Algorithms in Action

The Multiplicative Weights Update Method

Haim Kaplan, Uri Zwick

Tel Aviv University

May 2016

Last updated: June 7, 2017

“Using expert advice”

A basic binary setting

On each on of T days:

We need to make a binary decision (Up/Down).

n “*experts*” give us their prediction (Up/Down).

Based on their advice, we make a choice.

We then find out whether our choice is correct.

If our choice is *wrong*, we pay a *penalty* of 1.

If our choice is *right*, we do not pay anything.

Our goal, of course, is to pay as little as possible.

“Using expert advice”

A basic binary setting

Days

	1	2	3	4	5	Cost
Expert 1	U	U	D	U	U	1
Expert 2	D	D	D	D	D	3
Expert 3	U	D	U	U	D	4
Our decision	U	D	D	U	U	2
Outcome	U	U	D	D	U	

“Using expert advice”

A basic binary setting

How well can we do?

If all “experts” are bad,
we cannot do too well.

We would like to do
almost as well as the best expert,
with *hindsight*.

The Weighted Majority algorithm [Littlestone-Warmuth (1994)]

Choose a parameter $0 < \eta \leq \frac{1}{2}$.

Assign each expert a *weight*.

The weight of the i-th expert at day t is $w_i^{(t)}$.

On day 1, all weights are 1: $w_i^{(1)} = 1\ ,\ \ i = 1,2, \dots , n$.

At day t, predict Up or Down according to the *weighted majority* of the experts.

Predict up, if $\sum_{i \in U^{(t)}} w_i^{(t)} \geq \sum_{i \in D^{(t)}} w_i^{(t)}$,
$U^{(t)}, D^{(t)}$ – sets of experts predicting Up/Down at day t.

Update the weights:

$$w_i^{(t+1)} = \begin{cases} w_i^{(t)} & \text{if } i \text{ was correct on day } t \\ (1-\eta) w_i^{(t)} & \text{otherwise} \end{cases}$$

The Weighted Majority algorithm
[Littlestone-Warmuth (1994)]

$cost^{(T)}(WM_\eta)$ – Number of mistakes of WM_η up to time T.

$cost^{(T)}(EXP_i)$ – Number of mistakes of EXP_i up to time T.

Theorem: For every $i = 1,2, \ldots, n$,

$$cost^{(T)}(WM_\eta) \leq 2(1+\eta)cost^{(T)}(EXP_i) + \frac{2\ln n}{\eta}$$

In particular, the inequality holds for the *best* expert.

Thus, the cost of the Weighted Majority algorithm is only slightly larger than *twice* the cost of the *best* expert!

(We can do even better using a randomized algorithm.)

The Weighted Majority algorithm [Littlestone-Warmuth (1994)]

Theorem: For every $i = 1,2,\dots,n$,

$$cost^{(T)}(WM_\eta) \leq 2(1+\eta)cost^{(T)}(EXP_i) + \frac{2\ln n}{\eta}$$

Proof:

Let $W^{(t)} = \sum_{i=1}^{n} w_i^{(t)}$. Clearly, $W^{(1)} = n$.

If WM_η makes a mistake in day t, then

$$W^{(t+1)} \leq \left(\frac{1}{2} + \frac{1}{2}(1-\eta)\right)W^{(t)} = \left(1 - \frac{\eta}{2}\right)W^{(t)}.$$

$$W^{(T+1)} \leq n\left(1-\frac{\eta}{2}\right)^{cost^{(T)}(WM_\eta)}$$

$$W^{(T+1)} \geq w_i^{(T+1)} = (1-\eta)^{cost^{(T)}(EXP_i)}$$

The Weighted Majority algorithm

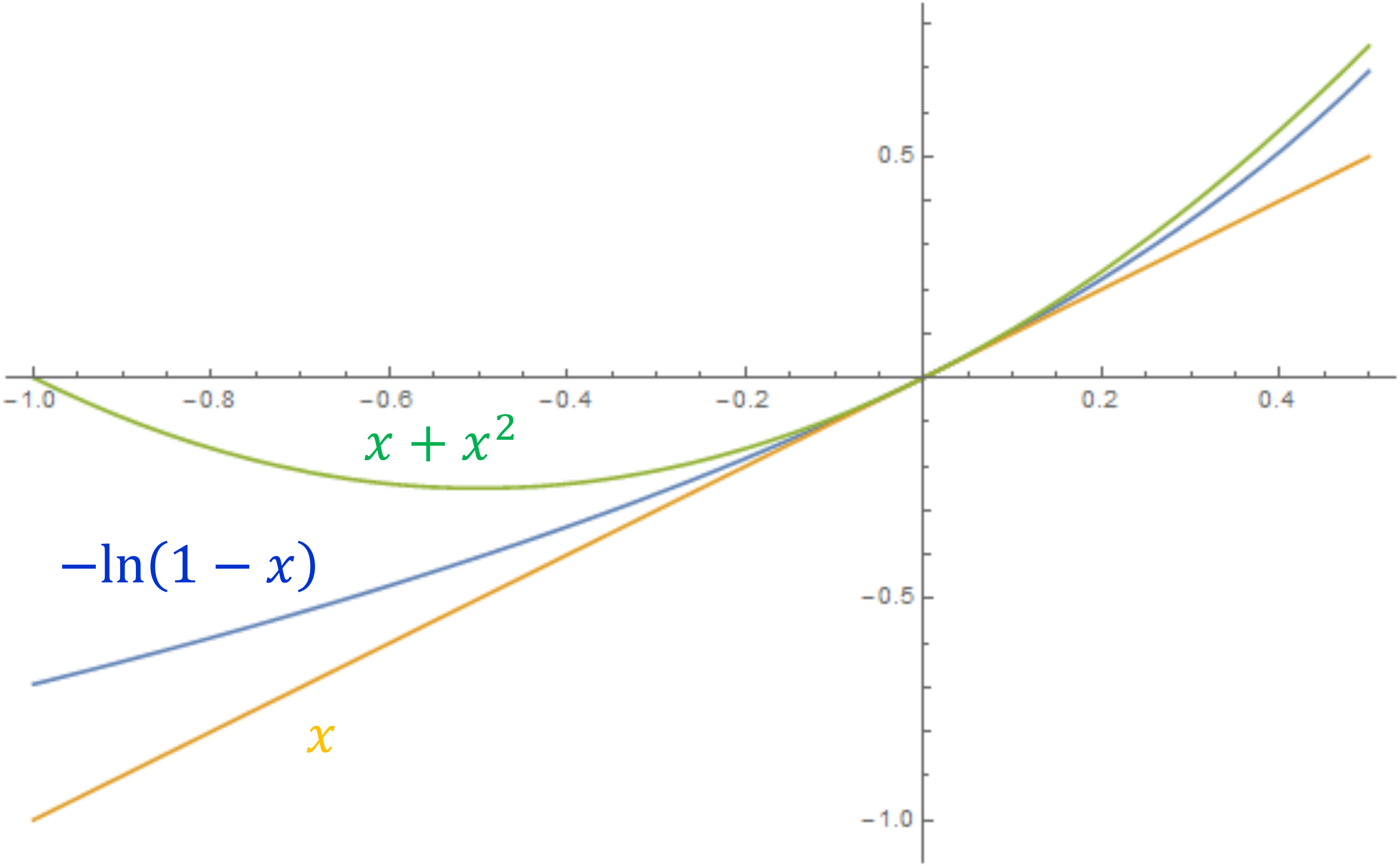
[Littlestone-Warmuth (1994)]

$$(1-\eta)^{cost^{(T)}(EXP_i)} \le W^{(T+1)} \le n\left(1-\frac{\eta}{2}\right)^{cost^{(T)}(WM_\eta)}$$

$$cost^{(T)}(EXP_i)\ln(1-\eta) \le cost^{(T)}(WM_\eta)\ln\left(1-\frac{\eta}{2}\right)+\ln n$$

$$cost^{(T)}(WM_\eta) \le \underbrace{\frac{\ln(1-\eta)}{\ln\left(1-\frac{\eta}{2}\right)}}_{\le \frac{\eta+\eta^2}{\frac{\eta}{2}} = 2(1+\eta)} cost^{(T)}(EXP_i) + \underbrace{\frac{\ln n}{-\ln\left(1-\frac{\eta}{2}\right)}}_{\le \frac{\ln n}{\frac{\eta}{2}} = \frac{2\ln n}{\eta}}$$

Using $x \le -\ln(1-x) \le x+x^2$, for $x \le \frac{1}{2}$.



$$x \leq -\ln(1-x) \leq x + x^2 \ , \quad \text{for } x \leq \frac{1}{2}.$$

“Using expert advice”

A more general setting

On each on of T days:

(Each one of n “experts” suggests a *course of action*.)

We choose a *(probability) distribution* over the experts.

The *costs* of choosing each expert are revealed.
All costs are in $[-1,1]$.

We pay the average cost according to the distribution chosen.

Our goal is to minimize our total cost.

Alternative interpretation: On each day a *random* expert is drawn according to the distribution chosen. We pay the expected cost.

“Using expert advice”

A more general setting

Days

Experts

	1		2		3		4		Cost
1	$\frac{1}{3}$	1	$\frac{1}{6}$	0	$\frac{1}{8}$	$\frac{1}{2}$	0	$\frac{1}{2}$	2
2	$\frac{1}{3}$	-1	$\frac{1}{2}$	1	$\frac{3}{8}$	$-\frac{3}{4}$	$\frac{1}{4}$	$-\frac{3}{4}$	$-\frac{3}{2}$
3	$\frac{1}{3}$	0	$\frac{1}{3}$	0	$\frac{1}{2}$	-1	$\frac{3}{4}$	-1	-2
Our cost		0		$\frac{1}{2}$		$-\frac{27}{32}$		$-\frac{15}{16}$	$-\frac{41}{32}$

The Multiplicative Weights algorithm

[Cesa-Bianchi, Mansour, Stoltz (2007)]

Choose a parameter $0 < \eta \le \frac{1}{2}$.

The weight of expert i at day t is $w_i^{(t)}$.

$$w_i^{(1)} = 1\,,\ \ i = 1,2,\ldots,n.$$

At day t use the distribution:

$$\boldsymbol{p}^{(t)} = \frac{\left(w_1^{(t)}, w_2^{(t)}, \ldots, w_n^{(t)}\right)}{W^{(t)}} \qquad W^{(t)} = \sum_{i=1}^{n} w_i^{(t)}$$

Let $\boldsymbol{m}^{(t)} = \left(m_1^{(t)}, m_2^{(t)}, \ldots, m_n^{(t)}\right)$ be the costs at day t.

Update the weights:

$$w_i^{(t+1)} = w_i^{(t)}\left(1 - \eta\, m_i^{(t)}\right)$$

The Multiplicative Weights algorithm
[Cesa-Bianchi, Mansour, Stoltz (2007)]

Theorem: Assume that $m_i^{(t)} \in [-1,1]$ and that $0 < \eta \le \frac{1}{2}$.
Let $\boldsymbol{p}^{(t)}$ be the distribution used by MW_η at day t.
Then, for every $i = 1,2, \dots, n$,

$$\underbrace{\sum_{t=1}^{T} \boldsymbol{p}^{(t)} \cdot \boldsymbol{m}^{(t)}}_{cost^{(T)}(MW_\eta)} \le \underbrace{\sum_{t=1}^{T} m_i^{(t)}}_{cost^{(T)}(EXP_i)} + \eta \underbrace{\sum_{t=1}^{T} \left(m_i^{(t)}\right)^2}_{\le \eta \sum_{t=1}^{T} \left|m_i^{(t)}\right|} + \frac{\ln n}{\eta}$$

The Multiplicative Weights algorithm

[Cesa-Bianchi, Mansour, Stoltz (2007)]

$$\ln \frac{W^{(T+1)}}{W^{(1)}} = \sum_{t=1}^{T} \ln \frac{W^{(t+1)}}{W^{(t)}} = \sum_{t=1}^{T} \ln \sum_{i=1}^{n} p_i^{(t)} \left(1 - \eta\, m_i^{(t)}\right)$$

$$= \sum_{t=1}^{T} \ln\left(1 - \eta\, \boldsymbol{p}^{(t)} \cdot \boldsymbol{m}^{(t)}\right) \;\le\; -\eta \sum_{t=1}^{T} \boldsymbol{p}^{(t)} \cdot \boldsymbol{m}^{(t)}$$

$$\ln \frac{W^{(T+1)}}{W^{(1)}} \;\ge\; \ln \frac{w_i^{(T+1)}}{n} = -\ln n + \sum_{t=1}^{T} \ln\left(1 - \eta\, m_i^{(t)}\right)$$

$$\ge\; -\ln n - \eta \sum_{t=1}^{T} m_i^{(t)} - \eta^2 \sum_{t=1}^{T} \left(m_i^{(t)}\right)^2$$

Using $-x - x^2 \;\le\; \ln(1-x) \le\; -x$, for $x \le \frac{1}{2}$.

Applications of the Multiplicative Weights algorithm

Learning a linear classifier (The Winnow algorithm)

Boosting the performance of weak learners (cf. Adaboost)

Approximately solving 0-sum 2-player games

Approximately solving packing Linear Programs

Approximately solving covering Linear Programs

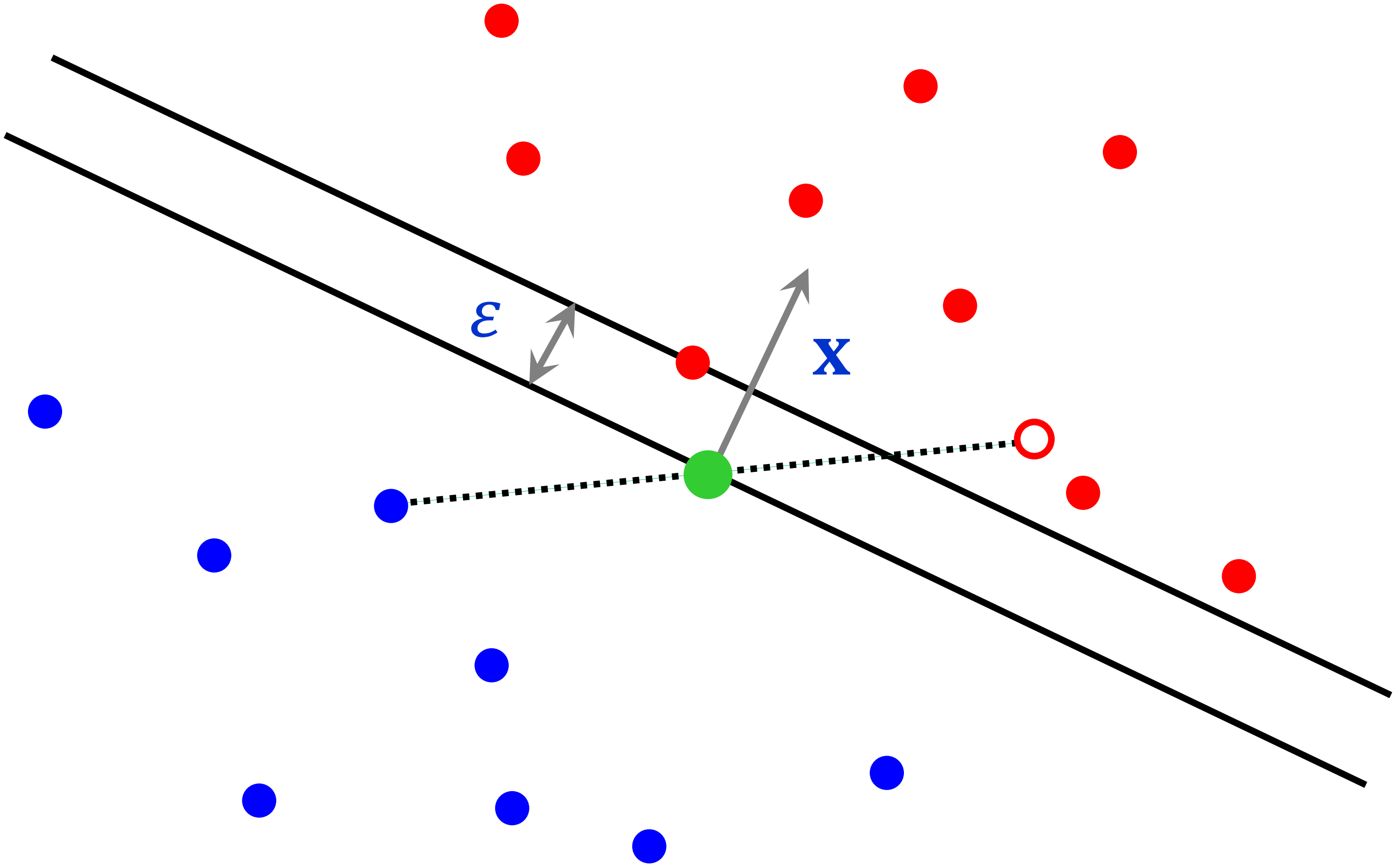
Special case: Multicommodity flow

Approximately solving Semidefinte Programs

Special case: SDP relaxation of MAX CUT

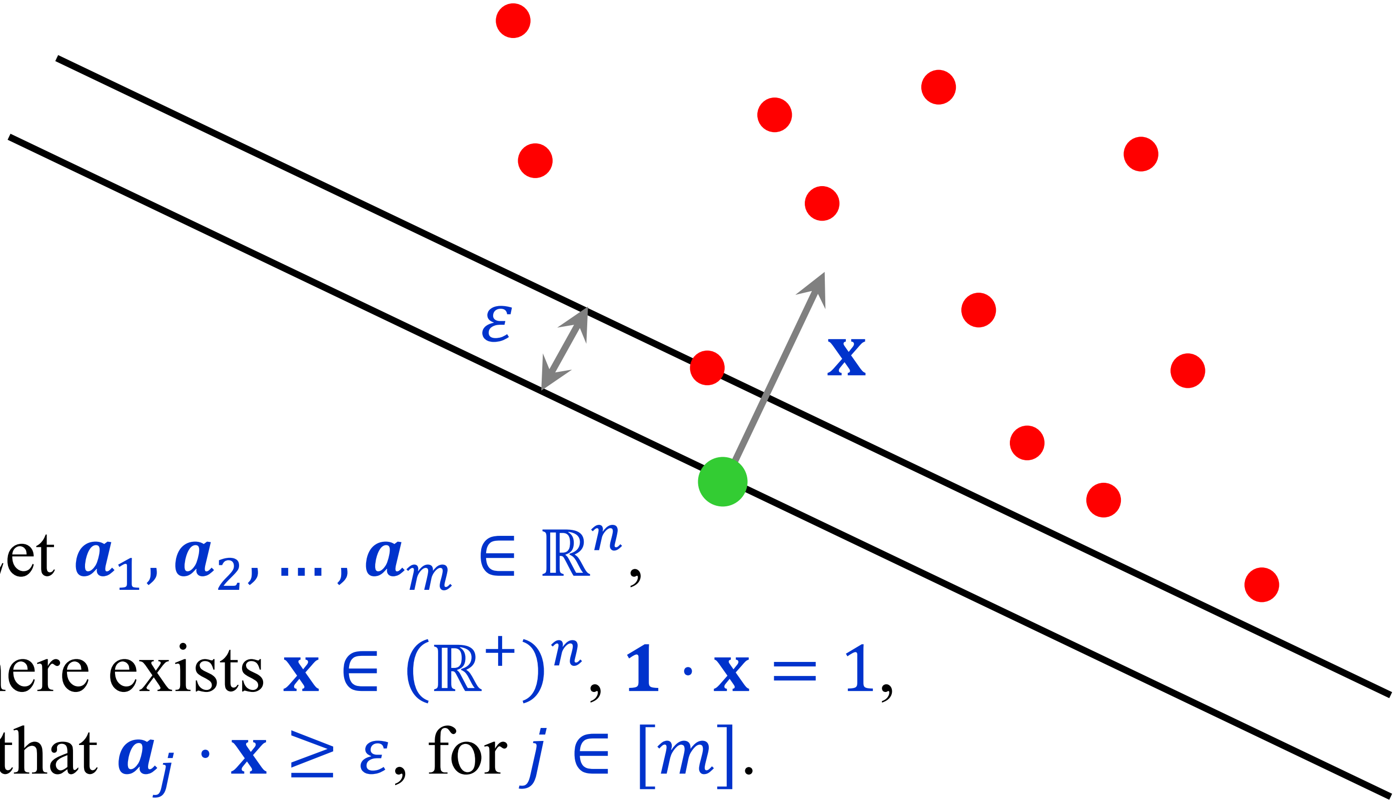
⋮

Learning a Linear Classifier



Assume, w.l.o.g., that the hyperplane passes through the origin and that $\mathbf{x} \in (\mathbb{R}^{+})^{n}$, $\mathbf{1} \cdot \mathbf{x} = 1$.

Learning a Linear Classifier



Let $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_m \in \mathbb{R}^n$,

Assume there exists $\mathbf{x} \in (\mathbb{R}^+)^n$, $\mathbf{1} \cdot \mathbf{x} = 1$,
such that $\boldsymbol{a}_j \cdot \mathbf{x} \geq \varepsilon$, for $j \in [m]$.

Find $\mathbf{x}' \in (\mathbb{R}^+)^n$, $\mathbf{1} \cdot \mathbf{x}' = 1$,
such that $\boldsymbol{a}_j \cdot \mathbf{x}' \geq 0$, for $j \in [m]$.

Let $\rho = \max_j \|a_j\|_\infty$.

Learning a Linear Classifier - The Winnow algorithm [Littlestone (1987)]

Experts correspond coordinates (also known as *features*).

Run MW_η with $\eta = \varepsilon/2\rho$.

In each iteration, if $\boldsymbol{p}^{(t)}$ is a good classifier, stop.

Otherwise, let j be such that $\boldsymbol{p}^{(t)} \cdot \boldsymbol{a}_j < 0$.

Let $\boldsymbol{m}^{(t)} = -\boldsymbol{a}_j/\rho$.

Theorem: If there exists a classifier $\mathbf{x}^*$ such $\boldsymbol{a}_j \cdot \mathbf{x}^* \geq \varepsilon$, $j \in [m]$, then Winnow finds a classifier $\mathbf{x}$ such that $\boldsymbol{a}_j \cdot \mathbf{x} \geq 0$, $j \in [m]$, after at most $T = \frac{4\rho^2}{\varepsilon^2} \ln n$ iterations.

Learning a Linear Classifier - The Winnow algorithm [Littlestone (1987)]

For every coordinate (expert) i we have:

$$\sum_{t=1}^{T} \boldsymbol{p}^{(t)} \cdot \boldsymbol{m}^{(t)} \leq \sum_{t=1}^{T} m_i^{(t)} + \eta \sum_{t=1}^{T} \left| m_i^{(t)} \right| + \frac{\ln n}{\eta}$$

Thus, for *every* distribution $\boldsymbol{p}$ we have:

$$\sum_{t=1}^{T} \boldsymbol{p}^{(t)} \cdot \boldsymbol{m}^{(t)} \leq \sum_{t=1}^{T} \boldsymbol{p} \cdot \boldsymbol{m}^{(t)} + \eta \sum_{t=1}^{T} \boldsymbol{p} \cdot \left| \boldsymbol{m}^{(t)} \right| + \frac{\ln n}{\eta}$$

We choose $\boldsymbol{p} = \mathbf{x}^*$.

The Winnow algorithm [Littlestone (1987)]

$$\sum_{t=1}^{T} \boldsymbol{p}^{(t)} \cdot \boldsymbol{m}^{(t)} \le \sum_{t=1}^{T} \mathbf{x}^{*} \cdot \boldsymbol{m}^{(t)} + \eta \sum_{t=1}^{T} \mathbf{x}^{*} \cdot \left|\boldsymbol{m}^{(t)}\right| + \frac{\ln n}{\eta}$$

$\boldsymbol{m}^{(t)} = \frac{-\boldsymbol{a}_{j_t}}{\rho}$, for some j_t such that $\boldsymbol{a}_{j_t} \cdot \boldsymbol{p}^{(t)} < 0.$

$$\underbrace{\sum_{t=1}^{T} \boldsymbol{p}^{(t)} \cdot \frac{-\boldsymbol{a}_{j_t}}{\rho}}_{0 <} \le \underbrace{\sum_{t=1}^{T} \mathbf{x}^{*} \cdot \frac{-\boldsymbol{a}_{j_t}}{\rho}}_{\le -\frac{\varepsilon T}{\rho} = -2\eta T} + \underbrace{\eta \sum_{t=1}^{T} \mathbf{x}^{*} \cdot \frac{\left|\boldsymbol{a}_{j_t}\right|}{\rho}}_{\le \eta T} + \frac{\ln n}{\eta}$$

$$\eta T \le \frac{\ln n}{\eta} \Rightarrow T \le \frac{\ln n}{\eta^{2}} = \left(\frac{2\rho}{\varepsilon}\right)^{2} \ln n$$

0-sum 2-player matrix games

$$A = \begin{bmatrix} 1 & 0 & 2 \\ 3 & 1 & -1 \\ -2 & 4 & 1 \end{bmatrix}$$

ROW chooses a row i.

COLUMN chooses a column j.

ROW pays COLUMN $A[i, j]$.

$$2 = \min_i \max_j A[i, j] \quad > \quad \max_j \min_i A[i, j] = 0$$

No player wants to go first…

Suppose the players play *simultaneously*.

Playing *deterministically* is similar to playing first.

Use *randomized* (*mixed*) strategies.

0-sum 2-player matrix games

Randomized (mixed) strategy for ROW:
A distribution $\boldsymbol{p}$ over the rows of A.

Randomized (mixed) strategy for COLUMN:
A distribution $\boldsymbol{q}$ over the columns of A.

If ROW uses $\boldsymbol{p}$ and COLUMN uses $\boldsymbol{q}$,
the expected payoff is:

$$A[\boldsymbol{p},\boldsymbol{q}] = \sum_{i,j} p_i q_j A[i,j] = \boldsymbol{p}^T A \boldsymbol{q}$$

0-sum 2-player matrix games

Von Neumann's min-max theorem:

$$\begin{array}{ccc}
\min\limits_{\boldsymbol{p}} \max\limits_{\boldsymbol{q}} A[\boldsymbol{p},\boldsymbol{q}] & = & \max\limits_{\boldsymbol{q}} \min\limits_{\boldsymbol{p}} A[\boldsymbol{p},\boldsymbol{q}] \\
\| & & \| \\
\min\limits_{\boldsymbol{p}} \max\limits_{j} A[\boldsymbol{p},j] & & \max\limits_{\boldsymbol{q}} \min\limits_{i} A[i,\boldsymbol{q}] \\
\| & & \| \\
\begin{array}{ll} \min & v \\ \text{s.t.} & \boldsymbol{p}^T A \leq v\mathbf{1}^T \\ & \boldsymbol{p}^T\mathbf{1} = 1 \\ & \boldsymbol{p} \geq 0 \end{array} & \underset{\substack{\uparrow \\ \text{LP} \\ \text{Duality}}}{=} & \begin{array}{ll} \max & v \\ \text{s.t.} & A\boldsymbol{q} \geq v\mathbf{1} \\ & \mathbf{1}^T\boldsymbol{q} = 1 \\ & \boldsymbol{q} \geq 0 \end{array}
\end{array}$$

0-sum 2-player matrix games

$$A = \begin{bmatrix} 1 & 0 & 2 \\ 3 & 1 & -1 \\ -2 & 4 & 1 \end{bmatrix}$$

ROW chooses a row i.

COLUMN chooses a column j.

ROW pays COLUMN $A[i,j]$.

$$2 = \min_i \max_j A[i,j] \quad > \quad \max_j \min_i A[i,j] = 0$$

What is the value and what are the optimal strategies?

$$\begin{array}{c|ccc} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \hline 6/11 & 1 & 0 & 2 \\ 3/11 & 3 & 1 & -1 \\ 2/11 & -2 & 4 & 1 \end{array}$$

value = 1

Solving 0-sum games approximately

Value and optimal strategies can be found by solving an LP.

Can be done in polynomial time, but relatively slowly.

In many situations a good approximation is sufficient.

W.l.o.g., assume that all entries of A are in $[0,1]$.

Let $v^* = val(A)$ be the value of A. Let $\varepsilon > 0$.

$\boldsymbol{p}$ and $\boldsymbol{q}$ are ε-optimal strategies iff:

$$\max_j A[\boldsymbol{p}, j] \leq v^* + \varepsilon \qquad \min_i A[i, \boldsymbol{q}] \geq v^* - \varepsilon$$

0-sum games using Multiplicative Updates [Freund-Schapire (1999)]

Experts correspond to the n rows of A.

A distribution over the experts is a mixed strategy for ROW.

In iteration t, the algorithm produces a distribution $\boldsymbol{p}^{(t)}$.

The cost vector $m^{(t)}$ is the column $j^{(t)}$ of A maximizing $A[\boldsymbol{p}^{(t)}, j]$.

Note that $\boldsymbol{p}^{(\mathrm{t})} \cdot \boldsymbol{m}^{(t)} = A\left[\boldsymbol{p}^{(t)}, j^{(t)}\right] \geq v^{*}$.

Theorem: If MW_{η} is run with $\eta = \varepsilon/2$ for $T = 4 \ln n / \varepsilon^{2}$ iterations, then the *best* strategy obtained is ε-optimal for ROW.

If A has m columns, the total running time is $O(\frac{mn \ln n}{\varepsilon^{2}})$.

An ε-optimal strategy for COLUMN can also be found.

0-sum games using Multiplicative Updates [Freund-Schapire (1999)]

For any distribution $\boldsymbol{p}$, and in particular $\boldsymbol{p} = \boldsymbol{p}^*$, we have

$$\sum_{t=1}^{T} \underbrace{A(\boldsymbol{p}^{(t)}, j^{(t)})}_{v^* \leq} \leq (1+\eta) \sum_{t=1}^{T} \underbrace{A(\boldsymbol{p}^*, j^{(t)})}_{\leq v^* \leq 1} + \frac{\ln n}{\eta}$$

$$v^* \leq \frac{1}{T} \sum_{i=1}^{T} A\left(\boldsymbol{p}^{(t)}, j^{(t)}\right) \leq v^* + \underbrace{\eta}_{= \frac{\varepsilon}{2}} + \underbrace{\frac{\ln n}{\eta T}}_{\leq \frac{\varepsilon}{2}} \leq \ v^* + \varepsilon$$

$$\eta = \varepsilon / 2$$

$$T = 4 \ln n \, / \varepsilon^2$$

0-sum games using Multiplicative Updates [Freund-Schapire (1999)]

$$v^* \;\le\; \frac{1}{T}\sum_{t=1}^{T} A\left(\boldsymbol{p}^{(t)}, j^{(t)}\right) \;\le\; v^* + \varepsilon$$

For at least one t we have:

$$A\left(\boldsymbol{p}^{(t)}, j^{(t)}\right) = \max_j A\left(\boldsymbol{p}^{(t)}, j\right) \le v^* + \varepsilon$$

Thus, if t minimizes $A\left(\boldsymbol{p}^{(t)}, j^{(t)}\right)$,
then $\boldsymbol{p}^{(t)}$ is ε-optimal for ROW.

($\frac{1}{T}\sum_{t=1}^{T} \boldsymbol{p}^{(t)}$ is also ε-optimal for ROW.)

0-sum games using Multiplicative Updates [Freund-Schapire (1999)]

Let $\boldsymbol{q}$ be such that $q_j = \left|\{\, t \mid j^{(t)} = j \}\right| / T$.

For every i, $$\frac{1}{T}\sum_{t=1}^{T} A\left(i, j^{(t)}\right) = A(i, \boldsymbol{q})$$

$$v^* \leq \frac{1}{T}\sum_{t=1}^{T} A\left(\boldsymbol{p}^{(t)}, j^{(t)}\right) \leq (1+\eta)\frac{1}{T}\sum_{t=1}^{T} A\left(i, j^{(t)}\right) + \frac{\ln n}{\eta T}$$

$$\leq A(i, \boldsymbol{q}) + \varepsilon$$

Hence, $v^* - \varepsilon \leq A(i, \boldsymbol{q})$, for every i,
so $\boldsymbol{q}$ is ε-optimal for COLUMN.

Rewards instead of costs

On day t we get a reward vector $\boldsymbol{r}^{(t)}$, instead of a cost vector $\boldsymbol{m}^{(t)}$.

Maximize reward instead of minimizing cost.

Simply let $\boldsymbol{m}^{(t)} = -\boldsymbol{r}^{(t)}$.

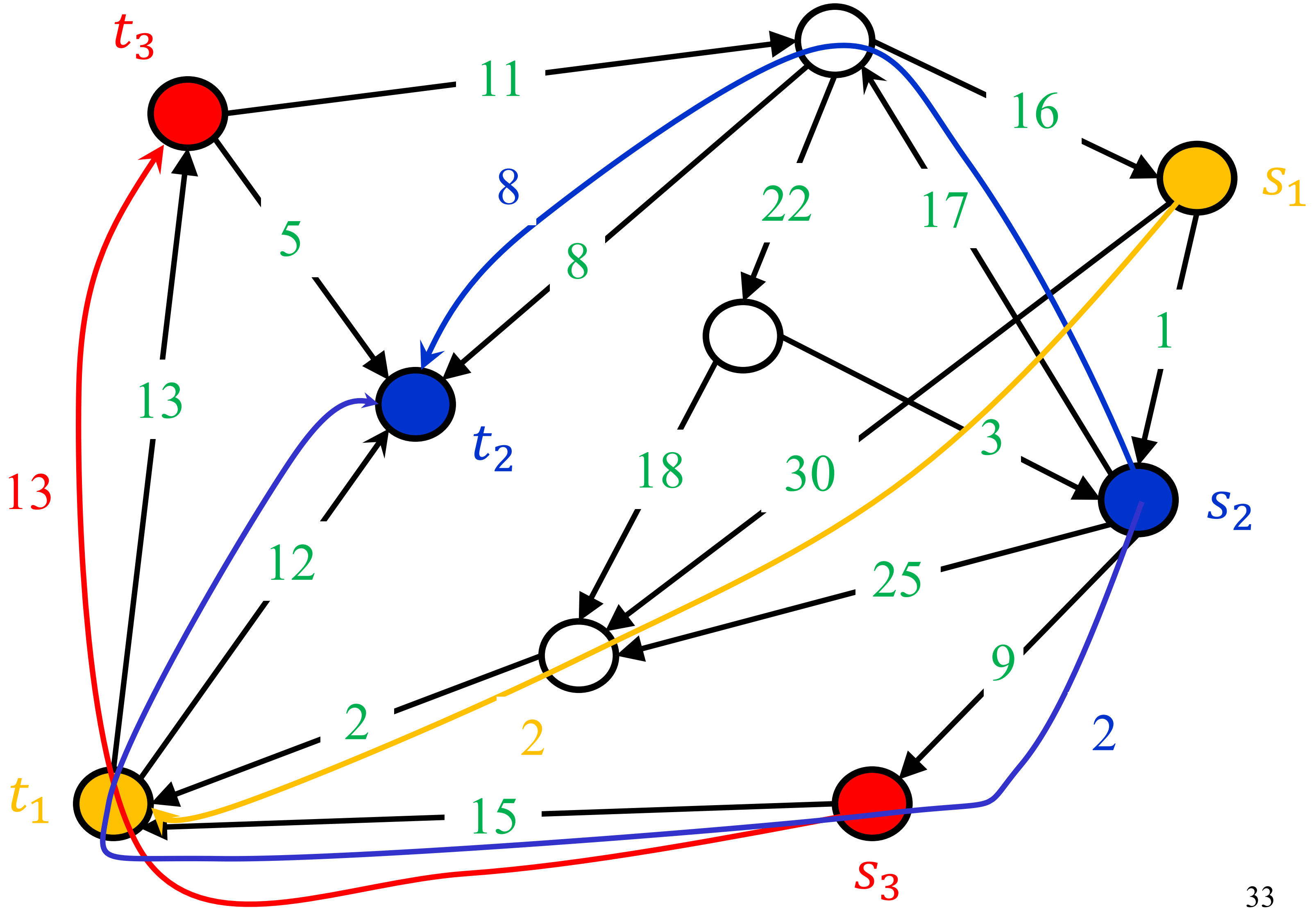
Multiplicative weight update:

$$w_i^{(t+1)} = w_i^{(t)} \left(1 + \eta\, r_i^{(t)}\right)$$

Theorem: Assume that $r_i^{(t)} \in [-1,1]$ and that $0 < \eta \leq \frac{1}{2}$.
Let $\boldsymbol{p}^{(t)}$ be the distribution used by MW_η at day t.
Then, for every $i = 1,2,\ldots,n$,

$$\sum_{t=1}^{T} \boldsymbol{p}^{(t)} \cdot \boldsymbol{r}^{(t)} \geq \sum_{t=1}^{T} r_i^{(t)} - \eta \sum_{t=1}^{T} \left(r_i^{(t)}\right)^2 - \frac{\ln n}{\eta}$$

Maximum Multicommodity Flow



Maximum Multicommodity Flow

$G = (V, E)$ – A directed graph (the flow network)

$c: E \to \mathbb{R}^+$ – A *capacity* function

$(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)$ - k source-sink pairs.

Maximize the total flow, i.e., the flow sent from s_1 to t_1, plus the flow sent from s_2 to t_2, etc.

Different commodities can share the edges of the network.

The total flow on an edge should not exceed its capacity.

Exercise: Express the maximum multicommodity flow as a linear program of polynomial size. (Hint: For every edge e introduce k flow variables $f_1(e), f_2(e), \ldots f_k(e)$.

Maximum Multicommodity Flow

We use a different LP formulation of the problem of possibly exponential size (!)

Let $\mathbb{P}$ be the set of simple directed paths from s_1 to t_1, and from s_2 to t_2, etc.

For $p \in \mathbb{P}$, let $f_p \geq 0$ be a variable that expresses the flow, of the appropriate commodity, on p.

We want to maximize $\sum_{p \in \mathbb{P}} f_p$ subject to $f_e = \sum_{p \ni e} f_p \leq c_e$, for every $e \in E$.

$$\begin{aligned} &\max \textstyle\sum_{p \in \mathbb{P}} f_p \\ &\text{s.t.} \quad \textstyle\sum_{p \ni e} f_p \leq c_e \,,\, e \in E \\ &\qquad\;\; f_p \geq 0 \,,\, p \in \mathbb{P} \end{aligned}$$

Maximum Multicommodity flow [Garg-Könemman (2007)]

A polynomial time $(1-\varepsilon)$-approximation algorithm.
Our presentation follows [Arora-Hazan-Kale (2012)].

Maintain a flow $\boldsymbol{f} = \boldsymbol{f}^{(t)}$ (may violate the capacity constraints).
Maintain a weight function $\boldsymbol{w}^{(t)}$ $(\sim \boldsymbol{p}^{(t)})$ on the edges.
Use Multiplicative weight updates with $\eta = \varepsilon/2$.

In each iteration:

Find a shortest path $p^{(t)} \in \mathbb{P}$ w.r.t. $w_e^{(t)}/c_e$.
Route $c^{(t)}$ units of flow on $p^{(t)}$, where $c^{(t)} = \min_{e \in p^{(t)}} c_e$.

Define $r_e^{(t)} = c^{(t)}/c_e \in [0,1]$, if $e \in p^{(t)}$, and $r_e^{(t)} = 0$, otherwise.

Let f_e be the total flow so far on e. Stop when $\exists e\ f_e/c_e \geq (\ln m)/\eta^2$.

Down-scale the flow $\boldsymbol{f}$ to establish all capacity constraints.

Maximum Multicommodity flow [Garg-Könemman (2007)]

$$\sum_{t=1}^{T} \boldsymbol{p}^{(t)} \cdot \boldsymbol{r}^{(t)} \ \ge\ (1-\eta)\underbrace{\sum_{t=1}^{T} r_e^{(t)}}_{= f_e/c_e} - \frac{\ln m}{\eta} \ , \qquad \forall\, e \in E$$

(See next slide.)

$$\sum_{t=1}^{T} \boldsymbol{p}^{(t)} \cdot \boldsymbol{r}^{(t)} = \sum_{t=1}^{T} \frac{\sum_{e\in p^{(t)}} w_e^{(t)} \cdot \frac{c^{(t)}}{c_e}}{\sum_{e\in E} w_e^{(t)}} = \sum_{t=1}^{T} c^{(t)} \underbrace{\boxed{\frac{\sum_{e\in p^{(t)}} \frac{w_e^{(t)}}{c_e}}{\sum_{e\in E} w_e^{(t)}}}}_{\le 1/F^{opt}}$$

$$\le \frac{1}{F^{opt}} \sum_{t=1}^{T} c^{(t)} = \frac{F}{F^{opt}}$$

Let $\boldsymbol{f}^{opt}$ be the optimal flow and let $F^{opt} = \sum_{p\in\mathbb{P}} f_p^{opt}$

Maximum Multicommodity flow [Garg-Könemman (2007)]

Let $\boldsymbol{f}^{opt}$ be the optimal flow and let $F^{opt} = \sum_{p\in\mathbb{P}} f_p^{opt}$

Let w_e be arbitrary (non-negative) edge weights.

Let $p \in \mathbb{P}$ be a shortest path w.r.t. edge lengths w_e/c_e.

$$\frac{\sum_{e\in E} w_e}{\sum_{e\in p} \frac{w_e}{c_e}} \geq \frac{\sum_{e\in E} w_e \cdot \sum_{p'\ni e} \frac{f_{p'}^{opt}}{c_e}}{\sum_{e\in p} \frac{w_e}{c_e}} \qquad \leq 1$$

$$= \frac{\sum_{p'\in\mathbb{P}} f_{p'}^{opt} \sum_{e\in p'} \frac{w_e}{c_e}}{\sum_{e\in p} \frac{w_e}{c_e}} \geq \sum_{p'\in\mathbb{P}} f_{p'}^{opt} = F^{opt}$$

≥ 1

Maximum Multicommodity flow [Garg-Könemman (2007)]

$$\frac{F}{F^{opt}} \geq \sum_{t=1}^{T} \boldsymbol{p}^{(t)} \cdot \boldsymbol{r}^{(t)} \geq (1-\eta) \underbrace{\max_e \frac{f_e}{c_e}} - \frac{\ln m}{\eta} \geq (1-2\eta)C$$

Maximum congestion $= C$

Upon termination:

$$C \geq \frac{\ln m}{\eta^2}$$

$$\frac{F}{F^{opt}} \geq (1-2\eta)C$$

Scale down the flow by C :

$$\frac{F}{C} \geq (1-2\eta)F^{opt} = (1-\varepsilon)F^{opt}$$

F/C is an $(1-\varepsilon)$-approximate maximal flow!

Maximum Multicommodity flow [Garg-Könemman (2007)]

How many iterations are needed?

We stop the algorithm when maximum congestion $C \geq \frac{\ln m}{\eta^2}$.

Each iteration adds 1 to the congestion of at least one edge.

Thus, number of iterations is at most $m \left\lceil \frac{\ln m}{\eta^2} \right\rceil$.

Total running time is $O\left(\frac{m \ln m}{\varepsilon^2}\ k\, T_{sp}(m)\right) = \tilde{O}\left(\frac{k\, m^2}{\varepsilon^2}\right)$.

[Fleisher (2000)] reduced the running time to $\tilde{O}\left(\frac{m^2}{\varepsilon^2}\right)$.

Positive Semidefinite Programming

$$\begin{aligned} \text{max } & C \bullet X \\ \text{s.t. } & A_j \bullet X \leq b_j \text{ , } j \in [m] \\ & X \succcurlyeq 0 \end{aligned}$$

$X, A_1, \dots, A_m \in \mathbb{R}^{n \times n}$, $b_1, \dots, b_m \in \mathbb{R}$

$A \bullet B = \sum_{i,j} a_{i,j} b_{i,j}$ (matrix inner product)

$A \succcurlyeq 0$ ((symmetric) positive semidefinite)

$\Leftrightarrow$ $\boldsymbol{x}^T A \boldsymbol{x} \geq 0$ for every $\boldsymbol{x} \in \mathbb{R}^n$

Can also be approximated using multiplicative updates.

Interesting application:
Approximation algorithm for MAX CUT

Bibliography

Sanjeev Arora, Elad Hazan, Satyen Kale,
The Multiplicative Weights Update Method:
A Meta-Algorithm and Applications,
Theory of Computing, Volume 8 (2012), pp. 121-164

Bonus material

Not covered in class this term

“Careful. We don’t want to learn from this.”
(Calvin in Bill Watterson’s “Calvin and Hobbes”)

Packing Linear Programs

$$A\mathbf{x} \le \boldsymbol{b}$$
$$\mathbf{x} \in \mathbb{K}$$

Find a feasible $\mathbf{x} \in \mathbb{R}^n$,
or show that none exists.

$A \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$, $\mathbb{K} \subseteq \mathbb{R}^n$ is a “simple” convex set

Packing: $A\mathbf{x} \ge 0$, for every $\mathbf{x} \in \mathbb{K}$.
$\boldsymbol{b} > 0$

By scaling, we sometimes assume that $\boldsymbol{b} = \mathbf{1}$.

Willing to settle for $\mathbf{x} \in \mathbb{K}$ such that $A\mathbf{x} \le \boldsymbol{b} + \varepsilon$

ORACLE: Given a distribution $\boldsymbol{p}$ on the rows of A,
return $\mathbf{x} \in \mathbb{K}$ such that $\boldsymbol{p}^{\mathrm{T}} A\mathbf{x} \le \boldsymbol{p}^{\mathrm{T}}\boldsymbol{b}$, or “no” if none exists.

If ORACLE returns “no” for any distribution $\boldsymbol{p}$,
then the problem is *infeasible*.

Packing Linear Programs

$$A\mathbf{x} \leq \boldsymbol{b}$$
$$\mathbf{x} \in \mathbb{K}$$

Find a feasible $\mathbf{x} \in \mathbb{R}^n$,
or show that none exists.

$A \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$, $\mathbb{K} \subseteq \mathbb{R}^n$ is a “simple” convex set

$\mathbb{K} \subseteq \mathbb{R}^n$ is convex iff

$$\boldsymbol{x}, \boldsymbol{y} \in \mathbb{K} \,,\, 0 \leq \alpha \leq 1 \quad \rightarrow \quad (1-\alpha)\boldsymbol{x} + \alpha\boldsymbol{y} \in \mathbb{K}$$

“Simple” is used informally. The only requirement is that ORACLE can be efficiently implemented.

Example: $\mathbb{K} = \{\, \mathbf{x} \in \mathbb{R}^n : \mathbf{x} \geq 0 \,,\, \boldsymbol{c}^T\mathbf{x} = f \,\}$.

Packing Linear Programs

$$A\mathbf{x} \le \boldsymbol{b}$$
$$\mathbf{x} \in \mathbb{K}$$

Find a feasible $\mathbf{x} \in \mathbb{R}^n$,
or show that none exists.

$A \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$, $\mathbb{K} \subseteq \mathbb{R}^n$ is a “simple” convex set

ORACLE: Given a distribution $\boldsymbol{p}$ on the rows of A, return $\mathbf{x} \in \mathbb{K}$ such that $\boldsymbol{p}^{\mathrm{T}} A\mathbf{x} \le \boldsymbol{p}^{\mathrm{T}} \boldsymbol{b}$, or “no” if none exists.

ORACLE is $(1,\rho)$-bounded iff
for every point $\mathbf{x} \in \mathbb{K}$ returned and every $i \in [m]$,

$$-1 \le A_i\mathbf{x} - b_i \le \rho$$

This is automatic,
as $\boldsymbol{A}\mathbf{x} \ge 0$, $\boldsymbol{b} = \mathbf{1}$.

The *width*.

Packing LPs using multiplicative weights
[Plotkin-Shmoys-Tardos (1995)]

Experts correspond to the m linear constraints (rows of A).

A distribution $\boldsymbol{p}$ corresponds to the constraint $\boldsymbol{p}^T A\mathbf{x} \leq \boldsymbol{p}^T \boldsymbol{b}$.

The costs at iteration t are determined by a point $\mathbf{x}^{(t)} \in \mathbb{K}$.

$$\boldsymbol{m}^{(t)} = \frac{1}{\rho}\left(\boldsymbol{b} - A\mathbf{x}^{(t)}\right) \in \left[-1, \frac{1}{\rho}\right]^m$$

Note: Satisfied constraints are more costly.

Use MW_η to produce distributions $\boldsymbol{p}^{(1)}, \boldsymbol{p}^{(2)}, \ldots, \boldsymbol{p}^{(T)}$.

In iteration t apply ORACLE to $\boldsymbol{p}^{(t)}$ to obtain $\mathbf{x}^{(t)}$ and $\mathbf{m}^{(t)}$.

If ORACLE returns “no” in any iteration, problem infeasible.

Run for $T = 8\rho \ln m / \varepsilon^2$ and return $\bar{\mathbf{x}} = \frac{1}{T}\sum_t \mathbf{x}^{(t)}$.

Packing LPs using multiplicative weights [Plotkin-Shmoys-Tardos (1995)]

Theorem: For any $\varepsilon \geq 0$, after $T = 8\rho \ln m \,/ \varepsilon^2$ iterations of MW_η, $\eta = \varepsilon/4$, with an $(1, \rho)$-ORACLE, the point $\bar{\mathbf{x}} = \frac{1}{T}\sum_t \mathbf{x}^{(t)}$ satisfies $A\mathbf{x} \ \leq \ \boldsymbol{b} + \frac{\varepsilon}{1-\eta}$ and $\mathbf{x} \in \mathbb{K}$.

As $\mathbf{x}^{(t)}$ is the ORACLE's response to $\boldsymbol{p}^{(t)}$, we have:

$$\boldsymbol{p}^{(t)} \cdot \boldsymbol{m}^{(t)} \ = \frac{1}{\rho}\left(\boldsymbol{p}^{(t)^T}\boldsymbol{b} - \boldsymbol{p}^{(t)^T}A\mathbf{x}^{(t)}\right) \ \geq \ 0$$

For every constraint ("expert") i we have:

$$0 \ \leq \ \sum_{t=1}^{T}\frac{1}{\rho}\left(b_i - A_i\mathbf{x}^{(t)}\right) \ + \ \eta\sum_{t=1}^{T}\frac{1}{\rho}\left|b_i - A_i\mathbf{x}^{(t)}\right| \ + \ \frac{\ln n}{\eta}$$

Useful fact:

$$\sum_{t=1}^{T} x_t + \eta \sum_{t=1}^{T} |x_t| = (1-\eta)\sum_{t=1}^{T} x_t + 2\eta \sum_{t=1}^{T} (x_t)^+$$

$$(x_t)^+ = \max\{0, x_t\}$$

Packing LPs using multiplicative weights
[Plotkin-Shmoys-Tardos (1995)]

$$0 \le \sum_{t=1}^{T} \frac{1}{\rho}\left(b_i - A_{\boldsymbol{i}}\mathbf{x}^{(t)}\right) + \eta \sum_{t=1}^{T} \frac{1}{\rho}\left|b_i - A_{\boldsymbol{i}}\mathbf{x}^{(t)}\right| + \frac{\ln n}{\eta}$$

$$= (1-\eta) \sum_{t=1}^{T} \frac{1}{\rho}\left(b_i - A_{\boldsymbol{i}}\mathbf{x}^{(t)}\right) + 2\eta \sum_{t=1}^{T} \frac{1}{\rho}\underbrace{\left(b_i - A_{\boldsymbol{i}}\mathbf{x}^{(t)}\right)^{+}}_{\le 1} + \frac{\ln n}{\eta}$$

ORACLE is $(1, \rho)$-bounded →

$\times \frac{\rho}{T}$

$$0 \le (1-\eta)\underbrace{\frac{1}{T}\sum_{t=1}^{T}\left(b_i - A_{\boldsymbol{i}}\mathbf{x}^{(t)}\right)}_{= b_i - A_i\bar{\mathbf{x}}} + \underbrace{2\eta}_{\le \frac{\varepsilon}{2}} + \underbrace{\frac{\rho \ln n}{\eta T}}_{\le \frac{\varepsilon}{2}}$$

Maximum Multicommodity Flow

$$\begin{aligned} \max \ & \textstyle\sum_{p \in \mathbb{P}} f_p \\ \text{s.t.} \ & \frac{1}{c_e} \textstyle\sum_{p \ni e} f_p \le 1 \,, e \in E \\ & f_p \ge 0 \,, p \in \mathbb{P} \end{aligned}$$

Using binary search can be essentially reduced to:
Is there a feasible multicommodity flow $\boldsymbol{f}$ of value $\sum_{p \in \mathbb{P}} f_p = F$?

$$\mathbb{K} = \{\, \boldsymbol{f} : \boldsymbol{f} \ge 0 \,, \textstyle\sum_{p \in \mathbb{P}} f_p = F \,\}$$

This is now a packing problem.

ORACLE is given a *weight* $w_e \ge 0$ for each edge and has to find a flow $\boldsymbol{f}$, if there is one, such that

$$\textstyle\sum_e w_e \frac{1}{c_e} \sum_{p \ni e} f_p \le \sum_e w_e \quad , \quad \sum_{p \in \mathbb{P}} f_p = F$$

Note: The flow $\boldsymbol{f}$ returned by ORACLE does not have to satisfy *all* the capacity constraints. Only *one* weighted capacity constraint.

Maximum Multicommodity flow

$$\frac{1}{c_e}\sum_{p \ni e} f_p \leq 1\,,\ e \in E$$

$$\boldsymbol{f} \in \mathbb{K} = \{\, \boldsymbol{f} : \boldsymbol{f} \geq 0\,, \sum_{p\in\mathbb{P}} f_p = F \,\}$$

ORACLE is given a weight $w_e \geq 0$ for each edge and has to find a flow $\boldsymbol{f}$, if there is one, such that

$$\sum_e w_e \frac{1}{c_e}\sum_{p \ni e} f_p \ \leq\ \sum_e w_e \quad , \quad \sum_{p\in\mathbb{P}} f_p = F$$

$$\sum_e w_e \frac{1}{c_e}\sum_{p \ni e} f_p = \sum_{p\in\mathbb{P}} f_p \sum_{e\in p} \frac{w_e}{c_e}$$

Find a path $p \in \mathbb{P}$ that minimizes $\sum_{e\in p} \frac{w_e}{c_e}$.

If $F\sum_{e\in p} \frac{w_e}{c_e} \leq\ \sum_e w_e$, send F units of flow on p, i.e., $f_p = F$.

Otherwise, return “no”.

ORACLE just needs to solve k shortest paths problems.

Maximum Multicommodity flow

How good is the algorithm obtained using the framework?

Number of iterations is $T = 8\rho \ln m / \varepsilon^2$

In each iteration, solve k shortest paths problems in $\tilde{O}(mk)$ time.

We also need to multiply by the cost of the binary search.

ORACLE is $(1, \rho)$-bounded iff
for every point $\mathbf{x} \in \mathbb{K}$ returned and every $i \in [m]$,

$$-1 \le A_i \mathbf{x} - b_i \le \rho$$

In our case: $\rho \le \frac{F}{c_{min}} - 1$, where $c_{min} = \min_{e \in E} c_e$

The running time is $\tilde{O}\left(\frac{F \cdot mk}{\varepsilon^2 c_{min}}\right)$

The running time is not polynomial!