Online Learning with Preference Feedback

Pannagadatta K. Shivaswamy
Department of Computer Science
Cornell University, Ithaca NY
pannaga@cs.cornell.edu

Thorsten Joachims
Department of Computer Science
Cornell University, Ithaca NY
tj@cs.cornell.edu

Abstract

We propose a new online learning model for learning with preference feedback. The model is especially suited for applications like web search and recommender systems, where preference data is readily available from implicit user feedback (e.g. clicks). In particular, at each time step a potentially structured object (e.g. a ranking) is presented to the user in response to a context (e.g. query), providing him or her with some unobserved amount of utility. As feedback the algorithm receives an improved object that would have provided higher utility. We propose a learning algorithm with provable regret bounds for this online learning setting and demonstrate its effectiveness on a web-search application. The new learning model also applies to many other interactive learning problems and admits several interesting extensions.

1 Introduction

Our new learning model is motivated by how users interact with a web-search engine or a recommender system. At each time step, the user issues a query and the system responds by supplying a list of results. The user views some of the results and selects those that he or she prefers. Here are two such examples:

Web Search: In response to a query, the search engine presents the ranking $[A, B, C, D, E, ...]$ and observes that the user clicks on documents C and D.

Movie Recommendation: An online service recommends movie A to a user. However, the user ignores the recommendation and instead rents another movie B after some browsing.

In both cases the user feedback comes in the form of a preference. In the web search example, we can infer that the user would have preferred the ranking $[C, D, A, B, E, ...]$ over the one we presented [6]. In the recommendation example, movie B was preferred over movie A. The cardinal utilities of the predictions, however, are never observed, and the algorithm typically does not get the optimal ranking/movie as feedback.

This preference feedback is different from conventional online learning models. In the simplest form of the multi-armed bandit problem [2, 1, 3], an algorithm chooses an action (out of K possible actions) and observes reward only for that action. Conversely, rewards of all possible actions are revealed in the case of learning with expert advice [3]. Our model, where the ordering of two arms is revealed (the one we presented and the one we receive as feedback), sits between the expert and the bandit setting. A similar relationship holds for online convex optimization [9] and online convex optimization in the bandit setting [5], which can be viewed as continuous extensions of the expert and the bandit problems respectively, since they rely on observing either a full convex function or the value of a convex functions after each iteration. Most closely related to our work is the dueling bandits setting [7, 8], but existing algorithms are known to converge rather slowly.

In the following, we formally define the online preference learning model and a notion of regret, propose a simple algorithm for which we prove a regret bound, and empirically evaluate the algorithm on a web-search problem.

2 Online Preference Learning Model

The online preference learning model is defined as follows. At each round t, the learning algorithm receives a context $\mathbf{x}_t \in \mathcal{X}$ and presents a (possibly structured) object $\mathbf{y}_t \in \mathcal{Y}$. In response, the user returns an object $\bar{\mathbf{y}}_t \in \mathcal{Y}$ which the algorithm receives as feedback. For example, in web-search, a user issues a query and is presented with a ranked list of URL's ($\mathbf{y}_t$). The user interacts with the ranking that was provided to her by clicking on results that are relevant to her. This user interaction allows us to infer a better ranking $\bar{\mathbf{y}}_t$ to this user.

We assume that the user evaluates rankings according to a utility function $U(\mathbf{x}, \mathbf{y})$ that is unknown to the learning algorithm. A natural way to define regret in this model is based on the difference in utility $U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t)$ between the object $\mathbf{y}_t$ we present and the best possible objects $\mathbf{y}_t^* = \text{argmax}_{\mathbf{y}} U(\mathbf{x}_t, \mathbf{y})$ that could have been presented. The goal of an algorithm is to minimize

$$\text{REGRET}_T := \frac{1}{T} \sum_{t=1}^{T} \left(U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t) \right). \tag{1}$$

To prove bounds on the regret, we specify the properties of the user's preference feedback more precisely. We say that user feedback is α-*informative*, if for some $\alpha \in (0, 1]$ and $\xi_t \geq 0$

$$\left(U(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U(\mathbf{x}_t, \mathbf{y}_t) \right) = \alpha \left(U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t) \right) - \xi_t. \tag{2}$$

Intuitively, the above definition describes the quality of feedback by how much the utility of the user feedback $\bar{\mathbf{y}}_t$ is higher than that of the algorithm's prediction $\mathbf{y}_t$ in terms of an (unknown) fraction α of the maximum possible utility range. Note that $\xi_t \geq 0$ is a slack variable that captures noise in the feedback. Also, note that (2) is written as an equality assuming that the slack variable ξ_t is set to its lowest possible value.

In the following, we use a linear model for the utility function

$$U(\mathbf{x}, \mathbf{y}) = \mathbf{w}^{*\top} \phi(\mathbf{x}, \mathbf{y}), \tag{3}$$

where $\mathbf{w}^* \in \mathbf{R}^N$ is an unknown parameter vector and $\phi : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbf{R}^N$ is a joint feature map such that $\|\phi(\mathbf{x}, \mathbf{y})\| \leq R$ for any $\mathbf{x} \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{Y}$.

3 Algorithm

We propose the algorithm in Figure 1 for the online preference learning problem. It maintains a vector $\mathbf{w}_t$ and predicts the object with the highest utility according to $\mathbf{w}_t$ in each iteration t. It then receives feedback $\bar{\mathbf{y}}_t$ and updates $\mathbf{w}_t$ in the direction $\phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t)$.

Initialize $\mathbf{w}_1 \leftarrow \mathbf{0}$
for $t = 1$ **to** T **do**
  Observe $\mathbf{x}_t$
  Present $\mathbf{y}_t \leftarrow \text{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y})$
  Obtain feedback $\bar{\mathbf{y}}_t$
  Update: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t)$
end for

Figure 1: Preference Perceptron.

Theorem 1 *Under α-informative feedback the algorithm in Figure 1 has regret*

$$\textit{REGRET}_T \leq \frac{1}{\alpha T} \sum_{t=1}^{T} \xi_t + \frac{2R\|\mathbf{w}^*\|}{\alpha \sqrt{T}}. \tag{4}$$

Proof of the above theorem is provided in the Appendix A. When the user feedback is noise free, the first term on the right hand side of the above bound vanishes. The average regret in this case approaches zero at the rate $1/\sqrt{T}$. In addition to this result, we have the following extensions which we cannot provide here due to space limitations:

- It is possible to further weaken the requirement on the feedback. Instead of requiring α-informative feedback, the user is required to give α-informative feedback in expectation. We can show a result similar to that in Theorem 1 in this case.
- It is also possible to show that an algorithm different from Algorithm 1 can minimize any convex loss (under mild assumptions) defined on the utility difference $\mathbf{w}^{*\top}(\phi(\mathbf{x}_t, \mathbf{y}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t^*))$.
- We can also minimize any convex loss defined on the utility difference using an exponentiated update rule. This algorithm considers convex combination (rather than linear combination) of features to make predictions.

4 Experiments

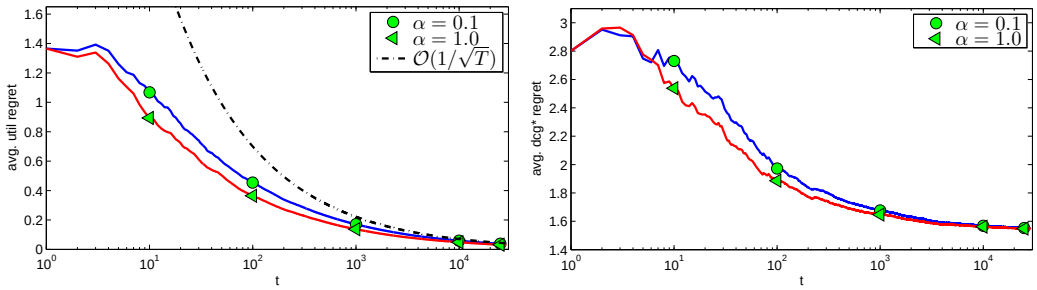

Figure 2: Average regret versus time based on noise free α-informative feedback.

We applied our Preference Perceptron algorithm to the Yahoo! learning to rank dataset [4]. This dataset consists of query-url features (denoted as $\mathbf{x}_i^q$ for query q and URL i for that particular query) with a relevance rating r_i^q which ranges from zero (irrelevant) to four (perfectly relevant). We first computed the best least squares fit to the relevance labels from the features using the entire dataset and all the utilities in our experiment are reported with respect to this $\mathbf{w}^*$.

To pose ranking as a structured prediction problem, we defined our joint feature map as follows:

$$\mathbf{w}^\top \phi(q, \mathbf{y}) = \sum_{i=1}^{5} \frac{\mathbf{w}^\top \mathbf{x}_{\mathbf{y}_i}^q}{\log(i+1)} \tag{5}$$

In the above equation, $\mathbf{y}$ denotes a ranking. In particular, $\mathbf{y}_i$ is the index of the URL which is placed at position i in the ranking. Thus, the above measure considers the top five URLs for a query q and computes a score based on a graded relevance. The above feature-map and utility are motivated from the definition: $\text{DCG@5}(q, \mathbf{y}) = \sum_{i=1}^{5} \frac{r_{\mathbf{y}_i}^q}{\log(i+1)}$. Effectively, our utility score (5) mimics DCG@5 by replacing the relevance label with a linear prediction based on the features.

For query q_t at time step t, the Preference Perceptron algorithm presents the ranking $\mathbf{y}_t^q$ that maximizes $\mathbf{w}_t^\top \phi(q, \mathbf{y})$. Note that this merely amounts to sorting documents by the scores $\mathbf{w}_t^\top \mathbf{x}_i^{q_t}$, which can be done very efficiently. Once a ranking ($\mathbf{y}^{q_t}$) was presented to a user, the user returns a ranking $\bar{\mathbf{y}}^{q_t}$. The exact nature of user feedback differed in the two experiments; the details of feedback can be found below. Query ordering was randomly permuted twenty times and all the results reported are an average over the runs.

The utility regret in Eqn. (1), based on the definition of utility in (5), is given by $\frac{1}{T}\sum_{t=1}^{T}(\mathbf{w}^{*\top}\phi(q_t, \mathbf{y}^{q_t*}) - \phi(q_t, \mathbf{y}^{q_t}))$. Here $\mathbf{y}^{q_t*}$ denotes the optimal ranking with respect to $\mathbf{w}^*$. We also present our results on another quantity which we refer to as the **DCG* regret**. Since for every query-URL pair there is a manual relevance judgment in the dataset, optimal DCG can be computed by sorting the relevance score. In DCG* regret, we measure the difference between the DCG of the optimal ranking and that of the rankings we present in each step.

α-informative feedback The goal of the first experiment was to see how the regret of the algorithm changes with α, assuming α-informative feedback without noise. Once a ranking was presented, the feedback was obtained as follows: given a ranked list, the simulated user would go down the list and would stop when she found five URL's such that, when they are placed at the top of

the list (in the order of their utilities), gave noise free α-informative feedback (i.e. $\xi_t = 0$) based on $\mathbf{w}^*$. Figure 2 shows the results for this experiment for two different α values. As expected, the regret with $\alpha = 1.0$ is lower compared to the regret with respect $\alpha = 0.1$. Note, however, that the difference between the two curves is much smaller than a factor of ten. This is because, strictly α-informative feedback is also strictly β-informative feedback for any $\beta \leq \alpha$. So, there could be several instances where user feedback was much stronger than what was required. Since the slack variables are zero, the average utility regret approaches zero as expected.

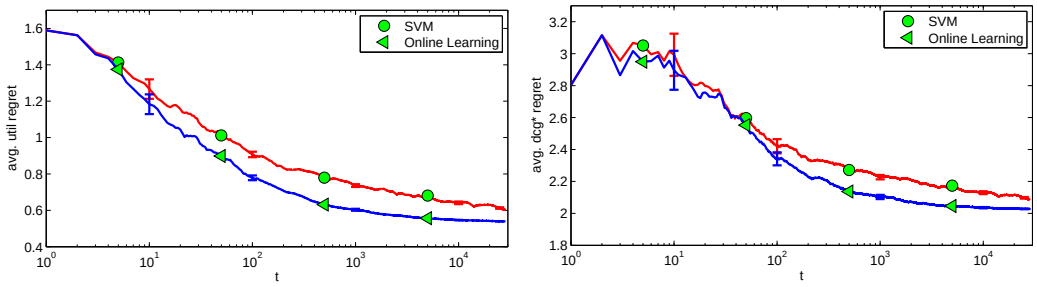

Figure 3: Regret versus time based on actual relevance labels.

Relevance label feedback In this experiment, feedback was based on the actual relevance labels in the dataset as follows: given a ranking for a query, the user would go down the list inspecting the top 25 (or all the URLs if the list is shorter) URLs. Five URL's with the highest relevance labels (r_i^q) are placed at the top five locations in the user feedback. Note that this is a noisy version of feedback since the linear fit cannot describe the labels exactly in this dataset.

As a baseline, a ranking SVM was trained repeatedly. In the first iteration, a random ranking was presented, the feedback ranking (as mentioned in the paragraph above) was obtained. An SVM was trained based on the pair of examples $((q_1, \mathbf{y}^{q_1}), (q_1, \bar{\mathbf{y}}^{q_1}))$. From then on, a ranking was presented based on the prediction from the previously trained ranking SVM. The user always returned a ranking based on the relevance labels as mentioned above; the pairs of examples were stored after every iteration. Note that training a ranking SVM after each iteration would be prohibitive since it involves cross-validating a parameter C that trades-off between the margin and the slacks. Thus, we trained an SVM whenever 10% more examples were added to the training set after the previous training. The value of the parameter C was obtained via a five-fold cross-validation.[1] Once a C value was determined, SVM was trained on all the training examples available at that time and used it to predict rankings until the next training.

Results of this experiment are presented in Figure 3. We have provided both the mean regret as well as one standard deviation for this experiment. Since the feedback is now based on relevance labels (and not on a linear fit), the utility regret converges to a non-zero value. It can also be noticed that our preference perceptron performs significantly better compared to the SVM. The perceptron algorithm took around 30 minutes to run (which was mostly inefficient Python IO), whereas the SVM version took about 20 hours (on the same machine). This shows that our preference perceptron algorithm can be empirically superior to SVM with significantly lower computational cost. We tried to retrain SVM more frequently (after 5% and 1% more training examples), while this was computationally very intensive, we never saw SVM performing better than our algorithm.

5 Conclusions

We proposed a new model of online learning with preferences that is especially suitable for implicit user feedback. An efficient algorithm was proposed that provably minimizes regret. Experiments showed its effectiveness for web-search ranking.

Acknowledgements We thank Bobby Kleinberg and Joshua Moore for helpful discussions. This work was funded in part under NSF award IIS-0905467.

[1] It was fixed at 100 when there were less than 50 examples.

References

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[2] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire. The non-stochastic multi-armed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

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

[4] O. Chapelle and Y. Chang. Yahoo! learning to rank challenge overview. *JMLR - Proceedings Track*, 14:1–24, 2011.

[5] A. Flaxman, A. T. Kalai, and H. B. McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. In *SODA*, 2005.

[6] T. Joachims, L. Granka, Bing Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems (TOIS)*, 25(2), April 2007.

[7] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. The k-armed dueling bandits problem. In *COLT*, 2009.

[8] Y. Yue and T. Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In *ICML*, 2009.

[9] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, 2003.

A Proof of theorem 1

Proof First, consider the inner product of $\mathbf{w}_{T+1}$ with itself. We have,

$$\begin{aligned}
\mathbf{w}_{T+1}^\top \mathbf{w}_{T+1} &= \mathbf{w}_T^\top \mathbf{w}_T + 2\mathbf{w}_T^\top(\phi(\mathbf{x}_T, \bar{\mathbf{y}}_T) - \phi(\mathbf{x}_T, \mathbf{y}_T)) \\
&\quad + (\phi(\mathbf{x}_T, \bar{\mathbf{y}}_T) - \phi(\mathbf{x}_T, \mathbf{y}_T))^\top (\phi(\mathbf{x}_T, \bar{\mathbf{y}}_T) - \phi(\mathbf{x}_T, \mathbf{y}_T)) \\
&\leq \mathbf{w}_T^\top \mathbf{w}_T + 4R^2 \\
&\leq 4R^2 T.
\end{aligned}$$

On the first line, we simply used our update rule from algorithm 1. On the second line, we used the fact that $\mathbf{w}_T^\top(\phi(\mathbf{x}_T, \bar{\mathbf{y}}_T) - \phi(\mathbf{x}_T, \mathbf{y}_T)) \leq 0$ from the choice of $\mathbf{y}_T$ in Algorithm 1 and that $\|\phi(\mathbf{x}_T, \bar{\mathbf{y}}_T) - \phi(\mathbf{x}_T, \mathbf{y}_T)\|^2 \leq 4R^2$. We obtain the last line inductively.

Further, from the update rule in algorithm 1, we have,

$$\begin{aligned}
\mathbf{w}_{T+1}^\top \mathbf{w}^* &= \mathbf{w}_T^\top \mathbf{w}^* + \mathbf{w}^{*\top} (\phi(\mathbf{x}_T, \bar{\mathbf{y}}_T) - \phi(\mathbf{x}_T, \mathbf{y}_T)) \\
&= \sum_{t=1}^{T} \mathbf{w}^{*\top} (\phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t)) \\
&= \sum_{t=1}^{T} (U(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U(\mathbf{x}_t, \mathbf{y}_t)) .
\end{aligned}$$

We now use the fact that $\mathbf{w}_{T+1}^\top \mathbf{w}^* \leq \|\mathbf{w}^*\| \|\mathbf{w}_{T+1}\|$ (Cauchy-Schwarz inequality) which implies,

$$\sum_{t=1}^{T} (U(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U(\mathbf{x}_t, \mathbf{y}_t)) \leq 2R\sqrt{T}\|\mathbf{w}^*\|.$$

The above inequality, along with the α-informative feedback (Eqn. (2)) gives,

$$\alpha \sum_{t=1}^{T} (U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t)) - \sum_{t=1}^{T} \xi_t \leq 2R\sqrt{T}\|\mathbf{w}^*\|.$$

from which the claimed result follows. ■