10-806 Foundations of Machine Learning and Data Science

Lecturer: Avrim Blum Lecture 12: October 19, 2015

---

1 Recap: Follow the Regularized Leader

The idea of FTRL is we run FTL but 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 $h_t = \arg\min_{h \in \mathcal{C}}[R(h) + (\ell_1(h) + \ldots + \ell_{t-1}(h))]$.

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 from last time immediately implies the following bound for Follow the Regularized Leader.

Theorem 1 *Let $h_t = \text{argmin}_{h \in \mathcal{C}}[R(h) + (\ell_1(h) + \ldots + \ell_{t-1}(h))]$, as in FTRL. Then for any T, for any sequence of T loss functions, for any $h^* \in \mathcal{C}$, we have:*

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

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 $\epsilon T + \max_{h \in \mathcal{C}} R(h)$.

1.1 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 $\mathcal{R}^n$, 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 $\mathcal{R}^n$ but instead a convex set in $\mathcal{R}^n$, 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.

2 Online Learning III: Follow the Perturbed Leader

2.1 Making this more tangible

The above setting looks fairly abstract. Let's now look at some more tangible problems we can model this way, or close to this way. This will then lead us to another algorithm that is sometimes more efficient. From now on we will assume the loss functions are *linear*, so we can think of them as vectors with $\ell_t(h) = \langle \ell_t, h \rangle$.

Online ranking: Imagine a search engine that given a query q outputs a list h_1 of web pages. The user clicks on one, and we define the loss of h_1 as the depth in the list of the web-page that was clicked. The next time q is queried we give a different ordering h_2 of these web pages, and the user this time clicks on (perhaps) a different page. Our goal is to do nearly as well as best fixed ordering of these pages in hindsight. Suppose there are n webpages in our list, and to keep the losses in $[0, 1]$ let's say the loss for the user clicking the top web page is $1/n$, for clicking the second web page on the list is $2/n$, and so on, down to n/n for the last one. We could model this as an "experts" problem with $n!$ experts but that would be too much computation. Instead, let $\mathcal{C}$ be the set of all $n!$ points in $\mathcal{R}^n$ with coordinates $\{1/n, 2/n, \ldots, n/n\}$ in some order, with the interpretation that each coordinate is one of the web-pages and the value of that coordinate is the depth of that page in the list (divided by n). If the user at time t clicks on web-page i, define $\ell_t = e_i$ to be the unit vector in coordinate i. Then $\langle \ell_t, h_t \rangle$ is the loss of ordering h_t.

Online shortest paths: Consider the following adaptive route-choosing problem. Imagine each day you have to drive between two points s and t. You have a map (a graph G) but you don't know what traffic will be like (what the cost of each edge will be that day). Say the costs are only revealed to you after you get to t. We can model this by having one dimension per edge, and each path is represented by the indicator vector listing the edges in the path. Then the loss vector ℓ_t is just the vector with the costs of each edge that day. Notice that you *could* represent this as an experts problem also, with one expert per path, but the number of s-t paths can be exponential

in the number of edges in the graph (e.g., think of the case of a grid). However, given any set of edge lengths, we can efficiently compute the best path for that cost vector, since that is just a shortest-path problem. You don't have to explicitly list all possible paths.

The standard expert setting: We can also model the standard experts setting by having one coordinate per expert, and defining $\mathcal{C}$ to be the simplex $\mathcal{C} = \{p = (p_1, \ldots, p_n) : \sum_i p_i = 1 \text{ and } p_j \geq 0 \; \forall j\}$. Note that now $||\ell_t||$ could be as large as $\sqrt{n}$ if all experts have loss of 1.

2.2 Follow the Perturbed Leader

Notice that in the ranking and shortest path examples above, the set $\mathcal{C}$ is discrete, and therefore not convex. However, both cases have the nice property that ERM can be performed efficiently: in the case of ranking, this is just ordering the items by frequency. In the case of online shortest paths, this is just summing all the loss vectors and then solving a shortest path problem.

This now motivates the Follow the Perturbed Leader algorithm, where instead of adding a convex regularizer over a convex set, we add a probabilistic linear regularizer over a discrete set. That is, we add in a fake "day 0" (which is linear just like the other days) where the losses for that day are chosen randomly from an appropriate probability distribution. From then on, we just run ERM with the fake day included which as we said can be done efficiently. We then show that our regret will be low in expectation. For this analysis, we will assume that the series of loss vectors $\ell_1, \ell_2, \ldots$ has been determined in advance by the world (but the future is just unknown to us), so that new loss vectors do not depend on our algorithm's previous actions. (E.g., we are not buying enough stocks to affect the market). This means that in analyzing regret, we can think of it as "for any sequence of loss vectors, our expected loss will be close to the loss of the best $h \in \mathcal{C}$ for that sequence". We'll see where specifically we use this in the argument.

We also assume the following boundedness conditions:

- The maximum L_1 length of any loss vector ℓ_t is 1.
- The maximum L_1 distance between any two $h, h' \in \mathcal{C}$ is D (i.e., D is the diameter of $\mathcal{C}$).

We will choose loss vector ℓ_0 at random in $[0, 2/\epsilon]^n$, where n is the dimension of the space.

2.3 Analysis

We can directly apply Theorem 1 to this scenario, using $R(h) = \langle \ell_0, h \rangle$. For any $h^* \in \mathcal{C}$, we have:

$$\sum_{t=1}^{T} \ell_t(h_t) - \sum_{t=1}^{T} \ell_t(h^*) \quad \leq \quad \sum_{t=1}^{T} [\ell_t(h_t) - \ell_t(h_{t+1})] + [\ell_0(h^*) - \ell_0(h_1)].$$

The last term on the RHS is easy to analyze. Since $\mathcal{C}$ has L_1 diameter D, and each coordinate in ℓ_0 is at most $2/\epsilon$, the value of $\ell_0(h^*) - \ell_0(h_1)$ is at most $2D/\epsilon$. So, what remains is to analyze the expected value of $\ell_t(h_t) - \ell_t(h_{t+1})$. By linearity of expectation, this is $\mathbf{E}[\ell_t(h_t)] - \mathbf{E}[\ell_t(h_{t+1})]$.

Now, to help us notationally, given two points $a, b \in \mathcal{R}^n$, define $Box(a, b) = \{x : a_i \leq x_i \leq b_i \; \forall i\}$. So, the algorithm is choosing ℓ_0 at random in $Box(\vec{0}, \frac{\vec{2}}{\epsilon})$ where $\frac{\vec{2}}{\epsilon} = (\frac{2}{\epsilon}, \ldots, \frac{2}{\epsilon})$.

Let p be the fraction of $Box(\vec{0}, \frac{\vec{2}}{\epsilon})$ that lies outside $Box(\ell_t, \frac{\vec{2}}{\epsilon})$. By symmetry, this is also the fraction of $Box(\vec{0}, \frac{\vec{2}}{\epsilon})$ that lies outside $Box(\vec{0}, \frac{\vec{2}}{\epsilon} - \ell_t)$. By the fact that ℓ_t has L_1 length at most 1, we have $p \leq \epsilon/2$, which can be seen by adding up the portion outside, coordinate by coordinate.

We can now write:

$$\begin{aligned}
\mathbf{E}[\ell_t(h_t)] &= (1-p)\mathbf{E}\left[\ell_t(h_t)|\ell_0 \in Box\left(\ell_t, \frac{\vec{2}}{\epsilon}\right)\right] + p\mathbf{E}\left[\ell_t(h_t)|\ell_0 \notin Box\left(\ell_t, \frac{\vec{2}}{\epsilon}\right)\right], \\
\mathbf{E}[\ell_t(h_{t+1})] &= (1-p)\mathbf{E}\left[\ell_t(h_{t+1})|\ell_0 \in Box\left(\vec{0}, \frac{\vec{2}}{\epsilon} - \ell_t\right)\right] + p\mathbf{E}\left[\ell_t(h_{t+1})|\ell_0 \notin Box\left(\vec{0}, \frac{\vec{2}}{\epsilon} - \ell_t\right)\right].
\end{aligned}$$

Here is the key: notice that the first terms on the RHS in the two equations above are equal! That is because the distribution on $\ell_0 + \ldots + \ell_{t-1}$ given that $\ell_0 \in Box(\ell_t, \frac{\vec{2}}{\epsilon})$ is identical to the distribution on $\ell_0 + \ldots + \ell_t$ given that $\ell_0 \in Box(\vec{0}, \frac{\vec{2}}{\epsilon} - \ell_t)$.[1] So, the distribution on h_t in the term in the first equation is equal to the distribution on h_{t+1} in the corresponding term in the second equation. So, all we need to worry about is the second term. But since $p \leq \epsilon/2$, in the worst case the difference here is at most $D\epsilon/2$ where D was the L_1 diameter of $\mathcal{C}$.[2]

So, overall, our expected regret is at most $2D/\epsilon + TD\epsilon/2$. Setting $\epsilon = 2/\sqrt{T}$, our expected regret is at most $2D\sqrt{T}$.

For more information on FPL, see [1].

References

[1] Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291 – 307, 2005.

[1] This is where we are using the fact that the quantities $\ell_1, \ldots, \ell_t$ have been determined in advance, before ℓ_0 is chosen.

[2] Technically, for this part we could use the L_∞ diameter of $\mathcal{C}$ since we've bounded the L_1 length of the loss vectors by 1.