An Online Algorithm for Hierarchical Phoneme Classification

Ofer Dekel, Joseph Keshet, and Yoram Singer

{oferd,jkeshet,singer}@cs.huji.ac.il
School of Computer Science and Engineering,
The Hebrew University, Jerusalem, 91904, Israel

Abstract. We present an algorithmic framework for phoneme classification where the set of phonemes is organized in a predefined hierarchical structure. This structure is encoded via a rooted tree which induces a metric over the set of phonemes. Our approach combines techniques from large margin kernel methods and Bayesian analysis. Extending the notion of large margin to hierarchical classification, we associate a prototype with each individual phoneme and with each phonetic group which corresponds to a node in the tree. We then formulate the learning task as an optimization problem with margin constraints over the phoneme set. In the spirit of Bayesian methods, we impose similarity requirements between the prototypes corresponding to adjacent phonemes in the phonetic hierarchy. We describe a new online algorithm for solving the hierarchical classification problem and provide worst-case loss analysis for the algorithm. We demonstrate the merits of our approach by applying the algorithm to synthetic data and as well as speech data.

1 Introduction

Phonemes classification is the task of deciding what is the phonetic identity of a (typically short) speech utterance. Work in speech recognition and in particular phoneme classification typically imposes the assumption that different classification errors are of the same importance. However, since the set of phoneme are embedded in a hierarchical structure some errors are likely to be more tolerable than others. For example, it seems less severe to classify an utterance as the phoneme /oy/ (as in *boy*) instead of /ow/ (as in *boat*), than predicting /w/ (as in *way*) instead of /ow/. Furthermore, often we cannot extended a high-confidence prediction for a given utterance, while still being able to accurately identify the phonetic group of the utterance. In this paper we propose and analyze a hierarchal model for classification that imposes a notion of "severity" of prediction errors which is in accordance with a pre-defined hierarchical structure.

Phonetic theory of spoken speech embeds the set of phonemes of western languages in a phonetic hierarchy where the phonemes constitute the leaves of the tree while broad phonetic groups, such as vowels and consonants, correspond to internal vertices. Such phonetic trees were described in [1, 2]. Motivated by this phonetic structure we propose a hierarchical model (depicted in Fig. 1) that incorporates the notion of the similarity (and analogously dissimilarity) between the phonemes and between phonetic groups and employs this notion in the learning procedure we describe and analyze below.

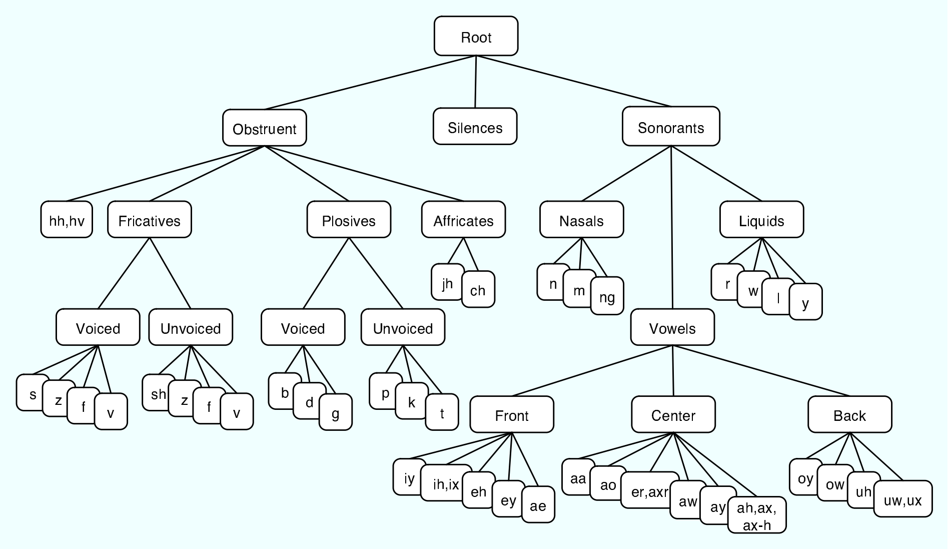

Fig. 1. The phonetic tree of American English.

Most of the previous work on phoneme classification sidestepped the hierarchical phonetic structure (see for instance, [3, 4]). Salomon [5] used a hierarchical clustering algorithm for phoneme classification. His algorithm generates a binary tree which is then used for constructing a phonetic classifier that employs multiple binary support vector machines (SVM). However, this construction was designed for efficiency reasons rather than for capturing the hierarchical phonetic structure. The problem of hierarchical classification in machine learning, in particular hierarchical document classification, was addressed by numerous researchers (see for instance [6–9]). Most previous work on hierarchical classification decoupled the problem into independent classification problems by assigning and training a classifier at each internal vertex in the hierarchy. To incorporate the semantics relayed by the hierarchical structure, few researchers imposed statistical similarity constraints between the probabilistic models for adjacent vertices in the hierarchy (e.g. [7]). In probabilistic settings, statistical similarities can be enforced using techniques such as back-off estimates [10] and shrinkage [7].

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 [11] and the references therein. In this paper, we describe, analyze, and apply a large margin approach to hierarchical classification which is in the spirit of statistical approaches. As in large margin methods, we associate a vector in a high dimensional space with each phoneme or phoneme group in the hierarchy. We call this vector the *prototype* of the phoneme or the phoneme group, and classify feature vectors 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 phonemes in the hierarchy. The result is an algorithmic solution that may tolerate minor mistakes, such as predicting a sibling of the correct phoneme, but avoids gross errors, such as predicting a vertex in a completely different part of the tree.

Speech corpora typically contain a very large number of examples. 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 [12–14]. 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. [15]), we show that the online algorithm can be used to devise a batch algorithm with good empirical performance.

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

2 Problem Setting

Let $\mathcal{X} \subseteq \mathbb{R}^n$ be an acoustic features domain and let $\mathcal{Y}$ be a set of phonemes and phoneme groups. In the hierarchical classification setting $\mathcal{Y}$ plays a double role: first, as in traditional multiclass problems, it encompasses the set of phonemes, namely each feature vector in $\mathcal{X}$ is associated with a phoneme $v \in \mathcal{Y}$. Second, $\mathcal{Y}$ defines a set of vertices, i.e., the phonemes and the phoneme groups, 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 phonemes $u, v \in \mathcal{Y}$, 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 phoneme or the phoneme group v when the correct phoneme is u is $\gamma(u,v)$.

We receive a training set $S=\{(\mathbf{x}_i,y_i)\}_{i=1}^m$ of feature vector-phoneme 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 phoneme $v\in\mathcal{Y}$ 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 phoneme or phoneme group 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 phoneme or phoneme group $v\in\mathcal{Y}$, define $\mathcal{P}(v)$ to be the set of phoneme groups 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 phoneme or phoneme group $v\in\mathcal{Y}\setminus 0$, 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 phonetic 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 supervised data.

3 An Online Algorithm

In this section we derive and analyze an efficient online learning algorithm for the hierarchical phoneme classification problem. In online settings, learning takes place in rounds. On round i, a feature vector, denoted $\mathbf{x}_i$, is presented to the learning algorithm. The algorithm 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 predicted phoneme or phoneme group of the algorithm for $\mathbf{x}_i$ is,

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

Then, the correct phoneme 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 feature vector-phoneme pair $(\mathbf{x}_i, y_i)$ and every $r \neq y_i$ it holds that,

$$\sum_{v\in\mathcal{P}(y_i)} \boldsymbol{\omega}^v \cdot \mathbf{x}_i - \sum_{u\in\mathcal{P}(r)} \boldsymbol{\omega}^u \cdot \mathbf{x}_i \geq \sqrt{\gamma(y_i, r)} \ . \tag{3}$$

The above difference between the projection onto the prototype corresponding to the correct phoneme and any other prototype is a generalization of the notion of margin employed by multiclass problems in machine learning literature [16]. Put informally, we require that the margin between the correct and each of the incorrect phonemes and phoneme groups be at least the square-root of the tree-based distance between them. The goal of the algorithm is to find a set of prototypes which fulfills the margin requirement of Eq. (3) 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

$$\ell\left(\{\mathbf{w}_i^v\}, \mathbf{x}_i, y_i\right) = \left[\sum_{v\in\mathcal{P}(\hat{y}_i)} \mathbf{w}_i^v \cdot \mathbf{x}_i - \sum_{v\in\mathcal{P}(y_i)} \mathbf{w}_i^v \cdot \mathbf{x}_i + \sqrt{\gamma(y_i, \hat{y}_i)}\right]_+ \ , \tag{4}$$

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

The online 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. (3) (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. (3). 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,

$$\begin{aligned} &\min_{\{\mathbf{w}^v\}} \ \frac{1}{2}\sum_{v\in\mathcal{Y}} \|\mathbf{w}^v - \mathbf{w}_i^v\|^2 \\ &\text{s.t.} \sum_{v\in\mathcal{P}(y_i)} \mathbf{w}^v\cdot\mathbf{x}_i \ - \sum_{u\in\mathcal{P}(\hat{y}_i)} \mathbf{w}^u\cdot\mathbf{x}_i \geq \sqrt{\gamma(y_i,\hat{y}_i)}\ . \end{aligned} \tag{5}$$

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. (5), 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. 2).

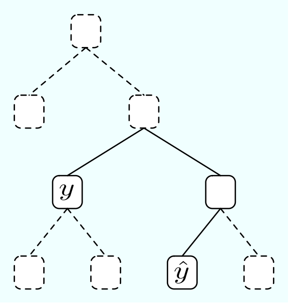

Fig. 2. An illustration of the update: only the vertices depicted using solid lines are updated.

To find the solution to Eq. (5) we introduce a Lagrange multiplier α_i, and formulate the optimization problem in the form of a Lagrangian. We set the derivative of the Lagrangian w.r.t. $\{\mathbf{w}^v\}$ to zero and get,

$$\mathbf{w}_{i+1}^v = \mathbf{w}_i^v + \alpha_i\mathbf{x}_i \quad v\in\mathcal{P}(y_i)\backslash\mathcal{P}(\hat{y}_i) \tag{6}$$

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

Since at the optimum the constraint of Eq. (5) 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. (4) 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,

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

```
INITIALIZE: $\forall v \in \mathcal{Y} : \mathbf{w}_v^1 = \mathbf{0}$
FOR    $t = 1, 2, \ldots, m$
  – Receive acoustic feature vector $\mathbf{x}_i$
  – Predict phoneme or phoneme group:
        $\hat{y}_i = \arg\max_{v \in \mathcal{Y}} \sum_{u \in \mathcal{P}(v)} \mathbf{w}_i^u \cdot \mathbf{x}_i$
  – Receive the correct phoneme $y_i$
  – Suffer loss: $\ell\left(\{\mathbf{w}_i^v\}, \mathbf{x}_i, y_i\right)$    [see Eq. (4)]
  – Update:
        $\mathbf{w}_{i+1}^v = \mathbf{w}_i^v + \alpha_i \mathbf{x}_i \quad v \in \mathcal{P}(y_i) \backslash \mathcal{P}(\hat{y}_i)$
        $\mathbf{w}_{i+1}^v = \mathbf{w}_i^v - \alpha_i \mathbf{x}_i \quad v \in \mathcal{P}(\hat{y}_i) \backslash \mathcal{P}(y_i)$
    where
        $\alpha_i = \frac{\ell\left(\{\mathbf{w}_i^v\}, \mathbf{x}_i, y_i\right)}{\gamma(y_i, \hat{y}_i) \, \|\mathbf{x}_i\|^2}$
```

Fig. 3. Online hierarchical phoneme classification algorithm.

The pseudo code of the online algorithm is given in Fig. 3. The following theorem implies that the cumulative loss suffered by the online algorithm is bounded as long as there exists a hierarchical phoneme 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. (3) 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 $\bar{\boldsymbol{\omega}}$,

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

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. (5) 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 [17], 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 $v \in \mathcal{P}(y_i) \mathit{\Delta} \mathcal{P}(\hat{y}_i)$ to get that,

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

Plugging Eqs. (6-7) into the above equation, we get

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

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

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

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

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

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 γ_{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. (4) 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,*

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

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,

$$f(\mathbf{x}) = \operatorname*{argmax}_{v \in \mathcal{Y}} \sum_{u \in \mathcal{P}(v)} \mathbf{w}_i^u \cdot \mathbf{x}_i \tag{11}$$

$$= \operatorname*{argmax}_{v \in \mathcal{Y}} \sum_{u \in \mathcal{P}(v)} \sum_{i=1}^{m} \alpha_i^u \, \mathbf{x}_i \cdot \mathbf{x} \quad . \tag{12}$$

We can replace the inner-products in Eq. (12) with a general kernel operator $K(\cdot, \cdot)$ that satisfies Mercer's conditions [11]. 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

$$\alpha_i = \frac{\left[\sum_{v \in \mathcal{P}(\hat{y}_i)} \sum_{j<i} \alpha_j^v K(\mathbf{x}_j, \mathbf{x}_i) - \sum_{v \in \mathcal{P}(y_i)} \sum_{j<i} \alpha_j^v K(\mathbf{x}_j, \mathbf{x}_i) + \gamma(y_i, \hat{y}_i)\right]_+}{\gamma(y_i, \hat{y}_i) \, K(\mathbf{x}_i, \mathbf{x}_i)} \; . \tag{13}$$

4 A Batch Conversion

In the previous section we presented an online algorithm for hierarchical phoneme classification. Often, the entire training set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ is available to the learning algorithm in advance, the batch setting is more natural. 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, in batch settings it is assumed that the examples are independently sampled from a distribution $\mathcal{D}$

over $\mathcal{X} \times \mathcal{Y}$. Therefore, our goal is to use S to obtain a hierarchical classifier f which attains a low *expected* tree-induced error, $\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\gamma(y, f(\mathbf{x}))\right]$, where expectation is taken over the random selection of examples from $\mathcal{D}$.

Perhaps the simplest idea is to use the online algorithm of Sec. 3 as a batch algorithm by applying it to the training set S in some 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 $\{\mathbf{w}^v_{m+1}\}_{v\in\mathcal{Y}}$. In practice, this idea works reasonably well, as demonstrated by our experiments (Sec. 5). However, a slight modification of this idea yields a significantly better classifier with an accompanying bound on expected tree-induced error. For every $v \in \mathcal{Y}$, define $\mathbf{w}^v = \frac{1}{m+1}\sum_{i=1}^{m+1}\mathbf{w}^v_i$, where $\{\mathbf{w}^v_i | 1 \leq i \leq m+1,\ v \in \mathcal{Y}\}$ is the set of vectors generated by the online algorithm when it is applied to S. Now, let f be the classifier defined by $\{\mathbf{w}^v\}_{v\in\mathcal{Y}}$. In words, we have defined the prototype for phoneme or phoneme group v to be the average over all prototypes generated by the online algorithm for v. For a general discussion on taking the average online hypothesis see [15].

5 Experiments

We begin this section with a comparison of the online algorithm and batch algorithm with standard multiclass classifiers which are oblivious to the hierarchical structure of the phoneme set. We conducted experiments with a synthetic dataset 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 "phonemes" 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 prototypes to be $\mathbf{W}^v = \sum_{u\in\mathcal{P}(v)} \mathbf{w}^u$. We generated 100 train vectors and 50 test vectors for each "phoneme". 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 is referred to as *synthetic* in the figures and tables appearing in this section. We didn't use any kernel on this dataset in any of the experiments.

The second dataset used in our experiments is a corpus of continuous natural speech for the task of phoneme classification. 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 [18]. 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 [19] 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 features vectors and 500 test feature vectors per each of the 40 phonemes. We normalized the data to have zero mean and unit variance and used an RBF kernel with $\sigma = 0.5$ in all the experiment with this dataset.

Table 1. Online algorithm results.

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

Table 2. Batch algorithm results.

Data set	Tree induced		Multiclass	
	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
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

We trained and tested the online and batch versions of our algorithm on the two datasets. To demonstrate the benefits of exploiting the hierarchal structure, we also trained and evaluated standard multiclass predictors which ignore the structure. These classifiers were trained using the algorithm but with a "flattened" version of the phoneme 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 algorithm train with knowledge of the hierarchical structure, 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, the hierarchical phoneme classifier achieved lower tree-induced error than its "flattened" counterpart. Furthermore, in most of the experiments the multiclass error of the algorithm 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 phoneme structure may prove useful even when the goal is not necessarily the minimization of some tree-based error.

Further examination of results demonstrates that the hierarchical phoneme classifier tends to tolerate small tree-induced errors while avoiding large ones. In Fig. 4 we depict the differences between the error rate of the batch algorithm 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

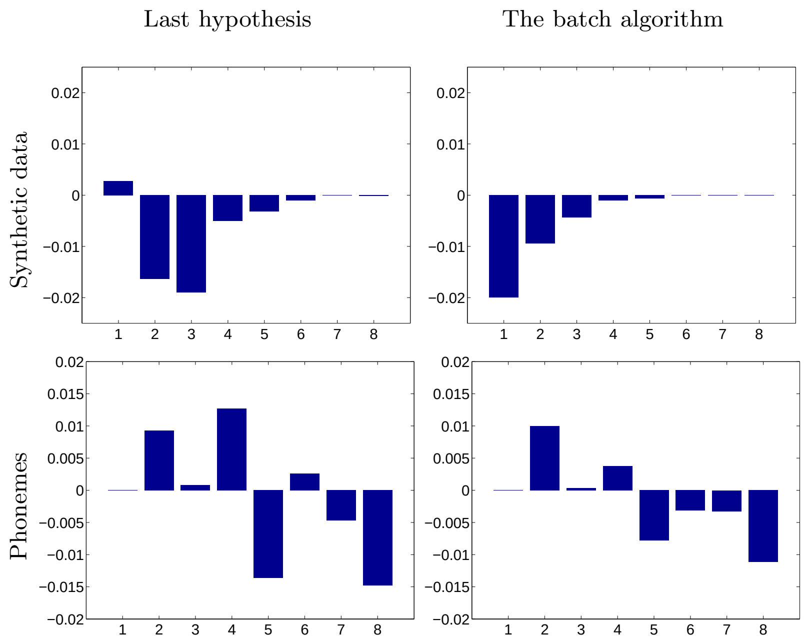

Fig. 4. The distribution of the tree induced-error for each dataset used in the experiments. Each bar corresponds to the difference between the error of the batch algorithm minus the error of a multiclass predictor.

on the right with the largest possible value of $\gamma(y, \hat{y})$. It is clear from the figure that the batch algorithm tends to make "small" errors by predicting the parent or a sibling of the correct phoneme. On the other hand the algorithm seldom chooses a phoneme or a phoneme group which is in an entirely different part of the tree, thus avoiding large tree induced errors. In the phoneme classification task, the algorithm seldom extends a prediction $\hat{y}$ such that $\gamma(y, \hat{y}) = 9$ while the errors of the multiclass predictor are uniformly distributed.

We conclude the experiments with a comparison of the hierarchical algorithm with a common construction of hierarchical classifiers (see for instance [6]), 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 feature vector 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 feature vector $\mathbf{x}$, its predicted phoneme 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, the batch algorithm 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 phoneme will impose a global prediction error.

Acknowledgment

This work was supported by EU PASCAL network of excellence.

References

1. Deller, J., Proakis, J., Hansen, J.: Discrete-Time Processing of Speech Signals. Prentice-Hall (1987)
2. Rabiner, L.R., Schafer, R.W.: Digital Processing of Speech Signals. Prentice-Hall (1978)
3. Robinson, A.J.: An application of recurrent nets to phone probability estimation. IEEE Transactions on Neural Networks **5** (1994) 298–305
4. Clarkson, P., Moreno, P.: On the use of support vector machines for phonetic classification. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing 1999, Phoenix, Arizona (1999)
5. Salomon, J.: Support vector machines for phoneme classification. Master's thesis, University of Edinburgh (2001)
6. Koller, D., Sahami, M.: Hierarchically classifying docuemnts using very few words. In: Machine Learning: Proceedings of the Fourteenth International Conference. (1997) 171–178
7. McCallum, A.K., Rosenfeld, R., Mitchell, T.M., Ng, A.Y.: Improving text classification by shrinkage in a hierarchy of classes. In: Proceedings of ICML-98. (1998) 359–367
8. Weigend, A.S., Wiener, E.D., Pedersen, J.O.: Exploiting hierarchy in text categorization. Information Retrieval **1** (1999) 193–216
9. Dumais, S.T., Chen, H.: Hierarchical classification of Web content. In: Proceedings of SIGIR-00. (2000) 256–263
10. Katz, S.: Estimation of probabilities from sparsedata for the language model component of a speech recognizer. IEEE Transactions on Acoustics, Speech and Signal Processing (ASSP) **35** (1987) 400–40
11. Vapnik, V.N.: Statistical Learning Theory. Wiley (1998)
12. Crammer, K., Dekel, O., Shalev-Shwartz, S., Singer, Y.: Online passive aggressive algorithms. In: Advances in Neural Information Processing Systems 16. (2003)
13. Herbster, M.: Learning additive models online with fast evaluating kernels. In: Proceedings of the Fourteenth Annual Conference on Computational Learning Theory. (2001) 444–460
14. Kivinen, J., Warmuth, M.K.: Exponentiated gradient versus gradient descent for linear predictors. Information and Computation **132** (1997) 1–64
15. Cesa-Bianchi, N., Conconi, A., C.Gentile: On the generalization ability of on-line learning algorithms. IEEE Transactions on Information Theory (2004) (to appear).
16. Weston, J., Watkins, C.: Support vector machines for multi-class pattern recognition. In: Proceedings of the Seventh European Symposium on Artificial Neural Networks. (1999)

17. Censor, Y., Zenios, S.: Parallel Optimization: Theory, Algorithms, and Applications. Oxford University Press, New York, NY, USA (1997)
18. Lemel, L., Kassel, R., Seneff, S.: Speech database development: Design and analysis . Report no. SAIC-86/1546, Proc. DARPA Speech Recognition Workshop (1986)
19. ETSI Standard, ETSI ES 201 108 (2000)