Notes for Lecture 11

1 Online Optimization

Online convex optimization deals with the following setup: we want to design an algorithm that, at each discrete time step $t = 1, 2, \ldots$, comes up with a solution $x_t \in K$, where K is a certain convex set of feasible solution. After the algorithm has selected its solution x_t, a convex cost function $f_t : K \to \mathbb{R}$, coming from a known restricted set of admissible cost functions $\mathcal{F}$, is revealed, and the algorithm pays the loss $f_t(x_t)$.

Again, the algorithm has to come up with a solution *without knowing what cost functions it is supposed to be optimizing.* Furthermore, we will think of the sequence of cost functions $f_1, f_2, \ldots, f_t, \ldots$ not as being fixed in advanced and unknown to the algorithm, but as being dynamically generated by an adversary, after seeing the solutions provided by the algorithm.

The *offline optimum* after T steps is the total cost that the best possible fixed solution would have incurred when evaluated against the cost functions seen by the algorithm, that is, it is a solution to

$$\min_{x \in K} \sum_{t=1}^{T} f_t(x)$$

The *regret* after T steps is the difference between the loss suffered by the algorithm and the offline optimum, that is,

$$\text{Regret}_T = \sum_{t=1}^{T} f_t(x_t) - \min_{x \in K} \sum_{t=1}^{T} f_t(x)$$

The remarkable results that we will review give algorithms that achieve regret

$$\text{Regret}_T \leq O_{K,\mathcal{F}}(\sqrt{T})$$

that is, for fixed K and $\mathcal{F}$, the regret-per-time-step goes to zero with the number of steps, as $O\left(\frac{1}{\sqrt{T}}\right)$. It is intuitive that our bounds will have to depend on how big is the "diameter" of K and how large is the "magnitude" and "smoothness" of the functions $f \in \mathcal{F}$, but depending on how we choose to formalize these quantities we will be led to define different algorithms.

2 Multiplicative Weights

The *multiplicative weights* or *hedge* algorithm is the most well known and most frequently rediscovered algorithm in online optimization.

The problem it solves is usually described in the following language: we want to design an algorithm that makes the best possible use of the advice coming from n self-described experts. At each time step $t = 1, 2, \ldots$, the algorithm has to decide with what probability

to follow the advice of each of the experts, that is, the algorithm has to come up with a probability distribution $x_t = (x_t(1), \ldots, x_t(n))$ where $x_t(i) \geq 0$ and $\sum_{i=1}^n x_t(i) = 1$. After the algorithm makes this choice, it is revealed that following the advice of expert i at time t leads to loss $\ell_t(i)$, so that the expected loss of the algorithm at time t is $\sum_{i=1}^n x_t(i)\ell_t(i)$. A loss can be negative, in which case its absolute value can be interpreted as a profit.

After T steps, the algorithm "regrets" that it did not just always follow the advice of the expert that, with hindsight, was the best one, so that the regret of the algorithm after T steps is

$$\text{Regret}_T = \left(\sum_{t=1}^T \sum_{i=1}^n x_t(i)\ell_t(i)\right) - \left(\min_{i=1,\ldots,n} \sum_{t=1}^T \ell_t(i)\right)$$

This corresponds to the instantiation of the framework we described in the previous post to the special case in which the set of feasible solutions K is the set $\Delta \subseteq \mathbb{R}^n$ of probability distributions over the sample space $\{1, \ldots, n\}$ and in which the loss functions $f_t(x)$ are linear functions of the form $f_t(x) = \sum_i x(i)\ell_t(i)$. In order to bound the regret, we also have to bound the "magnitude" of the loss functions, so in the following we will assume that for all t and all i we have $|\ell_t(i)| \leq 1$, and otherwise we can scale everything by a known upper bound on $\max_{t,i} |\ell_t|$.

We now describe the algorithm.

The algorithm maintains at each step t a vector of *weights* $w_t = (w_t(1), \ldots, w_t(n))$ which is initialized as $w_1 := (1, \ldots, 1)$. The algorithm performs the following operations at time t:

- $w_t(i) := w_{t-1}(i) \cdot e^{-\beta \ell_{t-1}(i)}$
- $x_t(i) := \dfrac{w_t(i)}{\sum_{j=1}^n w_t(j)}$

That is, the weight of expert i at time t is $e^{-\beta \sum_{k=1}^{t-1} \ell_k(i)}$, and the probability $x_t(i)$ of following the advice of expert i at time t is proportional to the weight. The parameter $\beta > 0$ is hardwired into the algorithm and we will optimize it later. Note that the algorithm gives higher weight to experts that produced small losses (or negative losses of large absolute value) in the past, and thus puts higher probability on such experts.

We will prove the following bound.

THEOREM 1
Assuming that for all t and i we have $|\ell_t(i)| \leq 1$, for every $0 < \beta < 1/2$, after T steps the multiplicative weight algorithm experiences a regret that is always bounded as

$$\text{Regret}_T \leq \beta \sum_{t=1}^T \sum_{i=1}^n x_t(i)\ell_t^2(i) + \frac{\ln n}{\beta} \leq \beta T + \frac{\ln n}{\beta}$$

In particular, if $T > 4 \ln n$, by setting $\beta = \sqrt{\frac{\ln n}{T}}$ we achieve a regret bound

$$\text{Regret}_T \leq 2\sqrt{T \ln n}$$

We will start by giving a short proof of the above theorem.

For each time step t, define the quantity

$$W_t := \sum_{i=1}^{n} w_t(i) \ .$$

We want to prove that, roughly speaking, the only way for an adversary to make the algorithm incur a large loss is to produce a sequence of loss functions such that *even the best expert incurs a large loss.* The proof will work by showing that if the algorithm incurs a large loss after T steps, then W_{T+1} is small, and that if W_{T+1} is small, then even the best expert incurs a large loss.

Let us define

$$L^* = \min_{i=1,\ldots,n} \sum_{t=1}^{T} \ell_t(i)$$

to be the loss of the best expert. Then we have

LEMMA 2 (IF W_{T+1} IS SMALL, THEN L^* IS LARGE)

$$W_{T+1} \geq e^{-\beta L^*}$$

PROOF: Let j be an index such that $L^* = \sum_{t=1}^{T} \ell_t(j)$. Then we have

$$W_{T+1} = \sum_{i=1}^{n} e^{-\beta \sum_{t=1}^{T} \ell_t(i)} \geq e^{-\beta \sum_{t=1}^{T} \ell_t(j)} = e^{-\beta L^*}$$

□

LEMMA 3 (IF THE LOSS OF THE ALGORITHM IS LARGE THEN W_{T+1} IS SMALL)

$$W_{T+1} \leq n \prod_{t=1}^{n} (1 - \beta \langle x_t, \ell_t \rangle + \beta^2 \langle x_t, \ell_t^2 \rangle)$$

where ℓ_t^2 is the vector whose i-th coordinate is $(\ell_t(i))^2$

PROOF: Since we know that $W_1 = n$, it is enough to prove that, for every $t = 1, \ldots, T$, we have

$$W_{t+1} \leq (1 - \beta \langle x_t, \ell_t \rangle + \beta^2 \langle x_t, \ell_t^2 \rangle) \cdot W_t \tag{1}$$

And we see that

$$\frac{W_{t+1}}{W_t} = \sum_{i=1}^{n} \frac{w_{t+1}(i)}{W_t}$$

$$= \sum_{i=1}^{n} \frac{w_t(i) \cdot e^{-\beta \ell_t(i)}}{W_t}$$

$$= \sum_{i=1}^{n} x_t(i) \cdot e^{-\beta \ell_t(i)}$$

$$\leq \sum_{i=1}^{n} x_t(i) \cdot (1 - \beta \ell_t(i) + \beta^2 \ell_t^2(i))$$
$$= 1 - \beta \langle x_t, \ell_t \rangle + \beta^2 \langle \ell_t^2, x_t \rangle$$

where we used the definitions of our quantities and the fact that $e^{-z} \leq 1 - z + z^2$ for $|z| \leq 1/2$. □

Using the fact that $1 - z \leq e^{-z}$ for all $|z| \leq 1$, the above lemmas can be restated as

$$\ln W_{T+1} \leq \ln n - \left(\sum_{t=1}^{T} \beta \langle \ell_t, x_t \rangle \right) + \left(\sum_{t=1}^{T} \beta^2 \langle \ell_t^2 x_t \rangle \right)$$

and

$$\ln W_{T+1} \geq -\beta L^*$$

which together imply

$$\left(\sum_{t=1}^{T} \langle \ell_t, x_t \rangle \right) - L^* \leq \frac{\ln n}{\beta} + \beta \sum_{t=1}^{T} \langle \ell_t^2, x_t \rangle$$

as desired.

Personally, I find all of the above very unsatisfactory, because both the algorithm and the analysis, but especially the analysis, seem to come out of nowhere. In fact, I never felt that I actually understood this analysis until I saw it presented as a special case of the *Follow The Regularized Leader* framework that we will discuss in the next lecture. (We will actually prove a slightly weaker bound, but with a much more satisfying proof.)

Here is, however, a story of how a statistical physicist might have invented the algorithm and might have come up with the analysis. Let's call the loss caused by expert i after $t-1$ steps the *energy* of expert i at time t:

$$E_t(i) = \sum_{k=1}^{t-1} \ell_k(i)$$

Note that we have defined it in such a way that the algorithm knows $E_t(i)$ at time t. Our offline optimum is the energy of the lowest energy expert at time $T+1$, that, is, the energy of the *ground state* at time $T+1$. When we have a collection of numbers $E_t(1), \ldots, E_t(n)$, a nice lower bound to their minimum is

$$\min_i E_t(i) \geq -\frac{1}{\beta} \ln \sum_{i=1}^{n} e^{-\beta E_t(i)}$$

which is true for every $\beta > 0$. The right-hand side above is the *free energy* at temperature $\frac{1}{\beta}$ at time t. This seems like the kind of expression that we could use to bound the offline optimum, so let's give it a name

$$\Phi_t := -\frac{1}{\beta} \ln \sum_{i=1}^{n} e^{-\beta E_t(i)}$$

In terms of coming up with an algorithm, all that we have got to work with at time t are the losses of the experts at times $1, \ldots, t-1$. If the adversary chooses to make one of the experts consistently much better than the others, it is clear that, in order to get any reasonable regret bound, the algorithm will have to put much of the probability mass in most of the steps on that expert. This suggests that the x_t should put higher probability on experts that have done well in the first $t-1$ steps, that is x_t should put higher probability on "lower-energy" experts. When we have a system in which, at time t, state i has energy $E_t(i)$, a standard distribution that puts higher probability on lower energy states is the *Gibbs distribution* at temperature $1/\beta$, defined as

$$x_t(i) = \frac{e^{-\beta E_t(i)}}{\sum_j e^{-\beta E_t(j)}}$$

where the denominator above is also called the *partition function* at time t

$$Z_t := \sum_{j=1}^{n} e^{-\beta E_t(j)}$$

So far we have "rediscovered" our multiplicative weights algorithm, and the quantity W_t that we had in our analysis gets interpreted as the partition function Z_t. The fact that Φ_{T+1} bounds the offline optimum suggests that we should use Φ_t as a potential function, and aim for an analysis involving a telescoping sum. Indeed some manipulations (the same as in the short proof above, but which are now more mechanical) give that the loss of the algorithm at time t is

$$\langle x_t, \ell_t \rangle \leq \Phi_{t+1} - \Phi_t + \langle x_t, \ell_t^2 \rangle$$

which telescopes to give

$$\sum_{t=1}^{T} \langle x_t, \ell_t \rangle \leq \Phi_{T+1} - \Phi_1 + \sum_{t=1}^{T} \langle x_t, \ell_t^2 \rangle$$

Recalling that

$$\Phi_1 = -\frac{1}{\beta} \ln n$$

and

$$\Phi_{T+1} \leq \min_{j=1,\ldots,n} \sum_{t=1}^{T} \ell_t(j)$$

we have again

$$\left(\sum_{t=1}^{T} \langle x_t, \ell_t \rangle \right) - \left(\min_{j=1,\ldots,n} \sum_{t=1}^{T} \ell_t(j) \right) \leq \frac{\ln n}{\beta} + \sum_{t=1}^{T} \langle x_t, \ell_t^2 \rangle$$