivative of the Lagrangian w.r.t. $\{\mathbf{w}^v\}$ to zero and get,

$$\begin{aligned} \mathbf{w}_{i+1}^v &= \mathbf{w}_i^v + \alpha_i \mathbf{x}_i \qquad v \in \mathcal{P}(y_i) \backslash \mathcal{P}(\hat{y}_i) \\ \mathbf{w}_{i+1}^v &= \mathbf{w}_i^v - \alpha_i \mathbf{x}_i \qquad v \in \mathcal{P}(\hat{y}_i) \backslash \mathcal{P}(y_i) \ . \end{aligned} \tag{7}$$

Since at the optimum the constraint of Eq. (6) is binding we get that,

$$\sum_{v \in \mathcal{P}(y_i)} (\mathbf{w}_i^v + \alpha_i \mathbf{x}_i) \cdot \mathbf{x}_i = \sum_{v \in \mathcal{P}(\hat{y}_i)} (\mathbf{w}_i^v - \alpha_i \mathbf{x}_i) \cdot \mathbf{x}_i + \sqrt{\gamma(y_i, \hat{y}_i)}.$$

Rearranging terms in the above equation and using the definition of the loss from Eq. (5) we get that,

$$\alpha_i \|\mathbf{x}_i\|^2 \, |\mathcal{P}(y_i) \Delta \mathcal{P}(\hat{y}_i)| \ = \ \ell\left(\{\mathbf{w}_i^v\}, x_i, y_i\right) \ .$$

Finally, noting that the cardinality of $\mathcal{P}(y_i) \Delta \mathcal{P}(\hat{y}_i)$ is equal to $\gamma(y_i, \hat{y}_i)$ we get that,

$$\alpha_i = \frac{\ell\left(\{\mathbf{w}_i^v\}, \mathbf{x}_i, y_i\right)}{\gamma(y_i, \hat{y}_i)\, \|\mathbf{x}_i\|^2} \tag{8}$$

The pseudo code of the online learning algorithm is given in Algorithm 1. The following theorem implies that the cumulative loss suffered by online Hieron is bounded as long as there exists a hierarchical classifier which fulfills the margin requirements on all of the examples.

Theorem 1. *Let $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ be a sequence of examples where $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^n$ and $y_i \in \mathcal{Y}$. Assume there exists a set $\{\boldsymbol{\omega}^v : \forall v \in \mathcal{Y}\}$ that satisfies Eq. (4) for all $1 \leq i \leq m$. Then, the following bound holds,*

$$\sum_{i=1}^m \ell^2\left(\{\mathbf{w}_i^v\}, \mathbf{x}_i, y_i\right) \leq \sum_{v \in \mathcal{Y}} \|\boldsymbol{\omega}^v\|^2 \, \gamma_{max} \, R^2$$

where for all i, $\|\mathbf{x}_i\| \leq R$ and $\gamma(y_i, \hat{y}_i) \leq \gamma_{max}$.

Proof. As a technical tool, we denote by $\bar{\boldsymbol{\omega}}$ the concatenation of the vectors in $\{\boldsymbol{\omega}^v\}$, $\bar{\boldsymbol{\omega}} = \left(\boldsymbol{\omega}^0, \ldots, \boldsymbol{\omega}^{k-1}\right)$ and similarly $\bar{\mathbf{w}}_i = \left(\mathbf{w}_i^0, \ldots, \mathbf{w}_i^{k-1}\right)$ for $i \geq 1$. We denote by δ_i the difference between the squared distance $\bar{\mathbf{w}}_i$ from $\bar{\boldsymbol{\omega}}$ and the squared distance of $\bar{\mathbf{w}}_{i+1}$ from $\bar{\boldsymbol{\omega}}$,

$$\delta_i = \|\bar{\mathbf{w}}_i - \bar{\boldsymbol{\omega}}\|^2 - \|\bar{\mathbf{w}}_{i+1} - \bar{\boldsymbol{\omega}}\|^2 \ .$$

We now derive upper and lower bounds on $\sum_{i=1}^m \delta_i$. First, note that by summing over i we obtain,

$$\begin{aligned} \sum_{i=1}^m \delta_i \ &= \ \sum_{i=1}^m \|\bar{\mathbf{w}}_i - \bar{\boldsymbol{\omega}}\|^2 - \|\bar{\mathbf{w}}_{i+1} - \bar{\boldsymbol{\omega}}\|^2 \\ &= \ \|\bar{\mathbf{w}}_1 - \bar{\boldsymbol{\omega}}\|^2 - \|\bar{\mathbf{w}}_m - \bar{\boldsymbol{\omega}}\|^2 \ \leq \ \|\bar{\mathbf{w}}_1 - \bar{\boldsymbol{\omega}}\|^2 \ . \end{aligned}$$

Our initialization sets $\bar{\mathbf{w}}_1 = \mathbf{0}$ and thus we get,

$$\sum_{i=1}^m \delta_i \ \leq \ \|\bar{\boldsymbol{\omega}}\|^2 \ = \ \sum_{v \in \mathcal{Y}} \|\boldsymbol{\omega}^v\|^2 \ . \tag{9}$$

This provides the upper bound on $\sum_i \delta_i$. We next derive a lower bound on each δ_i. The minimizer of the problem defined by Eq. (6) is obtained by projecting $\{\mathbf{w}_i^v\}$ onto the linear constraint corresponding to our margin requirement. The result is a new set $\{\mathbf{w}_{i+1}^v\}$ which in the above notation can be written as the vector $\bar{\mathbf{w}}_{i+1}$. A well known result (see for instance (Censor & Zenios, 1997), Thm. 2.4.1) states that this vector satisfies the following inequality,

$$\|\bar{\mathbf{w}}_i - \bar{\boldsymbol{\omega}}\|^2 - \|\bar{\mathbf{w}}_{i+1} - \bar{\boldsymbol{\omega}}\|^2 \geq \|\bar{\mathbf{w}}_i - \bar{\mathbf{w}}_{i+1}\|^2 \ .$$

Hence, we get that $\delta_i \geq \|\bar{\mathbf{w}}_i - \bar{\mathbf{w}}_{i+1}\|^2$. We can now take into account that $\mathbf{w}_i^v$ is updated if and only if $v \in \mathcal{P}(y_i) \Delta \mathcal{P}(\hat{y}_i)$ to get that,

$$\begin{aligned} \|\bar{\mathbf{w}}_i - \bar{\mathbf{w}}_{i+1}\|^2 \ &= \ \sum_{v \in \mathcal{Y}} \|\mathbf{w}_i^v - \mathbf{w}_{i+1}^v\|^2 \\ &= \sum_{v \in \mathcal{P}(y_i) \Delta \mathcal{P}(\hat{y}_i)} \|\mathbf{w}_i^v - \mathbf{w}_{i+1}^v\|^2 \ . \end{aligned}$$

Plugging Eq. (7) into the above, we get

$$\begin{aligned} \sum_{v \in \mathcal{P}(y_i) \Delta \mathcal{P}(\hat{y}_i)} \|\mathbf{w}_i^v - \mathbf{w}_{i+1}^v\|^2 \ &= \sum_{v \in \mathcal{P}(y_i) \Delta \mathcal{P}(\hat{y}_i)} \alpha_i^2 \, \|x_i\|^2 \\ &= \ |\mathcal{P}(y_i) \Delta \mathcal{P}(\hat{y}_i)| \, \alpha_i^2 \, \|x_i\|^2 \\ &= \ \gamma(y_i, \hat{y}_i) \, \alpha_i^2 \, \|x_i\|^2 \ . \end{aligned}$$

We now use the definition of α_i from Eq. (8) to obtain the lower bound,

$$\delta_i \ \geq \ \frac{\ell^2\left(\{\mathbf{w}_i^v\}, \mathbf{x}_i, y_i\right)}{\gamma(y_i, \hat{y}_i) \|\mathbf{x}_i\|^2} \ .$$

Using the assumptions $\|\mathbf{x}_i\| \leq R$ and $\gamma(y_i, \hat{y}_i) \leq \gamma_{max}$ we can further bound δ_i by,

$$\delta_i \ \geq \ \frac{\ell^2\left(\{\mathbf{w}_i^v\}, \mathbf{x}_i, y_i\right)}{\gamma_{max} \, R^2} \ .$$

Now, summing over all i and comparing the lower bound given above with the upper bound of Eq. (9) we get,

$$\frac{\sum_{t=1}^{m} \ell^2\left(\{\mathbf{w}_i^v\}, \mathbf{x}_i, y_i\right)}{\gamma_{max}\ R^2} \ \leq\ \sum_{t=1}^{m} \delta_i \ \leq\ \sum_{v\in\mathcal{Y}} \|\boldsymbol{\omega}^v\|^2 \ .$$

Multiplying both sides of the inequality above by $\gamma_{max}\ R^2$ gives the desired bound. □

The loss bound of Thm. 1 can be straightforwardly translated into a bound on the tree-induced error as follows. Note that whenever a prediction error occurs ($y_i \neq \hat{y}_i$), then $\sum_{v\in\mathcal{P}(\hat{y}_i)} \mathbf{w}_i^v \cdot \mathbf{x}_i \ \geq\ \sum_{v\in\mathcal{P}(y_i)} \mathbf{w}_i^v \cdot \mathbf{x}_i$. Thus, the hinge-loss defined by Eq. (5) is greater than $\sqrt{\gamma(y_i, \hat{y}_i)}$. Since we suffer a loss only on rounds were prediction errors were made, we get the following corollary.

Corollary 1. *Under the conditions of Thm. 1 the following bound on the cumulative tree-induced error holds,*

$$\sum_{t=1}^{m} \gamma(y_i, \hat{y}_i) \ \leq\ \sum_{v\in\mathcal{Y}} \|\boldsymbol{\omega}^v\|^2\ \gamma_{max}\ R^2 \ . \qquad (10)$$

To conclude the algorithmic part of the paper, we note that Mercer kernels can be easily incorporated into our algorithm. First, rewrite the update as $\mathbf{w}_{i+1}^v = \mathbf{w}_i^v + \alpha_i^v \mathbf{x}_i$ where,

$$\alpha_i^v = \begin{cases} \alpha_i & v \in \mathcal{P}(y_i)\backslash\mathcal{P}(\hat{y}_i) \\ -\alpha_i & v \in \mathcal{P}(\hat{y}_i)\backslash\mathcal{P}(y_i) \\ 0 & \text{otherwise} \end{cases} \ .$$

Using this notation, the resulting hierarchical classifier can be rewritten as,

$$\begin{aligned} f(\mathbf{x}) &= \operatorname*{argmax}_{v\in\mathcal{Y}} \sum_{u\in\mathcal{P}(v)} \mathbf{w}_i^u \cdot \mathbf{x}_i & (11) \\ &= \operatorname*{argmax}_{v\in\mathcal{Y}} \sum_{u\in\mathcal{P}(v)} \sum_{i=1}^{m} \alpha_i^u\, \mathbf{x}_i \cdot \mathbf{x} \ . & (12) \end{aligned}$$

We can replace the inner-products in Eq. (12) with a general kernel operator $K(\cdot,\cdot)$ that satisfies Mercer's conditions (Vapnik, 1998). It remains to show that α_i^v can be computed based on kernel operations whenever $\alpha_i^v \neq 0$. To see this, note that we can rewrite α_i from Eq. (8) as $\alpha_i = [\beta_i]_+ / \eta_i$ where

$$\begin{aligned} \beta_i \ = & \sum_{v\in\mathcal{P}(\hat{y}_i)} \sum_{j<i} \alpha_j^v K(\mathbf{x}_j, \mathbf{x}_i) \ - \\ & \sum_{v\in\mathcal{P}(y_i)} \sum_{j<i} \alpha_j^v K(\mathbf{x}_j, \mathbf{x}_i) + \gamma(y_i, \hat{y}_i) \ , \end{aligned}$$

and $\eta_i = \gamma(y_i, \hat{y}_i)\, K(\mathbf{x}_i, \mathbf{x}_i)$.

4. Batch Learning and Generalization

In the previous section we presented an online algorithm for hierarchical multiclass learning. However, many common hierarchical multiclass tasks fit more naturally in the batch learning setting, where the entire training set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ is available to the learning algorithm in advance. As before, the performance of a classifier f on a given example $(\mathbf{x}, y)$ is evaluated with respect to the tree-induced error $\gamma(y, f(\mathbf{x}))$. In contrast to online learning, where no assumptions are made on the distribution of examples, we now assume that the examples are independently sampled from a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. Our goal is to use S to obtain a hierarchical classifier f which attains a low *expected* tree-induced error, $\mathbb{E}\left[\gamma(y, f(\mathbf{x}))\right]$, where expectation is taken over the random selection of examples from $\mathcal{D}$.

Perhaps the simplest idea is to use the online Hieron algorithm of Sec. 3 as a batch algorithm by applying it to the training set S in an arbitrary order and defining f to be the last classifier obtained by this process. The resulting classifier is the one defined by the vector set $\{\mathbf{w}_{m+1}^v\}_{v\in\mathcal{Y}}$. In practice, this idea works reasonably well, as demonstrated by our experiments (Sec. 5). However, a variation of this idea yields a significantly better classifier with an accompanying generalization bound. First, we slightly modify the online algorithm by selecting $\hat{y}_i$ to be the label which maximizes Eq. (5) instead of selecting $\hat{y}_i$ according to Eq. (3). In other words, the modified algorithm predicts the label which causes it to suffer the greatest loss. This modification is possible since in the batch setting y_i is available to us before $\hat{y}_i$ is generated. It can be easily verified that Thm. 1 and its proof still hold after this modification. S is presented to the modified online algorithm, which generates the set of vectors $\{\mathbf{w}_i^v\}_{i,v}$. Now, for every $v \in \mathcal{Y}$ define

$$\mathbf{w}^v = \frac{1}{m+1} \sum_{i=1}^{m+1} \mathbf{w}_i^v \ , \qquad (13)$$

and let f be the multiclass classifier defined by $\{\mathbf{w}^v\}_{v\in\mathcal{Y}}$ with the standard prediction rule in Eq. (3). We have set the prototype for label v to be the average over all prototypes generated by the online algorithm for label v. We name this approach the *batch Hieron* algorithm. For a general discussion on taking the average online hypothesis see (Cesa-Bianchi et al., 2004).

In our analysis below, we use Eq. (13) to define the classifier generated by the batch Hieron algorithm. However, an equivalent definition can be given which is much easier to implement in practice. As stated in the previous section, each vector $\mathbf{w}_i^v$ can be represented in dual form by

$$\mathbf{w}_i^v = \sum_{j=1}^{i} \alpha_j^v \mathbf{x}_j \ . \qquad (14)$$

As a result, each of the vectors in $\{\mathbf{w}^v\}_{v\in\mathcal{Y}}$ can also be represented in dual form by

$$\mathbf{w}^v = \frac{1}{m+1}\sum_{i=1}^{m+1}(m+2-i)\,\alpha_i^v \mathbf{x}_i \ . \qquad (15)$$

Therefore, the output of the batch Hieron becomes

$$f(\mathbf{x}) \ = \ \operatorname*{argmax}_{v\in\mathcal{Y}} \sum_{u\in\mathcal{P}(v)} \sum_{i=1}^{m+1}(m+2-i)\,\alpha_i^u\,\mathbf{x}_i\cdot\mathbf{x}$$

Theorem 2. *Let $S=\{(\mathbf{x}_i,y_i)\}_{i=1}^m$ be a training set sampled i.i.d from the distribution $\mathcal{D}$. Let $\{\mathbf{w}^v\}_{v\in\mathcal{Y}}$ be the vectors obtained by applying batch Hieron to S, and let f denote the classifier they define. Assume there exist $\{\boldsymbol{\omega}^v\}_{v\in\mathcal{Y}}$ that define a classifier $f^\star$ which attains zero loss on S. Furthermore, assume that R, B and $\gamma_{\max}$ are constants such that $\|\mathbf{x}\|\le R$ for all $\mathbf{x}\in\mathcal{X}$, $\|\boldsymbol{\omega}^v\|\le B$ for all $v\in\mathcal{Y}$, $\gamma(\cdot,\cdot)$ is bounded by $\gamma_{\max}$. Then with probability of at least $1-\delta$,*

$$\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\gamma(y,f(\mathbf{x}))\right] \ \le \ \frac{\mathcal{L}+\lambda}{m+1}+\lambda\sqrt{\frac{2\log(1/\delta)}{m}} \ ,$$

where $\mathcal{L}=\sum_{i=1}^m \ell^2(\{\mathbf{w}_i^v\},\mathbf{x}_i,y_i)$ and $\lambda=kB^2R^2\gamma_{\max}$.

Proof. For any example $(\mathbf{x},y)$ it holds that $\gamma(y,f(\mathbf{x}))\le \ell^2(\{\mathbf{w}^v\},\mathbf{x},y)$, as discussed in the previous section. Using this fact, it suffices to prove a bound on $\mathbb{E}\left[\ell^2(\{\mathbf{w}^v\},\mathbf{x},y)\right]$ to prove the theorem. By definition, $\ell^2\left(\{\mathbf{w}^v\},\mathbf{x},y\right)$ equals

$$\left[\sum_{v\in\mathcal{P}(f(\mathbf{x}))}\mathbf{w}^v\cdot\mathbf{x} \ - \ \sum_{v\in\mathcal{P}(y)}\mathbf{w}^v\cdot\mathbf{x}+\sqrt{\gamma(y,f(\mathbf{x}))}\right]_+^2 .$$

By construction, $\mathbf{w}^v = \frac{1}{m+1}\sum_{i=1}^{m+1}\mathbf{w}_i^v$. Therefore this loss can be rewritten as

$$\left[\frac{1}{m+1}\sum_{i=1}^{m+1}\left(\sum_{v\in\mathcal{P}(f(\mathbf{x}))}\mathbf{w}_i^v-\sum_{v\in\mathcal{P}(y)}\mathbf{w}_i^v\right)\cdot\mathbf{x}+C\right]_+^2 ,$$

where $C=\sqrt{\gamma(y,f(\mathbf{x}))}$. Using the convexity of the function $g(a)=[a+C]_+^2$ together with Jensen's inequality, we can upper bound the above by

$$\frac{1}{m+1}\sum_{i=1}^{m+1}\left[\sum_{v\in\mathcal{P}(f(\mathbf{x}))}\mathbf{w}_i^v\cdot\mathbf{x}-\sum_{v\in\mathcal{P}(y)}\mathbf{w}_i^v\cdot\mathbf{x}+C\right]_+^2 .$$

Let $\ell_{\max}(\{\mathbf{w}^v\},\mathbf{x},y)$ denote the maximum of Eq. (5) over all $\hat{y}\in\mathcal{Y}$. We now use $\ell_{\max}$ to bound each of the summands in the expression above and obtain the bound,

$$\ell^2(\{\mathbf{w}^v\},\mathbf{x},y) \ \le \ \frac{1}{m+1}\sum_{i=1}^{m+1} \ \ell_{\max}^2(\{\mathbf{w}_i^v\},\mathbf{x},y) \ .$$

Taking expectations on both sides of this inequality, we get

$$\mathbb{E}\left[\ell^2\left(\{\mathbf{w}^v\},\mathbf{x},y\right)\right]\le\frac{1}{m+1}\sum_{i=1}^{m+1}\mathbb{E}\left[\ell_{\max}^2\left(\{\mathbf{w}_i^v\},\mathbf{x},y\right)\right] . \qquad (16)$$

Recall that the modified online Hieron suffers a loss of $\ell_{\max}(\{\mathbf{w}_i^v\},\mathbf{x}_i,y_i)$ on round i. As a direct consequence of Azuma's large deviation bound (see for instance Thm. 1 in (Cesa-Bianchi et al., 2004)), the sum $\sum_{i=1}^m\mathbb{E}\left[\ell_{\max}^2(\{\mathbf{w}_i^v\},\mathbf{x},y)\right]$ is bounded above with probability of at least $1-\delta$ by,

$$\mathcal{L}+m\lambda\sqrt{\frac{2\log(1/\delta)}{m}} \ ,$$

As previously stated, Thm. 1 also holds for the modified online update. It can therefore be used to obtain the bound $\ell_{\max}^2(\{\mathbf{w}_{m+1}^v\},\mathbf{x},y)\le\lambda$ and to conclude that,

$$\sum_{i=1}^{m+1}\mathbb{E}\left[\ell_{\max}^2(\{\mathbf{w}_i^v\},\mathbf{x},y)\right]\le\mathcal{L}+\lambda+m\lambda\sqrt{\frac{2\log(1/\delta)}{m}} \ .$$

Dividing both sides of the above inequality by $m+1$, we have obtained an upper bound on the right hand side of Eq. (16), which gives us the desired bound on $\mathbb{E}\left[\ell^2(\{\mathbf{w}^v\},\mathbf{x},y)\right]$. □

Thm. 2 is a data *dependent* error bound as it depends on $\mathcal{L}$. We would like to note in passing that a data *independent* bound on $\mathbb{E}[\gamma(y,f(\mathbf{x}))]$ can also be obtained by combining Thm. 2 with Thm. 1. As stated above, Thm. 1 holds for the modified version of the online Hieron described above. The data independent bound is derived by replacing $\mathcal{L}$ in Thm. 2 with its upper bound given in Thm. 1.

5. Experiments

We begin this section with a comparison of the online and batch variants of Hieron with standard multiclass classifiers which are oblivious to the hierarchical structure of the label set. We conducted experiments with a synthetic dataset, a dataset of web homepages taken from the Open Directory Project (ODP/DMOZ), and a data set of phonemes extracted from continuous natural speech. The synthetic data was generated as follows: we constructed a symmetric trinary tree of depth 4 and used it as the hierarchical structure. This tree contains 121 vertices which are the labels of our multiclass problem. We then set $\mathbf{w}^0,\ldots,\mathbf{w}^{120}$ to be some orthonormal set in $\mathbb{R}^{121}$, and defined the 121 label prototypes to be $\mathbf{W}^v=\sum_{u\in\mathcal{P}(v)}\mathbf{w}^u$. We generated 100 train instances and 50 test instances for each label. Each example was generated by setting $(\mathbf{x},y)=(\mathbf{W}^y+\boldsymbol{\eta},y)$, where $\boldsymbol{\eta}$ is a vector of Gaussian noise generated by randomly drawing each of its coordinates from a Gaussian distribution with expectation 0 and variance 0.16. This dataset

Table 1. Online Hieron results.

Data set	Tree induced error	Multiclass error
Synthetic data (tree)	**0.83**	**44.5**
Synthetic data (flat)	1.35	51.1
DMOZ (tree)	**3.62**	75.9
DMOZ (flat)	4.10	**75.4**
Phonemes (tree)	**1.64**	40.0
Phonemes (flat)	1.72	**39.7**

Table 2. Batch Hieron results.

Data set	Tree induced err.		Multiclass err.	
	Last	Batch	Last	Batch
Synthetic data (tree)	**0.04**	**0.05**	**4.1**	**5.0**
Synthetic data (flat)	0.14	0.11	10.8	8.6
Synthetic data (greedy)	0.57	0.52	37.4	34.9
DMOZ (tree)	**3.12**	**2.60**	**69.8**	62.6
DMOZ (flat)	3.56	2.89	70.2	**61.6**
DMOZ (greedy)	3.86	3.14	81.8	74.3
Phonemes (tree)	**1.88**	**1.30**	**48.0**	**40.6**
Phonemes (flat)	2.01	1.41	48.8	41.8
Phonemes (greedy)	3.22	2.48	73.9	58.2

is referred to as *synthetic* in the figures and tables appearing in this section.

The second dataset is a set of 8576 Internet homepages collected from the World Wide Web. We used the Open Directory Project (ODP/DMOZ) to construct the label hierarchy. DMOZ is a comprehensive human-edited directory of the web which defines a huge hierarchical structure over thousands of webpage topics. We extracted a subset of 316 vertices from the hierarchy, arranged in a tree of maximal depth 8. The top level topics are *Arts*, *Shopping*, *Sports* and *Computers*. An example of a typical path in our hierarchy is: *Top* → *Computers* → *Hardware* → *Peripherals* → *Printers* → *Supplies* → *Laser Toner*. Most of the internal nodes and all of the leaves in the hierarchy point to WWW documents. We used a bag-of-words variant to represent each document and used 4-fold cross validation to evaluate the performance of the algorithms on this dataset.

The last dataset used in our experiments is a corpus of continuous natural speech for the task of phoneme classification. It has been previously established that phonemes form an acoustic hierarchy (Deller et al., 1987; Rabiner & Schafer, 1978). For instance, the phoneme /b/ (as in *be*) is acoustically closer to the phoneme /d/ (as in *dad*), than for instance to the phoneme /ow/ (as in *oat*). In general, stop consonants are acoustically similar to each other and rather dissimilar to vowels. The data we used is a subset of the TIMIT acoustic-phonetic dataset, which is a phonetically transcribed corpus of high quality continuous speech spoken by North American speakers (Lemel et al., 1986). Mel-frequency cepstrum coefficients (MFCC) along with their first and the second derivatives were extracted from the speech in a standard way, based on the ETSI standard for distributed speech recognition (ETSI, 2000) and each feature vector was generated from 5 adjacent MFCC vectors (with overlap). The TIMIT corpus is divided into a training set and a test set in such a way that no speakers from the training set appear in the test set (speaker independent). We randomly selected 2000 training instances and 500 test instances per each of the 40 phonemes.

We trained and tested the online and batch versions of Hieron on all three datasets. To demonstrate the benefits of exploiting the label hierarchy, we also trained and evaluated standard multiclass predictors which ignore the hierarchical structure. These classifiers were trained using Hieron but with a "flattened" version of the label hierarchy. The (normalized) cumulative tree-induced error and the percentage of multiclass errors for each experiment are summarized in Table 1 (online experiments) and Table 2 (batch experiments). Rows marked by *tree* refer to the performance of the Hieron algorithm, while rows marked by *flat* refer to the performance of the classifier trained without knowledge of the hierarchy. The results clearly indicate that exploiting the hierarchical structure is beneficial in achieving low tree-induced errors. In all experiments, both online and batch, Hieron achieved lower tree-induced error than its "flattened" counterpart. Furthermore, in many cases the multiclass error of Hieron is also lower than the error of the corresponding multiclass predictor, although the latter was explicitly trained to minimize the error. This behavior exemplifies that employing a hierarchical label structure may prove useful even when the goal is not necessarily the minimization of some tree-based error.

The last examination of results further demonstrates that Hieron tends to tolerate small tree-induced errors while avoiding large ones. In Fig. 2 we depict the differences between the error rate of batch Hieron and the error rate of a standard multiclass predictor. Each bar corresponds to a different value of $\gamma(y, \hat{y})$, starting from the left with a value of 1 and ending on the right with the largest possible value of $\gamma(y, \hat{y})$. It is clear from the figure that Hieron tends to make "small" errors by predicting the parent or a sibling of the correct vertex. On the other hand Hieron seldom chooses a vertex which is in an entirely different part of the tree, thus avoiding large tree induced errors. Examining the results for the DMOZ dataset, we see that for $\gamma(y, \hat{y}) < 6$ the frequency of errors of Hieron is greater than that of the multiclass predictor while Hieron beats its multiclass counterpart for $\gamma(y, \hat{y}) \geq 6$. In the phoneme classification task, Hieron seldom extends a prediction $\hat{y}$ such that $\gamma(y, \hat{y}) = 9$ while the errors of the multiclass predictor are uniformly distributed.

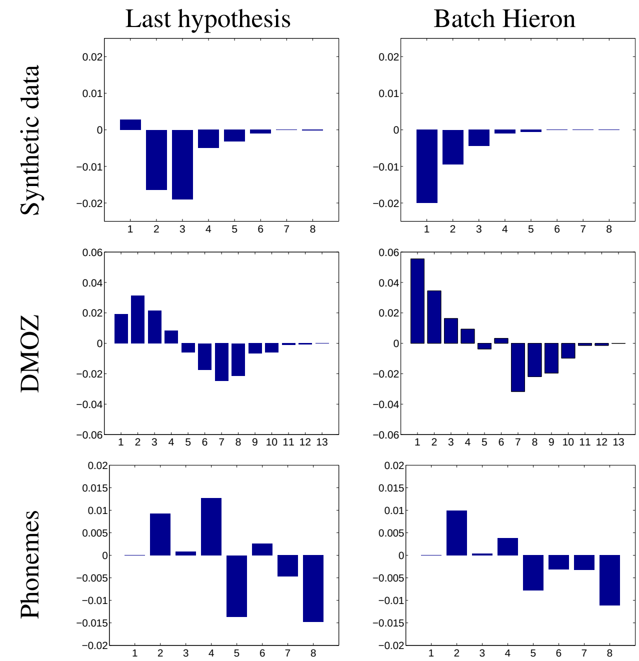

Figure 2. The distribution of the tree induced-error for each dataset used in the experiments. Each bar corresponds to the difference between the error of Hieron minus the error of a multiclass predictor.

We conclude the experiments with a comparison of Hieron with a common construction of hierarchical classifiers (see for instance (Koller & Sahami, 1997)), where separate classifiers are learned and applied at each internal vertex of the hierarchy independently. To compare the two approaches, we learned a multiclass predictor at each internal vertex of the tree hierarchy. Each such classifier routes an input instance to one of its children. Formally, for each internal vertex v of $\mathcal{T}$ we trained a classifier f_v using the training set $S_v = \{(\mathbf{x}_i, u_i) | u_i \in \mathcal{P}(y_i), v = \mathcal{A}(u_i), (\mathbf{x}_i, y_i) \in S\}$. Given a test instance $\mathbf{x}$, its predicted label is the leaf $\hat{y}$ such that for each $u \in \mathcal{P}(\hat{y})$ and its parent v we have $f_v(\mathbf{x}) = u$. In other words, to cast a prediction we start with the root vertex and move towards one of the leaves by progressing from a vertex v to $f_v(\mathbf{x})$. We refer to this hierarchical classification model in Table 2 simply as *greedy*. In all of the experiments, batch Hieron clearly outperforms greedy. This experiment underscores the usefulness of our approach which makes global decisions in contrast to the local decisions of the greedy construction. Indeed, any single prediction error at any of the vertices along the path to the correct label will impose a global prediction error.

References

Censor, Y., & Zenios, S. (1997). *Parallel optimization: Theory, algorithms, and applications*. Oxford University Press, New York, NY, USA.

Cesa-Bianchi, N., Conconi, A., & C.Gentile (2004). On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*. (to appear).

Crammer, K., Dekel, O., Shalev-Shwartz, S., & Singer, Y. (2003). Online passive aggressive algorithms. *Advances in Neural Information Processing Systems 16*.

Deller, J., Proakis, J., & Hansen, J. (1987). *Discrete-time processing of speech signals*. Prentice-Hall.

Dumais, S. T., & Chen, H. (2000). Hierarchical classification of Web content. *Proceedings of SIGIR-00* (pp. 256–263).

ETSI (2000). ETSI Standard, ETSI ES 201 108.

Herbster, M. (2001). Learning additive models online with fast evaluating kernels. *Proceedings of the Fourteenth Annual Conference on Computational Learning Theory* (pp. 444–460).

Katz, S. (1987). Estimation of probabilities from sparse-data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing (ASSP)*, *35*, 400–40.

Kivinen, J., & Warmuth, M. K. (1997). Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, *132*, 1–64.

Koller, D., & Sahami, M. (1997). Hierarchically classifying docuemnts using very few words. *Machine Learning: Proceedings of the Fourteenth International Conference* (pp. 171–178).

Lemel, L., Kassel, R., & Seneff, S. (1986). *Speech database development: Design and analysis* Report no. SAIC-86/1546). Proc. DARPA Speech Recognition Workshop.

McCallum, A. K., Rosenfeld, R., Mitchell, T. M., & Ng, A. Y. (1998). Improving text classification by shrinkage in a hierarchy of classes. *Proceedings of ICML-98* (pp. 359–367).

Rabiner, L. R., & Schafer, R. W. (1978). *Digital processing of speech signals*. Prentice-Hall.

Vapnik, V. N. (1998). *Statistical learning theory*. Wiley.

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

Weston, J., & Watkins, C. (1999). Support vector machines for multi-class pattern recognition. *Proceedings of the Seventh European Symposium on Artificial Neural Networks*.