ORIE 6334 Bridging Continuous and Discrete Optimization Sep 11th, 2019

Lecture 3

Lecturer: David P. Williamson *Scribe: Shengbo Wang*

1 The Multiplicative Weight Update Algorithm

Assume there are N possible different decisions that could be made at every time. Let $v_t(i) \in [0,1]$ denote the value resulting from making decision i at time t. Assume that $v_t(\cdot)$ is not known before one makes the decision at time t, and after making decision i, one observes $v_t(j)$ for all j. Let the time horizon be T.

Our goal is to show that over time one can make decision to obtain a total value which is almost the value one gets by using the best fixed decision, without any assumption on how v_t evolves in time. That is, the total value is almost

$$\max_{1\le j\le N} \sum_{t=1}^{T} v_t(j).$$

The randomized Algorithm 1 which we will define later is based on the idea that each decision i at time t will be associated with a weight $w_t(i)$. At each time t, i is chosen with probability $p_t(i) \propto w_t(i)$. Formally, let

$$W_t = \sum_{i=1}^{N} w_t(i), \qquad p_t(i) = \frac{w_t(i)}{W_t}.$$

Therefore, the expected total value under Algorithm 1 is

$$\sum_{t=1}^{T}\sum_{i=1}^{N} p_t(i)v_t(i).$$

Algorithm 1: Multiplicative Weights

$w_1(i) \leftarrow 1,\ \forall i = 1,\dots,N$
for $t \leftarrow 1$ **to** T **do**
  Pick decision i with probability $p_t(i)$ and get value $v_t(i)$
  $w_{t+1}(i) \leftarrow (1+\epsilon v_t(i))w_t(i),\ \forall i = 1,\dots,N$
end

Observe decision i is given higher weight at time $t+1$ if it gives higher value.
Note that the expected value gained by the algorithm is $\sum_{t=1}^{T}\sum_{i=1}^{N} p_t(i)v_t(i)$.
Now we prove that Algorithm 1 indeed achieves our goal.

[0]This lecture is based on a survey by Arora, Hazan, and Kale 2012 `http://theoryofcomputing.org/articles/v008a006/v008a006.pdf`.

Theorem 1 *Assume $\epsilon \leq 1/2$, then for all j,*

$$\sum_{t=1}^{T}\sum_{i=1}^{N} p_t(i)v_t(i) \geq (1-\epsilon)\sum_{t=1}^{T} v_t(j) - \frac{1}{\epsilon}\ln N.$$

Proof: We try to bridge the expectation that we care about and the optimal fixed decision by finding an upper and a lower bound on W_{T+1}. First, consider

$$\begin{aligned}
W_{t+1} &= \sum_{i=1}^{N} w_{t+1}(i) \\
&= \sum_{i=1}^{N} w_t(i)(1+\epsilon v_t(i)) \\
&= W_t + \epsilon W_t \sum_{i=1}^{N} p_t(i)v_t(i) \\
&= W_t\left(1+\epsilon\sum_{i=1}^{N} p_t(i)v_t(i)\right) \\
&\leq W_t \exp\left(\epsilon\sum_{i=1}^{N} p_t(i)v_t(i)\right).
\end{aligned}$$

where the last inequality follows from $1+x \leq e^x$ for $x \geq 0$. Therefore,

$$\begin{aligned}
W_{T+1} &\leq W_T \exp\left(\epsilon\sum_{i=1}^{N} p_T(i)v_T(i)\right) \\
&\leq W_{T-1} \exp\left(\epsilon\sum_{t=T-1}^{T}\sum_{i=1}^{N} p_t(i)v_t(i)\right) \\
&\leq \cdots \\
&\leq W_1 \exp\left(\epsilon\sum_{t=1}^{T}\sum_{i=1}^{N} p_t(i)v_t(i)\right) \\
&= N \exp\left(\epsilon\sum_{t=1}^{T}\sum_{i=1}^{N} p_t(i)v_t(i)\right).
\end{aligned}$$

where $W_1 = N$ since $w_1(i) = 1$ for all i.

On the other hand, for any given j,

$$W_{T+1} \geq w_{T+1}(j) = w_T(j)(1+\epsilon v_T(j)) = \prod_{t=1}^{T}(1+\epsilon v_t(j)) \geq (1+\epsilon)^{\sum_{t=1}^{T} v_t(j)},$$

using the result $1+\epsilon x \geq (1+\epsilon)^x$ for $x \in [0,1]$.

Combining the above two inequalities,

$$(1+\epsilon)^{\sum_{t=1}^{T} v_t(j)} \leq N \exp\left(\epsilon \sum_{t=1}^{T}\sum_{i=1}^{N} p_t(i)v_t(i)\right).$$

Taking the logarithm of each side we get

$$\ln(1+\epsilon)\sum_{t=1}^{T} v_t(j) \leq \ln N + \epsilon \sum_{t=1}^{T}\sum_{i=1}^{N} p_t(i)v_t(i),$$

which implies

$$\sum_{t=1}^{T}\sum_{i=1}^{N} p_t(i)v_t(i) \geq \frac{\ln(1+\epsilon)}{\epsilon}\sum_{t=1}^{T} v_t(j) - \frac{1}{\epsilon}\ln N \geq (1-\epsilon)\sum_{t=1}^{T} v_t(j) - \frac{1}{\epsilon}\ln N.$$

Here the last step follows from the inequality $\ln(1+x) \geq x - x^2$ for $x \leq 1/2$. □

2 Application: Finding ϵ-Feasible Solution

We now apply the multiplicative weights method to find ϵ-feasible solutions to the system:

$$Ax \leq e, \quad x \in Q. \tag{1}$$

Here $A \in \mathbb{R}^{m\times n}$, $e \in \mathbb{R}^m$ is the vector of all ones, and $Q \subseteq \mathbb{R}^n$ is a convex set. Assume $Ax \geq 0$ for each $x \in Q$. By ϵ-feasible, it means that $x \in Q$, $Ax \leq (1+\epsilon)e$.

We also assume that it is easy to optimize over Q, i.e., we have an *oracle*, given any $p \in \mathbb{R}^m_+$, it will find $x \in Q$ such that $p^T Ax \leq p^T e$, if such an x exists. If no such $x \in Q$ exists, then we can conclude that the system (1) is infeasible. Since $p^T Ax$ is a linear function in x, we have an oracle as long as we can optimize linear functions over Q.

We will base our Algorithm 2 on the multiplicative weights algorithm 1 defined above. For convenience, define the *width* of the oracle to be

$$\rho := \max_{i=1,\ldots,m} \max_{\substack{x\in Q \\ \text{returned} \\ \text{by oracle}}} (Ax)(i);$$

i.e. the furthest coordinate of Ax that the oracle will produce.

The idea in Algorithm 2 is to run multiplicative weights algorithm in which each decision corresponds to a row of A and its value is $\frac{1}{\rho}(Ax_t)(i)$, where x_t is a vector returned by the oracle.

Note that we didn't specify T; T depends on ϵ in the sense that to achieve ϵ-feasible, T iterations are needed; the relationship will be given below. The running time is $O(Tm)$ time plus $O(T)$ oracle calls and additional matrix-vector multiplications Ax_t which will be dependent on properties of the matrix; e.g. sparsity.

The intuition of Algorithm 2 is to increase the probability on the more violated coordinates, so in later iterations the oracle will produce more x_t satisfying the constraint on these particular coordinate.

Algorithm 2: Finding Feasible Solution to (1)

$w_1(i) \leftarrow 1,\ \forall i = 1, \ldots, m$
for $t \leftarrow 1$ **to** T **do**
  $W_t \leftarrow \sum_{i=1}^{m} w_t(i), \quad p_t(i) \leftarrow w_t(i)/W_t$
  Run oracle and obtain $x_t \in Q$ such that $p_t^T A x_t \leq p_t^T e$
  $v_t(i) \leftarrow (Ax_t)(i)/\rho$ (observe the value is in $[0,1]$ by the definition of width ρ)
  $w_{t+1}(i) \leftarrow (1 + \epsilon v_t(i)) w_t(i),\ \forall i = 1, \ldots, m$
end
return $\bar{x} := \frac{1}{T} \sum_{t=1}^{T} x_t$

Also note that the returned value $\bar{x}$ is always in Q by the convexity of Q. To show that the algorithm works, we would like to apply the theorem. First, by the algorithm, for all t

$$\sum_{i=1}^{m} p_t(i) v_t(i) = \frac{1}{\rho} p_t^T A x_t \leq \frac{1}{\rho} p_t^T e = \frac{1}{\rho},$$

since p_t gives a probability distribution.

By Theorem 1, for any j,

$$\begin{aligned}
\frac{T}{\rho} &\geq \sum_{t=1}^{T} \sum_{i=1}^{N} p_t(i) v_t(i) \\
&\geq (1-\epsilon) \sum_{t=1}^{T} v_t(j) - \frac{1}{\epsilon} \ln m \\
&= (1-\epsilon) \sum_{t=1}^{T} \frac{1}{\rho} (Ax_t)(j) - \frac{1}{\epsilon} \ln m \\
&= (1-\epsilon) \frac{T}{\rho} (A\bar{x})(j) - \frac{1}{\epsilon} \ln m.
\end{aligned}$$

Hence

$$(1-\epsilon) \frac{T}{\rho} (A\bar{x})(j) \leq \frac{T}{\rho} + \frac{1}{\epsilon} \ln m.$$

If we set $T = \rho \ln m / \epsilon^2$, then

$$(A\bar{x})(j) \leq \frac{1}{1-\epsilon} \left(1 + \frac{\rho \ln m}{\epsilon T} \right) = \frac{1+\epsilon}{1-\epsilon} \leq 1 + 4\epsilon$$

for $\epsilon \leq 1/3$, which gives $A\bar{x} \leq (1+4\epsilon)e$. The running time is

$$O\left(\frac{m\rho}{\epsilon^2} \ln m \right) + O\left(\frac{\rho}{\epsilon^2} \ln m \right) (\text{oracle calls} + \text{matrix multiplication}).$$

2.1 Application: Max Flow in Unit Capability Graphs

As a quick illustration, let's see how to apply the framework above to the maximum flow problem in unit capacity graphs. It won't give a very fast algorithm, but it will help us

understand what is going on. Let $G = (V, E)$ be a graph, and $u(i, j) = 1$ the capacity, $\forall (i, j) \in E$. Let $s \in V$ be the source and $t \in V$ be the sink. Our goal is to find a max flow from s to t.

The framework above is to decide if there is a feasible solution to a system, whereas the max flow problem is an optimization problem. In order to use the feasibility checker to find an optimal flow, we use the feasibility checker to check if there exists a flow of value k, and use a binary search for the max k that is feasible. Define $|m| = |E|$. Then an easy upper bound on the value of the maximum flow is m, and we need $\lceil \log_2(m) \rceil$ checks of feasibility.

We set up this feasibility checking problem. The A matrix in algorithm 2 checks the capacity constraints $x(i, j) \leq 1$, and hence is the identity matrix, where the variable $x \in \mathbb{R}^m$ is the flow vector; i.e. $x(i, j)$.

The convex set Q will encode the flow conservation and flow value constraints, and so is

$$Q = \left\{ x \geq 0 : \sum_{j:(i,j)\in E} x(i,j) - \sum_{j:(j,i)\in E} x(j,i) = 0, \forall i \neq s,t; \sum_{j:(s,j)\in E} x(s,j) - \sum_{j:(j,s)\in E} x(j,s) = k \right\}.$$

For any $p \geq 0$ and $\|p\|_1 = 1$, we can let the oracle to return $x \in Q$ that minimize $p^T Ax = p^T x$; if this is greater than $p^T e$ such k is not feasible. Observe that the minimum achieved on the shortest path with $d(i, j) = p(i, j)$, $\forall (i, j) \in E$. Because we can always shift the flow on longer path to the shortest one as there is no capacity imposed.