Learning Ensembles of Structured Prediction Rules

Corinna Cortes
Google Research
111 8th Avenue,
New York, NY 10011
corinna@google.com

Vitaly Kuznetsov
Courant Institute
251 Mercer Street,
New York, NY 10012
vitaly@cims.nyu.edu

Mehryar Mohri
Courant Institute and Google Research
251 Mercer Street,
New York, NY 10012
mohri@cims.nyu.edu

Abstract

We present a series of algorithms with theoretical guarantees for learning accurate ensembles of several structured prediction rules for which no prior knowledge is assumed. This includes a number of randomized and deterministic algorithms devised by converting on-line learning algorithms to batch ones, and a boosting-style algorithm applicable in the context of structured prediction with a large number of labels. We also report the results of extensive experiments with these algorithms.

1 Introduction

We study the problem of learning accurate ensembles of structured prediction experts. Ensemble methods are widely used in machine learning and have been shown to be often very effective (Breiman, 1996; Freund and Schapire, 1997; Smyth and Wolpert, 1999; MacKay, 1991; Freund et al., 2004). However, ensemble methods and their theory have been developed primarily for binary classification or regression tasks. Their techniques do not readily apply to structured prediction problems. While it is straightforward to combine scalar outputs for a classification or regression problem, it is less clear how to combine structured predictions such as phonemic pronunciation hypotheses, speech recognition lattices, parse trees, or alternative machine translations.

Consider for example the problem of devising an ensemble method for pronunciation, a critical component of modern speech recognition (Ghoshal et al., 2009). Often, several pronunciation models or experts are available for transcribing words into sequences of phonemes. These models may have been derived using other machine learning algorithms or they may be based on carefully hand-crafted rules. In general, none of these pronunciation experts is fully accurate and each expert may be making mistakes at different positions along the output sequence. One can hope that a model that *patches together* the pronunciation of different experts could achieve a superior performance.

Similar ensemble structured prediction problems arise in other tasks, including machine translation, part-of-speech tagging, optical character recognition and computer vision, with structures or substructures varying with each task. We seek to tackle all of these problems simultaneously and consider the general setting where the label or output associated to an input $\mathbf{x} \in \mathcal{X}$ is a structure $\mathbf{y} \in \mathcal{Y}$ that can be decomposed and represented by l substructures $y^1, \ldots, y^l$. For the pronunciation example just discussed, $\mathbf{x}$ is a specific word or word sequence and $\mathbf{y}$ its phonemic transcription. A natural choice for the substructures y^k is then the individual phonemes forming $\mathbf{y}$. Other possible choices include n-grams of consecutive phonemes or more general subsequences.

We will assume that the loss function considered admits an additive decomposition over the substructures, as is common in structured prediction. We also assume access to a set of structured prediction experts $h_1, \ldots, h_p$ that we treat as black boxes. Given an input $\mathbf{x} \in \mathcal{X}$, each expert predicts a structure $h_j(\mathbf{x}) = (h_j^1(\mathbf{x}), \ldots, h_j^l(\mathbf{x}))$. The hypotheses h_j may be the output of a structured prediction algorithm such as Conditional Random Fields (Lafferty et al., 2001), Averaged Perceptron (Collins, 2002), StructSVM (Tsochantaridis et al., 2005), Max Margin Markov Networks (Taskar et al., 2004) or the Regression Technique for Learning Transductions (Cortes et al., 2005), or some other algorithmic or human expert. Given a labeled training sample $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)$, our objective is to use the predictions of these experts

to form an accurate ensemble.

Variants of the ensemble problem just formulated have been studied in the past in the natural language processing and machine learning literature. One of the most recent, and possibly most relevant studies for sequence data is that of (Nguyen and Guo, 2007), which is based on the forward stepwise selection introduced by (Caruana et al., 2004). However, one disadvantage of this greedy approach is that it can be proven to fail to select an optimal ensemble of experts even in favorable cases where a specialized expert is available for each local prediction (Cortes et al., 2014a). Ensemble methods for structured prediction based on bagging, random forests and random subspaces have also been proposed in (Kocev et al., 2013). One of the limitations of this work is that it is applicable only to a very specific class of tree-based experts introduced in that paper. Similarly, a boosting approach was developed in (Wang et al., 2007) but it applies only to local experts. In the context of natural language processing, a variety of different re-ranking techniques have been proposed for somewhat related problems (Collins and Koo, 2005; Zeman and Žabokrtský, 2005; Sagae and Lavie, 2006; Zhang et al., 2009). But, re-ranking methods do not combine predictions at the level of substructures, thus the final prediction of the ensemble coincides with the prediction made by one of the experts, which can be shown to be suboptimal in many cases. Furthermore, these methods typically assume the use of probabilistic models, which is not a requirement in our learning scenario. Other ensembles of probabilistic models have also been considered in text and speech processing by forming a product of probabilistic models via the intersection of lattices (Mohri et al., 2008), or a straightforward combination of the posteriors from probabilistic grammars trained using EM with different starting points (Petrov, 2010), or some other rather intricate techniques in speech recognition (Fiscus, 1997). Finally, an algorithm of (MacKay, 1997) is another example of an ensemble method for structured prediction though it is not addressing directly the problem we are considering.

Most of the references just mentioned do not give a rigorous theoretical justification for the techniques proposed. We are not aware of any prior theoretical analysis for the ensemble structured prediction problem we consider. Here, we present two families of algorithms for learning ensembles of structured prediction rules that both perform well in practice and enjoy strong theoretical guarantees. In Section 3, we develop ensemble methods based on on-line algorithms. To do so, we extend existing on-line-to-batch conversions to our more general setting. In Section 4, we present a new boosting-style algorithm which is applicable even with a large set of classes as in the problem we consider, and for which we present margin-based learning guarantees. Section 5 reports the results of our extensive experiments.[1]

2 Learning scenario

As in standard supervised learning problems, we assume that the learner receives a training sample $S = ((\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)) \in \mathcal{X} \times \mathcal{Y}$ of m labeled points drawn i.i.d. according to the some distribution $\mathcal{D}$ used both for training and testing. We also assume that the learner has access to a set of p predictors $h_1, \ldots, h_p$ mapping $\mathcal{X}$ to $\mathcal{Y}$ to devise an accurate ensemble prediction. Thus, for any input $\mathbf{x} \in \mathcal{X}$, he can use the prediction of the p experts $h_1(\mathbf{x}), \ldots, h_p(\mathbf{x})$. No other information is available to the learner about these p experts, in particular the way they have been trained or derived is not known to the learner. But, we will assume that the training sample S is distinct from what may have been used for training the algorithms that generated $h_1(\mathbf{x}), \ldots, h_p(\mathbf{x})$.

To simplify our analysis, we assume that the number of substructures $l \geq 1$ is fixed. This does not cause any loss of generality so long as the maximum number of substructures is bounded, which is the case in all the applications we consider. The quality of the predictions is measured by a loss function $L \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ that can be decomposed as a sum of loss functions $\ell_k \colon \mathcal{Y}_k \to \mathbb{R}_+$ over the substructure sets $\mathcal{Y}_k$, that is, for all $\mathbf{y} = (y^1, \ldots, y^l) \in \mathcal{Y}$ with $y^k \in \mathcal{Y}_k$ and $\mathbf{y}' = (y'^1, \ldots, y'^l) \in \mathcal{Y}$ with $y'^k \in \mathcal{Y}_k$,

$$L(\mathbf{y}, \mathbf{y}') = \sum_{k=1}^{l} \ell_k(y^k, y'^k). \tag{1}$$

We will assume in all that follows that the loss function L is bounded: $L(\mathbf{y}, \mathbf{y}') \leq M$ for all

[1] This paper is a modified version of (Cortes et al., 2014a) to which we refer the reader for the proofs of the theorems stated and a more detailed discussion of our algorithms.

$(\mathbf{y}, \mathbf{y}')$ for some $M > 0$. A prototypical example of such loss functions is the normalized Hamming loss L_{Ham}, which is the fraction of substructures for which two labels $\mathbf{y}$ and $\mathbf{y}'$ disagree, thus in that case $\ell_k(y^k, y'^k) = \frac{1}{l} I_{y^k \neq y'^k}$ and $M = 1$.

3 On-line learning approach

In this section, we present an on-line learning solution to the ensemble structured prediction problem just discussed. We first give a new formulation of the problem as that of on-line learning with expert advice, where the experts correspond to the paths of an acyclic automaton. The on-line algorithm generates at each iteration a distribution over the path-experts. A critical component of our approach consists of using these distributions to define a prediction algorithm with favorable generalization guarantees. This requires an extension of the existing on-line-to-batch conversion techniques to the more general case of combining distributions over path-experts, as opposed to combining single hypotheses.

3.1 Path experts

Each expert h_j induces a set of substructure hypotheses $h_j^1, \ldots, h_j^l$. As already discussed, one particular expert may be better at predicting the kth substructure while some other expert may be more accurate at predicting another substructure. Therefore, it is desirable to combine the substructure predictions of all experts to derive a more accurate prediction. This leads us to considering an acyclic finite automaton G such as that of Figure 1 which admits all possible sequences of substructure hypotheses, or, more generally, a finite automaton such as that of Figure 2 which only allows a subset of these sequences.

An automaton such as G compactly represents a set of *path experts*: each path from the initial vertex 0 to the final vertex l is labeled with a sequence of substructure hypotheses $h_{j_1}^1, \ldots, h_{j_l}^l$ and defines a hypothesis which associates to input $\mathbf{x}$ the output $h_{j_1}^1(\mathbf{x}) \cdots h_{j_l}^l(\mathbf{x})$. We will denote by H the set of all path experts. We also denote by h each path expert defined by $h_{j_1}^1, \ldots, h_{j_l}^l$, with $j_k \in \{1, \ldots, p\}$, and denote by h^k its kth substructure hypothesis $h_{j_k}^k$. Our ensemble structure prediction problem can then be formulated as that of selecting the best path expert (or collection of path experts) in G. Note that, in general, the path expert selected does not coincide with any of the original experts $h_1, \ldots, h_p$.

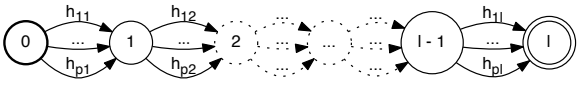

Figure 1: Finite automaton G of path experts.

3.2 On-line algorithm

Using an automaton G, the size of the pool of experts H we consider can be very large. For example, in the case of the automaton of Figure 1, the size of the pool of experts is p^l, and thus is exponentially large with respect to p. But, since learning guarantees in on-line learning admit only a logarithmic dependence on that size, they remain informative in this context. Nevertheless, the computational complexity of most on-line algorithms also directly depends on that size, which could make them impractical in this context. But, there exist several on-line solutions precisely designed to address this issue by exploiting the structure of the experts as in the case of our path experts. These include the algorithm of (Takimoto and Warmuth, 2003) denoted by WMWP, which is an extension of the (randomized) weighted-majority (WM) algorithm of (Littlestone and Warmuth, 1994) to more general bounded loss functions combined with the Weight Pushing (WP) algorithm of (Mohri, 1997); and the Follow the Perturbed Leader (FPL) algorithm of (Kalai and Vempala, 2005). The WMWP algorithm admits a more favorable regret guarantee than the FPL algorithm in our context and our discussion will focus on the use of WMWP for the design of our batch algorithm. However, we have also fully analyzed and implemented a batch algorithm based on FPL (Cortes et al., 2014a).

As in the standard WM algorithm (Littlestone and Warmuth, 1994), WMWP maintains at each round $t \in [1, T]$, a distribution p_t over the set of all experts, which in this context are the path experts $\mathsf{h} \in \mathsf{H}$. At each round $t \in [1, T]$, the algorithm receives an input sequence $\mathbf{x}_t$, incurs the loss $\mathbb{E}_{\mathsf{h} \sim \mathsf{p}_t}[L(\mathsf{h}(\mathbf{x}_t), \mathbf{y}_t)] = \sum_{\mathsf{h}} \mathsf{p}_t(\mathsf{h}) L(\mathsf{h}(\mathbf{x}_t), \mathbf{y}_t)$ and multiplicatively updates the distribution weight per expert:

$$\forall \mathsf{h} \in \mathsf{H}, \mathsf{p}_{t+1}(\mathsf{h}) = \frac{\mathsf{p}_t(\mathsf{h}) \beta^{L(\mathsf{h}(\mathbf{x}_t), \mathbf{y}_t)}}{\sum_{\mathsf{h}' \in \mathsf{H}} \mathsf{p}_t(\mathsf{h}') \beta^{L(\mathsf{h}'(\mathbf{x}_t), \mathbf{y}_t)}}, \quad (2)$$

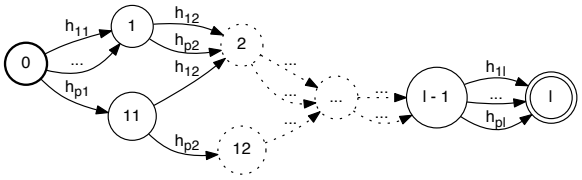

Figure 2: Alternative experts automaton.

where $\beta \in (0,1)$ is some fixed parameter. The number of paths is exponentially large in p and the cost of updating all paths is therefore prohibitive. However, since the loss function is additive in the substructures and the updates are multiplicative, it suffices to maintain instead a weight $w_t(e)$ per transition e, following the update

$$w_{t+1}(e) = \frac{w_t(e)\beta^{\ell_e(\mathbf{x}_t,\mathbf{y}_t)}}{\sum_{orig(e')=\mathrm{orig}(e)} w_t(e')\beta^{\ell_{e'}(\mathbf{x}_t,\mathbf{y}_t)}} \quad (3)$$

where $\ell_e(\mathbf{x}_t, \mathbf{y}_t)$ denotes the loss incurred by the substructure predictor labeling e for the input $\mathbf{x}_t$ and output $\mathbf{y}_t$, and $\mathrm{orig}(e')$ denotes the origin state of a transition e' (Takimoto and Warmuth, 2003). Thus, the cost of the update is then linear in the size of the automaton. To use the resulting weighted automaton for sampling, the weight pushing algorithm is used, whose complexity is also linear in the size of the automaton (Mohri, 1997).

3.3 On-line-to-batch conversion

The WMWP algorithm does not produce a sequence of path experts, rather, a sequence of distributions $\mathsf{p}_1, \ldots, \mathsf{p}_T$ over path experts. Thus, the on-line-to-batch conversion techniques described in (Littlestone, 1989; Cesa-Bianchi et al., 2004; Dekel and Singer, 2005) do not readily apply. Instead, we propose a generalization of the techniques of (Dekel and Singer, 2005). The conversion consists of two steps: extract a good collection of distributions $\mathcal{P} \subseteq \{\mathsf{p}_1, \ldots, \mathsf{p}_T\}$; next use $\mathcal{P}$ to define an accurate hypothesis for prediction. For a subset $\mathcal{P} \subseteq \{\mathsf{p}_1, \ldots, \mathsf{p}_T\}$, we define

$$\Gamma(\mathcal{P}) = \frac{1}{|\mathcal{P}|}\sum_{\mathsf{p}_t \in \mathcal{P}} \sum_{\mathsf{h} \in \mathsf{H}} \mathsf{p}_t(\mathsf{h}) L(\mathsf{h}(\mathbf{x}_t), \mathbf{y}_t) + M\sqrt{\frac{\log \frac{1}{\delta}}{|\mathcal{P}|}}$$
$$= \frac{1}{|\mathcal{P}|}\sum_{\mathsf{p}_t \in \mathcal{P}} \sum_{e} w_t(e)\ell_e(\mathbf{x}_t), \mathbf{y}_t) + M\sqrt{\frac{\log \frac{1}{\delta}}{|\mathcal{P}|}},$$

where $\delta > 0$ is a fixed parameter. With this definition, we choose $\mathcal{P}_\delta$ as a minimizer of $\Gamma(\mathcal{P})$ over some collection $\mathcal{P}$ of subsets of $\{\mathsf{p}_1, \ldots, \mathsf{p}_T\}$: $\mathcal{P}_\delta \in \mathrm{argmin}_{\mathcal{P} \in \mathcal{P}} \Gamma(\mathcal{P})$. The choice of $\mathcal{P}$ is restricted by computational considerations. One natural option is to let $\mathcal{P}$ be the union of the suffix sets $\{\mathsf{p}_t, \ldots, \mathsf{p}_T\}$, $t = 1, \ldots, T$. We will assume in what follows that $\mathcal{P}$ includes the set $\{\mathsf{p}_1, \ldots, \mathsf{p}_T\}$.

Next, we define a randomized algorithm based on $\mathcal{P}_\delta$. Given an input $\mathbf{x}$, the algorithm consists of randomly selecting a path h according to

$$\mathsf{p}(\mathsf{h}) = \frac{1}{|\mathcal{P}_\delta|}\sum_{\mathsf{p}_t \in \mathcal{P}_\delta} \mathsf{p}_t(\mathsf{h}), \quad (4)$$

and returning the prediction $\mathsf{h}(\mathbf{x})$. Note that computing and storing p directly is not efficient. To sample from p, we first choose $\mathsf{p}_t \in \mathcal{P}_\delta$ uniformly at random and then sample a path h according to that p_t. Sampling a path according to p_t can be done efficiently using the weight pushing algorithm. Note that once an input $\mathbf{x}$ is received, the distribution p over the path experts h induces a probability distribution $\mathsf{p}_\mathbf{x}$ over the output space $\mathcal{Y}$. It is not hard to see that sampling a prediction $\mathbf{y}$ according to $\mathsf{p}_\mathbf{x}$ is statistically equivalent to first sampling h according to p and then predicting $\mathsf{h}(\mathbf{x})$. We will denote by $\mathcal{H}_{\mathrm{Rand}}$ the randomized hypothesis thereby generated.

An inherent drawback of randomized solutions such as the one just described is that for the same input $\mathbf{x}$ the user can receive different predictions over time. Randomized solutions are also typically more costly to store. A collection of distributions $\mathcal{P}$ can also be used to define a deterministic prediction rule based on the scoring function approach. The majority vote scoring function is defined by

$$\widetilde{\mathsf{h}}_{\mathrm{MVote}}(\mathbf{x}, \mathbf{y}) = \prod_{k=1}^{l}\Big(\frac{1}{|\mathcal{P}_\delta|}\sum_{\mathsf{p}_t \in \mathcal{P}_\delta}\sum_{j=1}^{p} w_{t,kj}\mathbf{1}_{h_j^k(\mathbf{x})=y^k}\Big). \quad (5)$$

The majority vote algorithm denoted by $\mathcal{H}_{\mathrm{MVote}}$ is then defined for all $\mathbf{x} \in \mathcal{X}$, by $\mathcal{H}_{\mathrm{MVote}}(\mathbf{x}) = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}} \widetilde{\mathsf{h}}_{\mathrm{MVote}}(\mathbf{x}, \mathbf{y})$. For an expert automaton accepting all path experts such as that of Figure 1, the maximizer of $\widetilde{\mathsf{h}}_{\mathrm{MVote}}$ can be found very efficiently by choosing $\mathbf{y}$ such that y^k has the maximum weight in position k.

In the next section, we present learning guarantees for $\mathcal{H}_{\mathrm{Rand}}$ and $\mathcal{H}_{\mathrm{MVote}}$. For a more extensive dis-

cussion of alternative prediction rules, see (Cortes et al., 2014a).

3.4 Batch learning guarantees

We first present learning bounds for the randomized prediction rule $\mathcal{H}_{\text{Rand}}$. Next, we upper bound the generalization error of $\mathcal{H}_{\text{MVote}}$ in terms of that of $\mathcal{H}_{\text{Rand}}$.

Theorem 1. *For any $\delta > 0$, with probability at least $1-\delta$ over the choice of the sample $((\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_T, \mathbf{y}_T))$ drawn i.i.d. according to $\mathcal{D}$, the following inequalities hold:*

$$\mathbb{E}[L(\mathcal{H}_{\text{Rand}}(\mathbf{x}), \mathbf{y})] \leq \inf_{\mathsf{h} \in \mathsf{H}} \mathbb{E}[L(\mathsf{h}(\mathbf{x}), \mathbf{y})] + 2M\sqrt{\frac{l \log p}{T}} + 2M\sqrt{\frac{\log \frac{2T}{\delta}}{T}}.$$

For the normalized Hamming loss L_{Ham}, the bound of Theorem 1 holds with $M = 1$.

We now upper bound the generalization error of the majority-vote algorithm $\mathcal{H}_{\text{MVote}}$ in terms of that of the randomized algorithm $\mathcal{H}_{\text{Rand}}$, which, combined with Theorem 1, immediately yields generalization bounds for the majority-vote algorithm $\mathcal{H}_{\text{MVote}}$.

Proposition 2. *The following inequality relates the generalization error of the majority-vote algorithm to that of the randomized one:*

$$\mathbb{E}[L_{\text{Ham}}(\mathcal{H}_{\text{MVote}}(\mathbf{x}), \mathbf{y})] \leq 2\,\mathbb{E}[L_{\text{Ham}}(\mathcal{H}_{\text{Rand}}(\mathbf{x}), \mathbf{y})],$$

where the expectations are taken over $(\mathbf{x}, \mathbf{y}) \sim \mathcal{D}$ and $\mathsf{h} \sim \mathsf{p}$.

Proposition 2 suggests that the price to pay for derandomization is a factor of 2. More refined and more favorable guarantees can be proven for the majority-vote algorithm (Cortes et al., 2014a).

4 Boosting-style algorithm

In this section, we devise a boosting-style algorithm for our ensemble structured prediction problem. The variants of AdaBoost for multi-class classification such as AdaBoost.MH or AdaBoost.MR (Freund and Schapire, 1997; Schapire and Singer, 1999; Schapire and Singer, 2000) cannot be readily applied in this context. First, the number of classes to consider here is quite large, as in all structured prediction problems, since it is exponential in the number of substructures l. For example, in the case of the pronunciation problem where the number of phonemes for English is in the order of 50, the number of classes is 50^l. But, the objective function for AdaBoost.MH or AdaBoost.MR as well as the main steps of the algorithms include a sum over all possible labels, whose computational cost in this context would be prohibitive. Second, the loss function we consider is the normalized Hamming loss over the substructures predictions, which does not match the multi-class losses for the variants of AdaBoost.[2] Finally, the natural base hypotheses for this problem admit a structure that can be exploited to devise a more efficient solution, which of course was not part of the original considerations for the design of these variants of AdaBoost.

4.1 Hypothesis sets

The predictor $\mathcal{H}_{\text{Boost}}$ returned by our boosting algorithm is based on a scoring function $\widetilde{\mathsf{h}}\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, which, as for standard ensemble algorithms such as AdaBoost, is a convex combination of base scoring functions $\widetilde{\mathsf{h}}_t$: $\widetilde{\mathsf{h}} = \sum_{t=1}^{T} \alpha_t \widetilde{\mathsf{h}}_t$, with $\alpha_t \geq 0$. The base scoring functions used in our algorithm have the form

$$\forall (\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}, \quad \widetilde{\mathsf{h}}_t(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^{l} \widetilde{\mathsf{h}}_t^k(\mathbf{x}, \mathbf{y}).$$

In particular, these can be derived from the path experts in H by letting $\mathsf{h}_t^k(\mathbf{x}, \mathbf{y}) = \mathbf{1}_{\mathsf{h}_t^k(\mathbf{x}) = y^k}$. Thus, the score assigned to $\mathbf{y}$ by the base scoring function $\widetilde{\mathsf{h}}_t$ is the number of positions at which $\mathbf{y}$ matches the prediction of path expert h_t given input $\mathbf{x}$. $\mathcal{H}_{\text{Boost}}$ is defined as follows in terms of $\widetilde{\mathsf{h}}$ or h_ts:

$$\forall \mathbf{x} \in \mathcal{X}, \mathcal{H}_{\text{Boost}}(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \widetilde{\mathsf{h}}(\mathbf{x}, \mathbf{y})$$

We remark that the analysis and algorithm presented in this section are also applicable with a scoring function that is the product of the scores

[2] (Schapire and Singer, 1999) also present an algorithm using the Hamming loss for multi-class classification, but that is a Hamming loss over the set of classes and differs from the loss function relevant to our problem. Additionally, the main steps of that algorithm are also based on a sum over all classes.

at each substructure k as opposed to a sum, that is,

$$\widetilde{\mathsf{h}}(\mathbf{x}, \mathbf{y}) = \prod_{k=1}^{l} \left(\sum_{t=1}^{T} \alpha_t \widetilde{\mathsf{h}}_t^k(\mathbf{x}, \mathbf{y}) \right).$$

This can be used for example in the case where the experts are derived from probabilistic models.

4.2 ESPBoost algorithm

To simplify our exposition, the algorithm that we now present uses base learners of the form $\mathsf{h}_t^k(\mathbf{x}, \mathbf{y}) = \mathbf{1}_{\mathsf{h}_t^k(\mathbf{x}) = y^k}$. The general case can be handled in the same fashion with the only difference being the definition of the direction and step of the optimization procedure described below. For any $i \in [1, m]$ and $k \in [1, l]$, we define the *margin of* $\widetilde{\mathsf{h}}^k$ *for point* $(\mathbf{x}_i, \mathbf{y}_i)$ by $\rho(\widetilde{\mathsf{h}}^k, \mathbf{x}_i, \mathbf{y}_i) = \widetilde{\mathsf{h}}^k(\mathbf{x}_i, y_i^k) - \max_{y^k \neq y_i^k} \widetilde{\mathsf{h}}^k(\mathbf{x}_i, y^k)$. We first derive an upper bound on the empirical normalized Hamming loss of a hypothesis $\mathcal{H}_{\text{Boost}}$, with $\widetilde{\mathsf{h}} = \sum_{t=1}^{T} \alpha_t \widetilde{\mathsf{h}}_t$.

Lemma 3. *The following upper bound holds for the empirical normalized Hamming loss of the hypothesis* $\mathcal{H}_{\text{Boost}}$*:*

$$\mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\sim S}[L_{\text{Ham}}(\mathcal{H}_{\text{Boost}}(\mathbf{x}), \mathbf{y})] \leq \frac{1}{ml} \sum_{i=1}^{m} \sum_{k=1}^{l} \exp\Big(- \sum_{t=1}^{T} \alpha_t \rho(\widetilde{\mathsf{h}}_t^k, \mathbf{x}_i, \mathbf{y}_i) \Big).$$

The proof of this lemma as well as that of several other theorems related to this algorithm can be found in (Cortes et al., 2014a).

In view of this upper bound, we consider the objective function $F\colon \mathbb{R}^N \to \mathbb{R}$ defined for all $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N) \in \mathbb{R}^N$ by

$$F(\boldsymbol{\alpha}) = \frac{1}{ml} \sum_{i=1}^{m} \sum_{k=1}^{l} \exp\Big(- \sum_{j=1}^{N} \alpha_j \rho(\widetilde{\mathsf{h}}_j^k, \mathbf{x}_i, \mathbf{y}_i) \Big),$$

where $\mathsf{h}_1, \ldots, \mathsf{h}_N$ denote the set of all path experts in H. F is a convex and differentiable function of $\boldsymbol{\alpha}$. Our algorithm, ESPBoost (Ensemble Structured Prediction Boosting), is defined by the application of coordinate descent to the objective F. Algorithm 1 shows the pseudocode of the ESPBoost.

Algorithm 1 ESPBoost Algorithm

Inputs: $S = ((\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m))$; set of experts $\{h_1, \ldots, h_p\}$
for $i = 1$ **to** m **and** $k = 1$ **to** l **do**
  $\mathcal{D}_1(i, k) \leftarrow \frac{1}{ml}$
end for
for $t = 1$ **to** T **do**
  $\mathsf{h}_t \leftarrow \operatorname{argmin}_{\mathsf{h} \in \mathsf{H}} \mathbb{E}_{(i,k)\sim \mathcal{D}_t}[\mathbf{1}_{\mathsf{h}^k(\mathbf{x}_i) \neq y_i^k}]$
  $\epsilon_t \leftarrow \mathbb{E}_{(i,k)\sim \mathcal{D}_t}[\mathbf{1}_{\mathsf{h}_t^k(\mathbf{x}_i) \neq y_i^k}]$
  $\alpha_t \leftarrow \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
  $Z_t \leftarrow 2\sqrt{\epsilon_t(1-\epsilon_t)}$
  for $i = 1$ **to** m **and** $k = 1$ **to** l **do**
    $\mathcal{D}_{t+1}(i, k) \leftarrow \frac{\exp(-\alpha_t \rho(\widetilde{\mathsf{h}}_t^k, \mathbf{x}_i, \mathbf{y}_i)) \mathcal{D}_t(i,k)}{Z_t}$
  end for
end for
Return $\widetilde{\mathsf{h}} = \sum_{t=1}^{T} \alpha_t \widetilde{\mathsf{h}}_t$

Let $\boldsymbol{\alpha}_{t-1} \in \mathbb{R}^N$ denote the vector obtained after $t-1$ iterations and $\mathbf{e}_t$ the tth unit vector in $\mathbb{R}^N$. We denote by $\mathcal{D}_t$ the distribution over $[1, m] \times [1, l]$ defined by

$$\mathcal{D}_t(i, k) = \frac{\frac{1}{ml} \exp\Big(- \sum_{u=1}^{t-1} \alpha_u \rho(\widetilde{\mathsf{h}}_u^k, \mathbf{x}_i, \mathbf{y}_i) \Big)}{A_{t-1}}$$

where A_{t-1} is a normalization factor, $A_{t-1} = \frac{1}{ml} \sum_{i=1}^{m} \sum_{k=1}^{l} \exp\big(- \sum_{u=1}^{t-1} \alpha_u \rho(\widetilde{\mathsf{h}}_u^k, \mathbf{x}_i, \mathbf{y}_i) \big)$. The direction $\mathbf{e}_t$ selected at the tth round is the one minimizing the directional derivative, that is

$$\begin{aligned} &\frac{dF(\boldsymbol{\alpha}_{t-1} + \eta \mathbf{e}_t)}{d\eta}\bigg|_{\eta=0} \\ &= - \sum_{i=1}^{m} \sum_{k=1}^{l} \rho(\widetilde{\mathsf{h}}_t^k, \mathbf{x}_i, \mathbf{y}_i) \mathcal{D}_t(i, k) A_{t-1} \\ &= \Big[2 \sum_{i,k:\mathsf{h}_t^k(\mathbf{x}_i) \neq y_i^k} \mathcal{D}_t(i, k) - 1 \Big] A_{t-1} \\ &= (2\epsilon_t - 1) A_{t-1}, \end{aligned}$$

where ϵ_t is the average error of h_t given by

$$\begin{aligned} \epsilon_t &= \sum_{i=1}^{m} \sum_{k=1}^{l} \mathcal{D}_t(i, k) \mathbf{1}_{\mathsf{h}_t^k(\mathbf{x}_i) \neq y_i^k} \\ &= \mathop{\mathbb{E}}_{(i,k)\sim \mathcal{D}_t} [\mathbf{1}_{\mathsf{h}_t^k(\mathbf{x}_i) \neq y_i^k}]. \end{aligned}$$

The remaining steps of our algorithm can be determined as in the case of AdaBoost. In particular, given the direction $\mathbf{e}_t$, the best step α_t is obtained by solving the equation $\frac{dF(\boldsymbol{\alpha}_{t-1} + \alpha_t \mathbf{e}_t)}{d\alpha_t} =$

0, which admits the closed-form solution $\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$. The distribution $\mathcal{D}_{t+1}$ can be expressed in terms of $\mathcal{D}_t$ with the normalization factor $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$.

Our *weak learning assumption* in this context is that there exists $\gamma > 0$ such that at each round, ϵ_t verifies $\epsilon_t < \frac{1}{2} - \gamma$. Note that, at each round, the path expert h_t with the smallest error ϵ_t can be determined easily and efficiently by first finding for each substructure k, the h_t^k that is the best with respect to the distribution weights $\mathcal{D}_t(i,k)$.

Observe that, while the steps of our algorithm are syntactically close to those of AdaBoost and its multi-class variants, our algorithm is distinct and does not require sums over the exponential number of all possible labelings of the substructures and is quite efficient.

4.3 Learning guarantees

We have derived both a margin-based generalization bound in support of the ESPBoost algorithm and a bound on the empirical margin loss.

For any $\rho > 0$, define the empirical margin loss of $\mathcal{H}_{\text{Boost}}$ by the following:

$$\widehat{R}_\rho\left(\frac{\widetilde{\mathsf{h}}}{\|\boldsymbol{\alpha}\|_1}\right) = \frac{1}{ml}\sum_{i=1}^{m}\sum_{k=1}^{l}\mathbf{1}_{\rho(\widetilde{\mathsf{h}}^k,\mathbf{x}_i,\mathbf{y}_i)\leq\rho\|\boldsymbol{\alpha}\|_1},$$

where $\widetilde{\mathsf{h}}$ is the corresponding scoring function. The following theorem can be proven using the multi-class classification bounds of (Koltchinskii and Panchenko, 2002; Mohri et al., 2012) as can be shown in (Cortes et al., 2014a).

Theorem 4. *Let $\mathcal{F}$ denote the set of functions $\mathcal{H}_{\text{Boost}}$ with $\widetilde{\mathsf{h}} = \sum_{t=1}^{T}\alpha_t\mathsf{h}_t$ for some $\alpha_1,\ldots,\alpha_t \geq 0$ and $\mathsf{h}_t \in \mathsf{H}$ for all $t \in [1,T]$. Fix $\rho > 0$. Then, for any $\delta > 0$, with probability at least $1-\delta$, the following holds for all $\mathcal{H}_{\text{Boost}} \in \mathcal{F}$:*

$$\mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}[L_{\text{Ham}}(\mathcal{H}_{\text{Boost}}(\mathbf{x}),\mathbf{y})] \leq \widehat{R}_\rho\left(\frac{\widetilde{\mathsf{h}}}{\|\boldsymbol{\alpha}\|_1}\right) + \frac{2}{\rho l}\sum_{k=1}^{l}|\mathcal{Y}_k|^2\mathfrak{R}_m(H^k) + \sqrt{\frac{\log\frac{l}{\delta}}{2m}},$$

where $\mathfrak{R}_m(H^k)$ denotes the Rademacher complexity of the class of functions

$$H^k = \{\mathbf{x}\mapsto\widetilde{\mathsf{h}}_t^k : j\in[1,p], y\in\mathcal{Y}_k\}.$$

Table 1: Average Normalized Hamming Loss, ADS1 and ADS2. $\beta_{ADS1} = 0.95$, $\beta_{ADS2} = 0.95$, $T_{SLE} = 100$, $\delta = 0.05$.

	ADS1, $m = 200$	ADS2, $m = 200$
$\mathcal{H}_{\text{MVote}}$	**0.0197 ± 0.00002**	**0.2172 ± 0.00983**
$\mathcal{H}_{\text{FPL}}$	0.0228 ± 0.00947	0.2517 ± 0.05322
$\mathcal{H}_{\text{CV}}$	**0.0197 ± 0.00002**	0.2385 ± 0.00002
$\mathcal{H}_{\text{FPL-CV}}$	0.0741 ± 0.04087	0.4001 ± 0.00028
$\mathcal{H}_{\text{ESPBoost}}$	**0.0197 ± 0.00002**	**0.2267 ± 0.00834**
$\mathcal{H}_{\text{SLE}}$	0.5641 ± 0.00044	0.2500 ± 0.05003
$\mathcal{H}_{\text{Rand}}$	0.1112 ± 0.00540	0.4000 ± 0.00018
Best h_j	0.5635 ± 0.00004	0.4000

This theorem provides a margin-based guarantee for convex ensembles such as those returned by ESPBoost. The following theorem further provides an upper bound on the empirical margin loss for ESPBoost.

Theorem 5. *Let $\widetilde{\mathsf{h}}$ denote the scoring function returned by ESPBoost after $T \geq 1$ rounds. Then, for any $\rho > 0$, the following inequality holds:*

$$\widehat{R}_\rho\left(\frac{\widetilde{\mathsf{h}}}{\|\boldsymbol{\alpha}\|_1}\right) \leq 2^T\prod_{t=1}^{T}\sqrt{\epsilon_t^{1-\rho}(1-\epsilon_t)^{1+\rho}}.$$

As in the case of AdaBoost (Schapire et al., 1997), it can be shown that for $\rho < \gamma$, $\epsilon_t^{1-\rho}(1-\epsilon_t)^{1+\rho} \leq (1-2\gamma)^{1-\rho}(1+2\gamma)^{1+\rho} < 1$ and the right-hand side of this bound decreases exponentially with T.

5 Experiments

We used a number of artificial and real-world data sets for our experiments. For each data set, we performed 10-fold cross-validation with disjoint training sets.[3] We report the average error for each task. In addition to the $\mathcal{H}_{\text{MVote}}$, $\mathcal{H}_{\text{Rand}}$ and $\mathcal{H}_{\text{ESPBoost}}$ hypotheses, we experimented with two algorithms discussed in more detail in (Cortes et al., 2014a): a cross-validation on-line-to-batch conversion of the WMWP algorithm, $\mathcal{H}_{\text{CV}}$, a majority-vote on-line-to-batch conversion with FPL, $\mathcal{H}_{\text{FPL}}$, and a cross-validation on-line-to-batch conversion with FPL, $\mathcal{H}_{\text{FPL-CV}}$. Finally, we compare with the $\mathcal{H}_{\text{SLE}}$ algorithm of (Nguyen and Guo, 2007).

5.1 Artificial data sets

Our artificial data set, ADS1 and ADS2 simulate the scenarios described in Section 1. In ADS1 the

[3] For the OCR data set, these subsets are predefined.

kth expert has a high accuracy on the kth position, in ADS2 an expert has low accuracy in a fixed set of positions.

For the first artificial data set, ADS1, we used local experts $h_1, \ldots, h_p$ with $p = 5$. To generate the data we chose an arbitrary Markov chain over the English alphabet and sampled $40{,}000$ random sequences each consisting of 10 symbols. Each of the five experts was designed to have a certain probability of making a mistake at each position in the sequence. Expert h_j correctly predicted positions $2j-1$ and $2j$ with probability 0.97 and other positions with probability 0.5. We forced experts to make similar mistakes by making them select an adjacent alphabet symbol in case of an error. For example, when a mistake was made on a symbol b, the expert prediction was forced to be either a or c. The second artificial data set, ADS2, modeled the case of rather poor experts. ADS2 was generated in the same way as ADS1, but the expert predictions were different. This time each expert made mistakes at four out of the ten distinct random positions in each sequence.

Table 1 reports the results of our experiments. For all experiments with the algorithms $\mathcal{H}_{\text{Rand}}$, $\mathcal{H}_{\text{MVote}}$, and $\mathcal{H}_{\text{CV}}$, we ran the WMWP algorithm for $T = m$ rounds with the β values listed in the caption of Table 1, generating distributions $\mathcal{P} \subseteq \{\mathsf{p}_1, \ldots, \mathsf{p}_T\}$. For $\mathcal{P}$ we used the collection of all suffix sets $\{\mathsf{p}_t, \ldots, \mathsf{p}_T\}$ and $\delta = 0.05$. For the algorithms based on FPL, we used $\epsilon = 0.5/pl$. The same parameter choices were used for the subsequent experiments.

As can be seen from Table 1, in both cases, $\mathcal{H}_{\text{MVote}}$, our majority-vote algorithm based on our on-line-to-batch conversion using the WMWP algorithm (together with most of the other on-line based algorithms), yields a significant improvement over the best expert. It also outperforms $\mathcal{H}_{\text{SLE}}$, which in the case of ADS1 even fails to outperform the best h_j. After 100 iterations on ADS1, the ensemble learned by $\mathcal{H}_{\text{SLE}}$ consists of a single expert, which is why it leads to such a poor performance.

It is also worth pointing out that $\mathcal{H}_{\text{FPL-CV}}$ and $\mathcal{H}_{\text{Rand}}$ fail to outperform the best model on ADS2 set. This is in total agreement with our theoretical analysis since, in this case, any path expert has exactly the same performance and the error of the best path expert is an asymptotic upper bound on the errors of these algorithms. The superior performance of the majority-vote-based algorithms suggests that these algorithms may have an advantage over other prediction rules beyond what is suggested by our learning bounds.

Table 2: Average Normalized Hamming Loss for ADS3. $\beta_{ADS1} = 0.95$, $\beta_{ADS2} = 0.95$, $T_{SLE} = 100$, $\delta = 0.05$.

$\mathcal{H}_{\text{MVote}}$	$\mathbf{0.1788 \pm 0.00004}$
$\mathcal{H}_{\text{FPL}}$	0.2189 ± 0.04097
$\mathcal{H}_{\text{CV}}$	$\mathbf{0.1788 \pm 0.00004}$
$\mathcal{H}_{\text{FPL-CV}}$	0.3148 ± 0.00387
$\mathcal{H}_{\text{ESPBoost}}$	0.1831 ± 0.00240
$\mathcal{H}_{\text{SLE}}$	0.1954 ± 0.00185
$\mathcal{H}_{\text{Rand}}$	0.3196 ± 0.00018
Best h_j	0.2957 ± 0.00005

Table 3: Average Normalized Hamming Loss, PDS1 and PDS2. $\beta_{PDS1} = 0.85$, $\beta_{PDS2} = 0.97$, $T_{SLE} = 100$, $\delta = 0.05$.

	PDS1, $m = 130$	PDS2, $m = 400$
$\mathcal{H}_{\text{MVote}}$	$\mathbf{0.2225 \pm 0.00301}$	$\mathbf{0.2323 \pm 0.00069}$
$\mathcal{H}_{\text{FPL}}$	0.2657 ± 0.07947	0.2337 ± 0.00229
$\mathcal{H}_{\text{CV}}$	$\mathbf{0.2316 \pm 0.00189}$	$\mathbf{0.2364 \pm 0.00080}$
$\mathcal{H}_{\text{FPL-CV}}$	0.4451 ± 0.02743	0.4090 ± 0.01388
$\mathcal{H}_{\text{ESPBoost}}$	0.3625 ± 0.01054	0.3499 ± 0.00509
$\mathcal{H}_{\text{SLE}}$	0.3130 ± 0.05137	0.3308 ± 0.03182
$\mathcal{H}_{\text{Rand}}$	0.4713 ± 0.00360	0.4607 ± 0.00131
Best h_j	0.3449 ± 0.00368	0.3413 ± 0.00067

We also synthesized a third data set, ADS3. Here, we simulated the case where each expert specialized in predicting some subset of the labels. In particular, we generated $40{,}000$ random sequences over the English alphabet in the same way as for ADS1 and ADS2. To generate expert predictions, we partitioned the alphabet into 5 disjoint subsets A_j. Expert j always correctly predicted the label in A_j and the probability of correctly predicting the label not in A_j was set to 0.7. To train the ensemble algorithms, we used a training set of size $m = 200$.

The results are presented in Table 2. $\mathcal{H}_{\text{MVote}}$, $\mathcal{H}_{\text{CV}}$ and $\mathcal{H}_{\text{ESPBoost}}$ achieve the best performance on this data set with a considerable improvement in accuracy over the best expert h_j. We also observe as for the ADS2 experiment that $\mathcal{H}_{\text{Rand}}$ and $\mathcal{H}_{\text{FPL-CV}}$ fail to outperform the best model and approach the accuracy of the best path expert only asymptotically.

Table 4: Average edit distance, PDS1 and PDS2. $\beta_{PDS1} = 0.85$, $\beta_{PDS2} = 0.97$, $T_{SLE} = 100$, $\delta = 0.05$.

	PDS1, $m = 130$	PDS2, $m = 400$
$\mathcal{H}_{\text{MVote}}$	**0.8395 ± 0.01076**	**0.9626 ± 0.00341**
$\mathcal{H}_{\text{FPL}}$	1.0158 ± 0.34379	0.9744 ± 0.01277
$\mathcal{H}_{\text{CV}}$	**0.8668 ± 0.00553**	**0.9840 ± 0.00364**
$\mathcal{H}_{\text{FPL-CV}}$	1.8044 ± 0.09315	1.8625 ± 0.06016
$\mathcal{H}_{\text{ESPBoost}}$	1.3977 ± 0.06017	1.4092 ± 0.04352
$\mathcal{H}_{\text{SLE}}$	1.1762 ± 0.12530	1.2477 ± 0.12267
$\mathcal{H}_{\text{Rand}}$	1.8962 ± 0.01064	2.0838 ± 0.00518
Best h_j	1.2163 ± 0.00619	1.2883 ± 0.00219

5.2 Pronunciation data sets

We had access to two proprietary pronunciation data sets, PDS1 and PDS2. In both sets, each example is an English word, typically a proper name. For each word, 20 possible phonemic sequences are available, ranked by some pronunciation model. Since the true pronunciation was not available, we set the top sequence to be the target label and used the remaining as the predictions made by the experts. The only difference between PDS1 and PDS2 is their size: 1,313 words for PDS1 and 6,354 for PDS2.

In both cases, on-line based algorithms, specifically $\mathcal{H}_{\text{MVote}}$, significantly outperform the best model as well as $\mathcal{H}_{\text{SLE}}$, see Table 3. The poor performance of $\mathcal{H}_{\text{ESPBoost}}$ is due to the fact that the weak learning assumption is violated after 5-8 iterations and hence the algorithm terminates.

It can be argued that for this task the edit-distance is a more suitable measure of performance than the average Hamming loss. Thus, we also report the results of our experiments in terms of the edit-distance in Table 4. Remarkably, our on-line based algorithms achieve a comparable improvement over the performance of the best model in the case of edit-distance as well.

5.3 OCR data set

Rob Kassel's OCR data set is available for download from http://ai.stanford.edu/~btaskar/ocr/. It contains 6,877 word instances with a total of 52,152 characters. Each character is represented by $16 \times 8 = 128$ binary pixels. The task is to predict a word given its sequence of pixel vectors. To generate experts, we used several software packages: CRFsuite (Okazaki, 2007) and SVM^{struct}, SVM^{multiclass} (Joachims, 2008), and the Stanford Classifier (Rafferty et al., 2014). We trained these algorithms on each of the predefined folds of the data set and generated predictions on the test fold using the resulting models.

Table 5: Average Normalized Hamming Loss, TR1 and TR2. $\beta_{TR1} = 0.95$, $\beta_{TR2} = 0.98$, $T_{SLE} = 100$, $\delta = 0.05$.

	TR1, $m = 800$	TR2, $m = 1000$
$\mathcal{H}_{\text{MVote}}$	**0.0850 ± 0.00096**	**0.0746 ± 0.00014**
$\mathcal{H}_{\text{FPL}}$	**0.0859 ± 0.00110**	**0.0769 ± 0.00218**
$\mathcal{H}_{\text{CV}}$	**0.0843 ± 0.00006**	**0.0741 ± 0.00011**
$\mathcal{H}_{\text{FPL-CV}}$	0.1093 ± 0.00129	0.1550 ± 0.00182
$\mathcal{H}_{\text{ESPBoost}}$	0.1041 ± 0.00056	0.1414 ± 0.00233
$\mathcal{H}_{\text{SLE}}$	**0.0778 ± 0.00934**	**0.0814 ± 0.02558**
$\mathcal{H}_{\text{Rand}}$	0.1128 ± 0.00048	0.1652 ± 0.00077
Best h_j	0.1032 ± 0.00007	0.1415 ± 0.00005

Our results (see (Cortes et al., 2014a)) show that ensemble methods lead only to a small improvement in performance over the best h_j. This is because here the best model h_j dominates all other experts and ensemble methods cannot benefit from patching together different outputs.

5.4 Penn Treebank data set

The part-of-speech task, POS, consists of labeling each word of a sentence with its correct part-of-speech tag. The Penn Treebank 2 data set is available through LDC license at http://www.cis.upenn.edu/~treebank/ and contains 251,854 sentences with a total of 6,080,493 tokens and 45 different parts-of-speech.

For the first experiment, TR1, we used 4 disjoint training sets to produce 4 SVM^{multiclass} models and 4 maximum entropy models using the Stanford Classifier. We also used the union of these training sets to devise one CRFsuite model. For the second experiment, TR2, we trained 5 SVM^{struct} models. The same features were used for both experiments. For the SVM algorithms, we generated 267,214 bag-of-word binary features. The Stanford Classifier and CRFsuite packages use internal routines to generate features.

The results of the experiments are summarized in Table 5. For TR1, our on-line ensemble methods improve over the best model. Note that $\mathcal{H}_{\text{SLE}}$ has the best average loss over 10 runs for this experiment. This comes at a price of much higher standard deviation which does not allow us to conclude that the difference in performance between our methods and $\mathcal{H}_{\text{SLE}}$ is statistically significant.

Table 6: Average Normalized Hamming Loss, SDS. $l \geq 4$, $\beta = 0.97$, $\delta = 0.05$, $T_{SLE} = 100$.

	$p = 5, m = 1500$	$p = 10, m = 1200$
$\mathcal{H}_{\text{MVote}}$	**0.2465 ± 0.00248**	**0.2606 ± 0.00320**
$\mathcal{H}_{\text{FPL}}$	0.2500 ± 0.00248	**0.2622 ± 0.00316**
$\mathcal{H}_{\text{CV}}$	0.2504 ± 0.00576	**0.2755 ± 0.00212**
$\mathcal{H}_{\text{FPL-CV}}$	0.2726 ± 0.00839	0.3219 ± 0.01176
$\mathcal{H}_{\text{ESPBoost}}$	0.2572 ± 0.00062	0.2864 ± 0.00103
$\mathcal{H}_{\text{SLE}}$	0.2572 ± 0.00061	0.2864 ± 0.00102
$\mathcal{H}_{\text{Rand}}$	0.2877 ± 0.00480	0.3430 ± 0.00468
Best h_j	0.2573 ± 0.00060	0.2865 ± 0.00101

Table 7: Average Normalized Hamming Loss, SDS. $l \geq 4$, $\beta = 0.97$,$\delta = 0.05$, $T_{SLE} = 100$.

	$p = 20, m = 900$	$p = 50, m = 700$
$\mathcal{H}_{\text{MVote}}$	**0.2773 ± 0.00139**	**0.3217 ± 0.00375**
$\mathcal{H}_{\text{FPL}}$	**0.2797 ± 0.00154**	**0.3189 ± 0.00344**
$\mathcal{H}_{\text{CV}}$	0.2986 ± 0.00075	0.3401 ± 0.00054
$\mathcal{H}_{\text{FPL-CV}}$	0.3816 ± 0.01457	0.4451 ± 0.01360
$\mathcal{H}_{\text{ESPBoost}}$	0.3115 ± 0.00089	0.3426 ± 0.00071
$\mathcal{H}_{\text{SLE}}$	0.3114 ± 0.00087	0.3425 ± 0.00076
$\mathcal{H}_{\text{Rand}}$	0.3977 ± 0.00302	0.4608 ± 0.00303
Best h_j	0.3116 ± 0.00087	0.3427 ± 0.00077

In fact, on two runs, $\mathcal{H}_{\text{SLE}}$ chooses an ensemble consisting of a single expert and fails to outperform the best model.

5.5 Speech recognition data set

For our last set of experiments, we used another proprietary speech recognition data set, SDS. Each example in this data set is represented by a sequence of length $l \in [2, 15]$. Therefore, for training we padded the true labels and the expert predictions to normalize the sequence lengths. For each of the 22,298 examples, there are between 2 and 251 expert predictions available. Since the ensemble methods we presented assume that the predictions of all p experts are available for each example in the training and test sets, we needed to restrict ourselves to the subsets of the data where at least some fixed number of expert predictions were available. In particular, we considered $p = 5, 10, 20$ and 50. For each value of p we used only the top p experts in our ensembles.

Our initial experiments showed that, as in the case of OCR data set, ensemble methods offer only a modest increase in performance over the best h_j. This is again largely due to the dominant performance of the best expert h_j. However, it was observed that the accuracy of the best model is a decreasing function of l, suggesting that ensemble algorithm may be used to improve performance for longer sequences. Subsequent experiments show that this is indeed the case: when training and testing with $l \geq 4$, ensemble algorithms outperform the best model. Table 6 and Table 7 summarize these results for $p = 5, 10, 20, 50$.

Our results suggest that the following simple scheme can be used: for short sequences use the best expert model and for longer sequences, use the ensemble model. A more elaborate variant of this algorithm can be derived based on the observation that the improvement in accuracy of the ensemble model over the best expert increases with the number of experts available.

6 Conclusion

We presented a broad analysis of the problem of ensemble structured prediction, including a series of algorithms with learning guarantees and extensive experiments. Our results show that our algorithms, most notably $\mathcal{H}_{\text{MVote}}$, can result in significant benefits in several tasks, which can be of a critical practical importance. We also reported very favorable results for $\mathcal{H}_{\text{MVote}}$ when used with the edit-distance, which is the standard loss used in many applications. A natural extension of this work consists of devising new algorithms and providing learning guarantees specific to other loss functions such as the edit-distance. While we aimed for an exhaustive study, including multiple on-learning algorithms, different conversions to batch and derandomizations, we are aware that the problem we studied is very rich and admits many more facets and scenarios that we plan to investigate in the future. Finally, the boosting-style algorithm we presented can be enhanced using recent theoretical and algorithmic results on *deep boosting* (Cortes et al., 2014b).

Acknowledgments

We warmly thank our colleagues Francoise Beaufays and Fuchun Peng for kindly extracting and making available to us the pronunciation data sets, Cyril Allauzen for providing us with the speech recognition data, and Richard Sproat and Brian Roark for help with other data sets. This work was partly funded by the NSF award IIS-1117591 and the NSERC PGS D3 award.

References

[Breiman1996] Leo Breiman. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.

[Caruana et al.2004] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes. 2004. Ensemble selection from libraries of models. In *Proceedings of ICML*, pages 18–.

[Cesa-Bianchi et al.2004] N. Cesa-Bianchi, A. Conconi, and C. Gentile. 2004. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057.

[Collins and Koo2005] Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

[Collins2002] M. Collins. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of ACL*, pages 1–8.

[Cortes et al.2005] C. Cortes, M. Mohri, and J. Weston. 2005. A general regression technique for learning transductions. In *Proceedings of ICML 2005*, pages 153–160, New York, NY, USA. ACM.

[Cortes et al.2014a] Corinna Cortes, Vitaly Kuznetsov, and Mehryar Mohri. 2014a. Ensemble methods for structured prediction. In *Proceedings of ICML*.

[Cortes et al.2014b] Corinna Cortes, Mehryar Mohri, and Umar Syed. 2014b. Deep boosting. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML 2014)*.

[Dekel and Singer2005] O. Dekel and Y. Singer. 2005. Data-driven online to batch conversion. In *Advances in NIPS 18*, pages 1207–1216.

[Fiscus1997] Jonathan G Fiscus. 1997. Post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover). In *Proceedings of the 1997 IEEE ASRU Workshop*, pages 347–354, Santa Barbara, CA.

[Freund and Schapire1997] Y. Freund and R. Schapire. 1997. A decision-theoretic generalization of on-line learning and application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

[Freund et al.2004] Yoav Freund, Yishay Mansour, and Robert E. Schapire. 2004. Generalization bounds for averaged classifiers. *The Annals of Statistics*, 32:1698–1722.

[Ghoshal et al.2009] Arnab Ghoshal, Martin Jansche, Sanjeev Khudanpur, Michael Riley, and Morgan Ulinski. 2009. Web-derived pronunciations. In *Proceedings of ICASSP*, pages 4289–4292.

[Joachims2008] T. Joachims. 2008. Support vector machines for complex outputs.

[Kalai and Vempala2005] A. Kalai and S. Vempala. 2005. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307.

[Kocev et al.2013] D. Kocev, C. Vens, J. Struyf, and S. Deroski. 2013. Tree ensembles for predicting structured outputs. *Pattern Recognition*, 46(3):817–833, March.

[Koltchinskii and Panchenko2002] Vladmir Koltchinskii and Dmitry Panchenko. 2002. Empirical margin distributions and bounding the generalization error of combined classifiers. *Annals of Statistics*, 30.

[Lafferty et al.2001] J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289.

[Littlestone and Warmuth1994] N. Littlestone and M. Warmuth. 1994. The weighted majority algorithm. *Information and Computation*, 108(2):212–261.

[Littlestone1989] N. Littlestone. 1989. From on-line to batch learning. In *Proceedings of COLT 2*, pages 269–284.

[MacKay1991] David J. C. MacKay. 1991. *Bayesian methods for adaptive models*. Ph.D. thesis, California Institute of Technology.

[MacKay1997] David J.C. MacKay. 1997. Ensemble learning for hidden markov models. Technical report, Cavendish Laboratory, Cambridge UK.

[Mohri et al.2008] Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. 2008. Speech recognition with weighted finite-state transducers. In *Handbook on Speech Processing and Speech Communication, Part E: Speech recognition*. Springer-Verlag.

[Mohri et al.2012] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. 2012. *Foundations of Machine Learning*. The MIT Press.

[Mohri1997] Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311.

[Nguyen and Guo2007] N. Nguyen and Y. Guo. 2007. Comparison of sequence labeling algorithms and extensions. In *Proceedings of ICML*, pages 681–688.

[Okazaki2007] N. Okazaki. 2007. CRFsuite: a fast implementation of conditional random fields (crfs).

[Petrov2010] Slav Petrov. 2010. Products of random latent variable grammars. In *HLT-NAACL*, pages 19–27.

[Rafferty et al.2014] A. Rafferty, A. Kleeman, J. Finkel, and C. Manning. 2014. Stanford classifer.

[Sagae and Lavie2006] K. Sagae and A. Lavie. 2006. Parser combination by reparsing. In *Proceedings of HLT/NAACL*, pages 129–132.

[Schapire and Singer1999] Robert E. Schapire and Yoram Singer. 1999. Improved boosting algorithms

using confidence-rated predictions. *Machine Learning*, 37(3):297–336.

[Schapire and Singer2000] Robert E. Schapire and Yoram Singer. 2000. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2-3):135–168.

[Schapire et al.1997] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. 1997. Boosting the margin: A new explanation for the effectiveness of voting methods. In *ICML*, pages 322–330.

[Smyth and Wolpert1999] Padhraic Smyth and David Wolpert. 1999. Linearly combining density estimators via stacking. *Machine Learning*, 36:59–83, July.

[Takimoto and Warmuth2003] E. Takimoto and M. K. Warmuth. 2003. Path kernels and multiplicative updates. *JMLR*, 4:773–818.

[Taskar et al.2004] B. Taskar, C. Guestrin, and D. Koller. 2004. Max-margin Markov networks. In *Advances in NIPS 16*. MIT Press, Cambridge, MA.

[Tsochantaridis et al.2005] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. 2005. Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484, December.

[Wang et al.2007] Q. Wang, D. Lin, and D. Schuurmans. 2007. Simple training of dependency parsers via structured boosting. In *Proceedings of IJCAI 20*, pages 1756–1762.

[Zeman and Žabokrtský2005] D. Zeman and Z. Žabokrtský. 2005. Improving parsing accuracy by combining diverse dependency parsers. In *Proceedings of IWPT 9*, pages 171–178.

[Zhang et al.2009] H. Zhang, M. Zhang, C. Tan, and H. Li. 2009. K-best combination of syntactic parsers. In *Proceedings of EMNLP: Volume 3*, pages 1552–1560.