CS260: Machine Learning Theory
Lecture 10: Expert Advice & Randomized Weighted Majority
October 26, 2011

Lecturer: Jennifer Wortman Vaughan

1 The Expert Advice Framework

For the past few lectures we have been discussing online classification in scenarios in which there is a perfect target function. In this lecture, we remove the assumption of a perfect target, and study a more general learning framework, *learning from expert advice*, that applies to a wide range of problems beyond just classification.

Suppose that you are interested in making a sequence of predictions or decisions based on the advice of a set of n "experts." In this context, an expert could be a literal human being like a weather forecaster if you are interested in predicting tomorrow's weather, or a financial analyst if you are interested in trading stocks. More generally, experts could represent simple decision rules based on features of a learning problem, individual learning algorithms running in parallel, or any other predictors. On each round, you must choose an expert to follow.

On each round, each expert suffers a *loss* for its prediction. For example, in the case of binary classification, an expert might suffer a loss of 0 if its prediction is correct and 1 if its prediction is incorrect. For convenience, we assume that losses are bounded in $[0, 1]$, but it is easy to relax this assumption to any bounded range. The learner also suffers a loss based on the expert it chose to follow.

We could formalize this setting as follows. Note that we have abstracted away the particular predictions or decisions that are being made, and look only at the loss of each prediction or action.

Learning from Expert Advice (Preliminary Version)

At each round $t \in \{1, 2, \cdots, T\}$,

- The learner chooses an expert e_t to follow.
- Each expert $i \in \{1, \cdots, n\}$ suffers loss $\ell_{i,t} \in [0, 1]$.
- The learner suffers loss $\ell_{e_t,t}$.

We do not make any assumptions about the quality of the experts. The losses $\ell_{i,t}$ can take on any arbitrary values in $[0, 1]$. We can think about the sequence of losses as being selected by an adversary who would like to make the learner suffer a high cumulative loss. Because of this, we cannot make absolute guarantees about the cumulative loss of the learning algorithm, since all experts might have high cumulative

All CS260 lecture notes build on the scribes' notes written by UCLA students in the Fall 2010 offering of this course. Although they have been carefully reviewed, it is entirely possible that some of them contain errors. If you spot an error, please email Jenn.

loss. Instead, we evaluate the loss of the learning algorithm with respect to a particular comparator: the best expert in the set. To capture this idea, we (initially) define the *regret* of the learning algorithm as

$$\sum_{t=1}^{T} \ell_{e_t,t} - \min_{i \in \{1,\ldots,n\}} \sum_{t=1}^{T} \ell_{i,t} \ .$$

Our goal is to minimize this regret. In particular, we would like the regret to be sublinear in the number of rounds T so that our *average* regret goes to 0 as T gets large.

It is easy to see that this goal is impossible to achieve using a deterministic learning algorithm. An adversary who knows the algorithm could set $\ell_{e_t,t} = 1$ and $l_{i,t} = 0$ for all $i \neq e_t$ on each round t. In this case, at round T, the cumulative loss of the algorithm would be T. Additionally, there would be at least one expert having loss $\leq T/n$, leading to regret $\geq T - T/n$, which is $\Omega(T)$.

Therefore, instead of choosing a single expert, we allow the algorithm to choose a probability distribution over experts, and define the regret of the algorithm in terms of its *expected loss* according to this distribution.

Learning from Expert Advice

At each round $t \in \{1, 2, \cdots, T\}$,

- The learner chooses a distribution $\vec{p}_t$.
- Each expert $i \in \{1, \cdots, n\}$ suffers loss $\ell_{i,t} \in [0, 1]$.
- The learner suffers expected loss $\vec{p}_t \cdot \vec{\ell}_t$.

The regret is then

$$\sum_{t=1}^{T} \vec{p}_t \cdot \vec{\ell}_t - \min_{i \in \{1,\ldots,n\}} \sum_{t=1}^{T} \ell_{i,t}.$$

Additional Notation: Note that here and throughout these notes we use $\ell_{i,t}$ and $p_{i,t}$ to denote the ith components of the vectors $\vec{\ell}_t$ and $\vec{p}_t$ respectively. We will use $\ell_{\mathcal{A},t} = \vec{p}_t \cdot \vec{\ell}_t$ to denote the (expected) loss of the learning algorithm on round t, $L_{i,t}$ to denote the cumulative loss of expert i up to time t (i.e., $L_{i,t} = \sum_{s=1}^{t} \ell_{i,s}$), and $L_{\mathcal{A},t}$ to denote the cumulative loss of the algorithm up to time t. Let Δ_n denote the probability simplex in n dimensions, that is, the set of all probability distributions over n mutually exclusive and exhaustive events.

2 Randomized Weighted Majority

Perhaps the most popular algorithm for the expert advice framework is the *Randomized Weighted Majority* algorithm, sometimes called *Hedge*. Randomized Weighted Majority (RWM) is simple to describe. It takes a single parameter, $\eta > 0$. At each round t, it chooses $\vec{p}_t$ by setting

$$w_{i,t} = e^{-\eta L_{i,t-1}}$$

and

$$p_{i,t} = \frac{w_{i,t}}{\sum_{j=1}^{n} w_{j,t}}$$

for all experts $i \in \{1, \cdots, n\}$. Over the next few lectures, we will show that RWM has a regret of $O(\sqrt{T \log n})$ when η is set optimally. In this lecture, we will examine what this algorithm is doing in more detail to build some intuition before we jump into the bounds.

2.1 Multiplicative Updates

We first point out a similarity between Randomized Weighted Majority and Winnow: Both maintain a set of weights over experts/features, and update these weights multiplicatively. Indeed, for RWM we have that

$$w_{i,t+1} = e^{-\eta L_{i,t}} = e^{-\eta(L_{i,t-1} + \ell_{i,t})} = e^{-\eta L_{i,t-1}} e^{-\eta \ell_{i,t}} = w_{i,t} e^{-\eta \ell_{i,t}} .$$

Consider the binary classification setting in which losses are either 0 (if an expert's prediction is correct) or 1 (if an expert makes a mistake). If expert i makes a mistake at time t, RWM sets

$$w_{i,t+1} \leftarrow \frac{w_{i,t}}{e^{\eta}} .$$

This is very similar to the way that Winnow updates on a mistake, with a slightly different way of writing the parameters; e^{η} now plays the role of $1 + \beta$.

We next consider a different interpretation of RWM based on the idea of *regularization*.

3 Regularization

We will consider an alternative interpretation of RWM based on regularization. To motivate this, we first consider a different, very simple algorithm for the expert advice framework, and see why it fails.

3.1 Follow the Leader

In the expert advice framework, the learning algorithm is required to choose a distribution $\vec{p}_t$ over experts at each time t. it might seem to be a good idea to choose the distribution $\vec{p}$ with the best performance on previously observed data, i.e., set

$$\vec{p}_t = \arg \min_{\vec{p} \in \Delta_n} \sum_{s=1}^{t-1} \vec{\ell}_s \cdot \vec{p} = \arg \min_{\vec{p} \in \Delta_n} \vec{L}_{t-1} \cdot \vec{p} .$$

However, there are very simple sequences on which this algorithm has very bad performance. Suppose we have only two experts. Consider the following sequence of losses:

$$\vec{\ell}_1 = \langle 1, 0 \rangle, \vec{\ell}_2 = \langle 0, 1 \rangle, \vec{\ell}_3 = \langle 1, 0 \rangle, \vec{\ell}_4 = \langle 0, 1 \rangle, \cdots$$

Since our algorithm sets all the weight on the best expert in history, the distribution sequence generated by our algorithm becomes

$$\vec{p}_1 = \langle 0.5, 0.5 \rangle, \vec{p}_2 = \langle 0, 1 \rangle, \vec{p}_3 = \langle 1, 0 \rangle, \vec{p}_4 = \langle 0, 1 \rangle, \cdots$$

(The distribution at time 1 is arbitrary.)

We can easily see that our algorithm is tricked by the adversary. After T time steps, both experts have cumulative loss $T/2$, while the algorithm has loss (roughly) T. Thus, the regret for this algorithm would be $T - T/2 = T/2 = \Omega(T)$.

3.2 Adding Regularization

The Follow the Leader algorithm is inherently unstable. It can switch all of its weight from one expert to another on every time step. In order to add some stability, we can introduce a regularization term $R(\vec{p})$, where R is a convex function called the regularizer. The resulting algorithm is the so-called Follow the Regularized Leader algorithm, with

$$\vec{p}_t = \arg\min_{\vec{p} \in \Delta_n} \left(\eta \sum_{s=1}^{t-1} \vec{\ell}_s \cdot \vec{p} + R(\vec{p}) \right) .$$

The parameter $\eta > 0$ allows us to adjust the relative impact of the two terms.

In the next lecture, we will prove a regret bound for algorithms of this form. It turns out that RWM is one such algorithm. In particular, the distribution $\vec{p}_t$ selected by RWM at round t is precisely the distribution $\vec{p}$ that minimizes

$$\eta \sum_{s=1}^{t-1} \vec{\ell}_s \cdot \vec{p} - H(\vec{p})$$

where H is the entropy function,

$$H(\vec{p}) = \sum_{i=1}^{n} p_i \log \frac{1}{p_i}.$$

Here, by convention, $0 \log(1/0)$ is defined to be 0. Note that when $\vec{p}$ puts all of its weight on a single expert, we have $H(\vec{p}) = 0$. When $\vec{p}$ is uniform, we have $H(\vec{p}) = \log n$. In general, higher entropy implies "more uniform" weights. Because of this, using the negative entropy as the regularization term makes sense. Encouraging high entropy leads to more uniform weights, which leads to more stability in the algorithm.

4 Road Map

In the next lecture we will complete the proof that RWM is a Follow the Regularized Leader algorithm. We will then analyze the class of Follow the Regularized Leader algorithms and prove a general regret bound for this class, which can be used to derive a regret bound for RWM.