CSE 632 (Fall 2019): Analysis of Algorithms II : Randomized Algorithms

Lecture 21 (11/6/2019): Multiplicative Weight Update

Lecturer: Shi Li *Scriber: Chen Xu, Hao Shi*

1 Main Framework

We have n experts making predictions about if a stock index is going up or down each day (Let us use 0/1 for down/up). At day t, our algorithm make a prediction after seeing the predictions of the n experts about day 1 to t and the actual trend of the stock index from day 1 to day $t-1$. After the the experts and our algorithm made the predictions, the actual outcome for day t is then revealed. That is, the order of steps in the framework is as follows:

```
for i ← 1 to T do
    n experts make predictions about the outcome for day t
    algorithm makes a prediction
    outcome for day t is revealed
```

Let m^* be the number of mistakes made by the best expert, m be the number of mistakes made by the algorithm after T days. The goal is to make m not too big compared to m^*.

Simpler case Consider there is one expert that never make mistakes, namely, $m^* = 0$. We have an algorithm for this simpler model. We main a set S of experts who never made a mistake and everyday we use the majority prediction of over all experts in S. When the algorithm makes a mistake, the $|S|$ will decrease by a factor of $\frac{1}{2}$. The number of mistakes is $m \leq \log n$.

General case Now we consider the more general case where there is no perfect expert. We have a weighted majority algorithm:

```
Initialize w_1 = w_2 = ... = w_n = 1
for t ← 1 to T do
    if total weight of experts predicting 0 ≥ total weight of experts predicting 1 then
        predict 0
    else
        predict 1
    for every expert i who made a mistake w_i ← w_i/2.
```

Now we analyze the algorithm. Let w_i^t be the weight of expert i after time t. Let $\Phi^t = \sum_{i=1}^n w_i^t$. Whenever we made a mistake in day t, we then have $\Phi^t \leq \frac{3}{4}\Phi^{t-1}$ because we reduced the minority weight (which accounts for at most 1/2 fraction of the total weight) by a factor of 1/2. Moreover, we have $\Phi^T \geq \frac{1}{2^{m^*}}$ because this is the weight of the best expert at step T. Thus we have

$$m \leq \log_{\frac{4}{3}} \frac{n}{1/2^{m^*}} = (\log_{\frac{4}{3}} 2)(\log n + m^*) \leq 2.4(\log n + m^*).$$

Now assume that the multiplicative constant we choose is $(1-\epsilon)$ instead of $\frac{1}{2}$. We have

$$m \leq \log_{\frac{1}{1-\epsilon/2}} \frac{n}{(1-\epsilon)^{m^*}} = \left(\log_{\frac{1}{1-\epsilon/2}} 2\right) \log n + \left(\log_{\frac{1}{1-\epsilon/2}} \frac{1}{1-\epsilon}\right) m^*.$$

Notice that

$$\log_{\frac{1}{1-\epsilon/2}} 2 = \frac{1}{\log_2 \frac{1}{1-\epsilon/2}} = \frac{1}{(\log_2 e) \ln (\frac{1}{1-\epsilon/2})} = \Theta\left(\frac{1}{\epsilon}\right).$$

and

$$\log_{\frac{1}{1-\epsilon/2}} \frac{1}{1-\epsilon} = \frac{\ln \frac{1}{1-\epsilon}}{\ln \frac{1}{1-\epsilon/2}} \leq \frac{\epsilon + O(\epsilon^2)}{\epsilon/2} \leq 2 + O(\epsilon).$$

So

$$m \leq O\left(\frac{\log n}{\epsilon}\right) + (2 + O(\epsilon))m^*.$$

The following example shows that a multiplicative factor of 2 is the best we can do for deterministic algorithms. Suppose we we have only $n = 2$ experts and the predictions of them are always different. The first day we take prediction from expert 1, who makes a mistake. The weight of expert 1 decreases therefore the second day we will pick expert 2. Then expert 2 is wrong in day 2. The next day we will pick expert 1 again, who makes a mistake. This keeps going on, and expert makes $T/2$ mistakes but our algorithm makes T mistakes.

2 Randomized Multiplicative Weight Update Algorithm

The multiplicative factor of 2 can be improved via randomized algorithms. Instead of using the majority vote, we make our prediction randomly, so that the probability that we make the majority (resp. minority) prediction is exactly the total relative weight of majority experts. The algorithm is given as follows:

1: initialize $w_1 = w_2 = ... = w_n = 1$
2: **for** $t \leftarrow 1$ to T **do**
3: randomly choose an expert $i \in [n]$, such that i is chosen with probability $\frac{w_i}{\Phi}$, where $\Phi = \sum_{i'=1}^{n} w_{i'}$.
4: follow the expert i
5: for every expert i' who made a mistake $w_{i'} \leftarrow e^{-\epsilon} w_{i'}$

Before we analyze the above algorithm, we now make some modifications and at the same time make the framework slightly more general. Firstly, instead of obtaining a random expert i in Step 3, we simply define a distribution for all the experts, without giving an actual realization of i. This will be sufficient for the analysis and all the applications we will see. Secondly, the prediction of an expert does not need to be 0/1; it can be anything. If we follow the expert i in day t, then whether we make mistake in day t is the same as whether i makes a mistake in day t. Thus, in this new framework, the predictions do not matter any more. Finally, we can generalize the framework so that the number of mistakes an expert makes at any time t is not necessarily 0 or 1, but any *real number* in $[-\rho, \rho]$. So, instead of using the number of mistakes, we use the word "penalty". At any time, the penality for expert i is between $-\rho$ and ρ. When the penality is negative, we think of it as a "reward".

With this new framework, we can now reformulate our algorithm as follows:

Algorithm 1 Multiplicative Weight Update Method

1: initialize $w_1 = w_2 = ... = w_n = 1$
2: **for** $t \leftarrow 1$ to T **do**
3: follow the distribution $\frac{w}{\Phi}$, where $w = (w_1, w_2, \cdots, w_n)$ and $\Phi = \sum_{i=1}^{n} w_i$
4: the penalty vector $m = (m_1, m_2, \cdots, m_n)^{\mathrm{T}} \in [-\rho, \rho]^n$ is revealed: expert i pays penalty m_i
5: our algorithm pays the penalty $\frac{wm}{\Phi}$
6: **for** every expert i **do** $w_i \leftarrow w_i e^{-\epsilon m_i/\rho^2}$

We now analyze the algorithm, for notional convenience, we put superscript t everywhere to indicate the time step of the variables: m_i^t is the penalty of expert i at time t, $m^t = (m_1^t, m_2^t, \cdots, m_n^t)^{\mathrm{T}}$, w_i^t is the weight of expert i at the end of time t, $w^t = (w_1^t, w_2^t, \cdots, w_n^t)$, and $\Phi^t = \sum_{i=1}^{n} w_i^t$ is total weight of all experts at the end of time t. The theorem we shall try to prove is

Theorem 1. *Assume $\epsilon \in (0, \rho)$ and $T \geq \frac{\rho^2 \log n}{\epsilon^2}$. Then at the end of Algorithm 1, we have for every expert i,*

$$\frac{1}{T}\sum_{t=1}^{T}\frac{w^{t-1}m^t}{\Phi^{t-1}} \leq 2\epsilon + \frac{1}{T}\sum_{t=1}^{T} m_i^t.$$

Proof. For every $t \in [T]$, we have

$$\begin{aligned}
\Phi^t &= \sum_{i=1}^{n} w_i^t = \sum_{i=1}^{n} w_i^{t-1} e^{-\epsilon m_i^t/\rho^2} \leq \sum_{i=1}^{n} w_i^{t-1}\left(1 - \epsilon m_i^t/\rho^2 + \frac{\epsilon^2}{\rho^2}\right) \qquad \text{used that } |m_i^t| \leq \rho, \epsilon \leq \rho \\
&= \left(1 + \frac{\epsilon^2}{\rho^2}\right)\sum_{i=1}^{n} w_i^{t-1} - \frac{\epsilon}{\rho^2}\sum_{i=1}^{n} w_i^{t-1} m_i^t \\
&= \left(1 + \frac{\epsilon^2}{\rho^2}\right)\Phi^{t-1} - \frac{\epsilon}{\rho^2}\sum_{i=1}^{n}\Phi^{t-1}\frac{w_i^{t-1}}{\Phi^{t-1}} m_i^t \\
&= \left(1 + \frac{\epsilon^2}{\rho^2}\right)\Phi^{t-1} - \frac{\epsilon}{\rho^2}\Phi^{t-1}\frac{w^{t-1}m^t}{\Phi^{t-1}} \\
&= \left(1 + \frac{\epsilon^2}{\rho^2} - \frac{\epsilon}{\rho^2}\frac{w^{t-1}m^t}{\Phi^{t-1}}\right)\Phi^{t-1}.
\end{aligned}$$

Notice that $\frac{w^{t-1}m^t}{\Phi^{t-1}}$ is the penalty paid by our algorithm at time t. Take the logarithm on both sides we have:

$$\ln \Phi^t \leq \ln\left(1 + \frac{\epsilon^2}{\rho^2} - \frac{\epsilon}{\rho^2}\frac{w^{t-1}m^t}{\Phi^{t-1}}\right) + \ln \Phi^{t-1} \leq \frac{\epsilon^2}{\rho^2} - \frac{\epsilon}{\rho^2}\frac{w^{t-1}m^t}{\Phi^{t-1}} + \ln \Phi^{t-1}.$$

Solve the recursion we have:

$$\ln \Phi^T \leq \frac{T\epsilon^2}{\rho^2} - \frac{\epsilon}{\rho^2}\sum_{t=1}^{T}\frac{w^{t-1}m^t}{\Phi^{t-1}} + \ln \Phi^0.$$

For every expert $i \in [n]$, we have $\Phi^T \geq \exp(-\epsilon\sum_{i=1}^{T} m_i^t/\rho)$. Thus, the penalty paid by our algorithm is

$$\sum_{t=1}^{T}\frac{w^{t-1}m^t}{\Phi^{t-1}} \leq T\epsilon + \frac{\rho^2}{\epsilon}\ln\frac{\Phi^0}{\Phi^T} \leq T\epsilon + \frac{\rho^2}{\epsilon}\ln\frac{n}{\exp(-\epsilon\sum_{i=1}^{T} m_i^t/\rho^2)} = T\epsilon + \frac{\rho^2}{\epsilon}\ln n + \sum_{i=1}^{T} m_i^t.$$

So if $T \geq \frac{\rho^2 \ln n}{\epsilon^2}$ then $\frac{\rho^2}{\epsilon}\ln n \leq T\epsilon$. The theorem holds by dividing the above inequality by T. □

3 Application: Approximate Linear Program Solver

Our goal in the section is to solve the following linear program (LP):

$$\begin{aligned}
a_{1,1}x_1 + a_{1,2}x_2 + \ldots a_{1,m}x_m &\geq b_1 \\
a_{2,1}x_1 + a_{2,2}x_2 + \ldots a_{2,m}x_m &\geq b_2 \\
&\vdots \\
a_{n,1}x_1 + a_{n,2}x_2 + \ldots a_{n,m}x_m &\geq b_n \\
x &\in [0,1]^n
\end{aligned}$$

Let

$$A = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{pmatrix} = \begin{pmatrix} a_{1,1}, a_{1,2}, \ldots, a_{1,m} \\ a_{2,1}, a_{2,2}, \ldots, a_{2,m} \\ \cdots \\ a_{n,1}, a_{n,2}, \ldots, a_{n,m} \end{pmatrix}, \qquad B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}, \qquad \text{and } x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}.$$

Then, our goal can be concisely written as to find an $x \in [0,1]^n$ such that $Ax \geq b$, where "$\geq$" means coordinate-wise greater than or equal to.

In an approximate LP solver, the solution we find only satisfies the LP approximately. That is, assume the LP is feasible, then our algorithm will find an $x \in [0,1]^n$ such that $A_i x \geq b_i - \epsilon$ for every $i \in [n]$.

Define $\rho = \max_{i\in[n]} \sup_{x\in[0,1]^n} |A_i x - b_i|$. All the notations will be defined in the same way as those in Section 2. Our approximate LP solver is defined in the following procedure:

1: $w_1^0 = w_2^0 = w_3^0 = \cdots = w_n^0 = 1$
2: **for** $t \leftarrow 1$ to T, where $T = \left\lceil \frac{4\rho^2 \ln n}{\epsilon^2} \right\rceil$ **do**
3: find an $x^t \in [0,1]^n$ satisfying $\frac{w^{t-1}}{\Phi^{t-1}}(Ax^t - b) \geq 0$
4: expert i pays penalty $m_i^t := A_i x^t - b_i$
5: **for** $i \in [n]$ **do** $w_i^t = w_i^{t-1} \cdot e^{-\epsilon m_i^t/\rho^2}$
6: output $x^* = \frac{1}{T}\sum_{t=1}^{T} x^t$

Analysis of Algorithm We assume $\epsilon \leq \rho$ since otherwise the problem is trivial. Since $T \geq \frac{4\rho^2 \ln n}{\epsilon^2}$, by Theorem 1, we have for every $i \in [n]$,

$$0 \leq \frac{1}{T}\sum_{t=1}^{T} \frac{w^{t-1}}{\Phi^{t-1}}(Ax^t - b) \leq \frac{1}{T}\sum_{t=1}^{T}(A_i x^t - b_i) + \epsilon = A_i x^* - b_i + \epsilon$$

Thus, for every $i \in [n]$, we have

$$A_i x^* \geq b_i - \epsilon.$$

4 Application: Online Convex Optimization

In the online convex optimization problem, there is a polytope $\mathcal{P}$. In each iteration t of the algorithm, the algorithm first chooses a point $x^t \in P$ and then the penalty function $f^t : \mathcal{P} \to \mathbb{R}_{\geq 0}$ for iteration t is revealed. Overall the penalty paid by our algorithm is then $\sum_{t\in T} f^t(x_t)$. It will infeasible to require our penalty to be small compared to the best omniscient algorithm. However, we show that this is possible if we require the omniscient algorithm can must choose the same point x at all time steps. In other words, we need $\sum_{t\in T} f^t(x_t)$ to be small compared to $\inf_{x\in P} \sum_{t\in T} f^t(x)$.

In this section, we only show how to solve the problem when P is the $(n-1)$-dimensional simplex, namely, $\mathcal{P} = \{x \in [0,1]^n : x_1 + x_2 + \cdots + x_n = 1\}$. Then every point $x \in \mathcal{P}$ can be viewed as a distribution over the n experts. Also, we assume the functions f^t are convex and have the first and second order derivatives. The algorithm is as follows:

1: initialize $w_1^0 = w_2^0 = \cdots = x_n^0 = 1$
2: **for** $t \leftarrow 1$ to T **do**
3: choose $x^t = \frac{w^{t-1}}{\Phi^{t-1}}$
4: let the penalty of the expert i be $m_i^t = (\nabla f^t(x^t))_i$
5: **for** every $i \in n$ **do** $w_i^t \leftarrow w_i^{t-1} \cdot e^{-\epsilon m_i^t/\rho^2}$

In the algorithm, ρ is an upper bound on $|(\nabla f^t(x^t))_i|$ for every t, x^t and i.

Analysis of the algorithm If $T \geq \frac{\rho^2 \ln n}{\epsilon^2}$, then for every expert i, we have

$$\frac{1}{T}\sum_{t=1}^{T} x^t \nabla f^t(x^t) \leq \frac{1}{T}\sum_{t=1}^{T}(\nabla f^t(x^t))_i + 2\epsilon.$$

Notice that every $x^* \in \mathcal{P}$ is a convex combination of the n experts. Thus, we have for every $x^* \in \mathcal{P}$,

$$\frac{1}{T}\sum_{t=1}^{T} x^t \nabla f^t(x^t) \leq \frac{1}{T}\sum_{t=1}^{T} x^* \nabla f^t(x^t) + 2\epsilon.$$

Then we have

$$\frac{1}{T}\sum_{t=1}^{T} \left(f^t(x^t) - f^t(x^*)\right) \leq \frac{1}{T}\sum_{t=1}^{T} (x^t - x^*)\nabla f^t(x_t) \leq 2\epsilon,$$

where the first inequality used that each f^t is convex.