Scalable Large-Margin Online Learning for Structured Classification

Koby Crammer **Ryan McDonald** **Fernando Pereira**
Department of Computer and Information Science, University of Pennsylvania
Levine Hall, 3330 Walnut Street, Philadelphia, PA 19104
{crammer,ryantm,pereira}@cis.upenn.edu

Abstract

We investigate large-margin online learning algorithms for large-scale structured classification tasks, focusing on a structured-output extension of MIRA, the multi-class classification algorithm of Crammer and Singer [5]. The extension approximates the parameter updates in MIRA using k-best structural decoding. We evaluate the algorithm on several sequential classification tasks, showing that it offers a competitive, numerically stable, and computationally scalable alternative to conditional random fields and maximum margin Markov networks.

1 Introduction

Structured classification has seen much recent interest in machine learning. After the introduction of conditional random fields (CRFs) [9], several researchers developed margin-based learning alternatives, in particular maximum margin Markov networks (M^3N) [15] and the related methods of Tsochantaridis et al. [19]. All of these methods are in theory batch learning algorithms, in which the training objective is optimized with respect to all training instances simultaneously. However, in practice the large-margin methods require either purely online updates or semi-online updates in order to run tractably. These algorithms have proven successful in many real world applications including sequential classification [10, 13, 15], image labeling [7], natural language parsing [16, 19] and Web page classification [15]. However, it is still unclear how well the large-margin algorithms of Taskar et al. [15, 16] and Tsochantaridis et al. [19] scale computationally compared with other methods. For example, experiments on large-margin training of lexicalized natural-language parsers [16, 19] had to be restricted for computational reasons to sentences shorter than 15 words, which is only 22% of the Penn Treebank.

This paper focuses on pure online learning techniques. Unlike batch algorithms, online algorithms consider only one training instance at a time when optimizing parameters. This restriction to single-instance optimization might be seen as a weakness, since the algorithm uses less information about the objective function and constraints than batch algorithms. However, we will argue that this potential weakness balanced by the simplicity of online learning, which allows the use of more streamlined training methods. We focus here on variants of the perceptron algorithm [12], which inherit its conceptual and mathematical simplicity and scale up to large problems much better than batch algorithms.

Online learning with perceptron-style algorithms has recently gained popularity for struc-

tured classification due in large part to their simplicity and their generally robust empirical performance. Collins [1] uses an approximation to the voted perceptron algorithm [6], called here the averaged perceptron algorithm, to achieve state-of-the-art results for sequential classification problems. Collins and Roark [2] also use the averaged perceptron algorithm, but apply it to the much more computationally demanding problem of natural language lexicalized phrase-structure parsing with promising results.

In this work, we experiment with a large-margin online algorithm that generalizes the multi-class classification algorithm MIRA (Margin Infused Relaxed Algorithm, Crammer and Singer [5, 4]) to structured outputs. The generalization is achieved by using k-best structural decoding to approximate the large margin updates of MIRA. We evaluate the algorithm on handwriting recognition, base noun phrase chunking and named-entity recognition. All three tasks are large-scale benchmark language processing problems.

Our main argument is that the large-margin online method presented here represents a "sweet spot" in structured learning. Its accuracy is competitive with batch learners, but it is conceptually simpler conceptually and scales much better.

2 Online Inference Learning

We start with a general online framework for learning to make structured decisions. The learning algorithm observes a set of structured classification examples. Each example $(\boldsymbol{x}, \boldsymbol{y})$ is composed of an instance $\boldsymbol{x} \in \mathcal{X}$ and a vector of labels or *labeling* $\boldsymbol{y} \in \mathcal{Y}^{n_{\boldsymbol{x}}}$, where $n_{\boldsymbol{x}}$ is the *size* — number of classification tasks — of input $\boldsymbol{x}$ and $\mathcal{Y} = \{1, \ldots, q\}$ is the set of possible classification labels. [1] The ith component of $\boldsymbol{y}$ will be denoted by y_i. A *prediction rule* is a function of the form $h(\boldsymbol{x}) = \arg\max_{\boldsymbol{z} \in \mathcal{Y}^n} \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{z})$ parameterized by a vector $\boldsymbol{w} \in \mathbb{R}^D$, for some fixed *feature function* $\Phi : \mathcal{X} \times \mathcal{Y}^n \to \mathbb{R}^D$. The value of the inner product $\boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{z})$ is called the *score* of labeling $\boldsymbol{z}$. In general, the time required to make a single prediction is exponential in n. We continue previous line of research (see [20, 15] and the reference therein) and use the following factorization assumption. Each example $(\boldsymbol{x}, \boldsymbol{y})$ is associated with an undirected graph $G_{\boldsymbol{x}} = (V_{\boldsymbol{x}}, E_{\boldsymbol{x}})$.[1] The vertex set V is the index set for the labeling, $V = \{1, \ldots, n\}$; $E \subseteq V \times V$. Given G, we assume that the function Φ is a sum of simpler functions, each dependent only on a single vertex or an edge of G:

$$\Phi(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} \boldsymbol{\phi}_i(\boldsymbol{x}, y_i) + \sum_{\{i,j\} \in E} \boldsymbol{\phi}_{i,j}(\boldsymbol{x}, y_i, y_j) \; . \tag{1}$$

Finally, we denote by $N(i) \in V$ the set of neighbors of the vertex i, $N(i) = \{j \in V : \{i, j\} \in E\}$.

Given a single example, our algorithm is characterized by the choice of a set $\mathcal{C}$ of inequality constraints associated to the example. Each constraint in $\mathcal{C}$ has the form $\boldsymbol{w} \cdot \Psi \geq \eta$ for $\Psi \in \mathbb{R}^D$ and $\eta \geq 0$, and is thus linear in $\boldsymbol{w}$. We represent the constraint by the pair (Ψ, η). A prediction which satisfies $\mathcal{C}$ is called a *perfect* prediction. Consider the simplest case of binary classification in which we set $\mathcal{Y} = \{1, 2\}$ and $n = 1$. Given an example $(\boldsymbol{x}, \boldsymbol{y} = 1)$, we set $\mathcal{C} = \{(\Psi = \Phi(\boldsymbol{x}, 1) - \Phi(\boldsymbol{x}, 2), \eta = 1)\}$ (correct prediction with a large margin). Many online and batch learning algorithms can be stated using this notion of constraint set.

Online algorithms maintain a parameter vector $\boldsymbol{w}$ and work in rounds. At each round, the algorithm receives an instance $\boldsymbol{x}$ and outputs a prediction. It then receives the correct labeling $\boldsymbol{y}$. Depending on the disagreement between the prediction and the correct labeling, the algorithm updates its prediction rule by adjusting $\boldsymbol{w}$. We denote by $\boldsymbol{w}'$ the value of

[1] For conciseness, in what follows we will suppress the subscript $\boldsymbol{x}$ in $n_{\boldsymbol{x}}$ and other values that depend on $\boldsymbol{x}$.

Initialize: Set $\boldsymbol{w} = 0 \in \mathbb{R}^D$.
Loop: For $i = 1, 2, \ldots, m$

- Get a new instance: $\boldsymbol{x} \in \mathbb{R}^n$
- Get the correct labeling $\boldsymbol{y} \in \mathcal{Y}^k$.
- Let $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k\} = \text{best}_k(\boldsymbol{x})$.
- Generate the constraints $\mathcal{C}$
  - $\Psi_i = \Phi(\boldsymbol{x}, \boldsymbol{y}) - \Phi(\boldsymbol{x}, \boldsymbol{z}_i)$
  - $\eta_i = \text{Hamming}(\boldsymbol{y}, \boldsymbol{z}_i)$
- Set $\boldsymbol{w}$ to be the solution of the following problem:

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & \tfrac{1}{2} \|\boldsymbol{u} - \boldsymbol{w}\|^2 \\ \text{subject to :} \quad & \forall (\Psi, \eta) \in \mathcal{C} \quad \boldsymbol{u} \cdot \Psi \geq \eta \end{aligned}$$

Output : $h(\boldsymbol{x}) = \arg\max_{\boldsymbol{z}} \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{z})$.

Figure 1: k-best MIRA

the parameter vector before seeing the current example $\boldsymbol{x}$. Our approach is based on an inference oracle best_k, which given an example $\boldsymbol{x}$ and a current model $\boldsymbol{w}$ outputs the set of k labelings $\text{best}_k(\boldsymbol{x}) = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k\}$ which achieve the highest score. Given these k vectors, we define the constraint set to be $\mathcal{C} = \{(\Psi_i = \Phi(\boldsymbol{x}, \boldsymbol{y}) - \Phi(\boldsymbol{x}, \boldsymbol{z}_i), \eta_i) : \boldsymbol{z}_i \neq \boldsymbol{y}\}$.

To fully define our algorithm we need to provide both the threshold values η_i and how we preform the update. Before doing so we like to show how a variant of the perceptron algorithm [12] can be defined for such problems. This variant was first suggested by Collins [1]. In Collins's version, we set $k = 1$ and the threshold to be zero, $\eta_1 = 0$. The algorithm modifies $\boldsymbol{w}$ only if the constraints are violated. If so, it replaces $\boldsymbol{w}$ with a linear combination of itself and the constraint $\boldsymbol{w} \leftarrow \boldsymbol{w}' + \Psi$. This method is very simple and performs quite well. However, it is still not as good as state-of-the-art methods, and it is not clear how to generalize it to $k > 1$.

In this paper we build on the framework of the MIRA algorithm, which is a passive-aggressive learning algorithm [5, 4] designed originally for multi-class problems, and later extended to other problems [14, 3]. In this framework, given a set of constraints, we set $\boldsymbol{w}$ to be a solution of the optimization problem $\boldsymbol{w} = \arg\min_{\boldsymbol{u}} \frac{1}{2} \|\boldsymbol{u} - \boldsymbol{w}'\|^2$ subject to $\forall (\Psi, \eta) \in \mathcal{C} \quad \boldsymbol{u} \cdot \Psi \geq \eta$. For binary classification, in which $\eta = 1$ and $\mathcal{C}$ contains a single constraint, either $\Psi = \Phi(\boldsymbol{x}, 1) - \Phi(\boldsymbol{x}, 2)$ [if $\boldsymbol{y} = 1$] or $\Psi = \Phi(\boldsymbol{x}, 2) - \Phi(\boldsymbol{x}, 1)$ [if $\boldsymbol{y} = 2$]. In the case of structured learning, we set η_i to be the Hamming distance between $\boldsymbol{z}_i$ and the correct labeling $\boldsymbol{y}$. A skeleton of the general algorithm is given in Fig. 1, which we call k-best MIRA since it relies on the output of a k-best oracle to create a set of tractable margin constraints. A variant of the top-1 version was suggested independently [14].

Tsochantaridis et al. [19] use a somewhat related method. At each iteration, they find the most violated constraint for that iteration's instance, which for the Hamming loss can be done as efficiently as finding the highest-scoring instance. This constraint is added to a constraint set, and the dual parameters for all of those constraints are computed by dual quadratic programming. Hence, this method is semi-online since constraints are gathered in an online fashion, but optimization is made over constraints gathered for all instances. One major problem with this approach is that on the later iterations of the learning process, many large QPs will need to be solved since the set of constraints will have become quite large. Because of this, Tsochantaridis et al. were only able to run experiments on small subsets of standard named-entity extraction data sets.

Another option is to use the complete set of constraints. (There is then no need for the oracle.) The result will be an optimization problem with exponentially number of constraints.

(a) handwriting (degree 1/degree 2)

Method	Accuracy
Avg. Perc.	0.781 / 0.859
MIRA	0.785 / 0.870
CRF	0.802 / 0.818
M^3N	0.803 / 0.869

(b) NP chunking

Method	F-Meas
Avg. Perc.	0.941
MIRA	0.942
CRFs	0.943

(c) NE recognition

Method	F-Meas
Avg. Perc.	0.823
MIRA	0.830
CRFs	0.831

Table 1: Experimental results

In max-margin Markov networks (M^3N), Taskar et al. [15] use a clever factoring of this exponential number of constraints into a polynomial number of marginal constraints. However, this reduced set of constraints includes also equality constraints and slack variables and demands especially crafted, carefully tuned quadratic solvers.

Two observations are in order. First, k-best MIRA equates structured learning to multi-class classification with small number of classes, which are defined per example in an online setting. Second, two types of errors may arise by using the top k configurations only. The first is that incorrect classifications with low scores may not be separated from the correct classification with a margin proportional to their loss. The second, more alarming, problem is that the algorithm may make an update that will cause an incorrect classification with a low score to score higher then the correct one. Both problems arise because updates do not consider low scoring classifications. In practice, the second error does not seem to occur.

Note also that k-best MIRA does not use directly the factorization Φ assumed above. However, the factorization may be relied upon by the inference algorithm that finds the classifications with highest score. For the sequence tasks we experiment with, the factorization allows the use of a simple extension of Viterbi decoding to build a dynamic programming table that keeps the k-best ways of reaching each input position, instead of just the best way. This modification increases inference runtime by a factor of k.

3 Experiments and Results

Our first set of experiments are the on the handwriting recognition task used by Taskar et al. [15] to evaluate M^3N, which is a subset of an earlier evaluation collection [8]. We use exactly the same data set, training and test splits, instance representation, and first and second degree kernel data representations. For this experiment we compared averaged perceptron, k-best MIRA ($k = 20$), CRFs and M^3N. Looking at Table 1(a), we can see that both MIRA and M^3N outperform the averaged perceptron, most likely due to the fact that they more aggressively maximize margin. Furthermore, we can see the performance of MIRA is comparable to that of M^3N for the degree 2 kernel. In the impoverished degree 1 kernel, both CRFs and M^3N appear to be the best option. Interestingly, CRFs perform as well as M^3N for the degree 1 kernel, but do much worse than all methods for the degree 2 kernel. The reason for this is the use of feature normalization [15], which improves the performance of all methods dramatically except for CRFs, whose performance seems independent of normalization. Feature normalization artificially increases the value of the edge bias feature to prevent it from being overwhelmed by the overall magnitude of the vertex features in the model, in particular for the second-degree kernel.

Our text experiments involved two larger sequential classification problems: noun-phrase chunking and named-entity recognition. For these problems, we used a feature-based primal formulation, which is commonly used in natural-language processing and easier to implement. For noun-phrase chunking we implemented a first-order Markov model using the features of Sha and Pereira [13] on the CoNLL 2000 data set [17]. For named-entity recognition we used standard word and sub-word features on the CoNLL-2003 data set [18],

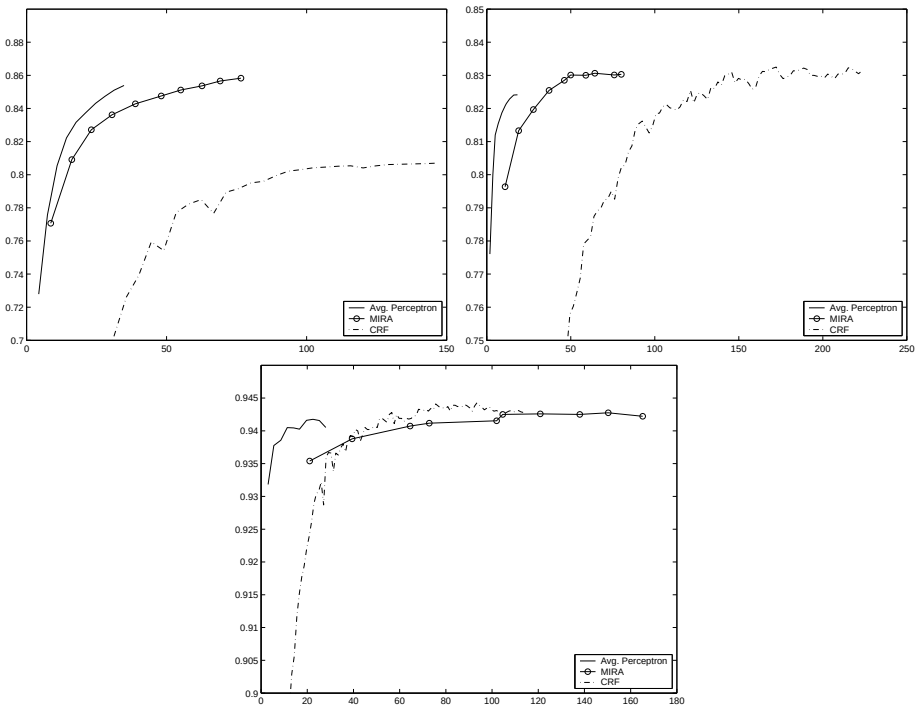

Figure 2: Hand writing recognition (degree 2 kernel), named-entity recognition and noun-phrase chunking. The plots show performance on testing vs. training time in CPU minutes.

including word, part-of-speech, suffix and prefix identity as well as standard orthographic features (e.g. word is capitalized), with all features over a window of two words previous to two words next.

These data sets contain thousands of sentences (roughly 9000 for chunking and 14,000 for named-entity recognition). For both tasks we compared three methods: averaged perceptron, k-best MIRA ($k = 5$) and CRFs. We implemented a version of M^{3}N but found that it took too long to train on the full data sets. Training takes under 5 hours for all the other algorithms. The results are shown in Table 1(b-c). We can see that MIRA slightly outperforms the averaged perceptron and begins to bridge the performance gap with batch learners such as CRFs. According to McNemar significance tests, the difference between MIRA and averaged perceptron is only significant for named-entity recognition ($p < 0.0001$) and the difference between MIRA and CRFs is not significant.

For the larger data sets (chunking and entity extraction) we observed that maximum performance is achieved with $5 \leq k \leq 10$, with sometimes diminishing returns when $k > 20$, providing evidence that the approximation is reasonable. The results for MIRA also include parameter averaging, which only slightly improved performance for MIRA. However, for perceptron, averaging is required to achieve the reported performance.

These experiments show that k-best MIRA performs as well as non-approximated batch methods such as CRFs and M^{3}N, while only depending on simple k-best inference to make updates to the parameters. As we will see in the next section, this results in faster training.

3.1 Perfomance vs. Training Time

Due to the complexity of the output space, learning with structured outputs is inherently more computationally demanding than simple classifier learning. In fact, many studies on batch learning for structured outputs use small or reduced data sets to fit within available computing budgets [15, 16, 19]. In contrast, the online learning methods studied here require only k-best inference on a single instance, and solving a small quadratic program. Fig. 2 plots the test accuracy for each method against CPU training time for all three tasks studied in this paper. We only compare averaged perceptron, MIRA and CRFs, since our implementation of M^3N could not be run on the entire chunking or entity extraction data sets in a reasonable amount of time. Results for these plots were gathered on a dual processor 1.3GHz 32-bit Pentium with 2GB of memory.

These plots show that learning with k-best MIRA and perceptron is typically much less costly than learning with CRFs. A training epoch has roughly the same computational cost, due to inference, for the three methods (Viterbi for perceptron and MIRA, forward-backward for CRFs). However, MIRA and perceptron quickly converge after a few iterations, whereas CRFs require many iterations before attaining high test accuracy.

4 Conclusions and Future Work

We have presented a scalable approach to large-margin online learning that directly optimizes the margin for each training instance individually. We have shown that these algorithms both scale well for complex structured outputs and perform competitively relative to standard batch learners and other online algorithms. We are also using these algorithms for more complex problems such as parsing [11], and we have started investigating their use in additional tasks, including recovering relations and semantic information from text for information extraction, and image segmentation.

References

[1] M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP*, 2002.

[2] M. Collins and B. Roark. Incremental parsing with the perceptron algorithm. In *Proc. of ACL*, 2004.

[3] K. Crammer. *Online Learning for Complex Categorial Problems*. PhD thesis, Hebrew University of Jerusalem, 2005. to appear.

[4] K. Crammer, O. Dekel, S. Shalev-Shwartz, and Y. Singer. Online passive aggressive algorithms. In *Proc. NIPS*, 2003.

[5] K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 2003.

[6] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

[7] X. He, R. Zemel, and M. Carreira-Perpinan. Multiscale conditional random fields for image labelling. In *Proc. of CVPR*, 2004.

[8] R. Kassel. *A comparison of approaches to on-line character handwritten digit recognition*. PhD thesis, MIT Spoken Language Systems Group, 1995.

[9] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*, 2001.

[10] A. McCallum and W. Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proc. of CoNLL*, 2003.

[11] R. McDonald, K. Crammer, and F. Pereira. Online large-margin training of dependency parsers. In *Proc. ACL*, 2005.

[12] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psych. Rev.*, 68:386–407, 1958.

[13] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proc. of HLT-NAACL*, 2003.

[14] S. Shalev-Shwartz, J. Keshet, and Y. Singer. Learning to align polyphonic music. In *5th International Conference on Music Information Retrieval*, 2004.

[15] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Proc. of NIPS*, 2003.

[16] B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. Max-margin parsing. In *Proc. of EMNLP*, 2004.

[17] E. F. Tjong Kim Sang and S. Buchholz. Introduction to the CoNLL-2000 shared task: Chunking. In *In Proc. CoNLL*, 2000.

[18] E. F. Tjong Kim Sang and F. De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proc. of CoNLL*, 2003.

[19] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector learning for interdependent and structured output spaces. In *Proc. of ICML*, 2004.

[20] J. Yedidia, W. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, 2001.