CSE5401 Lecture 5

Multiplicative Weights Method

1. No-regret on-line prediction
2. The Multiplicative Weights Method
3. Application: fast approximate linear programming

Today's algorithm has been re-discovered independently several times under different names in Economics, Machine Learning, and Optimization. We will see a couple different applications of it in this course. It is most naturally posed as solving the following on-line prediction problem: we are playing a game that proceeds in rounds, numbered $t=1,2,\dots,T$. On each round, we must allocate our budget in any way we wish across m options: we model this as a probability distribution $\vec{p} = (p_1,\dots,p_m) \in [0,1]^m$, $\sum_{i=1}^{m} p_i = 1$. We are then informed how much each option gains or loses in the round, $\vec{r}^{(t)} \in [-1,1]^m$, and our "score" adds $\langle \vec{p}^{(t)}, \vec{r}^{(t)} \rangle = \sum_{i=1}^{m} p_i^{(t)} r_i^{(t)}$ for the round. We make no assumption about how $\vec{r}^{(t)}$ is selected, and we will evaluate our algorithm's performance for an adversarially chosen sequence $\vec{r}^{(1)}, \vec{r}^{(2)}, \dots$ There is no way to guarantee the alg's score is greater than -1 (if $\vec{r}^{(t)} = (-1,\dots,-1)$ for all t), but in this case no algorithm can do well.

We evaluate the quality of the algorithm by comparing its score to the best single fixed option ("in hindsight"): $\max_i \sum_{t=1}^{T} r_i^{(t)}$. We wish to bound the regret: $\max_i \sum_{t=1}^{T} r_i^{(t)} - \langle \vec{p}^{(t)}, \vec{r}^{(t)} \rangle$. If we can obtain a bound that is $o(T)$, then over a

sufficiently large # of rounds our average score will be close to the average score of the best fixed option. This is called a "no-regret" algorithm. One of the simplest No-regret algorithms is the multiplicative weights algorithm. Given a parameter $\varepsilon \le 1/2$, the algorithm is:

Initialize $\vec{w}^{(1)} = (\overbrace{1,\dots,1}^{m})$.

For each round $t = 1, \dots, T$

Put $\vec{p}^{(t)} \leftarrow \frac{1}{\sum \vec{w}^{(t)}} \cdot \vec{w}^{(t)}$; play $\vec{p}^{(t)}$, receive $\vec{r}^{(t)}$.

Put $w_i^{(t+1)} \leftarrow \begin{cases} w_i^{(t)} \cdot (1-\varepsilon)^{-r_i^{(t)}} & \text{if } r_i^{(t)} < 0 \\ w_i^{(t)} \cdot (1+\varepsilon)^{r_i^{(t)}} & \text{if } r_i^{(t)} \ge 0 \end{cases}$

Intuitively, because the rewards are bounded, any option with a high average score (or low score) must be positive (resp. negative) over many rounds. Thus, the multiplicative weights algorithm has enough time to respond by shifting its weight to capture most of the gains/avoid the losses. Formally, we can show it is a no-regret algorithm:

Theorem After T rounds, for every option i:

$$\sum_{t=1}^{T} \langle \vec{p}^{(t)}, \vec{r}^{(t)} \rangle \ge \sum_{t=1}^{T} r_i^{(t)} - \varepsilon \sum_{t=1}^{T} |r_i^{(t)}| - \frac{\ln m}{\varepsilon}.$$

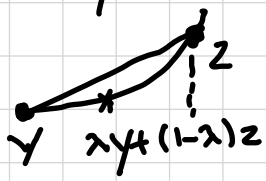
In particular, for $\varepsilon = \sqrt{\frac{\ln m}{T}}$, the regret is $\le 2\sqrt{T \ln m}$

For the analysis, we'll use the following bounds on the exponential

Lemma
1. $(1-\varepsilon)^{-x} \le 1 + \varepsilon x$ for $x \in [-1, 0]$
2. $(1+\varepsilon)^{x} \le 1 + \varepsilon x$ for $x \in [0, 1]$
3. $1 + x \le e^x$ for all x
4. $\ln(1+x) \ge x - x^2$ for $x \in [0, 1]$
5. $\ln\left(\frac{1}{1-x}\right) \le x + x^2$ for $x \in [0, 1/2]$

① + ② follow from the convexity of e^x: for any $\lambda \in [0,1]$

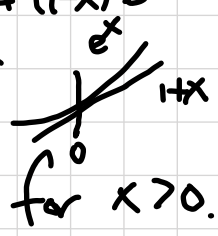
$e^{\lambda y + (1-\lambda)z} \le \lambda e^y + (1-\lambda)e^z$ for all $y \neq z$



For ① we consider $\lambda = -x$, $y = \ln(1-\varepsilon)$, $z = \ln 1$.

Then: $(1-\varepsilon)^{-x} = e^{-x\ln(1-\varepsilon) + (1+x)\ln 1} \le -x\, e^{\ln(1-\varepsilon)} + (1+x)\, e^{\ln 1} = 1 + \varepsilon x$

For ②, $\lambda = x$, $y = \ln(1+\varepsilon)$, $z = \ln 1$: $(1+\varepsilon)^x = e^{x\ln(1+\varepsilon) + (1-x)\ln 1} \le x\, e^{\ln(1+\varepsilon)} + (1-x)\, e^{\ln 1} = 1 + \varepsilon x$



For ③, $1 + x = e^x$ at $x = 0$, $\frac{d}{dx} e^x = e^x$ and $\frac{d}{dx}(1+x) = 1$.

e^x is monotone so $\frac{d}{dx} e^x < \frac{d}{dx}(1+x)$ for $x < 0$, $\frac{d}{dx} e^x > \frac{d}{dx}(1+x)$ for $x > 0$.

④ & ⑤ are Taylor's theorem, recalling $\frac{d}{dx}\ln(1+x) = \frac{1}{x}$ (so the const. for the remainder can be bounded by 1.)

Proof of Theorem

We examine a potential function $\Phi^{(t)} = \sum_{i=1}^m w_i^{(t)}$.

By def. $\Phi^{(t+1)} = \sum_{i=1}^m w_i^{(t+1)} = \sum_{i: r_i^{(t)} < 0} w_i^{(t)} (1-\varepsilon)^{-r_i^{(t)}} + \sum_{i: r_i^{(t)} \ge 0} w_i^{(t)} (1+\varepsilon)^{r_i^{(t)}}$

$\le \sum_{i=1}^m w_i^{(t)} (1 + \varepsilon r_i^{(t)})$ by ① & ②.

$= \Phi^{(t)} \left(1 + \varepsilon \sum_{i=1}^m \vec{p}_i^{(t)} \vec{r}_i^{(t)}\right)$

$\le \Phi^{(t)} e^{\varepsilon \langle \vec{p}^{(t)}, \vec{r}^{(t)} \rangle}$ by ③.

Therefore $\Phi^{(T)} \le \underbrace{\Phi^{(1)}}_{m} \cdot e^{\sum_{t=1}^T \varepsilon \langle \vec{p}^{(t)}, \vec{r}^{(t)} \rangle}$.

How large is $w_i^{(T)}$ for each i? $w_i^{(T)} = \prod_{t: r_i^{(t)} < 0} (1-\varepsilon)^{-r_i^{(t)}} \prod_{t: r_i^{(t)} \ge 0} (1+\varepsilon)^{r_i^{(t)}}$

and $w_i^{(T)} \le \Phi^{(T)}$ since all are nonnegative. Taking log's,

$\ln w_i^{(T)} = \sum_{t: r_i^{(t)} < 0} -r_i^{(t)} \ln(1-\varepsilon) + \sum_{t: r_i^{(t)} \ge 0} r_i^{(t)} \ln(1+\varepsilon) \overset{④+⑤}{\ge} \sum_{t=1}^T r_i^{(t)} \cdot \varepsilon - |r_i^{(t)}| \varepsilon^2$

So $\cdot \ln m + \varepsilon \sum_{t=1}^T \langle \vec{p}^{(t)}, \vec{r}^{(t)} \rangle \ge \ln \Phi^{(t)} \ge \ln w_i^{(T)} \ge \varepsilon \left(\sum_{t=1}^T r_i^{(t)} - \varepsilon |r_i^{(t)}| \right)$ ▨

Application to Linear Programming

Linear Programming is the following basic optimization problem:

Given a system of linear inequalities

$\langle \vec{a}^{(1)}, \vec{x} \rangle \ge b^{(1)}$

$\vdots$

$\langle \vec{a}^{(m)}, \vec{x} \rangle \ge b^{(m)}$

} We usually write $A\vec{x} \ge \vec{b}$

where $A \in \mathbb{R}^{m \times n}$, $\vec{x} \in \mathbb{R}^n$, $\vec{b} \in \mathbb{R}^m$

find a feasible solution $\vec{x}$ or report "infeasible".

Often we also include an objective vector $\vec{c} \in \mathbb{R}^n$, and ask for an $\vec{x}$ minimizing $\langle \vec{c}, \vec{x} \rangle$ s.t. $A\vec{x} \geq \vec{b}$, but by finding the largest C s.t. $\langle -\vec{c}, \vec{x} \rangle \geq C$ & $A\vec{x} \geq \vec{b}$ we can reduce such optimization to the above "feasibility" problem. One way to detect infeasibility is to find a nonnegative vector $\vec{p}$ such that $\vec{p}^T A\vec{x} < \vec{p}^T\vec{b}$ $\forall \vec{x}$ since if $\langle \vec{a}^{(i)}, \vec{x} \rangle \geq b^{(i)}$ for all i, $\vec{p}^T A\vec{x} = \sum_{i=1}^{m} p_i \langle \vec{a}^{(i)}, \vec{x} \rangle \geq \sum_{i=1}^{m} p_i b^{(i)} = \vec{p}^T\vec{b}$.

Here, we often are interested in a constrained set of $\vec{x}$'s, e.g., bounded ($\|\vec{x}\|_1$ or $\|\vec{x}\|_2$ or $\|\vec{x}\|_\infty \leq B$) or probability distns ($\|\vec{x}\|_1 = 1$ & $x_i \geq 0$ for all i). In each case, we can maximize $\vec{p}^T A\vec{x}$ efficiently:

$\|\vec{x}\|_1 \leq \beta$: for the index $i^* = \arg\max_i |\vec{p}^T A|_i$, we set $x_{i^*} = \begin{cases} \beta & \text{if } (\vec{p}^T A)_{i^*} \geq 0 \\ -\beta & \text{o.w.} \end{cases}$, $\vec{x}_{-i^*} = \vec{0}$.

$\|\vec{x}\|_2 \leq \beta$? $\vec{x} = \beta \frac{A^T\vec{p}}{\|\vec{p}^T A\|_2}$

$\|\vec{x}\|_\infty \leq \beta$? $x_i = \beta$ if $(\vec{p}^T A)_i \geq 0$, $x_i = -\beta$ o.w.

prob. dist'n?, for $i^* = \arg\max_i (\vec{p}^T A)_i$, set $x_{i^*} = 1$, $\vec{x}_{-i^*} = \vec{0}$.

In each case, if we obtain $\vec{x}$ s.t. $\vec{p}^T A\vec{x} < \vec{p}^T\vec{b}$, we know the linear system doesn't have a feasible solution in the corresp. set.

Notice also that the solutions can be computed fast in parallel.

It will actually be important for us to find $\vec{x}$ that minimize the "width": $\max_i |\langle \vec{a}^{(i)}, \vec{x} \rangle - b^{(i)}|$. In the case of prob. distns, our simple solution has width at most $\max_{i,j} |a^{(i)}_j| + \max_i |b^{(i)}|$.

By being a little more careful, we can get similar bounds for the others. Here's the guarantee:

Theorem Suppose that for A & $\vec{b}$ our single-inequality algorithm can always find a solution of width $\leq \gamma$. Let $\delta < \gamma$ be given. There

is an algorithm that makes $O\left(\frac{r^2 \log m}{\delta^2}\right)$ calls to the single-inequality alg. and takes an additional $O(m)$ steps per call, and returns $\vec{x}$ such that for all i $\langle \vec{a}^{(i)}, \vec{x} \rangle \geq b^{(i)} - \delta$.

So, $\vec{x}$ is "almost" feasible. In contrast to the usual algorithms for linear programming (which find true feasible solutions), the running time here scales with the value of the error in feasibility and the value of the width, which in the examples above, could be similar to the values of the input's entries. So, it is only pseudopolynomial time, in addition to being approximate.

What is good about this algorithm is that the dimension (n) only appears in the complexity of the algorithm for solving the single-inequality feasibility problem, which we noted can be solved fast in parallel, so the parallel time overall only scales, e.g., logarithmically with n.

Proof of theorem: Suppose we use the Multiplicative Weights algorithm as follows: we let the options correspond to constraints, and given a proposed allocation $\vec{p}^{(t)}$, use the single-inequality algorithm to find $\vec{x}^{(t)}$ s.t. $\vec{p}^{(t)T} A \vec{x}^{(t)} \geq \vec{p}^{(t)T} \vec{b}$. We then put $r_i^{(t)} = \frac{b^{(i)} - \langle \vec{a}^{(i)}, \vec{x}^{(t)} \rangle}{\rho}$.

Multiplicative weights indeed takes $O(m)$ time per round, and since the width is $\leq \rho$, $r_i^{(t)} \in [-1, 1]$ for all i and t. By our earlier analysis, for $T = \frac{\rho^2 \ln m}{\delta^2}$, we get

$\frac{1}{T}\sum_{t=1}^{T} \langle \vec{p}^{(t)}, \vec{r}^{(t)} \rangle \geq \frac{1}{T}\sum_{t=1}^{T} r_i^{(t)} - \sqrt{\frac{\log m}{T}}$, for all i, where

$0 \geq \frac{\vec{p}^{(t)T}\vec{b} - \vec{p}^{(t)T} A \vec{x}^{(t)}}{\rho}$ $\qquad$ $\frac{b^{(i)} - \langle \vec{a}^{(i)}, \vec{x}^{(t)} \rangle}{\rho}$ $\qquad$ $\frac{\delta}{\rho}$

so we find $\vec{x}^* = \frac{1}{T}\sum_{t=1}^{T} \vec{x}^{(t)}$ satisfies $0 \geq b^{(i)} - \langle \vec{a}^{(i)}, \underbrace{\frac{1}{T}\sum_{t=1}^{T} \vec{x}^{(t)}}_{\vec{x}^*} \rangle - \delta$ for all i. ▨

Back to the single-inequality problem, rather than using the

solution that maximizes the inner product, we can use the minimum rescaling that satisfies the inequality. For example, in the ∞-norm solution, we chose $\vec{x}$ s.t. $\vec{p}^T A\vec{x} = \beta \|\vec{p}^T A\|_1$, but we only needed $\vec{p}^T A\vec{x} = \vec{p}^T\vec{b}$. So, we could have instead used the solution equal to $\pm \frac{\vec{p}^T\vec{b}}{\|\vec{p}^T A\|_1}$.

The width of this solution is then, recalling $\langle \vec{x}, \vec{y}\rangle \le \|\vec{x}\|_\infty \|\vec{y}\|_1$, $\vec{p}^T\vec{b} \le \|\vec{b}\|_\infty$, + $\max_i |\langle \vec{a}^{(i)}, \vec{x}\rangle - b^{(i)}| \le \left(1 + \max_i \frac{\|\vec{a}^{(i)}\|_1}{\|\vec{p}^T A\|_1}\right)\|\vec{b}\|_\infty$. Similarly, for the Euclidean norm, rather than $\vec{p}^T A\vec{x} = \beta$, we could have put $\vec{x} = \frac{A^T\vec{p}}{\|\vec{p}^T A\|_2^2}\cdot \vec{p}^T\vec{b}$, thus getting width $\max_i |\langle \vec{a}^{(i)}, \vec{x}\rangle - b^{(i)}| \le \left(1 + \max_i \frac{\|\vec{a}^{(i)}\|_2}{\|\vec{p}^T A\|_2}\right)\|\vec{b}\|_\infty$

In general, the smallest singular value of A gives a lower bound on $\|p^T A\|_2$. This might not be great — it could be 0. The technique here is best suited to the 1-norm.