

The Multiplicative Weights Update Method: Applications to Linear Programming and Semidefinite Programming

Zheming Gao and Missy Gaddy

Based on paper by Arora, Hazan, and Kale (2012)

November 29, 2017

Outline

- ① Algorithm Overview
- ② MW Algorithm for Linear Programming
- ③ MW Algorithm for Semidefinite Programming

- ① Algorithm Overview
- ② MW Algorithm for Linear Programming
- ③ MW Algorithm for Semidefinite Programming

Multiplicative Weights (MW)

Algorithm Overview

- A decision maker has a set of n possible decisions
- Begin with equal probabilities of choosing each decision
- In each iteration:
 - ① Decision maker chooses a decision
 - ② Obtains a (possibly negative) payoff
 - ③ Adjusts the probabilities with a multiplicative factor based on payoff

- $t \in \mathbb{R}$: iteration number (aka - “round”)
- T : Number of iterations in total.
- $p^{(t)} \in \mathbb{R}^n$: probability vector for round t
 $p_i^{(t)}$ is the probability of selecting decision i in round t
- $m^{(t)} \in \mathbb{R}^n$: gain vector “revealed by nature” after the decision in round t is made
(assume $m_i^{(t)} \in [-1, 1], \forall i, \forall t$)

General MW Algorithm

Fix $\eta \leq \frac{1}{2}$. Initialize $w_i^{(0)} = 1 \forall i = 1, \dots, n$.

For $t = 1, \dots, T$:

- 1 Choose decision i with probability $p_i^{(t)} = \frac{w_i^{(t)}}{\sum_i w_i^{(t)}}$
- 2 Observe the payoff $m^{(t)}$
- 3 Update the weights: for each i ,

$$w_i^{(t+1)} = w_i^{(t)} \exp(\eta m_i^{(t)})$$

Performance

- Expected gain is “not too much less” than the gain of the best decisions

$$\sum_{t=1}^T m^{(t)} \cdot p^{(t)} \geq \sum_{t=1}^T m_i^{(t)} - \eta \sum_{t=1}^T (m_i^{(t)})^2 \cdot p^{(t)} - \frac{\ln n}{\eta}$$

$(\forall i = 1, \dots, n).$

- MW has many applications in machine learning, game theory, online decision making, etc.

Outline

- ① Algorithm Overview
- ② MW Algorithm for Linear Programming
- ③ MW Algorithm for Semidefinite Programming

Linear Classifier Problem

MW algorithm can be used to solve a linear classifier problem

- m labeled examples (ℓ_j, a_j) , with $\ell_j \in \{-1, 1\}$ and $a_j \in \mathbb{R}^n$
- Want to find a normal vector x to a hyperplane so that

$$\text{sgn}(a_j^T x) = \ell_j \quad \forall j$$

- Equivalently, we want to find x such that

$$\ell_j a_j^T x \geq 0 \quad \forall j$$

(Redefine $a_j \rightarrow \ell_j a_j$)

MW Algorithm to Solve an LP

MW Algorithm can be used to solve the following LP:

$$a_j^T x \geq 0 \quad \forall j = 1, \dots, m$$

$$\mathbf{1}^T x = 1$$

$$x_i \geq 0 \quad \forall i = 1, \dots, m$$

(x is analogous to the distribution p)

MW Algorithm to Solve the Linear Classifier Problem

Define $\rho = \max_j \|a_j\|_\infty$, $\eta = \frac{\epsilon}{2\rho}$, $\epsilon > 0$.

Initialize $x_i^{(0)} = \frac{1}{n}, \forall i$

While $\exists j$ s.t. $a_j^T x^{(t)} < 0$

① Find the index j such that $a_j^T x^{(t)} < 0$

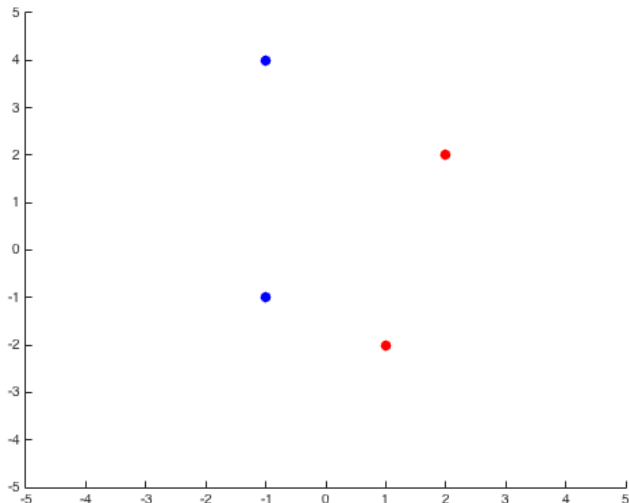
② Obtain payoff $m^{(t)} = \frac{a_j}{\rho}$

③ Update $x^{(t)}$: for each i ,

$$x_i^{(t+1)} = x_i^{(t)} \exp(\eta m_i^{(t)})$$

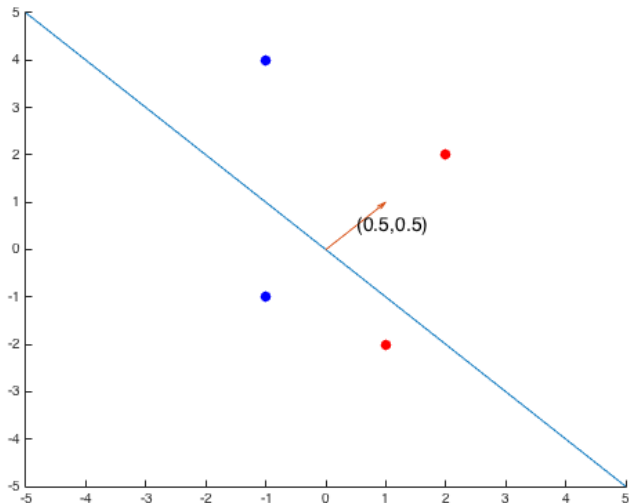
Linear Classifier: Toy Example

Find a separating hyperplane for the following points



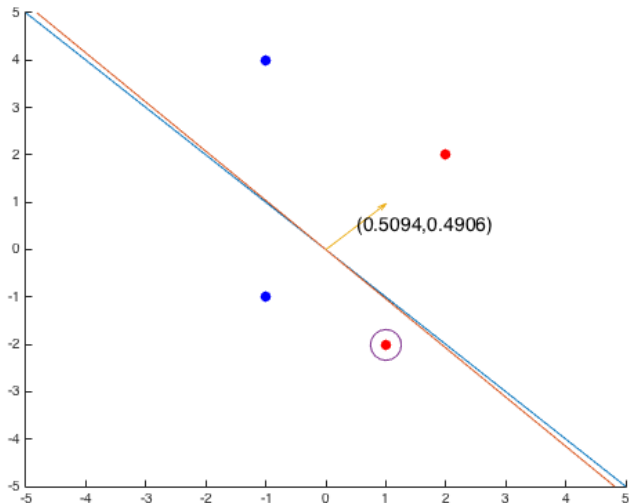
Linear Classifier: Toy Example

Initialize: $x^{(0)} = [\frac{1}{2}, \frac{1}{2}]$.



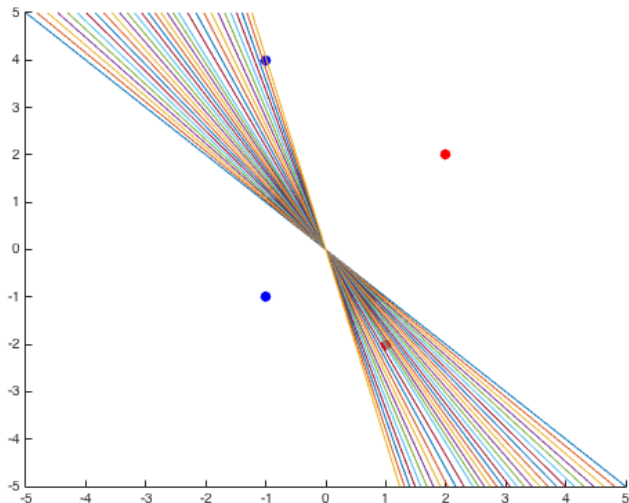
Linear Classifier: Toy Example

Misclassified example yields gain of $m^{(0)} = [\frac{1}{2}, -1]$.



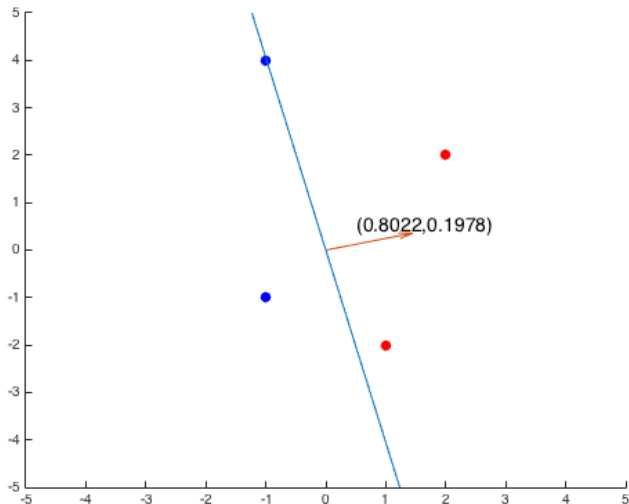
Linear Classifier: Toy Example

Continued iterations:



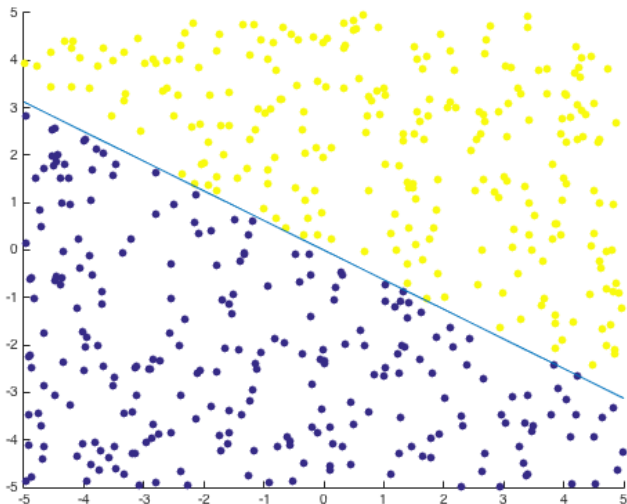
Linear Classifier: Toy Example

Final separating hyperplane, and solution to the LP:



Larger linear classifier problem

500 labeled examples



Outline

- ① Algorithm Overview
- ② MW Algorithm for Linear Programming
- ③ MW Algorithm for Semidefinite Programming

Matrix MW Algorithm

Solving an SDP using Multiplicative Weights algorithm:

Recall the LP:

$$\begin{aligned} a_j^T x &\geq 0 \quad \forall j \\ \mathbf{1}^T x &= 1 \\ x_i &\geq 0 \quad \forall i \end{aligned}$$

Now the SDP:

$$\begin{aligned} A_j \bullet x &\geq 0 \quad \forall j = 1, \dots, m \\ \text{tr}(X) &= 1 \\ X &\succcurlyeq 0 \end{aligned}$$

Matrix MW Algorithm

Define $\rho = \max_j \|A_j\|$, $\eta = -\ln(1 - \epsilon)$, $\epsilon > 0$.

Initialize $W^{(0)} = I_n$.

While $\exists j$ s.t. $A_j \bullet X^{(t)} < 0$:

① Find the index j such that $A_j \bullet X^{(t)} < 0$

② Obtain the payoff $M^{(t)} = \frac{A_j}{\rho}$

③ Update the weight matrix $W^{(t)}$:

$$W^{(t+1)} = \exp \left(\eta \sum_{\tau=1}^t M^{(\tau)} \right)$$

Recall the MAXCUT problem

Given a weighted graph G , find a partition of the vertices into two vertex sets V_1 and V_2 that maximizes the “cut” edge weights

(pic of graph)

Recall the MAXCUT problem

For a graph G with n vertices, the vector $x \in \mathbb{R}^n$ gives the partition:

$$x_i = \begin{cases} 1, & \text{if } v_i \in V_1 \\ -1, & \text{if } v_i \in V_2 \end{cases}$$

And w_{ij} = the edge weight between vertex i and vertex j

$$\max_{x \in \{-1, 1\}^n} \sum_{(i,j) \in E} w_{ij} \frac{1 - x_i x_j}{2}$$

The MAXCUT relaxation

Introduce the auxiliary matrix variable $X = xx^T$:

$$\begin{aligned} \max_{x, X} \quad & \sum_{(i,j) \in E} w_{ij} \frac{1 - x_i x_j}{2} \\ \text{s.t.} \quad & X = xx^T \\ & x_i^2 = 1 \end{aligned}$$

Then drop the rank 1 constraint for the relaxation:

$$\begin{aligned} \max_X \quad & \frac{1}{4} L \bullet X \\ \text{s.t.} \quad & X_{ii} = 1 \quad \forall i = 1, \dots, n \\ & X \succeq 0 \end{aligned}$$

Where L is the Laplacian matrix of the graph.

Reformulating the MAXCUT
relaxation

Decision form: Does there exist a b such that

$$\frac{1}{4}L \bullet X \geq b$$

$$X_{ii} = 1 \quad \forall i = 1, \dots, n$$

$$X \succeq 0$$

Observe that $X_{ii} = 1 \implies (e_i e_i^T) \bullet X = 1$. Rescale X by $\frac{1}{n}$.

Arrive at the formulation:

$$\left(\frac{n}{4b}L - I\right) \bullet X \geq 0$$

$$\left(n(e_i e_i^T) - I\right) \bullet X \geq 0 \quad \forall i = 1, \dots, n$$

$$\text{Tr}(X) = 1$$

$$X \succeq 0$$

Reformulating the MAXCUT relaxation

Let $A_j = n(e_j e_j^T) - I$, $j = 1, \dots, n$ and $A_{n+1} = \frac{n}{4b}L - I$.

$$\begin{aligned} A_j \bullet X &\geq 0 \quad \forall j = 1, \dots, n \\ A_{n+1} \bullet X &\geq 0 \\ \text{Tr}(X) &= 1 \\ X &\succeq 0 \end{aligned}$$

Matrix MW algorithm for
MAXCUT (Large-Margin)

Define $\rho = \max_j \|A_j\|$, $\eta = -\ln(1 - \epsilon)$, $\epsilon > 0$, $\delta > 0$.

Initialize $W^{(0)} = I_n$.

While $\exists j$ s.t. $A_j \bullet X^{(t)} < -\delta$:

① Find the index j such that $A_j \bullet X^{(t)} < -\delta$

② Obtain the payoff $M^{(t)} = \frac{A_j}{\rho}$

③ Update the weight matrix $W^{(t)}$:

$$W^{(t+1)} = \exp \left(\eta \sum_{\tau=1}^t M^{(\tau)} \right)$$

Toy MAXCUT problem

Number of nodes: 4. $b^* = 16$.

ϵ, δ	0.1	0.01	0.001	0.0001
T	20	328	3465	34827
$ b^* - b $	0.4595	0.0482	0.0039	0.0002
$\frac{ b^* - b }{ b^* }$	2.88×10^{-2}	3.01×10^{-3}	2.44×10^{-4}	1.25×10^{-5}

Table 1: toy example

MAXCUT problem with 10 vertices

Number of nodes: 10. $b^* = 80$.

ϵ, δ	0.1	0.01	0.001	0.0001
T	19	633	7230	73228
$ b^* - b $	1.9107	0.1908	0.0204	0.0011
$\frac{ b^* - b }{ b^* }$	2.39×10^{-2}	2.39×10^{-3}	2.55×10^{-4}	1.38×10^{-5}

Table 2: 10 nodes example

MAXCUT problem with 100 vertices

Number of nodes: 100. $b^* = 8190$.

ϵ, δ	0.1	0.01	0.001	0.0001
T	2777	49971		
$ b^* - b $	3.2430	0.7318		
$\frac{ b^* - b }{ b^* }$	3.96×10^{-4}	8.94×10^{-5}		

Table 3: 100 nodes example

Conclusions

- The MW algorithm can be used to solve LP and SDP feasibility problems of certain form.
- Use the MW algorithm to solve Linear Classification problems.
- Matrix MW algorithm performance depends on ϵ and δ . (Parameter sensitive).

References

- S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: A meta-algorithm and applications, *Theory of Computing*, 8 (2012), pp 121-164.
- S. Arora and S. Kale. A combinatorial, primal-dual approach to semidefinite programs. In *Journal of the ACM*, 63 (2016).