6.883 Learning with Combinatorial Structure
Notes for Lecture 20
Author: Tuhin Sarkar

1 Recap

Last class we talked about online learning problems. Today we will develop regret bounds for the regularized FTL algorithm, introduce online mirror descent and bandit feedback problems. We will also discuss optimal regret bounds for each of these problems.

2 Online Gradient Descent

In the last class, we talked about a few examples:

- Prediction from expert advice: In this decision maker chooses from the advice of n experts. Each expert has unknown loss function $l_t(i) \in [0, 1]$. To learn which one is good, we maintain a distribution w over the experts, and pick i to be the next expert with probability $w(i)$. The hedge algorithm updates those weights in each round multiplicatively:

$$y_{t+1}(i) = w_t(i)e^{-\eta l_t(i)}$$
$$w_{t+1}(i) = \frac{y_{t+1}(i)}{\sum_i y_{t+1}(i)}$$

- Online Shortest paths: In this method we use the regularized follow the leader (FTL) as discussed in the last class.

Next we will derive regret bounds for Regularized FTL given as:

$$w_t = \arg\min_{w \in \mathcal{K}} \eta \sum_{\tau=1}^{t-1} l_\tau(w) + \mathcal{R}(w) \tag{1}$$

here $\mathcal{K}$ is a convex set and $\mathcal{R}(\cdot)$ is a strongly smooth, convex function.

2.1 Online Gradient Descent and its generalization

Both of the above methods will turn out to be closely related to online gradient descent (analyzed in [5]). Online gradient updates its weight vector as follows:

$$\begin{aligned} w_t &= \Pi_{\mathcal{K}}(w_{t-1} - \eta l_{t-1}) \\ &= \arg\min_{w \in \mathcal{K}} \eta l_t(w) + \frac{1}{2}||w - w_{t-1}||^2 \end{aligned}$$

One may thing of this as a linear local approximation to l_t, and the last term encourages the next iterate to be close to the previous one, adding stability.

The connection comes from a generalization: replace the squared Euclidean distance by a Bregman divergence $D_R(\cdot)$:

$$w_t = \arg\min_{w \in \mathcal{K}} \quad \eta l_t(w) + D_R(w, w_{t-1}). \tag{2}$$

(This is also a Bregman Proximity Operator.)

2.2 Bregman Divergence

A Bregman divergence is determined by a strictly convex function R, and is defined as

$$D_R(x, y) = R(x) - R(y) - (x - y)^{\mathrm{T}} \nabla R(y)$$

Below are some properties of Bregman divergences:

- $D_R(x, y) \geq 0$ and, for a strictly convex function R, $D_R(x, y) = 0$ if and only if $x = y$
- $D_R(x, y)$ is convex in x
- The Bregman projection on a convex set exists and is unique:

$$w' = \arg\min_{w \in \mathcal{K}} D_R(w, y)$$

Examples of $D_R(x, y)$ for different R:

1. For $R(x) = \frac{1}{2}||x||^2$ we have $D_R(x, y) = \frac{1}{2}||x - y||^2$.
2. For the entropy function, $R(x) = \sum_{i=1}^n x_i \log x_i - x_i \implies D_R(x, y) = \sum_{i=1}^n x_i \log \frac{x_i}{y_i} - \sum_i x_i + \sum_i y_i$.

2.2.1 Rewriting the Bregman update step

Next, we rewrite the Bregman proximity operator (2). It can be decomposed into two sub-steps, using an auxiliary intermediate variable y_{t+1}:

1. Unconstrained step:
$$y_{t+1} = \arg\min_{w \in \mathbb{R}^d} \eta l_t(w) + D_R(w, w_t) \tag{3}$$

2. Projection step:
$$w_{t+1} = \arg\min_{w \in \mathcal{K}} D_R(w, y_{t+1}) \tag{4}$$

In the unconstrained step, we obtain y_{t+1} by setting the derivative to zero:

$$0 = \eta l_t + \nabla R(y_{t+1}) - \nabla R(w_t) \quad \Leftrightarrow \quad \nabla R(y_{t+1}) = \nabla R(w_t) - \eta l_t. \tag{5}$$

When $R(x)$ is the entropy function, and $\mathcal{K} = \Delta^d$, *i.e.*, the probability simplex, then $\nabla R(w) = \log w$. Then, in the unconstrained Step 1 we choose y_{t+1} such that $\log y_{t+1}(i) = \log w_t(i) - \eta l_t(i) \quad \Longrightarrow \quad y_{t+1}(i) = w_t(i) e^{\eta l_t(i)}$. Then the projection, Step 2, is $w_{t+1} = \frac{y_{t+1}}{\sum_i y_{t+1}(i)}$. This is exactly the Hedge algorithm!

Next, we write the algorithm in terms of the update rule (5) that we obtained from the optimality conditions (zero gradient).

3 Online Mirror Descent (OMD)

Online Mirror Descent is the following algorithm:

For $t = 1, \dots, T$:

- Gradient update: if $t = 1$, set y_t such that $\nabla R(y_t) = 0$. If $t > 1$, choose y_t such that
$$\begin{aligned} \nabla R(y_t) &= \nabla R(w_{t-1}) - \eta l_{t-1} \quad \text{(active version)} \\ \nabla R(y_t) &= \nabla R(y_{t-1}) - \eta l_{t-1} \quad \text{(passive version)} \end{aligned} \tag{6}$$

- Projection (link): $w_t = \arg\min_{w \in \mathcal{K}} D_R(w, y_t)$

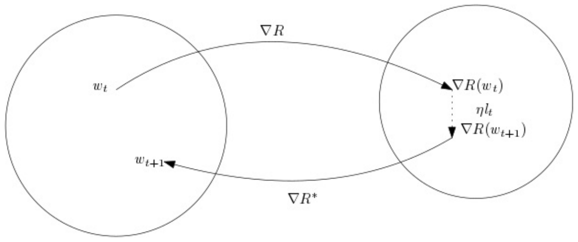

Figure 1: Descent in mirrored space

The active version is equivalent to online gradient descent, and the passive version to RFTL. This may be seen by setting derivatives to zero as we did above. If $\mathcal{K} = \mathbb{R}^n$, then passive and active are the same ($w_t = y_t$).

The updates (6) resemble gradient descent steps. Indeed, OMD may be seen as performing gradient descent steps in a dual space, and then going back to the original space via the projection (Step 2, also called a link function) that yields w_t. This is illustrated in Figure 1. If R is strictly convex, then there is a one-to-one mapping between gradients $\nabla R(y)$ and iterates y. The figure uses the language of conjugate functions. Indeed, the link / projection step 2 may be seen as a gradient mapping $\nabla R^* = (\nabla R)^{-1}$ using the conjugate dual function $R^*(u) = \sup_v (u^\top v - R(v))$.

Let us take the example when $R(x) = \frac{1}{2}||x||^2$. The active OMD gives

$$w_{t+1} = \arg\min_{w \in \mathcal{K}} ||w - (w_t - \eta l_t)||^2 = \arg\min_{w \in \mathcal{K}} \ \eta l_t^\top w + \tfrac{1}{2}\|w - w_t\|^2,$$

which is the online gradient descent.

The passive OMD collects the sum of all gradients in y_t: It uses the updates

$$\begin{aligned} &\nabla R(y_{t+1}) = \nabla R(y_t) - \eta l_t \\ \implies \ & y_{t+1} = y_t - \eta l_t = y_0 - \eta \sum_{\tau=1}^{t} l_\tau. \end{aligned}$$

Plugging this into the projection step 2, with $y_0 = 0$, we obtain

$$\begin{aligned} w_{t+1} &= \arg\min_{w \in \mathcal{K}} ||y_{t+1} - w||^2 = \arg\min_{w \in \mathcal{K}} \ \| - \eta \sum_{\tau=1}^{t} l_\tau - w\|^2 \\ &= \arg\min_{w \in \mathcal{K}} \sum_{\tau=1}^{t} \eta l_\tau^{\mathrm{T}} w + \frac{1}{2}||w||^2. \end{aligned}$$

This is the regularized FTL algorithm.

So, OMD captures both the RFTL and the Online gradient descent algorithms. In the previous section, we saw that it also captures multiplicative updates (the Hedge algorithm), when using the negative entropy for R. Hence, OMD provides a unified view on many of the existing online algorithms, which were derived originally as separate algorithms. Consequently, analyzing OMD yields an analysis of this entire family of algorithms.

3.1 Application to Combinatorial Structures

One of our initial motivations was to do online prediction of combinatorial structures (paths, subsets S of a certain size m, etc.). All of these structures can be viewed as subsets (e.g., a path is a set of edges in a graph). To apply OMD to this problem, we need a continuous vector w. Hence, we view $w \in \mathcal{K}$ as defining a distribution over structures, where we do a randomized prediction: we find a probability distribution p_w over structures such that $\mathbb{E}_{S \sim p_w}[1_S] = w$. The convex set $\mathcal{K}$ here is the convex hull of the indicator vectors of all feasible structures.

The expected loss (with linear loss functions) is then $\mathbb{E}[l^\top 1_S] = l^\top \mathbb{E}[1_S] = l^\top w$. This is useful since the same analysis as for continuous predictions w will hold, in expectation. We already did this for the experts problem, where we always pick exactly one expert (subset of size 1).

What do the OMD steps look like for combinatorial problems?

- Step a.
$$\nabla R(y_t) = \nabla R(w_{t-1}) - \eta l_{t-1} \qquad \text{(easy)}$$

- Step b.
$$w_t = \arg\min_{w \in \mathcal{K}} D_R(w, y_t)$$
  Step b. is a convex optimization over a convex hull $\mathcal{K}$. The difficulty of this problem depends on $\mathcal{K}$. In many cases, this can be solved e.g. with the Frank-Wolfe algorithm.

- Step c. Play randomized such that $\mathbb{E}[1_{S_t}] = w_t$.
  Finding the appropriate distribution p_w and then sampling from it may be hard to do sometimes.

It will turn out that using OMD with R to be the negaive entropy yields the optimal regret.

3.2 Regret Bounds for OMD

Theorem 1. *Let R be such that $D_R(x,y) \geq ||x-y||^2$ in some norm $||\cdot||$, and $||l_t||_* \leq G$ for all $t \geq 0$. Further, if $D_R(w,w_1) \leq D^2$ $\forall w \in \mathcal{K}$, then the regrent of OMD with $\eta = \dfrac{D}{2G\sqrt{T}}$ is bounded as $R_T \leq DG\sqrt{T}$.*

The proofs can be found in Chapter 2 of [1].

Example: Let us use the entropy function as R, and a scaled version of the probability simplex, $\mathcal{K} = \alpha\Delta = \{w \mid ||w||_1 = \alpha \text{ and } w \geq 0\}$. By Pinsker's inequality, we have $D_R(x,y) \geq \dfrac{1}{\alpha}||x-y||_1^2$ on $\alpha\Delta = \mathcal{K}$. Hence, our norm will be a scaled version of the ℓ_1-nomr. The dual norm of $\frac{1}{\sqrt{\alpha}}\|\cdot\|_1$ is $\sqrt{\alpha}\|\cdot\|_\infty$. So we can set the constant G to be $G = \sqrt{\alpha}\max_t ||l_t||_\infty$.

For the special case of $\mathcal{K}$ being the probability simplex Δ_d, we obtain the "diameter" constant $D^2 = \log d$.

Now consider the combinatorial setting, and assume for simplicity that we pick sets of cardinality m, i.e., $|S_t| \leq m$. The convex hull of the corresponding indicator vectors is $\mathcal{K} = m\Delta$ (setting $\alpha = m$ above). In this case, it can be shown that $D^2 = m\log(d/m)$ (the proof is in [6]). If for the loss functions we have that $l_t \in [0,1]^d$, then we can set $G = \sqrt{m}$ and obtain $R_T = m\sqrt{T\log(d/m)}$, which is optimal.

4 Bandit Feedback

So far, we assumed the *full information* setting, where in each step we observe the full loss function l_t after making a choice w_t (S_t in the discrete setting). In the *bandit setting*, in step t we only observe the loss evaluated at the particular point we picked, i.e., $l_t(w_t)$ (or $l_t(S_t)$ in the discrete setting): the feedback is reduced from an entire function to a scalar.

We run into two challenges:

- OMD assumes knowledge of the full loss to compute gradients; but we don't know the full loss function
- We need to trade off between exploring and exploiting our knowledge on what has been good so far

The basic idea to tackle these challenges is to use a perturbed version of w_t (exploring) and replace the gradient in the descent by a one-shot estimate $\hat{l}_t$ that matches the actual gradient in expectation.

4.1 Online stochastic mirror descent

If we have an estimate $\hat{l}_t$ of the gradient at step t, we can plug this into our OMD and proceed as before:

$w_1 = \arg\min_{w \in \mathcal{K}} \mathcal{R}(w)$
For $t = 1, \dots, T$

- play a random $w_t^{'} \sim p_t$ based on w_t
  - observe feedback
- compute random estimate $\widehat{l_t}$ of l_t
- use estimate as gradient:

$$\begin{aligned} \nabla R(y_{t+1}) &= \nabla R(w_t) - \eta \widehat{l_t} \\ w_{t+1} &= \arg\min_{w \in \mathcal{K}} D_R(w, y_{t+1}) \end{aligned}$$

The key question is how to obtain the gradient estimate. We will explore this with an example.

4.2 Example: The multi armed bandit problem

The multi-armed bandit problem is the bandit version of the experts problem: there are d arms (experts), and in each online round the learner chooses one of the arms, denoted by p_t. Then the learner only receives the cost of $l_t(p_t)$. We use OMD and plug in an estimate of the gradient since true gradient of l_t is unknown. In round t, we proceed as follows:

- play arm a_t based on the current weight vector w_t.
- estimated gradient:

$$\widehat{l_t}(i) = \begin{cases} l_t(i)/w_t(i) & \text{if } a_t = i \\ 0 & \text{otherwise.} \end{cases}$$

- Using $\hat{l}_t$ leads to the weight updates

$$y_{t+1}(i) = \begin{cases} w_t(a_t)e^{-\eta \widehat{l}_t(a_t)} & \text{if } i = a_t \\ w_t(i) & \text{otherwise} \end{cases}, \qquad w_{t+1} = y_{t+1} / \sum_{i=1}^{d} y_{t+1}(i) \tag{7}$$

Since the learner picks arm i with probability $w_t(i)$, we have that

$$\mathbb{E}[\widehat{l}_t(j)|\widehat{l}_1, \ldots, \widehat{l}_{t-1}] = w_t(j)\tfrac{l_t(j)}{w_t(j)} = l_t(j).$$

Therefore, $\widehat{l}_t$ is an unbiased estimator of l_t.

Let us show a regret bound for this method. First, by a similar proof as the one we saw in the last lecture for the full-information case, one can show that for any $u \in \mathcal{K}$:

$$R_T \leq \sum_{t=1}^{T} \langle w_t - u, l_t \rangle \leq \log(d)/\eta + \eta \sum_{t=1}^{T} \sum_i w_t(i) l_t^2(i).$$

This holds for any non-negative vector l_t (Theorem 2.22 in [1]). The part $\log(d)/\eta$ comes from the "diameter". Taking expectations on both sides we get (Theorem 4.1 in [1]),

$$\begin{aligned} \mathbb{E}\left[\sum_{t=1}^{T}(f_t(w_t) - f_t(u))\right] &\leq \log(d)/\eta + \eta \sum_{t=1}^{T} \mathbb{E}\left[\sum_i w_t(i) \hat{l}_t^2(i)\right] \\ \mathbb{E}\left[\sum_i w_t(i) l_t^2(i) \mid l_{t-1}, \ldots, l_1\right] &= \sum_{j=1}^{d} \mathbb{P}(a_t = j) \sum_{i=1}^{d} w_t(i) \tfrac{l_t^2(i)}{w_t(i)^2} \mathbf{1}[i = j] \\ &= \sum_i l_t^2(i) \\ &\leq d \end{aligned}$$

Thus the regret bound is $\log d/\eta + \eta dT$. Note that there is a bump up of d because we only observe $1/d$ of complete feedback. Plugging in $\eta = \sqrt{\log d / dT}$ (which minimizes the bound with respect to η) yields $\mathbb{E}[R_T] \leq \sqrt{dT \log d}$.

References

[1] S. Shalev-Shwartz. Online learning and online convex optimization. Foundations and Trends in ML, 2011.

[2] E. Hazan. Introduction to Online convex Optimization.

	Full Information	Semi-Bandit	Bandit
Lower Bound	$m\sqrt{T\log\frac{d}{m}}$	$\sqrt{mdT}$	$m\sqrt{dT}$
Upper Bound	$m\sqrt{T\log\frac{d}{m}}$	$\sqrt{mdT}$	$m^{3/2}\sqrt{dT\log\frac{d}{m}}$

$\mathcal{K} \subseteq \{0,1\}^d \qquad \|w\|_1 = m \quad \forall w \in \mathcal{K}$

Figure 2: Regret Bounds in different settings (adapted from [4])

[3] S. Bubeck, N. Cesa-Bianchi. Regret Analysis of stochastic and non-stochastic multi-armed bandit problems. Foundations and Trends in ML, 2012.

[4] J. Audibert, S. Bubeck, G. Lugosi. Regret in Online Combinatorial Optimization. Math. of OR 2014

[5] M. Zinkevich. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. ICML 2003.

[6] S. Bubeck. Introduction to Online Optimization (Lecture notes). 2011.