On-line Learning Approach to Ensemble Methods for Structured Prediction [*]

Corinna Cortes
Google Research
76 Ninth 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 & 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. We also report the results of experiments with these algorithms on various structured prediction tasks.

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 [Breiman1996, Freund and Schapire1997, Smyth and Wolpert1999, MacKay1991, 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.

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 example if we want to devise a correct pronunciation for a text transcription, $\mathbf{x}$ 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 $\mathbf{y}$.

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 have been derived using other machine learning algorithms or they may be based on carefully hand-crafted rules. 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.

A number of ensemble methods for structured prediction has been previously proposed in machine learning and natural language processing literature [Nguyen and Guo2007, Kocev et al.2013,

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

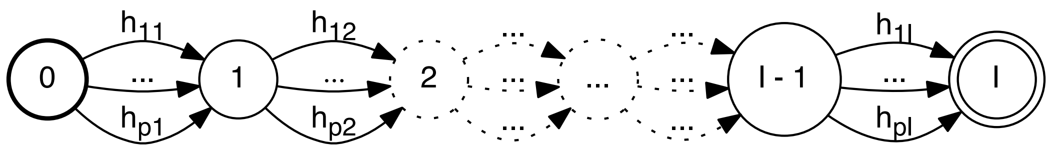

Figure 1: Finite automaton G of path experts.

Wang et al.2007, Collins and Koo2005, Zeman and Žabokrtský2005, Sagae and Lavie2006, Zhang et al.2009, Mohri et al.2008, Petrov2010, Fiscus1997, MacKay1997]. Most of the references just mentioned do not give a rigorous theoretical justification for the techniques proposed. See [Cortes et al.2014a, Cortes et al.2014b] for the detailed overview. We are not aware of any prior theoretical analysis for the ensemble structured prediction problem we consider. Here, we present a family of algorithms for learning ensembles of structured prediction rules that both perform well in practice and enjoy strong theoretical guarantees. Also, see [Cortes et al.2014a, Cortes et al.2014b] for a boosting-style algorithm.

We adopt a standard supervised learning scenario, assuming 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. 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. For a fixed $l \geq 1$, 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 $(\mathbf{y}, \mathbf{y}')$ 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 $M = 1$.

2 On-line learning approach

In this section, we present an on-line learning solution to the ensemble structured prediction problem just discussed. Observe that each expert h_j induces a set of substructure hypotheses $h_j^1, \ldots, h_j^l$. 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, any acyclic finite automaton. 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 $h_1, \ldots, h_p$.

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 $\mathsf{p}_t(\mathsf{h})$. Since the loss function is additive in the substructures and the updates are multiplicative, it suffices to maintain instead a weight $w_t(e)$ for each transition e in G, following the update

$$w_{t+1}(e) = \frac{w_t(e)\beta^{\ell_e(\mathbf{x}_t,\mathbf{y}_t)}}{\sum_{orig(e')=\mathrm{orig}(e)} w_t(e')\beta^{\ell_{e'}(\mathbf{x}_t,\mathbf{y}_t)}} \tag{2}$$

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 Warmuth2003]. 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 [Mohri1997].

This on-line 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 [Littlestone1989, Cesa-Bianchi et al.2004, Dekel and Singer2005] do not readily apply. Instead, we propose a generalization of the techniques of [Dekel and Singer2005]. 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 $\mathscr{P}$ of subsets of $\{\mathsf{p}_1, \ldots, \mathsf{p}_T\}$: $\mathcal{P}_\delta \in \mathrm{argmin}_{\mathcal{P}\in\mathscr{P}} \Gamma(\mathcal{P})$. The choice of $\mathscr{P}$ is restricted by computational considerations. One natural option is to let $\mathscr{P}$ be the union of the suffix sets $\{\mathsf{p}_t, \ldots, \mathsf{p}_T\}$, $t \in [1, T]$. We will assume in what follows that $\mathscr{P}$ 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 $\mathsf{p}(\mathsf{h}) = |\mathcal{P}_\delta|^{-1}\sum_{\mathsf{p}_t\in\mathcal{P}_\delta} \mathsf{p}_t(\mathsf{h})$, and returning the prediction $\mathsf{h}(\mathbf{x})$. 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.

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

$$\widetilde{\mathsf{h}}_{\text{MVote}}(\mathbf{x},\mathbf{y}) = \prod_{k=1}^{l}\Big(\frac{1}{|\mathcal{P}_\delta|}\sum_{\mathsf{p}_t\in\mathcal{P}_\delta}\sum_{j=1}^{p} w_{t,kj}\mathbf{1}_{h_j^k(\mathbf{x})=y^k}\Big). \tag{3}$$

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

We conclude this section with learning bounds for prediction rules $\mathcal{H}_{\text{Rand}}$ and $\mathcal{H}_{\text{MVote}}$.

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:*

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

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

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

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

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

More refined and more favorable guarantees as well as the proofs can be found in [Cortes et al.2014a].

Table 1: Average Normalized Hamming Loss (left) and Average edit-distance (right), PDS1 and PDS2.

	PDS1, $m = 130$	PDS2, $m = 400$
$\mathcal{H}_{\text{MVote}}$	**0.2225 ± 0.00301**	**0.2323 ± 0.00069**
$\mathcal{H}_{\text{SLE}}$	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

	PDS1, $m = 130$	PDS2, $m = 400$
$\mathcal{H}_{\text{MVote}}$	**0.8395 ± 0.01076**	**0.9626 ± 0.00341**
$\mathcal{H}_{\text{SLE}}$	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

Table 2: 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**
$\mathcal{H}_{\text{SLE}}$	**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

3 Experiments

We used a number of artificial and real-world data sets for our experiments. Here we present results for $\mathcal{H}_{\text{MVote}}$ and $\mathcal{H}_{\text{Rand}}$ on two of these data sets: Penn Treebank and a pronunciation data set and compare it to state-of-the-art results of $\mathcal{H}_{\text{SLE}}$ of [Nguyen and Guo2007]. Results for other data sets, comparison against other algorithms, as well as further details on the experimental setup can be found in [Cortes et al.2014a, Cortes et al.2014b].

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/` For our experiments we have generated a number of different base experts by training SVM^{struct}, SVM^{multiclass}, CRF and maximum entropy models. TR1 and TR2 experiments only differ by the collection of models that was used as base experts. The results of the experiments are summarized in Table 2. For both TR1 and TR2, our on-line ensemble methods improve over the best model.

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. The only difference between PDS1 and PDS2 is their size: 1,313 words for PDS1 and 6,354 for PDS2. In both cases $\mathcal{H}_{\text{MVote}}$, significantly outperform the best model as well as $\mathcal{H}_{\text{SLE}}$, see Table 1. 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. 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.

4 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.

Acknowledgments

We warmly thank our colleagues Francoise Beaufays and Fuchun Peng for kindly extracting and making available to us the pronunciation data sets, and Cyril Allauzen, 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.

[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.

[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, Vitaly Kuznetsov, and Mehryar Mohri. 2014b. Learning ensembles of structured prediction rules. In *Proceedings of ACL*.

[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.

[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.

[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.

[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.

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

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

[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.

[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.