Logistic Online Learning Methods and Their Application to Incremental Dependency Parsing

Richard Johansson
Department of Computer Science
Lund University
Lund, Sweden
richard@cs.lth.se

Abstract

We investigate a family of update methods for online machine learning algorithms for cost-sensitive multiclass and structured classification problems. The update rules are based on multinomial logistic models. The most interesting question for such an approach is how to integrate the cost function into the learning paradigm. We propose a number of solutions to this problem.

To demonstrate the applicability of the algorithms, we evaluated them on a number of classification tasks related to incremental dependency parsing. These tasks were conventional multiclass classification, hiearchical classification, and a structured classification task: complete labeled dependency tree prediction. The performance figures of the logistic algorithms range from slightly lower to slightly higher than margin-based online algorithms.

1 Introduction

Natural language consists of complex structures, such as sequences of phonemes, parse trees, and discourse or temporal graphs. Researchers in NLP have started to realize that this complexity should be reflected in their statistical models. This intuition has spurred a growing interest of related research in the machine learning community, which in turn has led to improved results in a wide range of applications in NLP, including sequence labeling (Lafferty et al., 2001; Taskar et al., 2006), constituent and dependency parsing (Collins and Duffy, 2002; McDonald et al., 2005), and logical form extraction (Zettlemoyer and Collins, 2005).

Machine learning research for structured problems have generally used margin-based formulations. These include global batch methods such as Max-margin Markov Networks (M^{3}N) (Taskar et al., 2006) and SVM^{struct} (Tsochantaridis et al., 2005) as well as online methods such as Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003) and the Online Passive-Aggressive Algorithm (OPA) (Crammer et al., 2006). Although the batch methods are formulated very elegantly, they do not seem to scale well to the large training sets prevalent in NLP contexts. The online methods on the other hand, although less theoretically appealing, can handle realistically sized data sets.

In this work, we investigate whether logistic online learning performs as well as margin-based methods. Logistic models are easily extended to using kernels; that this is theoretically well-justified was shown by Zhu and Hastie (2005), who also made an elegant argument that margin-based methods are in fact related to regularized logistic models. For batch learning, there exist several learning algorithms in a logistic framework for conventional multiclass classification but few for structured problems.

Prediction of complex structures is conventionally treated as a cost-sensitive multiclass classification problem, although special care has to be taken to handle the large space of possible outputs. The integration of the cost function into the logistic framework leads to two distinct (although related) update methods: the Scaled Prior Variance (SPV) and the Minimum Expected Cost (MEC) updates.

Apart from its use in structured prediction, cost-sensitive classification is useful for *hierachical* classification, which we briefly consider here in an experiment. This type of classification has useful applications in NLP. Apart from the obvious use in classification of concepts in an ontology, it is also useful for prediction of complex morphological or named-entity tags. Cost-sensitive learning is also required in the SEARN algorithm (Daumé III et al., 2006), which is a method to decompose the prediction problem of a complex structure into a sequence of actions, and train the search in the space of action sequences to maximize global performance.

2 Algorithm

We model the learning problem as finding a discriminant function F that assigns a score to each possible output y given an input $\boldsymbol{x}$. Classification in this setting is done by finding the $\hat{y}$ that maximizes $F(\boldsymbol{x}, y)$. In this work, we consider linear discriminants of the following form:

$$F(\boldsymbol{x}, y) = \langle \boldsymbol{w}, \boldsymbol{\Psi}(\boldsymbol{x}, y) \rangle$$

Here, $\boldsymbol{\Psi}(\boldsymbol{x}, y)$ is a numeric feature representation of the pair $(\boldsymbol{x}, y)$ and $\boldsymbol{w}$ a vector of feature weights. Learning in this case is equivalent to assigning appropriate weights in the vector $\boldsymbol{w}$.

In the online learning framework, the weight vector is constructed incrementally. Algorithm 1 shows the general form of the algorithm. It proceeds a number of times through the training set. In each step, it computes an update to the weight vector based on the current example. The resulting weight vector tends to be overfit to the last few examples; one way to reduce overfitting is to use the average of all successive weight vectors as the result of the training (Freund and Schapire, 1999).

Algorithm 1 General form of online algorithms

```
input Training set $\mathcal{T} = \{(\boldsymbol{x}_t, y_t)\}_{t=1}^{T}$
      Number of iterations $N$
for $n$ in $1..N$
  for $(\boldsymbol{x}_t, y_t)$ in $\mathcal{T}$
    Compute update vector $\delta\boldsymbol{w}$ for $(\boldsymbol{x}_t, y_t)$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \delta\boldsymbol{w}$
return $\boldsymbol{w}_{\text{average}}$
```

Following earlier online learning methods such as the Perceptron, we assume that in each update step, we adjust the weight vector by incrementally adding feature vectors. For stability, we impose the constraint that the sum of the updates in each step should be zero. We assume that the possible output values are $\{y_i\}_{i=0}^{m}$ and, for convenience, that y_0 is the correct value. This leads to the following *ansatz*:

$$\delta\boldsymbol{w} = \sum_{j=1}^{m} \alpha_j (\boldsymbol{\Psi}(\boldsymbol{x}, y_0) - \boldsymbol{\Psi}(\boldsymbol{x}, y_j))$$

Here, α_j defines how much F is shifted to favor y_0 instead of y_j. This is also the approach (implicitly) used by other algorithms such as MIRA and OPA.

The following two subsections present two ways of creating the weight update $\delta\boldsymbol{w}$, differing in how the cost function is integrated into the model. Both are based on a multinomial logistic framework, where we model the probability of the class y being assigned to an input $\boldsymbol{x}$ using a "soft-max" function as follows:

$$P(y|\boldsymbol{x}) = \frac{e^{F(\boldsymbol{x},y)}}{\sum_{j=0}^{m} e^{F(\boldsymbol{x},y_j)}}$$

2.1 Scaled Prior Variance Approach

The first update method, Scaled Prior Variance (SPV), directly uses the probability of the correct output. It uses a maximum a posteriori approach, where the cost function is used by the prior.

Naïvely, the update could be done by maximizing the likelihood with respect to $\boldsymbol{\alpha}$ in each step. However, this would lead to overfitting – in the case of separability, a maximum does not even exist. We thus introduce a regularizing prior that penalizes large values of $\boldsymbol{\alpha}$. We introduce variance-controlling hyperparameters s_j for each α_j, and with a Gaussian prior we obtain (disregarding constants) the following log posterior:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\alpha}) &= \sum_{j=1}^{m} \alpha_j (K_{00} - K_{j0}) - \sum_{j=1}^{m} s_j \alpha_j^2 \\ &- \log \sum_{k=0}^{m} e^{f_k + \sum_{j=1}^{m} \alpha_j (K_{0k} - K_{jk})}\end{aligned}$$

where $K_{ij} = \langle \boldsymbol{\Psi}(\boldsymbol{x}, y_i), \boldsymbol{\Psi}(\boldsymbol{x}, y_j) \rangle$ and $f_k = F(\boldsymbol{x}, y_k)$ (i.e. the output before $\boldsymbol{w}$ is updated). As usual, the feature vectors occur only in inner products, allowing us to use kernels if appropriate.

We could have used any prior; however, in practice we will require it to be log-concave to avoid suboptimal local maxima. A Laplacian prior (i.e. $-\sum_{j=1}^{m} s_j|\alpha_j|$) will also be considered in this work – the discontinuity of its gradient at the origin seems to pose no problem in practice.

Costs are incorporated into the model by associating them to the prior variances. We tried two variants of variance scaling. In the first case, we let the variance be directly proportional to the cost (C-SPV):

$$s_j = \frac{\gamma}{c(y_j)}$$

where γ is a tradeoff parameter controlling the relative weight of the prior with respect to the likelihood. Intuitively, this model allows the algorithm more freedom to adjust an α_j associated with a y_j with a high cost.

In the second case, inspired by margin-based learning we instead scaled the variance by the *loss*, i.e. the scoring error plus the cost (L-SPV):

$$s_j = \frac{\gamma}{\max(0, f_j - f_0) + c(y_j)}$$

Here, the intuition is instead that the algorithm is allowed more freedom for "dangerous" outputs that are ranked high but have high costs.

2.2 Minimum Expected Cost Approach

In the second approach to integrating the cost function, the Minimum Expected Cost (MEC) update, the method seeks to minimize the expected cost in each step. Once again using the soft-max probability, we get the following expectation of the cost:

$$\begin{aligned}\mathbf{E}(c(y)|\boldsymbol{x}) &= \sum_{k=0}^{m} c(y_k)P(y_k|\boldsymbol{x}) \\ &= \frac{\sum_{k=0}^{m} c(y_k)e^{f_k+\sum_{j=1}^{m}\alpha_j(K_{0k}-K_{jk})}}{\sum_{k=0}^{m} e^{f_k+\sum_{j=1}^{m}\alpha_j(K_{0k}-K_{jk})}}\end{aligned}$$

This quantity is easily minimized in the same way as the SPV posterior was maximized, although we had to add a constant 1 to the expectation to avoid numerical instability. To avoid overfitting, we added a quadratic regularizer $\gamma\sum_{j=1}^{m}\alpha_j^2$ to $\log(1 + \mathbf{E}(c(y)|\boldsymbol{x}))$ just like the prior in the SPV method, although this regularizer does not have an interpretation as a prior.

The MEC update is closely related to SPV: for cost-insensitive classification (i.e. the cost of every misclassified instance is 1), the expectation is equal to one minus the likelihood in the SPV model.

2.3 Handling Complex Prediction Problems

The algorithm can thus be used for any cost-sensitive classification problem. This class of problems includes prediction of complex structures such as trees or graphs. However, for those problems the set of possible outputs is typically very large. Two broad categories of solutions to this problem have been common in literature, both of which rely on the structure of the domain:

- *Subset selection*: instead of working with the complete range of outputs, only an "interesting" subset is used, for instance by repeatedly finding the most violated constraints (Tsochantaridis et al., 2005) or by using N-best search (McDonald et al., 2005).
- *Decomposition*: the inherent structure of the problem is used to factorize the optimization problem. Examples include Markov decompositions in M^{3}N (Taskar et al., 2006) and dependency-based factorization for MIRA (McDonald et al., 2005).

In principle, both methods could be used in our framework. In this work, we use subset selection since it is easy to implement for many domains (in the form of an N-best search) and allows a looser coupling between the domain and the learning algorithm.

2.4 Implementation Issues

Since we typically work with only a few variables in each iteration, maximizing the log posterior or minimizing the expectation is easy (assuming, of course, that we chose a log-concave prior). We used gradient ascent and did not try to use more sophisticated optimization procedures like BFGS or Newton's method. Typically, only a few iterations were needed to reach the optimum. The running time of the update step is almost identical to that of MIRA, which solves a small quadratic program in each step, but longer than for the Perceptron algorithm or OPA.

Actions	**Parser actions**	**Conditions**
Initialize	$(nil, W, \emptyset)$	
Terminate	(S, nil, A)	
Left-arc	$(n\|S, n'\|I, A) \to (S, n'\|I, A \cup \{(n', n)\})$	$\neg\exists n''(n'', n) \in A$
Right-arc	$(n\|S, n'\|I, A) \to (n'\|n\|S, I, A \cup \{(n, n')\})$	$\neg\exists n''(n'', n') \in A$
Reduce	$(n\|S, I, A) \to (S, I, A)$	$\exists n'(n', n) \in A$
Shift	$(S, n\|I, A) \to (n\|S, I, A)$	

Table 1: Nivre's parser transitions where W is the initial word list; I, the current input word list; A, the graph of dependencies; and S, the stack. (n', n) denotes a dependency relations between n' and n, where n' is the head and n the dependent.

3 Experiments

To compare the logistic online algorithms against other learning algorithms, we performed a set of experiments in incremental dependency parsing using the Nivre algorithm (Nivre, 2003).

The algorithm is a variant of the shift–reduce algorithm and creates a projective and acyclic graph. As with the regular shift–reduce, it uses a stack S and a list of input words W, and builds the parse tree incrementally using a set of parsing actions (see Table 1). However, instead of finding constituents, it builds a set of arcs representing the graph of dependencies. It can be shown that every projective dependency graph can be produced by a sequence of parser actions, and that the worst-case number of actions is linear with respect to the number of words in the sentence.

3.1 Multiclass Classification

In the first experiment, we trained multiclass classifiers to choose an action in a given parser state (see (Nivre, 2003) for a description of the feature set). We stress that this is true multiclass classification rather than a decomposed method (such as one-versus-all or pairwise binarization).

As a training set, we randomly selected 50,000 instances of state–action pairs generated for a dependency-converted version of Penn Treebank. This training set contained 22 types of actions (such as SHIFT, REDUCE, LEFT-ARC(SUBJECT), and RIGHT-ARC(OBJECT). The test set was also randomly selected and contained 10,000 instances.

We trained classifiers using the logistic updates (C-SPV, L-SPV, and MEC) with Gaussian and Laplacian priors. Additionally, we trained OPA and MIRA classifiers, as well as an Additive Ultraconservative (AU) classifier (Crammer and Singer, 2003), a variant of the Perceptron.

For all algorithms, we tried to find the best values of the respective regularization parameter using cross-validation. All training algorithms iterated five times through the training set and used an expanded quadratic kernel.

Table 2 shows the classification error for all algorithms. As can be seen, the performance was lower for the logistic algorithms, although the difference was slight. Both the logistic (MEC and SPV) and the margin-based classifiers (OPA and MIRA) outperformed the AU classifier.

Method	Test error
MIRA	6.05%
OPA	6.17%
C-SPV, Laplace	6.20%
MEC, Laplace	6.21%
C-SPV, Gauss	6.22%
MEC, Gauss	6.23%
L-SPV, Laplace	6.25%
L-SPV, Gauss	6.26%
AU	6.39%

Table 2: Multiclass classification results.

3.2 Hierarchical Classification

In the second experiment, we used the same training and test set, but considered the selection of the parsing action as a hierarchical classficiation task, i.e. the predicted value has a main type (SHIFT, REDUCE, LEFT-ARC, and RIGHT-ARC) and possibly also a subtype (such as LEFT-ARC(SUBJECT) or

RIGHT-ARC(OBJECT)).

To predict the class in this experiment, we used the same feature function but a new cost function: the cost of misclassification was 1 for an incorrect parsing action, and 0.5 if the action was correct but the arc label incorrect.

We used the same experimental setup as in the multiclass experiment. Table 3 shows the average cost on the test set for all algorithms. Here, the MEC update outperformed the margin-based ones by a negligible difference. We did not use AU in this experiment since it does not optimize for cost.

Method	Average cost
MEC, Gauss	0.0573
MEC, Laplace	0.0576
OPA	0.0577
C-SPV, Gauss	0.0582
C-SPV, Laplace	0.0587
MIRA	0.0590
L-SPV, Gauss	0.0590
L-SPV, Laplace	0.0632

Table 3: Hierarchical classification results.

3.3 Prediction of Complex Structures

Finally, we made an experiment in prediction of dependency trees. We created a global model where the discriminant function was trained to assign high scores to the correct parse tree. A similar model was previously used by McDonald et al. (2005), with the difference that we here represent the parse tree as a sequence of actions in the incremental algorithm rather than using the dependency links directly.

For a sentence $\boldsymbol{x}$ and a parse tree y, we defined the feature representation by finding the sequence $((S_1, I_1), a_1), ((S_2, I_2), a_2) \ldots$ of states and their corresponding actions, and creating a feature vector for each state/action pair. The discriminant function was thus written

$$\langle \boldsymbol{\Psi}(\boldsymbol{x}, y), \boldsymbol{w} \rangle = \sum_i \langle \psi((S_i, I_i), a_i), \boldsymbol{w} \rangle$$

where ψ is the feature function from the previous two experiments, which assigns a feature vector to a state (S_i, I_i) and the action a_i taken in that state.

The cost function was defined as the sum of link costs, where the link cost was 0 for a correct dependency link with a correct label, 0.5 for a correct link with an incorrect label, and 1 for an incorrect link.

Since the history-based feature set used in the parsing algorithm makes it impossible to use independence to factorize the scoring function, an exact search to find the best-scoring action sequence is not possible. We used a beam search of width 2 in this experiment.

We trained models on a 5000-word subset of the Basque Treebank (Aduriz et al., 2003) and evaluated them on a 8000-word subset of the same corpus. As before, we used an expanded quadratic kernel, and all algorithms iterated five times through the training set.

Table 4 shows the results of this experiment. We show labeled accuracy instead of cost for ease of interpretation. Here, the loss-based SPV outperformed MIRA, and two other logistic updates also outperformed OPA. The differences between the first four scores are however not statistically significant. Interestingly, all updates with Laplacian prior resulted in low performance. The reason for this may be that Laplacian priors tend to promote sparse solutions (see Krishnapuram et al. (2005), *inter alia*), and that this sparsity is detrimental for this highly lexicalized feature set.

Method	Labeled Accuracy
L-SPV, Gauss	66.24
MIRA	66.19
MEC, Gauss	65.99
C-SPV, Gauss	65.84
OPA	65.45
MEC, Laplace	64.81
C-SPV, Laplace	64.73
L-SPV, Laplace	64.50

Table 4: Results for dependency tree prediction.

4 Conclusion and Future Work

This paper presented new update methods for online machine learning algorithms. The update methods are based on a multinomial logistic model. Their performance is on par with other state-of-the-art online learning algorithms for cost-sensitive problems.

We investigated two main approaches to integrating the cost function into the logistic model. In the first method, the cost was linked to the prior variances, while in the second method, the update rule sets the weights to minimize the expected cost. We tried a few different priors. Which update method and which prior was the best varied between experiments. For instance, the update where the prior variances were scaled by the costs was the best-performing in the multiclass experiment but the worst-performing in the dependency tree prediction experiment.

In the SPV update, the cost was incorporated into the MAP model in a rather ad-hoc fashion. Although this seems to work well, we would like to investigate this further and possibly devise a cost-based prior that is both theoretically well-grounded and performs well in practice.

To achieve a good classification performance using the updates presented in this article, there is a considerable need for cross-validation to find the best value for the regularization parameter. This is true for most other classification methods as well, including SVM, MIRA, and OPA. There has been some work on machine learning methods where this parameter is tuned automatically (Tipping, 2001), and a possible extension to our work could be to adapt those models to the multinomial and cost-sensitive setting.

We applied the learning models to three problems in incremental dependency parsing, the last of which being prediction of full labeled dependency trees. Our system can be seen as a unification of the two best-performing parsers presented at the CoNLL-X Shared Task (Buchholz and Marsi, 2006).

References

Itzair Aduriz, Maria Jesus Aranzabe, Jose Mari Arriola, Aitziber Atutxa, Arantza Diaz de Ilarraza, Aitzpea Garmendia, and Maite Oronoz. 2003. Construction of a Basque dependency treebank. In *Proceedings of the TLT*, pages 201–204.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the CoNLL-X*.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the ACL*.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 2003(3):951–991.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Schwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 2006(7):551–585.

Hal Daumé III, John Langford, and Daniel Marcu. 2006. Search-based structured prediction. *Submitted*.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Balaji Krishnapuram, Lawrence Carin, Mário A. T. Figueiredo, and Alexander J. Hartemink. 2005. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6).

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT-EMNLP-2005*.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*, pages 149–160, Nancy, France, 23-25 April.

Ben Taskar, Carlos Guestrin, Vassil Chatalbashev, and Daphne Koller. 2006. Max-margin Markov networks. *Journal of Machine Learning Research*, to appear.

Michael E. Tipping. 2001. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211 – 244.

Iannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(Sep):1453–1484.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI 2005*.

Ji Zhu and Trevor Hastie. 2005. Kernel logistic regression and the import vector machine. *Journal of Computational and Graphical Statistics*, 14(1):185–205.