Ensemble Methods for Structured Prediction

Corinna Cortes CORINNA@GOOGLE.COM
Google Research, 111 8th Avenue, New York, NY 10011

Vitaly Kuznetsov VITALY@CIMS.NYU.EDU
Courant Institute of Mathematical Sciences, 251 Mercer Street, New York, NY 10012

Mehryar Mohri MOHRI@CIMS.NYU.EDU
Courant Institute and Google Research, 251 Mercer Street, New York, NY 10012

Abstract

We present a series of learning algorithms and theoretical guarantees for designing accurate ensembles of structured prediction tasks. This includes several 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 give a detailed study of all these algorithms, including the description of new on-line-to-batch conversions and learning guarantees. We also report the results of extensive experiments with these algorithms in several structured prediction tasks.

1. Introduction

Ensemble methods are general techniques in machine learning for combining several hypotheses to create a more accurate predictor (Breiman, 1996; Freund & Schapire, 1997; Smyth & Wolpert, 1999; MacKay, 1991; Freund et al., 2004). These methods often significantly improve the performance in practice and additionally benefit from favorable learning guarantees. However, ensemble methods and their theory have been developed primarily for the binary classification problem or regression tasks. These 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 outputs of several machine translation systems.

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 in 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 of these experts predicts l substructures $h_j(\mathbf{x}) = (h_j^1(\mathbf{x}), \ldots, h_j^l(\mathbf{x}))$. The hypotheses h_j may be the output of other structured prediction algorithms 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), 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 combine the predictions of these experts to form an accurate ensemble.

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

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 & Guo (2007), which is based on the forward stepwise selection procedure introduced by Caruana et al. (2004). Starting with a possibly empty collection of experts, E_0, that algorithm performs T iterations. To make predictions using a collection of models, E_t, a variant of a majority-vote scheme per position is proposed, and at each iteration t, a new expert h_j from $\{h_1, \ldots, h_p\}$ is added to the collection E_{t-1} in such a way that $E_t = E_{t-1} \cup \{h_j\}$ has the best performance on the training set among all sets $E_{t-1} \cup \{h_j\}$, $j = 1, \ldots, p$. This algorithm always performs at least as well as the best expert among $h_1, \ldots, h_p$ on the training set. If the initial collection E_0 of experts is empty, then E_1 simply contains the expert with the smallest error on the training set. Further additions to E_t only decrease that error, hence the performance of this algorithm on the training set cannot be worse than the performance of the best expert.

One disadvantage of this greedy approach is that it may fail to select an optimal ensemble of experts in cases where experts specialize in local predictions. Consider the case where expert h_k is a strong predictor for the kth substructure but does not perform well on other substructures. Assume further that expert h_0 is a jack-of-all-trades and performs better than any of $h_1, \ldots, h_p$ on average, but each h_k beats h_0 at position k. Then, one can show that the stepwise selection routine may end up with an ensemble consisting of only h_0, while an optimal solution would use expert h_k to predict the kth substructure. We provide an explicit construction of such an example in Appendix I and report similar empirical observations in Section 5.

Ensemble methods for structured prediction based on bagging, random forests and random subspaces have 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 & Koo, 2005; Zeman & Žabokrtský, 2005; Sagae & 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). See Appendix J for a brief discussion of other related work.

Most of the references 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 aim to bridge this gap and develop ensemble methods that both perform well in practice and enjoy strong theoretical guarantees. Two families of algorithms are introduced. 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. A boosting-type algorithm is also presented and analyzed in Section 4. Section 5 reports the results of our extensive experiments.

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 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 available to learn the ensemble is distinct from what may 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 by some $M > 0$: $L(\mathbf{y}, \mathbf{y}') \leq M$ for all $(\mathbf{y}, \mathbf{y}')$.

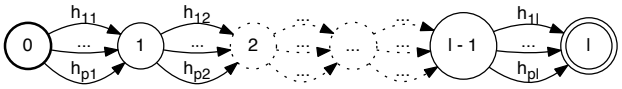

Figure 1. Directed graph G of path experts.

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.

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 formulate the problem as that of on-line learning with expert advice, where the experts correspond to the paths of a directed graph. The on-line algorithm generates at each iteration a distribution over the path-experts. A critical component of our approach consists of using the distributions to define a prediction algorithm with good 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 (instead of combining intermediate 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 a directed graph G such as that of Figure 1 which admits $l+1$ vertices $0, 1, \ldots, l$ and an edge from vertex k to vertex $k+1$ labeled with each of the p hypotheses $h_1^k, \ldots, h_p^k$ induced by the experts $h_1, \ldots, h_p$ for the kth substructure. Graph 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 p^l 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 the graph G. Note that, in general, the path expert selected does not coincide with any of the original experts $h_1, \ldots, h_p$.

More generally, our paths experts can be selected from a directed acyclic graph of experts G' distinct from G, as illustrated by Figure 2. This can be motivated by scenarios

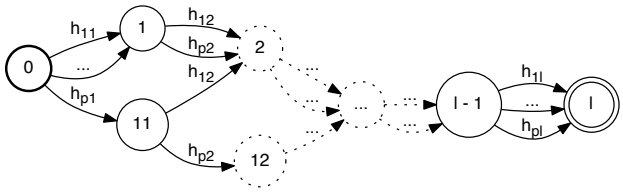

Figure 2. Alternative graph G'.

where some prior knowledge is available about the expert predictions for different substructures (see Appendix A), which could be related to phonotactic constraints, as in the example of pronunciation sequences, or any other prior constraint on illegal n-grams or other subsequences that would result in ruling out certain paths of graph G.

For convenience, in what follows, we will discuss our algorithms and solutions in the specific case of the graph G. However, the on-line learning algorithms we use apply in the same way to an arbitrary directed acyclic graph G'. The randomized algorithm we describe can also be used in a similar way and our batch learning guarantees for our randomized algorithm can be straightforwardly extended to an arbitrary graph G'. In fact, those guarantees are then somewhat more favorable since the number of path experts in G' will be smaller than in G.

3.2. On-line algorithm

Using G, the size of the pool of experts H we consider 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. However, the computational complexity of most on-line algorithms also directly depends on that size, which would 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 & Warmuth (2003) denoted by WMWP, which is an extension of the (randomized) weighted-majority (WM) algorithm of Littlestone & Warmuth (1994) (see also (Vovk, 1990)) to more general bounded loss functions[1] combined with the directed graph Weight Pushing (WP) algorithm of Mohri (1997), and the Follow the Perturbed Leader (FPL) algorithm of Kalai & Vempala (2005).

The basis for the design of our batch algorithms is the WMWP algorithm since it admits a more favorable regret guarantee than the FPL algorithm in our context. However, we have also carried out a full analysis based on the FPL

[1]The extension of the weighted majority algorithm to other losses is also known as the *Hedge algorithm* (Freund & Schapire, 1997) or the *exponentially weighted averaged algorithm* (Cesa-Bianchi & Lugosi, 2006).

Algorithm 1 WMWP algorithm.

Inputs: sample $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_T, \mathbf{y}_T)\}$; set of experts $\{h_1, \ldots, h_p\}$; parameter $\beta \in (0,1)$.
for $j = 1$ **to** p **and** $k = 1$ **to** l **do**
  $w_{1,kj} \leftarrow \frac{1}{p}$
end for
for $t = 1$ **to** T **and** $j = 1$ **to** p **and** $k = 1$ **to** l **do**
  $w_{t+1,kj} \leftarrow \frac{w_{t,kj}\beta^{\ell_k(h_j^k(\mathbf{x}_t),\mathbf{y}_t)}}{\sum_{j=1}^p w_{t,kj}\beta^{\ell_k(h_j^k(\mathbf{x}_t),\mathbf{y}_t)}}$
end for
Return matrices $\{\mathbf{W}_1, \ldots, \mathbf{W}_T\}$

algorithm which can be found in Appendix D.

As in the standard WM algorithm (Littlestone & Warmuth, 1994), WMWP maintains a distribution p_t, $t \in [1,T]$, 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:

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

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, the updates are multiplicative, and p_t can be compactly represented and updated by maintaining a potential value stored at each vertex (Takimoto & Warmuth, 2003). The cost of the update is then linear in the size of the graph.

The graph G we consider has a specific structure, thus, our description of the algorithm can be further simplified by maintaining at any round $t \in [1,T]$, an edge weight $w_{t,kj}$ for the jth edge, $j \in [1,p]$, between vertices $k-1$ and k. This defines a matrix $\mathbf{W}_t = (w_{t,kj})_{kj} \in \mathbb{R}^{l\times p}$ with the following properties:

1. for any path expert h defined by $h_{j_1 1}, \ldots, h_{j_l l}$, $\mathsf{p}_t(\mathsf{h}) = \prod_{k=1}^l w_{t,kj_k}$;
2. the weights of outgoing edges sum to one at any vertex $k \in [0, l-1]$: $\sum_{j=1}^p w_{t,kj} = 1$.

This clearly ensures that $\sum_{\mathsf{h}\in\mathsf{H}} \mathsf{p}_t(\mathsf{h}) = 1$ with the update rule (2) replaced by the following equivalent and more efficient edge weight update:

$$w_{t+1,kj} = \frac{w_{t,kj}\beta^{\ell_k(h_j^k(\mathbf{x}_t),\mathbf{y}_t)}}{\sum_{j=1}^p w_{t,kj}\beta^{\ell_k(h_j^k(\mathbf{x}_t),\mathbf{y}_t)}}. \quad (3)$$

Algorithm 1 gives the pseudocode of WMWP.

3.3. On-line-to-batch conversion

The WMWP algorithm does not produce a sequence of path experts, rather, it produces a sequence of distributions $\mathsf{p}_1, \ldots, \mathsf{p}_T$ over path experts, or equivalently a sequence of matrices $\mathbf{W}_1, \ldots, \mathbf{W}_T$. Thus, the on-line-to-batch conversion techniques described in (Littlestone, 1989; Cesa-Bianchi et al., 2004; Dekel & Singer, 2005) do not readily apply. Instead, we propose a generalization of the techniques of Dekel & Singer (2005). The conversion consists of two steps: first 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

$$\Gamma(\mathcal{P}) = \frac{1}{|\mathcal{P}|}\sum_{\mathsf{p}_t\in\mathcal{P}}\sum_{\mathsf{h}\in\mathsf{H}} \mathsf{p}_t(\mathsf{h}) L(\mathsf{h}(\mathbf{x}_t), \mathbf{y}_t) + M\sqrt{\frac{\log\frac{1}{\delta}}{|\mathcal{P}|}} \quad (4)$$
$$= \frac{1}{|\mathcal{P}|}\sum_{\mathsf{p}_t\in\mathcal{P}}\sum_{k=1}^l\sum_{j=1}^p w_{t,kj}\ell_k(h_j^k(\mathbf{x}_t), y_t^k) + M\sqrt{\frac{\log\frac{1}{\delta}}{|\mathcal{P}|}},$$

where $\delta > 0$ is a fixed parameter. The second equality in (4) is a straightforward consequence of the identity $\mathsf{p}_t(\mathsf{h}) = \prod_{k=1}^l w_{t,kj_k}$ and the additive decomposition of L in terms of l_ks (see Lemma 6 in the appendix). With this definition, we choose $\mathcal{P}_\delta$ as a minimizer of $\Gamma(\mathcal{P})$ over some collection $\mathcal{P}$ of subsets of $\{\mathsf{p}_1, \ldots, \mathsf{p}_T\}$: $\mathcal{P}_\delta \in \operatorname{argmin}_{\mathcal{P}\in\mathcal{P}} \Gamma(\mathcal{P})$. 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

$$\mathsf{p}(\mathsf{h}) = \frac{1}{|\mathcal{P}_\delta|}\sum_{\mathsf{p}_t\in\mathcal{P}_\delta} \mathsf{p}_t(\mathsf{h}). \quad (5)$$

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. Observe that for any fixed $k \in [1,l]$, $\sum_{j=1}^l w_{t,kj} = 1$, thus the non-negative weights $w_{t,kj}$ define a distribution over the edges leaving vertex k that we denote by $w_{t,k\cdot}$. Thus, to sample h from p_t we can simply draw an edge from each of the l distributions $w_{t,k\cdot}$ (the probability mass of a path is the product of the probability masses of its edges). 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 $\mathcal{H}_{\text{Rand}}$ 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, however, also be used to define a deterministic prediction rule based on the scoring function approach. The majority vote scoring function is defined by

$$\widetilde{\mathsf{h}}_{\text{MVote}}(\mathbf{x}, \mathbf{y}) = \prod_{k=1}^{l} \Big(\frac{1}{|\mathcal{P}_\delta|} \sum_{\mathsf{p}_t \in \mathcal{P}_\delta} \sum_{j=1}^{p} w_{t,kj} \mathbf{1}_{h_j^k(\mathbf{x}) = y^k} \Big). \quad (6)$$

The majority vote algorithm denoted by $\mathcal{H}_{\text{MVote}}$ is then defined by $\mathcal{H}_{\text{MVote}}(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} \widetilde{\mathsf{h}}_{\text{MVote}}(\mathbf{x}, \mathbf{y}), \mathbf{x} \in \mathcal{X}$. In the case of the graph G, the maximizer of $\widetilde{\mathsf{h}}_{\text{MVote}}$ is found 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 $\mathcal{H}_{\text{Rand}}$ and $\mathcal{H}_{\text{MVote}}$. We also briefly discuss alternative prediction rules in Appendix E.

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 $\mathcal{H}_{\text{Rand}}$.

Proposition 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 inequality holds:*

$$\mathbb{E}[L(\mathcal{H}_{\text{Rand}}(\mathbf{x}), \mathbf{y})] \leq \frac{1}{T} \sum_{t=1}^{T} L_t + M \sqrt{\frac{\log \frac{1}{\delta}}{T}},$$

where $L_t = \mathbb{E}_{\mathsf{h} \sim \mathsf{p}_t}[L(\mathsf{h}(\mathbf{x}_t), \mathbf{y}_t)]$.

Proof. Let $\mathcal{P} = \{\mathsf{p}_{t_1}, \ldots, \mathsf{p}_{t_{|\mathcal{P}|}}\}$. Observe that

$$\begin{aligned} &\mathbb{E}[L(\mathcal{H}_{\text{Rand}}(\mathbf{x}), \mathbf{y})] - \frac{1}{|\mathcal{P}|} \sum_{s=1}^{|\mathcal{P}|} L_{t_s} \\ &= \sum_{s=1}^{|\mathcal{P}|} \sum_{\mathsf{h} \in \mathsf{H}} \frac{\mathsf{p}_{t_s}(\mathsf{h})}{|\mathcal{P}|} (\mathbb{E}[L(\mathsf{h}(\mathbf{x}), \mathbf{y})] - L(\mathsf{h}(\mathbf{x})_{t_s}, \mathbf{y}_{t_s})). \end{aligned}$$

We denote the inner summand by A_s and observe that A_s forms a martingale difference with respect to the filtration $\mathcal{G}_s = \mathcal{F}_{t_s}$ associated with the process $(\mathbf{x}_t, \mathbf{y}_t)$, i.e. $\mathcal{F}_t$ is a σ-algebra generated by $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_t, \mathbf{y}_t)$. Indeed,

$$\begin{aligned} \mathbb{E}[A_s | \mathcal{G}_{s-1}] = \frac{1}{|\mathcal{P}|} \sum_{\mathsf{h}} &\mathbb{E}[\mathsf{p}_{t_s}(\mathsf{h})\, \mathbb{E}[L(\mathsf{h}(\mathbf{x}), \mathbf{y})] | \mathcal{G}_{s-1}] \\ &- \mathbb{E}[\mathsf{p}_{t_s}(\mathsf{h}) L(\mathsf{h}(\mathbf{x}_{t_s}), \mathbf{y}_{t_s}) | \mathcal{G}_{s-1}]. \end{aligned}$$

Since p_t is determined by $\mathcal{F}_{t-1}$ and $(\mathbf{x}_t, \mathbf{y}_t)$ is independent of $\mathcal{F}_{t-1}$, we can write

$$\begin{aligned} &\mathbb{E}[\mathsf{p}_{t_s}(\mathsf{h}) L(\mathsf{h}(\mathbf{x}_{t_s}), \mathbf{y}_{t_s}) | \mathcal{G}_{s-1}] \\ &\quad = \mathop{\mathbb{E}}_{1:t_s-1}[\mathop{\mathbb{E}}_{t_s}[\mathsf{p}_{t_s}(\mathsf{h}) L(\mathsf{h}(\mathbf{x}_{t_s}), \mathbf{y}_{t_s})] | \mathcal{G}_{s-1}] \\ &\quad = \mathop{\mathbb{E}}_{1:t_s-1}[\mathsf{p}_{t_s}(\mathsf{h}) \mathop{\mathbb{E}}_{t_s}[L(\mathsf{h}(\mathbf{x}_{t_s}), \mathbf{y}_{t_s})] | \mathcal{G}_{s-1}] \end{aligned}$$

where $\mathbb{E}_{1:q}$ indicates that the expectation is taken with respect to $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_q, \mathbf{y}_q)$. This shows that $\mathbb{E}[A_s | \mathcal{G}_{s-1}] = 0$, which implies that A_s is a martingale difference sequence. Since $|A_s| \leq M/|\mathcal{P}|$, it follows from Azuma's inequality that the probability of the event

$$\left\{ \mathbb{E}[L(\mathcal{H}_{\text{Rand}}(\mathbf{x}), \mathbf{y})] - \frac{1}{|\mathcal{P}|} \sum_{s=1}^{|\mathcal{P}|} L_{t_s} > M \sqrt{\frac{\log \frac{1}{\delta}}{|\mathcal{P}|}} \right\}$$

is at most δ. Since $\mathcal{P}_\delta$ is a minimizer of $\frac{1}{|\mathcal{P}|} \sum_{s=1}^{|\mathcal{P}|} L_{t_s} + M \sqrt{\frac{\log \frac{1}{\delta}}{|\mathcal{P}|}}$ over $\mathcal{P}$ and $\mathcal{P}$ contains $\{\mathsf{p}_1, \ldots, \mathsf{p}_T\}$, the desired conclusion follows. □

The next step consists of relating the expected loss of $\mathcal{H}_{\text{Rand}}$ to the regret R_T of the WMWP algorithm:

$$R_T = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\mathsf{h} \sim \mathsf{p}_t} [L(\mathsf{h}(\mathbf{x}_t), \mathbf{y}_t)] - \inf_{\mathsf{h} \in \mathsf{H}} \sum_{t=1}^{T} L(\mathsf{h}(\mathbf{x}_t), \mathbf{y}_t). \quad (7)$$

Theorem 2. *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:*

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

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

See Appendix B for a proof of this result. 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 2, immediately yields generalization bounds for the majority-vote algorithm $\mathcal{H}_{\text{MVote}}$. The first proposition, which admits a simple proof, relates the expected loss of the majority vote algorithm to that of a randomized algorithm in the case of the normalized Hamming loss.

Proposition 3. *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 $\mathsf{h} \sim \mathsf{p}$.

Proof. By definition of the majority vote, if $\mathcal{H}_{\text{MVote}}$ makes an error at position k on example $(\mathbf{x}, \mathbf{y})$, then, the total weight of incorrect labels at that position must be at least half of the total weight of labels in that position. In other words, the following inequality holds for any k:

$$\mathbf{1}_{\mathcal{H}^k_{\text{MVote}}(\mathbf{x}) \neq y^k} \leq 2 \frac{1}{|\mathcal{P}_\delta|} \sum_{\mathsf{p}_t \in \mathcal{P}_\delta} \sum_{j=1}^{p} w_{t,kj} \mathbf{1}_{h^k_j(\mathbf{x}) \neq y^k}.$$

Summing up these inequalities over k and taking expectations yields the desired bound. □

Proposition 3 suggests that the price to pay for derandomization is a factor of 2. However, this may be too pessimistic. A more refined result presented in the following proposition shows that often this price is lower.

Proposition 4. *The following bound holds for any distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$:*

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

where $\gamma(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^{l} \gamma_k(\mathbf{x}, \mathbf{y})$ *with* $\gamma_k(\mathbf{x}, \mathbf{y}) = \max\left(0, \frac{1}{|\mathcal{P}_\delta|} \sum_{\mathsf{p}_t \in \mathcal{P}_\delta} \sum_{j=1}^{p} w_{t,kj} \mathbf{1}_{h^k_j(\mathbf{x}) \neq y^k} - \frac{1}{2}\right)$.

The proof is a refinement of the proof of Proposition 3 and can be found in Appendix B. Each γ_k in Proposition 4 can be interpreted as the edge of incorrect labels and this result implies that any additional edge of an incorrect hypothesis (beyond $\frac{1}{2}$) should not be included in the bound.

Our methods generalize the results of Dekel & Singer (2005) where $l = 1$ and each p_t is a probability point mass at a hypothesis h_t produced by an on-line algorithm on the tth iteration. It is also possible to extend the cross-validation approach of Cesa-Bianchi et al. (2004) to our setting, but the learning guarantees for this algorithm end up being less favorable than those just given (see Appendix C for a full description and analysis). Our results and algorithms can be extended to the case of other directed acyclic graphs of path experts and other derandomization methods (see Appendix E for a more detailed discussion).

4. Boosting approach

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 & Schapire, 1997; Schapire & Singer, 1999; 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 and loss function

The predictor $\mathcal{H}_{\text{ESPBoost}}$ 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 we consider for our problem are derived from the path experts in H. For each path expert $\mathsf{h}_t \in \mathsf{H}$, we define a scoring function $\widetilde{\mathsf{h}}_t$ as follows:

$$\forall (\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}, \quad \widetilde{\mathsf{h}}_t(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^{l} \mathbf{1}_{\mathsf{h}^k_t(\mathbf{x}) = y^k}. \tag{8}$$

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 $\mathbf{x}$. $\mathcal{H}_{\text{ESPBoost}}$ is defined as follows in terms of $\widetilde{\mathsf{h}}$ or h_ts:

$$\begin{aligned} \forall \mathbf{x} \in \mathcal{X},\ \mathcal{H}_{\text{ESPBoost}}(\mathbf{x}) &= \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \widetilde{\mathsf{h}}(\mathbf{x}, \mathbf{y}) \\ &= \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \sum_{k=1}^{l} \sum_{t=1}^{T} \alpha_t \mathbf{1}_{\mathsf{h}^k_t(\mathbf{x}) = y^k}. \end{aligned} \tag{9}$$

4.2. ESPBoost algorithm

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 $\rho(\widetilde{\mathsf{h}}^k, \mathbf{x}_i, \mathbf{y}_i) = \widetilde{\mathsf{h}}^k(\mathbf{x}_i, y^k_i) - \max_{y^k \neq y^k_i} \widetilde{\mathsf{h}}^k(\mathbf{x}_i, y^k)$.

Lemma 5. *The following upper bound holds for the empirical normalized Hamming loss of the hypothesis $\mathcal{H}_{\text{ESPBoost}}$:*

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

[2] Schapire & 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.

Algorithm 2 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 $\widetilde{\mathsf{h}} = \sum_{t=1}^{T}\alpha_t\widetilde{\mathsf{h}}_t$

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

$$F(\boldsymbol{\alpha}) = \frac{1}{ml}\sum_{i=1}^{m}\sum_{k=1}^{l}\exp\Big(-\sum_{j=1}^{N}\alpha_j\rho(\widetilde{\mathsf{h}}_j^k, \mathbf{x}_i, \mathbf{y}_i)\Big), \quad (10)$$

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 2 shows the pseudocode of the ESPBoost (see Appendix G.2 for the details of the derivation of the coordinate descent algorithm).

Our *weak learning assumption* in this context is that there exists $\gamma > 0$ such that at each round, ϵ_t verifies $\epsilon_t < \frac{1}{2} - \gamma$. For the graph G, 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. We have derived margin-based learning guarantees for ESPBoost which are presented in detail and proven in Appendix G.3.

5. Experiments

We used a number of artificial and real-life data sets for our experiments. For each data set, we performed 10-fold cross-validation with disjoint training sets.[3] We report the average test 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 the appendix: a cross-validation on-line-to-batch conversion of the WMWP algorithm, $\mathcal{H}_{\text{CV}}$, and 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 & Guo (2007).

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

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

Table 2. 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, $m = 400$
$\mathcal{H}_{\text{MVote}}$	**0.2225 ± 0.00301**	**0.2323 ± 0.00069**
$\mathcal{H}_{\text{FPL}}$	0.2657 ± 0.07947	0.2337 ± 0.00229
$\mathcal{H}_{\text{CV}}$	**0.2316 ± 0.00189**	**0.2364 ± 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

5.1. Artificial data sets

Our artificial data set, ADS1 and ADS2 simulate the scenarios described in Section 1. In ADS1 the kth expert has a high accuracy on the kth position, in ADS2 an expert has low accuracy in a fixed set of positions. More details on the data set and the experimental parameters can be found in Appendix H.1.

Table 1 reports the results of our experiments. 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.

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 outperformed the best model as well as $\mathcal{H}_{\text{SLE}}$, see Table 2. 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.

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 used the resulting models to generate expert predictions.

The results reported in Table 7 in Appendix H show that ensemble methods lead only to a small improvement in performance over the best h_j. This is because 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 their features. For more detail, see Appendix H.

Table 3. 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

The results of the experiments are summarized in Table 3. 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. In fact, on two runs $\mathcal{H}_{\text{SLE}}$ chooses an ensemble consisting of a single expert and fails to outperform the best model.

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. In Appendix H, we also report very favorable results for $\mathcal{H}_{\text{MVote}}$ when used with the edit-distance, which is the natural measure 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.

The extension of our algorithms and solutions to other directed graphs, as discussed in Appendix E, can further increase the applicability of our methods and enhance performance. 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.

Acknowledgments

We warmly thank our colleagues Francoise Beaufays and Fuchun Peng for kindly extracting and making available to us the pronunciation data sets used for our experiments 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

Breiman, L. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

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

Cesa-Bianchi, N. and Lugosi, G. *Prediction, learning, and games*. Cambridge University Press, 2006.

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

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

Collins, M. and Koo, T. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70, 2005.

Cortes, C., Mohri, M., and Weston, J. A general regression technique for learning transductions. In *Proceedings of ICML 2005*, pp. 153–160, 2005.

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

Fiscus, J. G. Post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER). In *Proceedings of ASRU*, pp. 347–354, 1997.

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

Freund, Y., Mansour, Y., and Schapire, R. Generalization bounds for averaged classifiers. *Ann. Stat.*, 32:1698–1722, 2004.

Ghoshal, A., Jansche, M., Khudanpur, S., Riley, M., and Ulinski, M. Web-derived pronunciations. In *Proceedings of ICASSP*, pp. 4289–4292, 2009.

Joachims, T. Support vector machines for complex outputs, 2008. URL http://www.cs.cornell.edu/people/tj/svm_light/svm_struct.html.

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

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

Koltchinskii, V. and Panchenko, D. Empirical margin distributions and bounding the generalization error of combined classifiers. *Ann. Stat.*, 30:1–50, 2002.

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

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

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

MacKay, D. J. C. *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology, 1991.

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

Mohri, M. and Riley, M. An efficient algorithm for the n-best-strings problem. In *Interspeech*, 2002.

Mohri, M., Pereira, F. C. N., and Riley, M. Speech recognition with weighted finite-state transducers. In *Handbook on Speech Processing and Speech Communication, Part E: Speech recognition*. Springer-Verlag, 2008.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. *Foundations of Machine Learning*. The MIT Press, 2012.

Nguyen, N. and Guo, Y. Comparison of sequence labeling algorithms and extensions. In *Proc. of ICML*, pp. 681–688, 2007.

Okazaki, N. CRFsuite: a fast implementation of conditional random fields (crfs), 2007. URL http://www.chokkan.org/software/crfsuite/.

Petrov, S. Products of random latent variable grammars. In *HLT-NAACL*, pp. 19–27, 2010.

Rafferty, A., Kleeman, A., Finkel, J., and Manning, C. Stanford classifer, 2014. URL http://nlp.stanford.edu/downloads/classifier.shtml.

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

Schapire, R. and Singer, Y. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

Schapire, R. and Singer, Y. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39:135–168, 2000.

Schapire, R., Freund, Y., Bartlett, P., and Lee, W. S. Boosting the margin: A new explanation for the effectiveness of voting methods. In *ICML*, pp. 322–330, 1997.

Smyth, P. and Wolpert, D. Linearly combining density estimators via stacking. *Machine Learning*, 36:59–83, 1999.

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

Taskar, B., Guestrin, C., and Koller, D. Max-margin Markov networks. In *Advances in NIPS 16*. MIT Press, 2004.

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

Vovk, V. G. Aggregating strategies. In *COLT*, pp. 371–386, 1990.

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

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

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