Multiplicative weights method: A meta algorithm with applications to linear and semi-definite programming

Sanjeev Arora
Princeton University

Based upon:
Fast algorithms for Approximate SDP [FOCS ‘05]
$\sqrt{\log(n)}$ approximation to SPARSEST CUT in $\tilde{O}(n^2)$ time [FOCS ‘04]
The multiplicative weights update method and it’s applications [’05]
See also recent papers by Hazan and Kale.

Multiplicative update rule (long history)

n agents weights

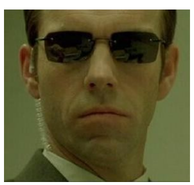
w_1

w_2

.

.

.



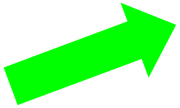
w_n

Update weights according to performance:

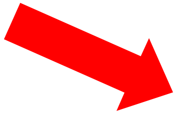
w_i^{t+1} Ã w_i^t (1 + ε ¢ performance of i)

Applications: approximate solutions to LPs and SDPs, flow problems, online learning (boosting), derandomization & chernoff bounds, online convex optimization, computational geometry, metricembeddongs, portfolio management… (see our survey)

Simplest setting – predicting the market



1$ for correct prediction



0$ for incorrect

- N “experts” on TV
- Can we perform as good as the best expert ?

Weighted majority algorithm [LW '94]

"Predict according to the weighted majority."

Multiplicative update (initially all w_i =1):

- If expert predicted correctly: w_i^{t+1} Ã w_i^t
- If incorrectly, w_i^{t+1} Ã $w_i^t(1 - \varepsilon)$

Claim: #mistakes by algorithm ¼ 2(1+ε)(#mistakes by best expert)

- Potential: ϕ_t = Sum of weights= $\sum_i w_i^t$ (initially n)
- If algorithm predicts incorrectly) ϕ_{t+1} · $\phi_t - \varepsilon\, \phi_t /2$
- ϕ_T · $(1-\varepsilon/2)^{m(A)}\, n$ m(A) =# mistakes by algorithm
- ϕ_T ¸ $(1-\varepsilon)^{m_i}$
-) m(A) · $2(1+\varepsilon)m_i + O(\log n/\varepsilon)$

Generalized Weighted majority

[A.,Hazan, Kale '05]

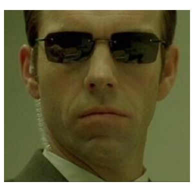
Set of events (possibly infinite)

n agents

event j

expert i

payoff
M(i,j)



Generalized Weighted majority [AHK '05]

Set of events (possibly infinite)

n agents

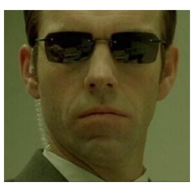
p_1

p_2

.

.

.



p_n

Algorithm: plays distribution on experts $(p_1,\ldots,p_n)$

Payoff for event j: $\sum_i p_i M(i,j)$

Update rule:
p_i^{t+1} Ã $p_i^t (1 + \varepsilon$ ¢ $M(i,j))$

Claim: After T iterations,

Algorithm payoff ¸ $(1-\varepsilon)$ best expert – $O(\log n / \varepsilon)$

Game playing,
Online optimization

Lagrangean relaxation

Gradient descent

Chernoff bounds

Games with Matrix
Payoffs

Fast soln to
LPs, SDPs

Common features of MW algorithms

- "competition" amongst n experts
- Appearance of terms like

  $\exp(-\sum_t \text{(performance at time t)})$

- Time to get ε-approximate solutions is proportional to $1/\varepsilon^2$.

Application1 : Approximate solutions to LPs ("Combinatorial")

- Plotkin Shmoys Tardos '91
- Young'97
- Garg Koenemann'99
- Fleischer'99

MW Meta-Algorithm gives unified view

Solving LPs (feasibility)

w_1 $\quad$ $\mathbf{a}_1$¢ x ¸ b_1 - ε

w_2 $\quad$ $\mathbf{a}_2$¢ x ¸ b_2 - ε

⋮ $\quad$ ⋮ $\quad$ ⋮

w_m $\quad$ $\mathbf{a}_m$¢ x ¸ b_m - ε

x 2 **P**

P = convex domain

Oracle

$\sum_k w_k(\mathbf{a}_k$¢ $x - b_k)$ ¸ 0

x 2 **P**

Solving LPs (feasibility)

$[1 - \varepsilon(\mathbf{a}_1 \text{ ¢ } x_0 - b_1)/\rho] \pounds\ w_1$ $\quad$ $\mathbf{a}_1\text{¢ } x \text{ ¸ } b_1$

$[1 - \varepsilon(\mathbf{a}_2 \text{ ¢ } x_0 - b_2)/\rho] \pounds\ w_2$ $\quad$ $\mathbf{a}_2\text{¢ } x \text{ ¸ } b_2$

$\vdots \qquad \vdots \qquad \vdots$

Final solution = Average x vector

$[1 - \varepsilon(\mathbf{a}_m \text{ ¢ } x_0 - b_m)/\rho] \pounds\ w_m$ $\quad$ $\mathbf{a}_m\text{¢ } x \text{ ¸ } b_m$

$x\ 2\ \mathbf{P}$

x_0

$|\mathbf{a}_k \text{ ¢ } x_0 - b_k| \cdot \rho$

ρ = width

Oracle

$\sum_k w_k(\mathbf{a}_k\text{¢ } x - b_k) \text{ ¸ } 0$

$x\ 2\ \mathbf{P}$

Performance guarantees

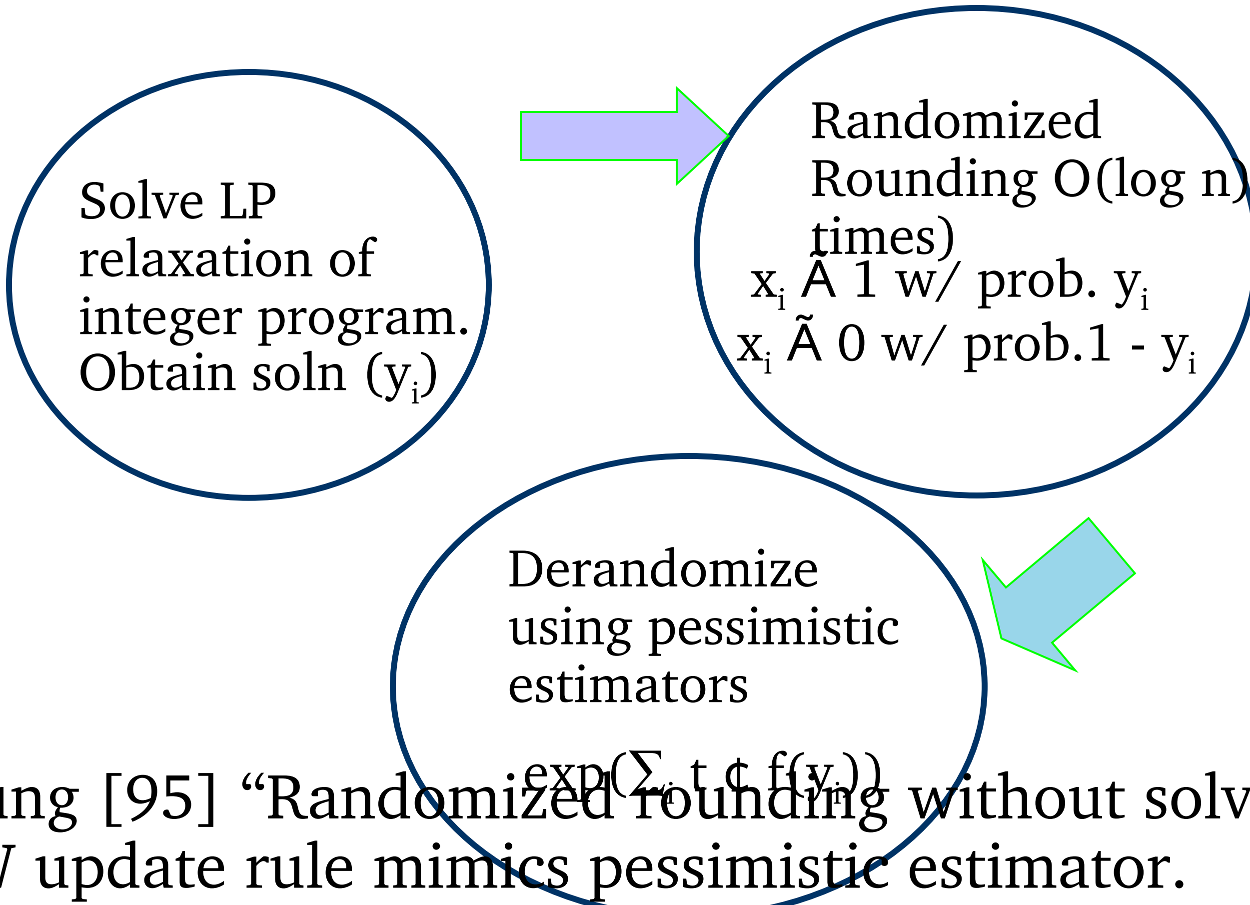
- In O(ρ² log(n)/ε²) iterations, average x is ε feasible.
- Packing-Covering LPs: [Plotkin, Shmoys, Tardos '91]
  - 9? x 2 **P**:

    j = 1, 2, … m: **a**ⱼ ¢ x ¸ 1

    Covering problem
  - Want to find x 2 P s.t. **a**ⱼ ¢ x ¸ 1 - ε
  - Assume: 8 x 2 P: 0 · **a**ⱼ ¢ x · ρ
  - MW algorithm gets ε feasible x in O(ρ log(n)/ε²) iterations

Connection to Chernoff bounds and derandomization

Deterministic approximation algorithms for 0/1 packing/covering problem *a la* Raghavan-Thompson



Young [95] "Randomized rounding without solving the LP."
MW update rule mimics pessimistic estimator.

Application 2:

Semidefinite programming (Klein-Lu'97)

$$\mathbf{a}_1 \cdot x \geq b_1$$
$$\mathbf{a}_2 \cdot x \geq b_2$$
$$\vdots$$
$$\mathbf{a}_m \cdot x \geq b_m$$
$$x \in \mathbf{P}$$

$\mathbf{a}_j$ and x : symmetric matrices in $\mathbf{R}^{n \times n}$

$\mathbf{P} =$ $\{x\!: x \text{ is psd}; \; tr(x) = 1\}$

Oracle: max $\sum_j w_j (\mathbf{a}_j \cdot x)$ over $\mathbf{P}$

(eigenvalue computation!)

Next few slides: New Results (AHK'04, AHK'05)

Key difference between efficient and
not-so-efficient implementations
of the MW idea: Width management.

(e.g., the difference between PST'91 and
GK'99)

Solving SDP relaxations more efficiently

[AHK'05]

Problem	Using Interior Point	Our result
MAXQP (e.g. MAX-CUT)	$\tilde{O}(n^{3.5})$	$\tilde{O}(n^{1.5}N/\varepsilon^{2.5})$ or $\tilde{O}(n^{3}/\alpha^{*}\varepsilon^{3.5})$
HAPLOFREQ	$\tilde{O}(n^{4})$	$\tilde{O}(n^{2.5}/\varepsilon^{2.5})$
SCP	$\tilde{O}(n^{4})$	$\tilde{O}(n^{1.5}N/\varepsilon^{4.5})$
EMBEDDING	$\tilde{O}(n^{4})$	$\tilde{O}(n^{3}/d^{5}\varepsilon^{3.5})$
SPARSEST CUT	$\tilde{O}(n^{4.5})$	$\tilde{O}(n^{3}/\varepsilon^{2})$
MIN UNCUT *etc*	$\tilde{O}(n^{4.5})$	$\tilde{O}(n^{3.5}/\varepsilon^{2})$

Recall: issue of width

MW

$\mathbf{a}_1$¢ x ¸ b_1

$\mathbf{a}_2$¢ x ¸ b_2

⋮

$\mathbf{a}_m$¢ x ¸ b_m

Oracle

$\sum_k w_k(\mathbf{a}_k$¢ $x - b_k)$ ¸ 0

x 2 $\mathbf{P}$

- $\tilde{O}(\rho^2/\varepsilon^2)$ iterations to obtain ε feasible x
- $\rho = \max_k |\mathbf{a}_k$ ¢ $x - b_k|$
- ρ is too large!!

Issue 1:Dealing with width

MW

$\mathbf{a}_1 \cdot x \geq b_1$

$\mathbf{a}_2 \cdot x \geq b_2$

$\vdots$

$\mathbf{a}_m \cdot x \geq b_m$

Oracle

$\sum_k w_k(\mathbf{a}_k \cdot x - b_k) \geq 0$

$x \in \mathbf{P}$

- A few high width constraints
- Oracle: separation hyperplane for dual
- Can run ellipsoid/Vaidya
- poly(m, log(ρ/ε)) iterations to obtain ε feasible x

Dealing with width (contd)

MW

$\mathbf{a}_1 ¢\ x\ ¸\ b_1$

$\mathbf{a}_2 ¢\ x\ ¸\ b_2$

$\vdots$

Dual ellipsoid/Vaidya

$\mathbf{a}_m ¢\ x\ ¸\ b_m$

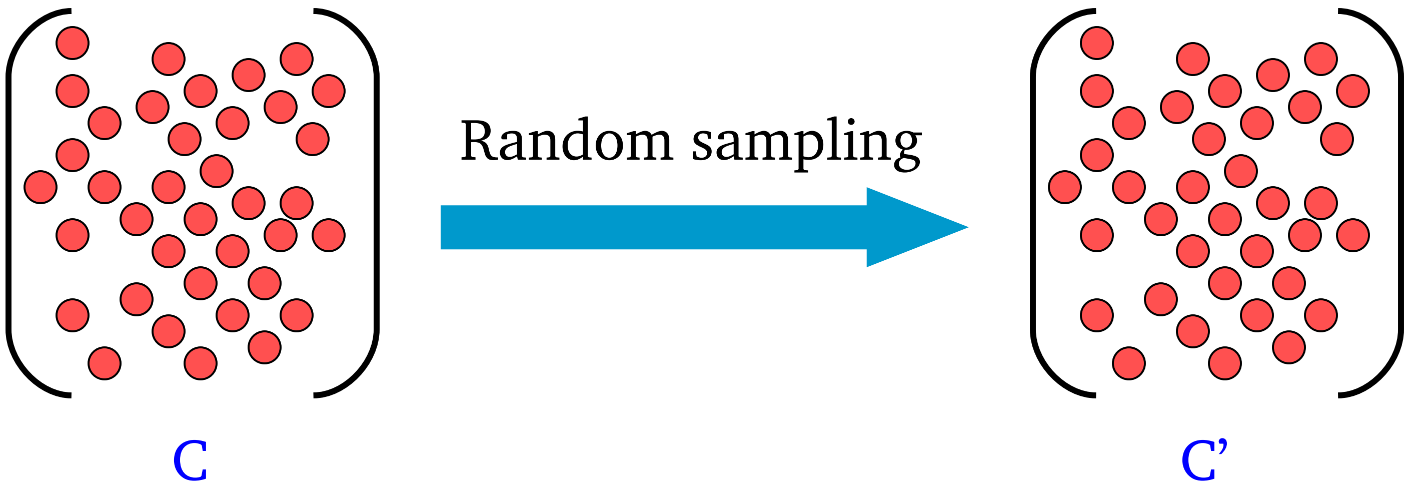
Oracle

$\sum_k w_k(\mathbf{a}_k ¢\ x - b_k)\ ¸\ 0$

$x\ 2\ \mathbf{P}$

- Hybrid of MW and Vaidya
- $\tilde{O}(\rho_L{}^2/\varepsilon^2)$ iterations to obtain ε feasible x
- ρ_L ¿ ρ

Issue 2: Efficient implementation of Oracle: fast eigenvalues via matrix sparsification



$O(\sqrt{n}\sum_{ij}|C_{ij}|/\varepsilon)$ non-zero entries

$$\|C - C'\| \le \varepsilon$$

- Lanczos effectively uses sparsity of C
- Similar to Achlioptas, McSherry ['01], but better in some situations (also easier analysis)

Online games with matrix payoffs (Satyen Kale'06)

Payoff is a matrix, and so is the "distribution" on experts!

Uses matrix analogues of usual inequalities

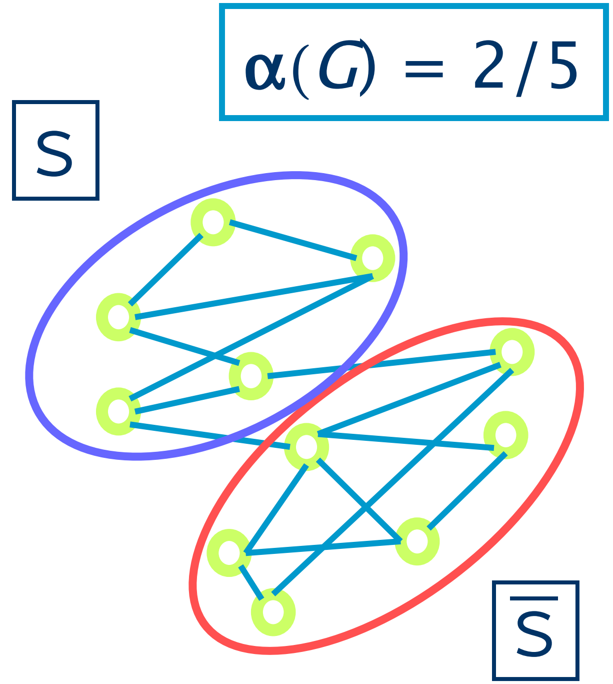
$1+ x \cdot e^{x}$ $\quad\quad$ $I + A \cdot e^{A}$

Used (together with many other tricks) to solve "triangle inequality" SDPs in $O(n^3)$ time.

$O(n^2)$-time algorithm to compute O(plog n)-approximation to SPARSEST CUT

(v/s $O(n^{4.5})$ using interior point methods)

Sparsest Cut



The sparsest cut:

$$\alpha := \min_{S \subseteq V,\ |S| < V/2} \frac{|E(S, \bar{S})|}{|S|}$$

- $O(\log n)$ approximation [Leighton Rao ’88]
- $O(\sqrt{\log n})$ approximation [A., Rao, Vazirani’04]
- $O(\sqrt{\log n})$ approximation in $O(n^2)$ time. (Actually, finds expander flows) [A., Hazan, Kale’05]

MW algorithm to find expander flows

- Events – { (s,w,z) | weights on vertices, edges, cuts}
- Experts – pairs of vertices (i,j)
- Payoff: (for weights $d_{i,j}$ on experts)

$$\sum_{ij} d_{ij} (s_i + s_j + l_{ij} - z_{ij})$$

shortest path according to weights w_e

Cuts separating i and j

Fact: If events are chosen optimally, the distribution on experts $d_{i,j}$ converges to a demand graph which is an “expander flow”

[by results of Arora-Rao-Vazirani ’04 suffices to produce approx. sparsest cut]

Faster algorithms for online learning and portfolio management

(Agarwal-Hazan'06, Agarwal-Hazan-Kalai-Kale'06)

- Framework for online optimization inspired by Newton's method (2nd order optimization). (Note: MW ¼ gradient descent)

- Fast algorithms for Portfolio management and other online optimization problems

Open problems

- Better approaches to width management?
- Faster run times?

THANK YOU

Connection to Chernoff bounds and derandomization

- Deterministic approximation algorithms *a la* Raghavan-Thompson
- Packing/covering IP with variables $x_i = 0/1$

  9? x 2 P: 8 j 2 [m], $f_j(x)$ ¸ 0

  - Solve LP relaxation using variables y_i 2 [0, 1]
  - Randomized rounding:

$$x_i = \begin{cases} 1 & \text{w.p. } y_i \\ 0 & \text{w.p. } 1 - y_i \end{cases}$$

- Chernoff: $O(\log m)$ sampling iterations suffice

Derandomization [Young, '95]

- Can derandomize the rounding using $\exp(t\sum_j f_j(x))$ as a pessimistic estimator of failure probability
- By minimizing the estimator in every iteration, we mimic the random expt, so $O(\log m)$ iterations suffice
- The structure of the estimator obviates the need to solve the LP: *Randomized rounding without solving the Linear Program*
- Punchline: resulting algorithm is the MW algorithm!

Weighted majority [LW '94]

- If lost at t, $\phi_{t+1} \cdot (1-½\, \varepsilon)\, \phi_t$
- At time T: $\phi_T \cdot (1-½\, \varepsilon)^{\#\text{mistakes}}$ ¢ ϕ_0

$$(1 ¡ \;")^{m_i} = w_i^T \cdot \underset{i}{X} \; w_i^T = ©_T$$

#mistakes of expert i

- Overall:

#mistakes $\cdot \log(n)/\varepsilon + (1+\varepsilon)\, m_i$

Semidefinite programming

- Vectors $\mathbf{a}_j$ and x: symmetric matrices in $\mathbf{R}^{n\,£\,n}$
- x ° 0
- Assume: $\mathbf{Tr}(x) \cdot 1$
- Set $\mathbf{P} = \{x\colon x$ ° $0,\ \mathbf{Tr}(x) \cdot 1\}$
- Oracle: max $\sum_j w_j(\mathbf{a}_j$ ¢ $x)$ over $\mathbf{P}$
- Optimum: $x = vv^T$ where v is the largest eigenvector of $\sum_j w_j\ \mathbf{a}_j$

Efficiently implementing the oracle

- Optimum: $x = vv^T$
  - v is the largest eigenvector of some matrix C
- Suffices to find a vector v such that $v^T C v$ ¸ 0
- Lanczos algorithm with a random starting vector is ideal for this
- Advantage: uses only matrix-vector products
  - Exploits sparsity (also: sparsification procedure)
- Use analysis of Kuczynski and Wozniakowski ['92]