Online Multiclass Learning by Interclass Hypothesis Sharing

Michael Fink FINK@CS.HUJI.AC.IL
Center for Neural Computation, The Hebrew University, Israel

Shai Shalev-Shwartz SHAIS@CS.HUJI.AC.IL
School of Computer Science & Engineering, The Hebrew University, Israel

Yoram Singer SINGER@CS.HUJI.AC.IL
Google Inc., USA

Shimon Ullman SHIMON.ULLMAN@WEIZMANN.AC.IL
Weizmann institute, Israel

Abstract

We describe a general framework for online multiclass learning based on the notion of hypothesis sharing. In our framework sets of classes are associated with hypotheses. Thus, all classes within a given set share the same hypothesis. This framework includes as special cases commonly used constructions for multiclass categorization such as allocating a unique hypothesis for each class and allocating a single common hypothesis for all classes. We generalize the multiclass Perceptron to our framework and derive a unifying mistake bound analysis. Our construction naturally extends to settings where the number of classes is not known in advance but rather is revealed along the online learning process. We demonstrate the merits of our approach by comparing it to previous methods on both synthetic and natural datasets.

1. Introduction

A Zoologist in a research expedition is required to identify beetle species. There are over $350,000$ different known beetle species and new species are being discovered all the time. In this paper we describe, analyze, and experiment with a framework for multiclass learning aimed at addressing our Zoologist's classification task. In the multiclass problem we discuss, the learner is required to make predictions on-the-fly while the identity of the target classes is incrementally revealed as the learning proceeds. We thus use *online* learning as the learning apparatus and analyze our algorithms within the mistake bound model. In online learning we observe instances in a sequence of trials. After each observation, we need to predict the class of the observed instance. To do so, we maintain a hypothesis which scores each of the candidate classes and the predicted label is the one attaining the highest score. Once a prediction is made, we receive the correct class label. Then, we may update our hypothesis in order to improve the chance of making an accurate prediction on subsequent trials. Our goal is to minimize the number of online prediction mistakes.

Our solution builds on two commonly used constructions for multiclass categorization problems. The first dedicates an individual hypothesis for each target class (Duda & Hart, 1973; Vapnik, 1998) and is common in applications where the input instance is class independent. We refer to this construction as the *multi-vector* model. The second construction, abbreviated as the *single-vector* model, maintains a single hypothesis shared by all the classes while the input is class dependent. The latter construction is used in generalized additive models (Hastie & Tibshirani, 1995), boosting algorithms (Freund & Schapire, 1997), and structured multiclass problems (Collins, 2002). A common thread of the single-vector and the multi-vector models is that both were developed under the assumption that the set of target classes is known in advance. One of the goals of this paper is to provide a unified framework which encompasses these two models as special cases while lifting the requirement that the set of classes is known before learning takes place.

In the multiclass learning paradigm we study in this paper, sets of classes are associated with hypotheses. Thus, all classes within a given set share the same hypothe-

Appearing in *Proceedings of the 23^{rd} International Conference on Machine Learning*, Pittsburgh, PA, 2006.

sis. This framework naturally includes as special cases the two models discussed above. After introducing our new multiclass learning framework, we describe a generalization of the Perceptron algorithm (Rosenblatt, 1958) to our framework and derive a unifying mistake bound analysis. Our construction naturally extends to settings where the number of classes is not known in advance but rather revealed along the online learning process. The analysis we present is applicable to both the single-vector model and the multi-vector model and underscores a natural complexity-performance tradeoff. The complexity of the multi-vector model increases linearly with the number of classes while the model complexity of the single-vector approach is invariant to the number of classes. However, the higher complexity of the multi-vector model is occasionally necessary for achieving more accurate predictions.

The generalized Perceptron algorithm we derive can be viewed as an automatic mixing mechanism between the single-vector and the multi-vector models, as well as any model that shares hypotheses between classes. The performance of the generalized Perceptron is competitive with any hypothesis sharing model and in particular the single and multi vector models. Our construction also allows to share features across classes via a feature mapping mechanism. For example, our dextrous Zoologist can share feature mappings and hypotheses between groups of classes such as desert dweller beetles or terrestrial beetles. While our framework is especially appealing in settings where the classes are revealed on-the-fly, it can be used verbatim in standard multiclass problems. We illustrate the merits of our hypotheses sharing framework in a series of experiments with synthetic and natural datasets.

2. Problem Setting

Online learning is performed in a sequence of trials. At trial t the algorithm first receives an instance $\mathbf{x}_t \in \mathbb{R}^n$ and is required to predict a class label associated with that instance. The set of all possible labels constitutes a finite set denoted by $\mathcal{Y}$. Most if not all online classification algorithms assume that $\mathcal{Y}$ is known in advance. In contrast, in our setting the set $\mathcal{Y}$ is incrementally revealed as the online learning proceeds. We denote by $\mathcal{Y}_t$ the set of unique labels observed on rounds 1 through $t-1$. After the online learning algorithm predicts the class $\hat{y}_t$, the true class $y_t \in \mathcal{Y}$ is revealed and the set of known classes is updated accordingly, $\mathcal{Y}_{t+1} = \mathcal{Y}_t \cup \{y_t\}$. We say that the algorithm makes a prediction mistake if $\hat{y}_t \neq y_t$ and the class y_t is not a novel class, $y_t \in \mathcal{Y}_t$. We thus exclude from our mistake analysis all the rounds on which a label is observed for the first time. The goal of the algorithm is to minimize the total number of prediction mistakes it makes, denoted by M. To achieve this goal, the algorithm may update its prediction mechanism at the end of each trial.

The prediction of the algorithm at trial t is determined by a hypothesis, $h_t : \mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}$, which induces a score for each of the possible classes in $\mathcal{Y}$. The predicted label is defined as, $\hat{y}_t = \arg\max_{r \in \mathcal{Y}_t} h_t(\mathbf{x}_t, r)$. To evaluate the performance of a hypothesis h on the example $(\mathbf{x}_t, y_t)$ we need to check whether h makes a prediction mistake, namely determine if $\hat{y}_t \neq y_t \in \mathcal{Y}_t$. To derive bounds on prediction mistakes we use a second way for evaluating the performance of h which is based on the multiclass *hinge-loss* function, defined as follows. If the current class is not novel ($y_t \in \mathcal{Y}_t$), then we set

$$\ell_t(h) = \left(1 - h(\mathbf{x}_t, y_t) + \max_{r \in \mathcal{Y}_t \setminus \{y_t\}} h(\mathbf{x}_t, r)\right)_+ ,$$

where $(a)_+ = \max\{a, 0\}$. Since in our setting the algorithm is not penalized for the first instance of each class, we simply set $\ell_t(h) = 0$ whenever $y_t \notin \mathcal{Y}_t$. The term $h(\mathbf{x}_t, y_t) - \max_r h(\mathbf{x}_t, r)$ in the definition of the hinge-loss is a generalization of the notion of *margin* from binary classification. The hinge-loss penalizes a hypothesis for any margin less than 1. Additionally, if $\hat{y}_t \neq y_t$ then $\ell_t(h) \geq 1$. Thus, the *cumulative hinge-loss* suffered over a sequence of examples upper bounds the number of prediction mistakes, M.

Recall that the prediction on each trial is based on a hypothesis which is a function from $\mathbb{R}^n \times \mathcal{Y}$ into the reals. In this paper we focus on hypotheses which are parameterized by weight vectors. A common construction (Duda & Hart, 1973; Vapnik, 1998; Crammer & Singer, 2003) of a hypothesis space is the set of functions parameterized by $|\mathcal{Y}|$ vectors $W = \{\mathbf{w}^r : r \in \mathcal{Y}\}$ where,

$$h(\mathbf{x}, r) = \langle \mathbf{w}^r, \mathbf{x} \rangle .$$

That is, h associates a different weight vector with each class and the prediction at trial t is,

$$\hat{y}_t = \operatorname*{argmax}_{r \in \mathcal{Y}_t} \langle \mathbf{w}_t^r, \mathbf{x}_t \rangle .$$

To obtain a concrete online learning algorithm we must determine the initial value of each weight vector and the update rule used to modify the weight vectors at the end of each trial. Following Kesler's construction (Duda & Hart, 1973; Crammer & Singer, 2003), we address the multiclass setting using a Perceptron update. The multiclass Perceptron algorithm initializes all the weight vectors to be zero. On trial t, if the algorithm makes a prediction mistake, $\hat{y}_t \neq y_t \in \mathcal{Y}_t$, then the weight vectors are updated as follows,

$$\mathbf{w}_{t+1}^{y_t} = \mathbf{w}_t^{y_t} + \mathbf{x}_t \ , \ \mathbf{w}_{t+1}^{\hat{y}_t} = \mathbf{w}_t^{\hat{y}_t} - \mathbf{x}_t \ ,$$

and $\mathbf{w}^r_{t+1} = \mathbf{w}^r_t$ for all $r \in \mathcal{Y}_t \setminus \{y_t, \hat{y}_t\}$. In words, we add the instance $\mathbf{x}_t$ to the weight vector of the correct class and subtract $\mathbf{x}_t$ from the weight vector of the (wrongly) predicted class. We would like to note in passing that other Perceptron-style updates can be devised for multiclass problems (Crammer & Singer, 2003). Finally, if we do not make a prediction mistake then the weight vectors are kept intact. We refer to the above construction as the multi-vector method.

Several mistake bounds have been derived for the multi-vector method. In this paper we obtain the following mistake bound, which follows as a corollary from our analysis in Sec. 4. Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$ be a sequence of examples and define $R = 2 \max_t \|\mathbf{x}_t\|_2$. Let $h^\star$ be a fixed hypothesis defined by any set of weight vectors $U = \{\mathbf{u}^r : r \in \mathcal{Y}\}$. We denote by

$$L = \sum_{t=1}^{m} \ell_t(h^\star) \ , \tag{1}$$

the cumulative hinge-loss of $h^\star$ over the sequence of examples and by

$$C = R^2 \sum_{r \in \mathcal{Y}} \|\mathbf{u}^r\|^2 \ , \tag{2}$$

the *complexity* of $h^\star$. Then the number of prediction mistakes of the multi-vector method is at most,

$$M \ \leq \ L + C + \sqrt{L\,C} \ . \tag{3}$$

The mistake bound in Eq. (3) consists of three terms: the loss of $h^\star$, the complexity of $h^\star$, and a sub-linear term which is often negligible. We would like to underscore that the complexity term increases with the number of classes since we have a different weight vector for each class.

We now describe an alternative construction and an accompanying learning algorithm which maintains a *single* vector. We show in the sequel that the second construction entertains a mistake bound of the form given in Eq. (3). However, the complexity term in this bound does not increase with the number of classes, in contrast to the complexity term for the multi-vector method given in Eq. (2). The second multiclass construction uses a *single* weight vector, denoted w, for all the classes, paired with a class-specific feature mapping, $\boldsymbol{\phi} : \mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}^d$. That is, the score given by a hypothesis h for class r is,

$$h(\mathbf{x}, r) \ = \ \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, r) \rangle \ .$$

We denote by $\mathbf{w}_t$ the single weight vector of the algorithm at trial t and its prediction is thus,

$$\hat{y}_t \ = \ \operatorname*{argmax}_{r \in \mathcal{Y}_t} \langle \mathbf{w}_t, \boldsymbol{\phi}(\mathbf{x}, r) \rangle \ .$$

This construction is common in generalized additive models (Hastie & Tibshirani, 1995), multiclass versions of boosting (Freund & Schapire, 1997), and has been popularized lately due to its role in prediction with structured output where the number of classes is exponentially large (Collins, 2002; Taskar et al., 2003; Tsochantaridis et al., 2004; Shalev-Shwartz et al., 2004). Following a simple Perceptron-based mechanism we initialize $\mathbf{w}_1 = \mathbf{0}$ and only update $\mathbf{w}$ if we have a prediction mistake, $\hat{y}_t \neq y_t \in \mathcal{Y}_t$. The update takes the form,

$$\mathbf{w}_{t+1} \ = \ \mathbf{w}_t + \boldsymbol{\phi}(\mathbf{x}_t, y_t) - \boldsymbol{\phi}(\mathbf{x}_t, \hat{y}_t) \ .$$

We refer to the above construction as the single-vector method.

The single-vector method is based on a class specific feature mapping $\boldsymbol{\phi}$. Usually, this class specific mapping relies on an a-priori knowledge of the set of possible classes $\mathcal{Y}$. This paper emphasizes the setting where the identity of the target classes is incrementally revealed only during the online stream. Since $\mathcal{Y}$ is not known a-priori, we apply a class specific feature mapping which is *data dependent*. For each class $r \in \mathcal{Y}$, let $\mathbf{p}^r \in \mathbb{R}^n$ be the first instance of class r in the sequence of examples. We define $\boldsymbol{\phi}(\mathbf{x}_t, r)$ to be the vector in $\mathbb{R}^n$ whose i'th element is,

$$\phi_i(\mathbf{x}_t, r) \ = \ x_{t,i}\, p^r_i \ . \tag{4}$$

That is, $\boldsymbol{\phi}(\mathbf{x}_t, r)$ is the coordinate-wise product between $\mathbf{x}_t$ and $\mathbf{p}^r$. In Sec. 5 we describe additional data-dependent constructions of $\boldsymbol{\phi}$.

A relative mistake bound can also be derived for the single-vector method. Specifically, in Sec. 4 we show that the bound in Eq. (3) holds where now $R = 2 \max_{t,r} \|\boldsymbol{\phi}(\mathbf{x}_t, r)\|_2$, the competing hypothesis $h^\star$ is parameterized by any *single* weight vector $\mathbf{u}$, and the complexity of $h^\star$ is,

$$C = R^2 \|\mathbf{u}\|^2 \ . \tag{5}$$

The complexity term for the single-vector method does not increase with the number of classes, in contrast to the complexity term for the multi-vector method given in Eq. (2). However, the value of the cumulative loss, L, in the multi-vector method is upper bounded by the cumulative loss in the single-vector method. This follows from the fact that the hypotheses space employed by the multi-vector method is richer than that of the single-vector method. To see this, note that given any single weight vector $\mathbf{u}$, we can construct the set of multiple weight vectors $U = \{\mathbf{u}^r : r \in \mathcal{Y}\}$ where $u^r_i = u_i p^r_i$. Using this construction we observe that $\langle \mathbf{u}, \boldsymbol{\phi}(\mathbf{x}_t, r) \rangle = \langle \mathbf{u}^r, \mathbf{x}_t \rangle$ and therefore the cumulative loss of $\mathbf{u}$ in the single-vector method equals to the cumulative loss of $U = \{\mathbf{u}^r : r \in \mathcal{Y}\}$ in the multi-vector method.

The prevailing question is which of the two approaches would perform better in practical applications. Indeed, our

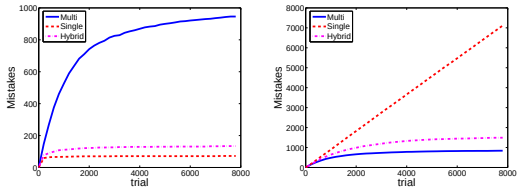

Figure 1. The number of mistakes of the single-vector and the multi-vector methods described in Sec. 2, and a hybrid method described in Sec. 3 on two synthetic datasets.

experiments indicate that on certain datasets the single-vector method outperforms the multi-vector approach while on other datasets an opposite effect is exhibited and the richer model complexity of the multi-vector method is necessary. One of the main contributions of this paper is a mixing method, whose performance on any dataset is competitive with the best of the aforementioned alternatives.

To illustrate the difference between the single-vector and multi-vector methods we have constructed two synthetic datasets. Both datasets contain $8,000$ instances from $\{-1,1\}^{64}$ and the set of classes is $\mathcal{Y} = \{0,\ldots,15\}$. In the first dataset, the class of an instance $\mathbf{x}$ is the value of the binary number (x_1, x_2, x_3, x_4). In the second dataset, the class of an instance $\mathbf{x}$ is r if $x_{4r+1} = \ldots = x_{4r+4} = 1$ (we made sure that for each instance, only one class satisfies the above). We presented both datasets to the single-vector and multi-vector methods. The cumulative number of mistakes of the two algorithms as a function of the trial number is depicted in Fig. 1. As can be seen from the figure, the single-vector method clearly outperforms the multi-vector method on the first dataset while the opposite phenomenon is exhibited in the second dataset. This difference can be attributed to the interplay between the loss and the complexity terms in our mistake bounds. Indeed, in the first dataset, the single-vector model is capable of achieving zero cumulative loss by setting the first 4 elements of $\mathbf{u}$ to be $\frac{1}{2}$ and the rest to be zero. Our mistake bound for the single-vector method reduces to $2 \cdot 64 \cdot 1 = 128$. In contrast, the mistake bound for the multi-vector method is 16 times higher and equals to 2048. In the second dataset, the single-vector model is not rich enough for perfectly predicting the correct labels. Therefore, the number of mistakes sustained by the single-vector method increases linearly with the number of examples. In this dataset, the opulent complexity of the multi-vector method is beneficial.

3. Mixing the single and multi vector methods

In this section we describe a *hybrid* method whose performance on any dataset is competitive with the best of the two alternative multiclass approaches described in the previous section. Moreover, we show that on certain datasets the hybrid method outperforms both the single-vector and multi-vector methods.

The hypotheses of the hybrid method are parameterized by a set of $|\mathcal{Y}| + 1$ weight vectors. As in the single-vector method we maintain one weight vector, denoted $\mathbf{w}^{\mathcal{Y}}$, which is shared among all classes in $\mathcal{Y}$. As in the multi-vector method the remaining $|\mathcal{Y}|$ weight vectors are specific to each of the classes. The score of h for class r is,

$$h(\mathbf{x}, r) \;=\; \langle \mathbf{w}^{\mathcal{Y}}, \boldsymbol{\phi}(\mathbf{x}, r) \rangle + \langle \mathbf{w}^{r}, \mathbf{x} \rangle \;\; .$$

We denote by $\{\mathbf{w}_t^{\mathcal{Y}}\} \cup \{\mathbf{w}_t^r : r \in \mathcal{Y}_t\}$ the weight vectors of the algorithm at trial t and its prediction is thus,

$$\hat{y}_t \;=\; \operatorname*{argmax}_{r \in \mathcal{Y}_t} \left(\langle \mathbf{w}_t^{\mathcal{Y}}, \boldsymbol{\phi}(\mathbf{x}_t, r) \rangle \;+\; \langle \mathbf{w}_t^{r}, \mathbf{x}_t \rangle \right) \;\; .$$

We now describe a Perceptron-style update for the hybrid method. Initially, all the weight vectors are set to zero. On trial t, the weight vectors are updated only if the algorithm made a prediction mistake ($\hat{y}_t \neq y_t \in \mathcal{Y}_t$) by using the update rule,

$$\begin{aligned} \mathbf{w}_{t+1}^{\mathcal{Y}} &= \mathbf{w}_t^{\mathcal{Y}} \;+\; \boldsymbol{\phi}(\mathbf{x}_t, y_t) \;-\; \boldsymbol{\phi}(\mathbf{x}_t, \hat{y}_t) \;\; , \\ \mathbf{w}_{t+1}^{y_t} &= \mathbf{w}_t^{y_t} + \mathbf{x}_t \;\; , \\ \mathbf{w}_{t+1}^{\hat{y}_t} &= \mathbf{w}_t^{\hat{y}_t} - \mathbf{x}_t \;\; , \end{aligned}$$

and for all $r \in \mathcal{Y}_t \setminus \{y_t, \hat{y}_t\}$, $\mathbf{w}_{t+1}^r = \mathbf{w}_t^r$.

A relative mistake bound can be proven for the hybrid method as well. In particular, in Sec. 4 we show that a bound of the same form given in Eq. (3) holds for the hybrid method. That is, given a hypothesis $h^\star$, parameterized by any set of vectors $U = \{\mathbf{u}^{\mathcal{Y}}\} \cup \{\mathbf{u}^r : r \in \mathcal{Y}\}$, the following bound holds, $M \leq L + C + \sqrt{L\,C}$, where C is defined to be,

$$C = 2\,R^2 \left(\|\mathbf{u}^{\mathcal{Y}}\|^2 + \sum_{r \in \mathcal{Y}} \|\mathbf{u}^r\|^2 \right) \;\; , \qquad (6)$$

and R is now the maximal value between $2 \max_t \|\mathbf{x}_t\|$ and $2 \max_{t,r} \|\boldsymbol{\phi}(\mathbf{x}_t, r)\|_2$.

We now compare the above mistake bound of the hybrid method to the mistake bounds of the single-vector and multi-vector methods. The cumulative loss of the hybrid method is bounded above by both the loss of the single-vector method and the loss of the multi-vector method. This follows directly from the fact that the hypothesis space of the hybrid method includes both the hypothesis space of the single-vector method and that of the multi-vector method. To facilitate a clear comparison of the complexity term, let us assume that $\max_t \|\mathbf{x}_t\| = \max_{t,r} \|\boldsymbol{\phi}(\mathbf{x}_t, r)\|$ and thus the value of R for all methods is identical. This equality indeed holds for the datasets described in Sec. 2. Moreover, if the norm of $\boldsymbol{\phi}$ is not restricted relatively to

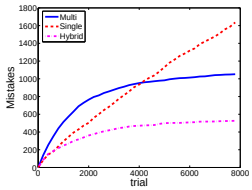

Figure 2. The number of mistakes of the hybrid method, the single-vector method, and the multi-vector method on the third synthetic dataset.

$\|\mathbf{x}_t\|$ and is allowed to grow with the number of classes then by concatenating class vectors we can reduce the multi-vector method to the single-vector method. Therefore, throughout the paper we focus on constructions in which the norm of $\boldsymbol{\phi}$ is of the same order of magnitude of $\|\mathbf{x}_t\|$. Equipped with this assumption we note that the complexity term of the hybrid method is at most twice the minimum between the complexity of the single-vector method and the multi-vector method.

In Fig. 1 we compare the performance of the hybrid method to the performance of the single-vector and the multi-vector methods on the two synthetic datasets described in Sec. 2. As expected, the performance of the hybrid method is comparable to the best of the two alternatives. The two synthetic datasets we constructed in Sec. 2 represent two extremes: the relevant components for each class are either common (first dataset) or completely disjoint (second dataset). In practical situations, it might be the case that while most of the classes share the same relevant dimensions several of the classes might depend on other dimensions. For example, if the task in hand is bird classification, the features used in recognizing most birds are common but are not applicable to penguins. To illustrate this point we have generated a third dataset as follows. As in our previous datasets, we chose $8,000$ instances from $\{+1,-1\}^{64}$ and the set of classes was set to $\mathcal{Y} = \{0,\ldots,15\}$. Instances of the first 15 classes have been generated as in the first dataset, that is, the class label was the value of the binary number $(x_1,\ldots,x_4)$. Instances of the last class ($r = 15$) were generated as in the second dataset by setting $x_{61} = \ldots = x_{64} = 1$. We have presented this dataset to the hybrid method and to the single-vector and multi-vector methods. The performance of the different algorithms is depicted in Fig. 2. It is clear from the figure that the hybrid method outperforms the two alternatives. It should also be noted that in the first half of the input sequence the single-vector method errs less than the multi-vector method while in the second half the multi-vector method outperforms the single-vector method. These effects can be explained in the light of our analysis. Our mistake bounds depend on a fixed complexity term and on a loss term which depends on the number of trials. The complexity term in the bound of the multi-vector method is higher than the complexity term in the bound of the single-vector method while an opposite trend characterizes the loss term.

4. A general mixing framework

In the previous sections we described the single-vector method, the multi-vector method and the hybrid method. In this section we propose a general mixing framework of which the above three methods are special cases. We also utilize this framework for deriving new mixing algorithms. Finally, we provide a unified analysis for our general mixing framework and in particular obtain the mistake bounds for the three methods described in previous sections.

Our general mixing framework assumes the existence of a collection of indicator functions, denoted $\mathcal{T}$, where each $\tau \in \mathcal{T}$ is a function from $\mathcal{Y}$ into $\{0,1\}$. Thus, each function τ corresponds to the set $S^\tau = \{r \in \mathcal{Y} : \tau(r) = 1\}$, which includes all the classes in $\mathcal{Y}$ for which $\tau(r) = 1$. The hypotheses of the general mixing framework are parameterized by a set of $|\mathcal{T}|$ weight vectors. For each $\tau \in \mathcal{T}$ we maintain one weight vector, $\mathbf{w}^\tau$, which is shared among all classes in S^τ. In addition, we assume that there exists a feature mapping function $\boldsymbol{\phi}^\tau(\mathbf{x}, r)$ for each $\tau \in \mathcal{T}$. The score given by a hypothesis h for class r is,

$$h(\mathbf{x}, r) \;=\; \sum_{\tau \in \mathcal{T}} \tau(r) \left\langle \mathbf{w}^\tau, \boldsymbol{\phi}^\tau(\mathbf{x}, r) \right\rangle \;. \tag{7}$$

We denote by $\{\mathbf{w}_t^\tau : \tau \in \mathcal{T}\}$ the weight vectors of the algorithm at trial t and its prediction is thus,

$$\hat{y}_t \;=\; \operatorname*{argmax}_{r \in \mathcal{Y}_t} \sum_{\tau \in \mathcal{T}} \tau(r) \left\langle \mathbf{w}^\tau, \boldsymbol{\phi}^\tau(\mathbf{x}, r) \right\rangle \;.$$

In our beetle recognition example, a function $\tau \in \mathcal{T}$ might indicate whether a beetle is a desert dweller. This information is known before a zoologist might encounter a new species and is beneficial for transferring representational knowledge from previously learned distinctions.

We now describe a Perceptron-style update for the general mixing framework. Initially, all the weight vectors are set to zero. If there was a prediction mistake on trial t, $\hat{y}_t \neq y_t \in \mathcal{Y}_t$, then we update each of the vectors in $\{\mathbf{w}^\tau : \tau \in \mathcal{T}\}$ as follows,

$$\mathbf{w}_{t+1}^\tau = \mathbf{w}_t^\tau + \tau(y_t)\, \boldsymbol{\phi}^\tau(\mathbf{x}_t, y_t) - \tau(\hat{y}_t)\, \boldsymbol{\phi}^\tau(\mathbf{x}_t, \hat{y}_t) \;.$$

If the algorithm does not err then $\mathbf{w}_{t+1}^\tau = \mathbf{w}_t^\tau$ for all $\tau \in \mathcal{T}$. A pseudo-code summarizing the general mixing method is given in Fig. 3.

The single-vector method is a special case of the general mixing framework that can be derived by setting $\mathcal{T} =$

INPUT: Collection of indicator functions $\mathcal{T}$ and corresponding feature mappings $\{\boldsymbol{\phi}^\tau : \tau \in \mathcal{T}\}$

INITIALIZE: $\mathcal{Y}_1 = \emptyset$; $\forall \tau \in \mathcal{T}, \mathbf{w}^\tau = \mathbf{0}$

For $t = 1, 2, \ldots$

receive an instance $\mathbf{x}_t$

predict: $\hat{y}_t = \operatorname{argmax}_{r \in \mathcal{Y}_t} \sum_{\tau \in \mathcal{T}} \tau(r) \langle \mathbf{w}^\tau, \boldsymbol{\phi}^\tau(\mathbf{x}_t, r) \rangle$

receive correct label y_t

If $\hat{y}_t \neq y_t$ and $y_t \in \mathcal{Y}_t$

forall $\tau \in \mathcal{T}$,

$\mathbf{w}^\tau \leftarrow \mathbf{w}^\tau + \tau(y_t)\, \boldsymbol{\phi}^\tau(\mathbf{x}_t, y_t) - \tau(\hat{y}_t)\, \boldsymbol{\phi}^\tau(\mathbf{x}_t, \hat{y}_t)$

$\mathcal{Y}_{t+1} = \mathcal{Y}_t \cup \{y_t\}$

Figure 3. The general mixing algorithm.

$\{\tau^{\mathcal{Y}}\}$ where $\tau^{\mathcal{Y}}(r) = 1$ for all $r \in \mathcal{Y}$. Thus, $S^{\tau^{\mathcal{Y}}} = \mathcal{Y}$ and we obtain a single weight vector $\mathbf{w}^{\tau^{\mathcal{Y}}}$ which is shared by all the classes in $\mathcal{Y}$. The multi-vector method can also be derived from the general mixing framework by setting $\mathcal{T} = \{\tau^r : r \in Y\}$, where $\tau^r(k)$ is one if $k = r$ and zero otherwise. We therefore associate a different weight vector with each label in $\mathcal{Y}$. In the multi-vector method, the value of $\boldsymbol{\phi}^\tau(\mathbf{x}, r)$ reduces to $\mathbf{x}$. The hybrid method can be derived in a similar manner by a simple conjunction of the above indicator functions.

The algorithm from Fig. 3 can be adjusted to incorporate Mercer kernels. Note that each vector $\mathbf{w}^\tau$ can be represented as a sum of vectors of the form $\boldsymbol{\phi}^\tau(\mathbf{x}_i, r)$ where $i < t$. Furthermore, the inner-product $\langle \mathbf{w}^\tau, \boldsymbol{\phi}^\tau(\mathbf{x}_t, r) \rangle$ can be rewritten as a sum of inner-products each taking the form $\langle \boldsymbol{\phi}^\tau(\mathbf{x}_t, r), \boldsymbol{\phi}^\tau(\mathbf{x}', r') \rangle$. We can replace the inner-products in this sum with a general Mercer kernel operator, $K^\tau((\mathbf{x}_t, r), (\mathbf{x}', r'))$, and leave the rest of the derivation intact. The formal analysis presented in the sequel can be extended verbatim and applied with Mercer kernels.

We now turn to the analysis of the algorithm in Fig. 3. Our analysis is based on the following lemma.

Lemma 1 *Let $\mathbf{a}_1, \ldots, \mathbf{a}_M$ be a sequence of vectors and define $R = \max_i \|\mathbf{a}_i\|_2$. Assume that for all $i \in \{1, \ldots, M\}$ we have that $\langle \mathbf{w}_i, \mathbf{a}_i \rangle \leq 0$ where $\mathbf{w}_i = \sum_{t=1}^{i-1} \mathbf{a}_i$. Let $\mathbf{u}$ be an arbitrary vector and let C and L be two scalars such that $C = R^2 \|\mathbf{u}\|^2$ and $L \geq \sum_{i=1}^{M} (1 - \langle \mathbf{a}_i, \mathbf{u} \rangle)_+$. Then, $M \leq L + C + \sqrt{L\,C}$.*

The proof of this lemma can be derived from the analysis of the Perceptron algorithm for binary classification given in (Gentile, 2002), and is omitted due to the lack of space. Equipped with the above lemma we now prove a mistake bound for the algorithm. Let $h^\star$ be any competing hypothesis defined by a set of vectors $U = \{\mathbf{u}^\tau : \tau \in \mathcal{T}\}$. As in previous sections, our mistake bound takes the form $M \leq L + C + \sqrt{L\,C}$, where L is the cumulative loss of $h^\star$ defined by Eq. (1) and C is the complexity of $h^\star$ which we now define. Let ρ be the maximal number of sets S^τ which include r, that is, $\rho = \max_{r \in \mathcal{Y}} \sum_{\tau \in \mathcal{T}} \tau(r)$. For example, in the single-vector and multi-vector methods the value of ρ is one while in the hybrid method $\rho = 2$. The complexity of $h^\star$ is formally defined to be,

$$C \;=\; \rho\, R^2 \sum_{\tau \in \mathcal{T}} \|\mathbf{u}^\tau\|^2 \;, \tag{8}$$

where $R \;=\; 2 \max_{t,r,\tau} \tau(r)\, \|\boldsymbol{\phi}^\tau(\mathbf{x}_t, r)\|_2$.

Theorem 1 *Let $((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)) \in (\mathbb{R}^n \times \mathcal{Y})^m$ be a sequence of examples and assume that this sequence is presented to the general mixing algorithm given in Fig. 3. Let $h^\star$ be any competing hypothesis defined by a set of weight vectors $U = \{\mathbf{u}^\tau : \tau \in \mathcal{T}\}$. Define L and C as given by Eq. (1) and Eq. (8). Then, the number of prediction mistakes the general mixing algorithm makes on the sequence is upper bounded by,*

$$M \;\leq\; L + C + \sqrt{L\,C} \;.$$

Proof Let $i_1, \ldots, i_M$ be the indices of trials in which the algorithm makes a prediction mistake. We prove the theorem by constructing a sequence of vectors $A_{i_1}, \ldots, A_{i_M}$ in a Hilbert space $\mathcal{H}$, which satisfies the condition given in Lemma 1. For each $\tau \in \mathcal{T}$, the function $\boldsymbol{\phi}^\tau$ maps an instance $\mathbf{x}_t$ and a label r into a Hilbert space, denoted $\mathcal{H}^\tau$. Let $\mathcal{H} = \bigotimes_{\tau \in \mathcal{T}} \mathcal{H}^\tau$ be the product of these feature spaces. Let $V_1 = \{\mathbf{v}_1^\tau \in \mathcal{H}^\tau : \tau \in \mathcal{T}\}$ and $V_2 = \{\mathbf{v}_2^\tau \in \mathcal{H}^\tau : \tau \in \mathcal{T}\}$ be two vectors in $\mathcal{H}$. Then, the vector addition V_1 and V_2 in $\mathcal{H}$ is defined as $V_1 + V_2 = \{\mathbf{v}_1^\tau + \mathbf{v}_2^\tau : \tau \in \mathcal{T}\}$ and their inner-product as $\langle V_1, V_2 \rangle = \sum_{\tau \in \mathcal{T}} \langle \mathbf{v}_1^\tau, \mathbf{v}_2^\tau \rangle$. The sets $W_t = \{\mathbf{w}_t^\tau : \tau \in \mathcal{T}\}$ and $U = \{\mathbf{u}^\tau : \tau \in \mathcal{T}\}$ are vectors in $\mathcal{H}$. For a trial t and a label $r \in \mathcal{Y}$, define $V_t^r \in \mathcal{H}$ to be, $V_t^r = \{\tau(r)\, \boldsymbol{\phi}^\tau(\mathbf{x}_t, r) : \tau \in \mathcal{T}\}$. Thus, the prediction of the algorithm can be rewritten as, $\hat{y}_t \;=\; \arg\max_{r \in \mathcal{Y}_t} \langle W_t, V_t^r \rangle$. Let t be a trial in which the algorithm makes a prediction mistake ($\hat{y}_t \neq y_t \in \mathcal{Y}_t$) and define $A_t = V_t^{y_t} - V_t^{\hat{y}_t}$. From the definitions of $\hat{y}_t$ and A_t and the fact that the algorithm makes a prediction mistake on this trial we get that, $\langle W_t, A_t \rangle \leq 0$. In addition, the update of the algorithm can be rewritten as a vector addition in $\mathcal{H}$, $W_{t+1} = W_t + A_t$. The definition of the hinge-loss of $h^\star$ gives that,

$$\begin{aligned} \ell_t(h^\star) \;&=\; \max_{r \neq y_t} \left(1 - \langle U, V_t^{y_t} - V_t^r \rangle\right)_+ \\ &\geq\; \left(1 - \langle U, V_t^{y_t} - V_t^{\hat{y}_t} \rangle\right)_+ = \left(1 - \langle U, A_t \rangle\right)_+ \;. \end{aligned}$$

Next, we upper bound the norm of A_t as follows. For all r,

$$\|V_t^r\|^2 \;=\; \sum_{\tau \in \mathcal{T}} \tau(r)\, \|\boldsymbol{\phi}^\tau(\mathbf{x}_t, r)\|^2 \;\leq\; \rho\, (R/2)^2 \;.$$

Thus,

$$\|A_t\| \;\leq\; \|V_t^{y_t}\| + \|V_t^{\hat{y}_t}\| \;\leq\; 2\sqrt{\rho}\,R/2 = \sqrt{\rho}\,R \;\;.$$

We can now apply Lemma 1 to the sequence $A_{i_1}, \ldots, A_{i_M}$ in conjunction with U and obtain the mistake bound in the theorem. ■

5. Experiments

In this section we present experimental results that demonstrate different aspects of our proposed framework. All experiments compare the multi-vector and single-vector methods to the hybrid method. Our first experiment was performed with the Enron email dataset (available at http://www.cs.umass.edu/∼ronb/datasets/enron_flat.tar.gz). The task is to automatically classify email messages into user defined folders. Thus, the instances in this dataset are email messages while the set of classes is the email folders. Note that the set of folders is not known in advance and the user can define new folders on-the-fly. Therefore, our online setting, in which the set of classes is revealed as the online learning proceeds, naturally captures the essence of this email classification task. We represented each email message as a binary vector $\mathbf{x} \in \{0,1\}^n$ with a coordinate for each word, so that $x_i = 1$ if the word corresponding to the index i appears in the email message and zero otherwise. At each trial, we constructed class specific mappings $\boldsymbol{\phi}(\mathbf{x}_t, r)$, for each class $r \in \mathcal{Y}_t$, as follows. Let $I_t^r = \{i < t : y_i = r\}$ be the set of previous trials in which the class label is r and define $\mathbf{p}_t^r$ to be the average instance over I_t^r,

$$\mathbf{p}_t^r = \frac{1}{|I_t^r|} \sum_{i \in I_t^r} \mathbf{x}_i \;\;. \tag{9}$$

We define $\boldsymbol{\phi}(\mathbf{x}_t, r)$ to be the vector in $\mathbb{R}^n$ whose i'th element is,

$$\phi_i(\mathbf{x}_t, r) = \begin{cases} 2 & x_{t,i} = 1 \;\wedge\; p_{t,i}^r \geq 0.2 \\ -1 & x_{t,i} = 1 \;\wedge\; p_{t,i}^r \leq 0.02 \\ 0 & \text{otherwise} \end{cases} \;\;. \tag{10}$$

That is, $\phi_i(\mathbf{x}_t, r) = 2$ if the word corresponding to index i appears in the current email message and also appears in at least fifth of the previously observed messages of class r. If the word appears in the current message but is very rare in previous messages of class r, then $\phi_i(\mathbf{x}_t, r) = -1$. In all other cases, $\phi_i(\mathbf{x}_r, t) = 0$. We ran the various algorithms on sequences of email messages from 7 users. The results are summarized in Table 1. As can be seen, the hybrid method consistently outperforms both the multi-vector and single-vector methods. It should also be noted that for 4 users the multi-vector method outperforms the single-vector method while for the remaining 3 users an opposite trend is apparent.

dataset	$\lvert\mathcal{Y}\rvert$	m	**multi**	**single**	**hybrid**
Enron1	101	1971	60.0	52.3	47.7
Enron2	25	3672	29.5	35.5	26.3
Enron3	41	4477	50.9	58.4	46.0
Enron4	47	4015	48.4	53.6	42.5
Enron5	11	2489	25.2	27.7	22.3
Enron6	30	1188	30.2	23.7	21.8
Enron7	18	2769	4.84	3.54	3.35
Office	51	362	8.01	6.63	4.42
YaleB	30	1920	18.3	15.2	12.9
ISOLET	26	6238	12.7	8.52	9.08
LETTER	26	20000	11.8	16.7	11.4

Table 1. The average number of online mistakes of the multi-vector method, the single-vector method, and the hybrid method on various datasets. The datasets labeled Enron1-Enron7, correspond to email messages of the users beck-s, farmer-d, kaminski-v, kitchen-l, lokay-m, sanders-r, and williams-w3 in the Enron dataset.

Our second experiment was performed with a dataset of office workspace images (available at http://www.cs.huji.ac.il/∼fink/office.html). To motivate the learning task, imagine a robot that is required to deliver packages in a large office building. Every day the robot must wander throughout the building and upon reaching a person's desk, deliver the appropriate package. Here again the identity of the classes is not known in advance. An office complex with 51 different desks was selected for constructing the dataset. The dataset contains 362 images of the different desks. Images were taken while the camera was facing the desk typically 1m away from the target and at an approximate height of 1.5m. The variation in the images due to changing pose and lighting conditions suggests that a representation based on sets of local descriptors might be suitable for our task. This representation choice seems to be especially appropriate since the characteristic components of each workspace, e.g. a telephone, mug or briefcase, might appear in any location within the image. We therefore chose a representation of images which is based on SIFT key descriptors (Lowe, 2004). Similarly to email messages in the Enron dataset, we represented each instance as a binary vector $\mathbf{x} \in \{0,1\}^n$ with a coordinate for each possible SIFT key, where $x_i = 1$ if the SIFT key corresponding to index i matches a SIFT key in the image $\mathbf{x}$. As suggested in the SIFT key literature, we declare a match between two SIFT keys if the Euclidean distance between them is significantly lower than any other key extracted from the image. The set of SIFT keys is incrementally constructed by adding all the SIFT keys of each new image that were not matched with previous images in the sequence of

examples. As in the Enron dataset, we used the class specific mapping given in Eq. (10). The performance of the various algorithms on the office dataset is given in Table 1. Here too the hybrid method outperforms the other two alternatives. It should be noted that similar results are obtained when averaging the performance of the algorithm over different permutations of the examples in the dataset.

Our next experiment was performed with the YaleB dataset containing 1920 face images of 30 different people under various illumination conditions. Following (Hertz et al., 2004), we automatically centered all images using optical flow and converted each image to a vector using its first 60 PCA coefficients. We normalized the resulting vectors so that the standard deviation of each coordinate of an instance will be 1. For the single-vector method we defined $\phi_i(\mathbf{x}_t, r) = |x_{t,i} - p^r_{t,i}|$, where $\mathbf{p}^t_r$ is as defined in Eq. (9). The performance of the three methods is given in Table 1. The single-vector method outperforms the multi-vector method while the hybrid method achieves the best results.

Our last experiment was performed with two standard multiclass datasets: ISOLET and LETTER taken from the UCI repository. Here, we implemented the various algorithms using Mercer kernels. The classes in both datasets are the 26 English letters. However, the instances are represented in two different modalities: the ISOLET instances encode auditory recordings of subjects pronouncing the names of the 26 letters, while the LETTER instances encode features derived from black and white images of the 26 uppercase letters. For the multi-vector method we used a Gaussian kernel. The value of σ was set to 0.16 for the ISOLET dataset and to 0.07 for the LETTER dataset. In the single-vector method we define the kernel,

$$K((\mathbf{x}_t, r), (\mathbf{x}_j, s)) \;=\; e^{-\frac{1}{2\sigma}\|(\mathbf{x}_t - \mathbf{p}^r_t) - (\mathbf{x}_j - \mathbf{p}^s_j)\|^2} \quad , \quad (11)$$

where $\mathbf{p}^r_t$ is as defined in Eq. (9) and σ was again 0.16 for ISOLET and 0.07 for LETTER. As can be seen in Table 1, the multi-vector method outperforms the single-vector method on LETTER while an opposite trend is apparent on ISOLET. This experiment emphasizes the fact that although the classes are known a-priori, we cannot determine in advance which of the two methods will be better. The hybrid method is comparable to the best of the two alternatives and thus relieves us from the necessity to make an early choice between the multi-vector and single-vector methods.

6. Discussion

In this paper we introduced a framework for online multiclass learning by hypothesis sharing. We described the hypothesis sharing model which, together with a feature mapping mechanism, enables learning without prior knowledge of class labels. Our analysis and experiments indicate that the proposed framework is a viable alternative to the common multiclass learning approaches. The hypothesis sharing approach relies heavily on the set of indicator functions defined by $\mathcal{T}$. In certain applications, it is natural to assume that these indicator functions are provided in advance. For example, in the beetle recognition task, a function in $\mathcal{T}$ can indicate whether a beetle is a member of the set of desert dwelling beetles. In the specific case where the set of indicator functions $\mathcal{T}$ reflects a hierarchical structure, the general mixing model can be viewed as a generalization of the model described in (Dekel et al., 2004). In general, the indicator functions might not be provided in advance and thus, learning $\mathcal{T}$ is a worthwhile challenge which is differed to future work.

Acknowledgments This work was supported by Grant 7-0369 from the Israeli Science Foundation and by EU IST Grant FP6-2005-015803

References

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

Crammer, K., & Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *JMLR, 3*.

Dekel, O., Keshet, J., & Singer, Y. (2004). Large margin hierarchical classification. *Proceedings of the Twenty-First International Conference on Machine Learning*.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. Wiley.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55*.

Gentile, C. (2002). The robustness of the p-norm algorithms. *Machine Learning, 53*.

Hastie, T., & Tibshirani, R. (1995). *Generalized additive models*. Chapman & Hall.

Hertz, T., Bar-Hillel, A., & Weinshall, D. (2004). Learning distance functions for image retrieval. *CVPR*.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *IJCV*.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review, 65*.

Shalev-Shwartz, S., Keshet, J., & Singer, Y. (2004). Learning to align polyphonic music. *ISMIR*.

Taskar, B., Guestrin, C., & Koller, D. (2003). Max-margin markov networks. *NIPS*.

Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. *ICML*.

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