CS229T/STATS231: Statistical Learning Theory

Lecturer: Tengyu Ma
Scribe: Chris Cundy, Ananya Kumar

Lecture # 16
November 14, 2018

1 Review and Overview

Last time, we discussed how the `Follow the Leader` algorithm does not perform very well as an online convex optimization algorithm, and how we could introduce the `Follow the Regularized Leader` algorithm to fix some of the shortcomings of FTL. In this lecture we will show the rates of FTRL on the concrete examples we discussed earlier to build some intuition. To use FTRL on the expert problem we will introduce the entropic regularizer, a useful regularizer when we have to make actions that are probability distributions.

Recall that on each timestep FTRL chooses

$$\omega_t = \operatorname*{argmin}_{\omega \in \Omega} \sum_{i=1}^{t-1} f_i(\omega) + \frac{1}{\eta} \underbrace{\ell(\omega)}_{\text{s.c}}, \tag{1}$$

where ω is in the set of actions the algorithm can take, and ℓ is strongly convex with respect to some norm. (Good to expand what each term is here)

Now last time we proved that the regret at time t, R_t is bounded as follows:

$$R_t \leq \frac{D}{\eta} + \eta \sum_{t=1}^{T} \|\nabla f_t(\omega_t)\|_*^2, \tag{2}$$

where $\|\cdot\|_*$ is the dual norm of the norm that ℓ is strongly convex with respect to and the parameter D is given by the range of values that the regularizer φ can output:

$$D = \max_{x \in \Omega} \varphi(x) - \min_{x \in \Omega} \varphi(x). \tag{3}$$

So if we can show that the dual norm of the gradient is bounded, i.e. $\|\nabla f_t(\omega)\|_* \leq G$ for all t and $\omega \in \Omega$, we can find the value of η which gives the tightest bound. This is $\eta = \sqrt{D/TG^2}$ and substituting in we see

$$R_t \leq \frac{D}{\eta} + \eta T G^2 \in \mathcal{O}\left(G\sqrt{TD}\right). \tag{4}$$

Today we will show concretely the values of D, G, etc on the problems we motivated our discussion of online learning with, and investigate which regularizers we might want to use.

2 Online Regression

We are given data points $\{x \in \mathbb{R}^d : \|x\|_2 \leq 1\}$ and a label $y_t \in [-1, 1]$. We can output a weight vector in $\Omega = \{\omega : \|\omega\|_2 \leq 1\}$. This is a bit more concrete than when we introduced the problem last time, as we didn't have the norm-constraints on the variables.

We receive (x_t, y_t) and have $f_t(\omega) = \frac{1}{2}(\langle \omega, x_t \rangle - y_t)^2$. We choose the regularizer as $\varphi(\omega) = \frac{1}{2}\|\omega\|_2^2$ which is 1-strongly convex with respect to the ℓ_2-norm.

To see the strong convexity, we first argue that a differentiable function $\varphi(\omega)$ is α-strongly convex w.r.t. ℓ_2 norm if and only if $\varphi(\omega) - \frac{\alpha}{2}\|\omega\|_2^2$ is convex. This follows from rearranging: by definition, the convexity of $\varphi(\omega) - \frac{\alpha}{2}\|\omega\|_2^2$ is equivalent to

$$\begin{aligned}\varphi(\omega) - \frac{\alpha}{2}\|\omega\|_2^2 - \varphi(\omega') + \frac{\alpha}{2}\|\omega'\|_2^2 &\geq \langle \nabla\varphi(\omega) - \alpha\omega, \omega - \omega'\rangle \\ \Leftrightarrow \varphi(\omega) - \varphi(\omega') &\geq \langle \nabla\varphi(\omega), \omega - \omega'\rangle + \frac{\alpha}{2}\|\omega\|_2^2 + \frac{\alpha}{2}\|\omega'\|_2^2 - \alpha\langle\omega, \omega'\rangle \\ \Leftrightarrow \varphi(\omega) - \varphi(\omega') &\geq \langle \nabla\varphi(\omega), \omega - \omega'\rangle + \frac{\alpha}{2}\|\omega - \omega'\|_2^2\end{aligned}$$

Now we can apply this since $\frac{1}{2}\|\omega\|_2^2 - \frac{1}{2}\|\omega\|_2^2 \geq 0$ to conclude that $\frac{1}{2}\|\omega\|_2^2$ is 1-strongly convex.

What is G? We will show that $G = 2$ in this case, using the fact that the dual norm of the ℓ_2-norm is the ℓ_2-norm:

$$\begin{aligned}\|\nabla f_t(\omega_t)\|_2 = \|x(\langle\omega_t, x_t\rangle - y_t)\|_2 &= \underbrace{|\langle\omega_t, x_t\rangle - y_t|}_{\leq 2} \cdot \|x_t\|_2 \\ &\leq 2.\end{aligned} \tag{5}$$

Now D is max $\varphi(\omega) - \min\ \varphi(\omega)$. So $D \leq 1/2 - 0 = 1/2$. This gives us a bound on the regret:

$$R_T \leq \mathcal{O}\left(\sqrt{T}\right) \tag{6}$$

Recall the 'batch to online reduction': we can convert a batch algorithm to an online one by feeding each data point sequentially to the online algorithm. This involves running the online system for T timesteps, getting average loss R_T/T. For our regression case, this gives us an $\mathcal{O}\left(n^{-1/2}\right)$ rate for the loss, which is a reasonable rate. This is typically the best rate we can hope for without some special knowledge on the problem (such as realizability, etc). The fact that we get a reasonable rate should be an indication that we haven't made a big mistake along the way.

3 Expert Problem

Recall the expert problem, where we have N 'experts'. At each timestep we output a probability distribution, describing a stochastic policy of the form 'choose expert 1 with probability p_1, expert 2 with probability p_2', etc. We represent that choice of stochastic policy with a vector p which lies in the N-simplex $\Delta(N)$, i.e. we have an action space

$$\Omega = \Delta(N) = \left\{p : \sum p = 1, 0 \leq p \leq 1\right\}. \tag{7}$$

We receive a loss $f_t(p) = \langle\ell_t, p\rangle$ where ℓ_t is the loss vector $\ell_t \in [0,1]^N$, i.e. ℓ_t quantifies how well each of the experts performed at step t. We will apply FTRL with two choices of the regularizer: the simple choice of the ℓ_2-norm, and the more problem-specific entropic regularizer. We will see that the second choice of the regularizer gives a better bound on the regret.

3.1 ℓ_2-regularized

If we use the ℓ_2-norm with a regularizer $\varphi(p) = \frac{1}{2}\|p\|_2^2$ then as before this is strongly convex wrt the ℓ_2-norm. Computing D, we see

$$D = \sup_{p\in\Omega} \varphi(p) - \inf_{p\in\Omega} \varphi(p) \leq \sup_{p\in\Delta(N)} \tfrac{1}{2}\|p\|_2^2 \leq \frac{1}{2}, \tag{8}$$

as the sup over a simplex has to be at the vertices, and $\|p\|_2 \leq \|p\|_1$. Turning to G, we see that

$$\|\nabla f_t(p_t)\|_2 = \|\ell_t\|_2 \leq \sqrt{N} \qquad (\text{as } p \in \mathbb{R}^d, p_i \leq 1). \tag{9}$$

This holds because $\ell_t \in [0,1]^N$. Plugging into equation 4, we achieve a rate of $\mathcal{O}(\sqrt{NT})$. As discussed above, this is likely a 'good' rate with respect to T. However, it is not a great rate in N. It isn't immediately clear why $\sqrt{N}$ is a bad rate in N, but we will show soon that we are able to get $\log N$ dependence on N if we use a different regularizer. In some sense the worse rate that we achieve is because we use a regularizer that isn't very 'compatible' with the actions. We can hope to do better by choosing a regularizer that fits with the constraint on p to lie in the N-simplex.

3.2 Entropic Regularization

Since p is a probability distribution, we might aim for a regularizer with an information-theoretic basis. We want to penalize p that are too certain, concentrating probability around a small number of experts. A natural choice is to penalize p if the entropy of p is too low.

$$\varphi(p) = -H(p) = \sum_{i=1}^{N} p(i)\log p(i). \tag{10}$$

Now we can show that $\varphi(\cdot)$ is 1-strongly convex with respect to the ℓ_1-norm: we want to show that

$$\varphi(p) - \varphi(q) \geq \langle \nabla\varphi(q), p-q\rangle + \tfrac{1}{2}\|p-q\|_1^2. \tag{11}$$

First note that

$$\nabla\varphi(q) = \begin{pmatrix} \log q_i + 1 \\ \vdots \\ \log q_N + 1 \end{pmatrix}. \tag{12}$$

So we can re-arrange equation 11 to get

$$\sum_{i=1}^{N} p_i \log p_i - \sum_{i=1}^{N} q_i \log q_i \;\geq \sum_{i=1}^{N} (\log q_i + 1)\cdot(p_i - q_i) + \tfrac{1}{2}\|p-q\|_1^2 \tag{13}$$

.

We can make further simplifications, as the $q_i \log q_i$ cancels on both sides, and we know that $\sum 1 \cdot (p_i - q_i) = 0$ since p, q are probability distributions. Finally we see that convexity of the entropic regularizer is equivalent to the following statement:

$$\underbrace{\sum_{i=1}^{N} p_i \log \frac{p_i}{q_i}}_{\mathrm{KL}(p\|q)} \geq \underbrace{\tfrac{1}{2}\|p-q\|_1^2}_{\mathrm{TV}(p,q)}, \tag{14}$$

which is true from the Pinsker inequality.

All that remains is to compute D and G:

$$D = \underbrace{\sup_{p\in\Omega} \varphi(p)}_{\leq 0 \text{ as } \varphi \leq 0} - \inf_{p\in\Omega} \varphi(p) \leq \log N, \tag{15}$$

because $H(p) \leq \log N$. Then for G we need to bound the gradient under the dual norm:

$$\|\nabla f_t(\omega_t)\|_*^2 = \|\ell_t\|_\infty \leq 1, \qquad (\text{so } G = 1). \tag{16}$$

where the last part holds because $\ell_t \in [0,1]^N$. Here we see that using the ℓ_∞-norm was useful as it allowed us to remove any N from G. Plugging in, we see that

$$R_t \leq \mathcal{O}(G\sqrt{TD}) = \mathcal{O}(\sqrt{T \log N}). \tag{17}$$

It's interesting that we can get regret that's logarithmic in N, but why is this algorithm able to work, while the unregularized follow the leader algorithm doesn't work? In the following section we will gain some intuition for the behaviour of FTRL by calculating the updates in the expert problem.

4 Computing the Updates for FTRL

Let's take a look at what the FTRL algorithm with entropic regularization is actually doing. We begin by deriving a closed form for p_t, the probability distribution over experts that we select at the t^{th} step.

$$
\begin{aligned}
p_t &= \operatorname*{argmin}_{p\in\Omega} \sum_{i=1}^{t-1} f_i(p) + \frac{1}{\eta}\varphi(p) \\
&= \operatorname*{argmin}_{p\in\Omega} \sum_{i=1}^{t-1} \langle l_i, p\rangle + \frac{1}{\eta}\varphi(p) \\
&= \operatorname*{argmin}_{p\in\Omega} \langle \sum_{i=1}^{t-1} l_i, p\rangle + \frac{1}{\eta}\varphi(p) \\
&= \operatorname*{argmin}_{p\in\Omega} \langle \eta \sum_{i=1}^{t-1} l_i, p\rangle - H(p) \\
&= \operatorname*{argmax}_{p\in\Omega} \langle -\eta \sum_{i=1}^{t-1} l_i, p\rangle + H(p)
\end{aligned}
$$

In homework 0 we dervied the closed form to this using Lagrange multipliers. That is, $p_t = \text{softmax}(-\eta \sum_{i=1}^{t-1} l_i)$. An alternative way we can do this is using Gibbs' variational principle. Gibbs variational principle is as follows, where μ and γ are probability distributions:

$$
\log \mathop{\mathbb{E}}_{x\sim\mu} e^{g(x)} = \sup_{\gamma} \left[\mathop{\mathbb{E}}_{x\sim\gamma}[g(x)] - \mathrm{KL}(\gamma\|\mu) \right]
$$

And the supremum is achieved when $\gamma(x) \propto \mu(x) e^{f(x)}$. To apply this to our problem, select μ to be the uniform distribution over $[N]$ and $\gamma = p$. Define $g(j) = -\eta(\sum_{i=1}^{t-1} l_i)_j$. Then, we have,

$$
\left\langle -\eta \sum_{i=1}^{t-1} l_i, p \right\rangle = \mathop{\mathbb{E}}_{j\sim p} \left[-\eta (\sum_{i=1}^{t-1} l_i)_j \right].
$$

Additionally, we have,

$$
\begin{aligned}
KL(\gamma\|\mu) &= \sum_{i=1}^{N} p_i \log \frac{p_i}{q_i} \\
&= \sum_{i=1}^{N} p_i \log \frac{p_i}{1/N} \\
&= \log N + \sum_{i=1}^{N} p_i \log p_i \\
&= -H(p) + \log N.
\end{aligned}
$$

Finally, we can compute the desired result, where μ and g are defined above:

$$
\operatorname*{argmax}_{p\in\Omega} \left[\left\langle -\eta \sum_{i=1}^{t-1} l_i, p \right\rangle + H(p) \right] = \operatorname*{argsup}_{p} \left(\mathop{\mathbb{E}}_{x\sim p}[g(x)] - \mathrm{KL}(p\|\mu) \right) + \log N
$$

So by Gibbs variational result, the argsup is achieved when $p_j \propto \exp[-\eta(\sum_{i=1}^{t-1} l_i)_j]$, which is precisely the softmax. This can be viewed as a *multiplicative weights update*. We begin with a uniform weight over all the experts. Whenever an expert makes a mistake, and incures loss 1, we divide the weight of that expert by a factor of $\exp(\eta)$ and re-normalize p. This is a very common procedure that is used in many algorithms in machine learning (e.g. boosting) and other fields.

5 The 'Convex to Linear Reduction'

We will now tackle the more general problem of online convex optimization. Let Ω be a set of parameters. Let G_Ω be a set of linear functions on Ω. By linear, we mean that for all $g \in G_\Omega$ there exists a s.t. $g(w) = \langle a, w \rangle$ for all $w \in \Omega$. Suppose we have an online learning algorithm A that achieves regret upper bounded by $R(T)$ on G_Ω.

Now, suppose we have a set F of differentiable convex functions on Ω such that for all $f \in F$ and all $w \in \Omega$, $\nabla f(w) \in G_\Omega$. We will show that we can achieve regret upper bounded by $R(T)$ on F. Our high level approach will be to reduce this online convex setting to the online linear setting, by taking linear approximations. We will then upper bound the regret by the regret in the online linear setting. To refresh our memories about convex functions, we begin with a simple result on convex functions that we will use. From the first order characterization of convex functions,

$$f(y) \geq f(x) + \langle \nabla f(x), (y - x) \rangle$$

With some manipulation, this gives us,

$$f(x) - f(y) \leq \langle \nabla f(x), (x - y) \rangle$$

In other words, we can upper bound the difference between function values using the gradient. This will allow us to bound the regret. We now give the online convex optimization algorithm that has regret $\leq R(T)$.

On each iteration $t \in [T]$:

1. Ask online linear algorithm A to pick action w_t.
2. Obtain a loss function $f_t \in F$.
3. As usual in the online setting, we incur loss $f_t(w_t)$.
4. Construct a linear approximation to the loss $g_t(w) = \langle \nabla f_t(w_t), w \rangle$.
5. Give g_t as the loss function to A.

We can now prove that the regret of this algorithm is bounded by $R(T)$, beginning from the definition of regret.

$$\begin{aligned}
\max_w \sum_{t=1}^{T} f_t(w_t) - f_t(w) &\leq \max_w \sum_{t=1}^{T} \langle \nabla f_t(w_t), (w_t - w) \rangle \\
&= \sum_{t=1}^{T} g_t(w_t) - g_t(w) \\
&\leq R(T)
\end{aligned}$$

Where the first step used the warm up result above on convex functions. If our regularization function $\varphi(w) = ||w||_2$, then this algorithm becomes online gradient descent. This is because,

$$w_t = \operatorname*{argmin}_{w \in \Omega} \sum_{i=1}^{t-1} \langle \nabla f_i(w_i), w \rangle + \frac{1}{\eta} ||w||_2$$

We can show that the solution to this is:

$$w_t = \prod_{\Omega} \left(-\eta \sum_{i=1}^{t-1} \nabla f_i(w_i) \right)$$

where Π_Ω projects the input to Ω (the projection is in ℓ_2 norm). This can be viewed as doing gradient descent, starting from $w = 0$, and then finally projecting the result onto Ω. Since we project after applying all the updates, this is known as lazy projection. This is in contrast to what we do in standard projected gradient descent, for example, where we project the parameters back into the constraint set at each step. That is known as eager projection.