10-806 Foundations of Machine Learning and Data Science

Lecturer: Avrim Blum Lecture 11: October 14, 2015

1 Online Learning II: ERM and Follow the (Regularized) Leader

1.1 Overview

We saw last time that ERM doesn't always do well in online learning, even in simple settings. This then led us to formulate and analyze the *halving algorithm* for the realizable case, and the *randomized weighted majority* algorithm for the general (non-realizable / agnostic) case. For example, for learning a finite class $\mathcal{C}$, we saw a setting where ERM could make up to $|\mathcal{C}| - 1$ mistakes,[1] whereas the halving algorithm always makes at most $\lg |\mathcal{C}|$ mistakes. In the general (non-realizable) case, the randomized weighted majority algorithm makes at most

$$\min_{h \in \mathcal{C}} \left[(1+\epsilon) L(h) + \frac{\log |\mathcal{C}|}{\epsilon} \right]$$

mistakes in expectation, where $L(h)$ is the total loss (number of mistakes) of h. Tuning ϵ based on the total number of rounds T, we get an expected total loss at most

$$\min_{h \in \mathcal{C}} \left[L(h) + O(\sqrt{T \log |\mathcal{C}|}) \right],$$

i.e., a regret with respect to $\mathcal{C}$ of $O(\sqrt{T \log |\mathcal{C}|})$. Randomized weighted majority is also called the *hedge* or *multiplicative weights* algorithm.

One problem with randomized weighted majority is that it requires keeping explicit track of weights across all $h \in \mathcal{C}$. So, even though the regret is logarithmic in $|\mathcal{C}|$, the running time is linear in $\mathcal{C}$. Today we will look at ways of getting around this problem, and getting a good regret bound efficiently, at least in some interesting cases.

We begin by talking about ERM, which in the online learning scenario is often called "Follow the Leader" (since you are picking the $h \in \mathcal{C}$ that performed best in the past and using it today). As we saw, ERM doesn't always do well, but we will see a useful theorem we *can* prove about it. We'll then see how, motivated by this, we can modify ERM to yield algorithms with interesting worst-case guarantees.

1.2 The general setting

We assume we have a class $\mathcal{C}$ of hypotheses, or "experts" or actions. At each time step $t = 1, 2, 3, \ldots$, our algorithm chooses some $h_t \in \mathcal{C}$. Then the world chooses a loss function $\ell_t : \mathcal{C} \to [0, 1]$, and we pay $\ell_t(h_t)$. We will soon put constraints on what ℓ_t can look like, but for now let's allow it to be an arbitrary function from $\mathcal{C}$ to $[0, 1]$. Also, define

$$L_T(h) \quad = \quad \ell_1(h) + \ell_2(h) + \ldots + \ell_T(h) \tag{1}$$

[1] E.g., learning an interval in the set of n-bit integers $\{0, 1, 2, \ldots, 2^n - 1\}$.

to be the total loss of h from time 1 to T. ERM, which is more typically called *Follow the Leader* (FTL) in the online setting, is the algorithm that chooses

$$h_t \quad = \quad \arg\min_{h \in \mathcal{C}} L_{t-1}(h), \tag{2}$$

with ties broken in some deterministic manner.

In this general setting, ERM/FTL can do very badly. Suppose that ℓ_t is defined as $\ell_t(h_t) = 1$ and $\ell_t(h) = 0$ for all $h \in \mathcal{C}, h \neq h_t$, where h_t is as defined in equation (2). Then in T time steps, we get $L_T(FTL) = T$ but we can see by the pigeon-hole principle that $\min_{h \in \mathcal{C}} L_T(h) = \lfloor T/|\mathcal{C}| \rfloor$. So, we have a factor of $|\mathcal{C}|$ ratio between the loss of the algorithm and the loss of the best $h \in \mathcal{C}$.[2]

However, it turns out there is something interesting we can say.

1.3 A useful theorem about ERM / FTL

Theorem 1 *Let $h_t = \text{argmin}_{h \in \mathcal{C}} L_{t-1}(h)$, as in FTL. Then for any T, for any sequence of T loss functions, we have:*

$$\sum_{t=1}^{T} \ell_t(h_t) - \min_{h \in \mathcal{C}} \left[\sum_{t=1}^{T} \ell_t(h) \right] \quad \leq \quad \sum_{t=1}^{T} [\ell_t(h_t) - \ell_t(h_{t+1})] \tag{3}$$

Notice that the left-hand-side of equation (3) is the exactly regret of FTL with respect to the best hypothesis in $\mathcal{C}$. So, what Theorem 1 is saying is that so long as the quantity "$\ell_t(h_t) - \ell_t(h_{t+1})$" is small on average, then regret will be low. For example, one immediate corollary is:

Corollary 1 *If FTL changes its hypothesis at most k times on some sequence of loss functions $\ell_1, \ell_2, \ldots$, then its regret on this sequence is at most k.*

Another case we will be particularly interested in is where there is structure to the loss functions and the class $\mathcal{C}$. For instance, if the hypotheses and loss functions can be viewed as vectors in the unit ball in R^n, with $\ell(h) = \langle \ell, h \rangle$, then so long as we can somehow ensure that the hypotheses of FTL do not change too radically from time t to time $t+1$, the regret will be small.

Proof of Theorem 1: First, by subtracting $\sum_{t=1}^{T} \ell_t(h_t)$ from both sides and moving the remaining terms, we can rewrite the statement we wish to prove as:

$$\sum_{t=1}^{T} \ell_t(h_{t+1}) \quad \leq \quad \min_{h \in \mathcal{C}} \left[\sum_{t=1}^{T} \ell_t(h) \right].$$

In other words, if we could see into the future and use at time t the hypothesis that FTL is planning to use at time $t+1$, then we would have zero regret.[3] We can prove this by induction on T.

[2] One can see that this fact actually holds for any deterministic algorithm that must choose $h_t \in \mathcal{C}$, when we have no assumptions about the loss functions. Note that this does not apply to the randomized weighted majority algorithm because that algorithm is not deterministic, nor to the halving algorithm, because that algorithm does not choose functions in $\mathcal{C}$.

[3] This "algorithm", which uses today what FTL would do tomorrow, is sometimes called *Be the Leader* (BTL). Of course, we cannot run this algorithm without seeing the future.

Base case: For $T = 1$, by definition, $h_2 = \arg\min_{h \in \mathcal{C}} \ell_1(h)$, so our formula holds at equality.

Inductive case: Assume by induction that

$$\sum_{t=1}^{T-1} \ell_t(h_{t+1}) \leq \min_{h \in \mathcal{C}} \sum_{t=1}^{T-1} \ell_t(h).$$

Note that this certainly implies (instantiating $h = h_{T+1}$ on the RHS) that

$$\sum_{t=1}^{T-1} \ell_t(h_{t+1}) \leq \sum_{t=1}^{T-1} \ell_t(h_{T+1}).$$

Adding $\ell_T(h_{T+1})$ to both sides, we have:

$$\sum_{t=1}^{T} \ell_t(h_{t+1}) \leq \sum_{t=1}^{T} \ell_t(h_{T+1}).$$

Finally, by definition, $h_{T+1} = \arg\min_{h \in \mathcal{C}} \sum_{t=1}^{T} \ell_t(h)$. So we have:

$$\sum_{t=1}^{T} \ell_t(h_{t+1}) \leq \min_{h \in \mathcal{C}} \sum_{t=1}^{T} \ell_t(h)$$

as desired. ■

1.4 Follow the Regularized Leader

Theorem 1 suggests the following idea for achieving a good regret bound in situations where the loss functions ℓ_t and the hypothesis class $\mathcal{C}$ have structure. Suppose we add in a fake "day 0", where the loss on day 0 is given by a penalty function on hypotheses called a *regularizer* $R : \mathcal{C} \to [0, \infty)$. We now run FTL where we include the fake day 0. So, now the algorithm looks as follows:

Follow the Regularized Leader (FTRL):

- Begin with $h_1 = \arg\min_{h \in \mathcal{C}} R(h)$.
- For $t = 2, 3, \ldots$ let $h_t = \arg\min_{h \in \mathcal{C}} [R(h) + L_{t-1}(h)]$.

If we can find a function R such that hypotheses don't change quickly and yet R's penalties do not get too large, we will perform well. Specifically, Theorem 1 immediately implies the following bound for Follow the Regularized Leader.[4]

Theorem 2 *Let $h_t = \operatorname{argmin}_{h \in \mathcal{C}} [R(h) + L_{t-1}(h)]$, as in FTRL. Then for any T, for any sequence of T loss functions, for any $h^* \in \mathcal{C}$, we have:*

$$\sum_{t=1}^{T} \ell_t(h_t) - \sum_{t=1}^{T} \ell_t(h^*) \quad \leq \quad \sum_{t=1}^{T} [\ell_t(h_t) - \ell_t(h_{t+1})] + [R(h^*) - R(h_1)]. \tag{4}$$

[4]We have changed "min over $h \in \mathcal{C}$" to the equivalent "for any $h^* \in \mathcal{C}$" in order to be able to move $R(h^*)$ to the other side of the equation without confusion.

We can interpret this as follows. Suppose there is structure on the loss functions so that similar hypotheses have similar losses (e.g., perhaps they are linear functions) and suppose we can come up with a regularizer that guarantees that hypotheses change slowly, guaranteeing that $\ell_t(h_t) - \ell_t(h_{t+1}) \leq \epsilon$. Then our regret will be at most $\epsilon T + \max_{h \in \mathcal{C}} R(h)$.

1.5 FTRL with a quadratic regularizer for linear optimization

Let's apply this to the following natural scenario. Suppose that we are doing online linear optimization. $\mathcal{C}$ is the set of all points in $\mathcal{R}^d$, and the losses are linear functions. That is, $\ell_t(h) = \langle \ell_t, h \rangle$. We need to keep losses (or gains) bounded so let's assume $||\ell_t|| \leq 1$ and also that we are going to compete with the optimal h^* such that $||h^*|| \leq 1$. Let's use a regularizer $R(h) = \sqrt{\frac{T}{2}}||h||^2$. So we will start off with $h_1 = \vec{0}$ and run FTRL from there.

We need to figure out how far apart h_{t+1} is from h_t. To do this, let's look at the formula for $h_{t+1} = \text{argmin}_h[\sqrt{\frac{T}{2}}||h||^2 + \sum_{\tau=1}^{t} \langle \ell_\tau, h \rangle]$. Taking partial derivatives in each coordinate and setting them to 0 we get that in each coordinate j we have $\sqrt{2T} h_{t+1,j} + \sum_{\tau=1}^{t} \ell_{\tau,j} = 0$ which implies that $h_{t+1} = \frac{-1}{\sqrt{2T}} \sum_{\tau=1}^{t} \ell_\tau$. Equivalently, $h_{t+1} = h_t - \frac{1}{\sqrt{2T}} \ell_t$.

This tells us two things: first of all, this algorithm is equivalent to doing online gradient descent. Secondly, $\ell_t(h_t) - \ell_t(h_{t+1}) = \langle \ell_t, h_t - h_{t+1} \rangle = \frac{1}{\sqrt{2T}} \langle \ell_t, \ell_t \rangle \leq \frac{1}{\sqrt{2T}}$. So, overall, our total regret is at most $R(h^*) + \sum_{t=1}^{T} \frac{1}{\sqrt{2T}} \leq \sqrt{\frac{T}{2}} + \sqrt{\frac{T}{2}} = \sqrt{2T}$.

Extensions: convex sets, convex functions

If $\mathcal{C}$ is not all of $\mathcal{R}^d$ but instead a convex set in $\mathcal{R}^d$, then for this regularizer each h_t will just be the projection of the overall minimizer onto $\mathcal{C}$ (the nearest point to it in $\mathcal{C}$). Since projecting points to a convex set can only decrease their distance, this can only shrink the distance between h_t and h_{t+1}.

Another extension is that if each $\ell_t(h)$ is not a linear function but is convex, then notice that if we just replace ℓ_t with a tangent plane to ℓ_t at h_t this will only make the problem harder (h_t pays the same, and h^* perhaps pays less). Since translations don't change regret, we can translate that tangent plane to go through the origin, making it a linear function. Another way to say this is that if we have convex loss functions, we can reduce to the linear case by running FTRL on subgradients of the loss functions.

More information: For more information on these and related methods, see [1, 2].

References

[1] Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2), 2012. http://www.cs.huji.ac.il/~shais/papers/OLsurvey.pdf.

[2] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, pages 928–936, 2003.