“MULTIPLE WEIGHTS UPDATE ALGORITHM”

CS 602 PAPER PRESENTATION BY

210050081 LOKESH
210050090 RAHUL DEEPAK
210050160 RAJA GOPAL

INTRODUCTION AND HISTORY

INTRODUCTION

- The Multiplicative Weights Update (MWU) method is a simple and powerful algorithm that can be used to solve a wide range of optimization problems.
- This method is an algorithmic technique which maintains a distribution on a certain set of interest and updates it iteratively by multiplying the probability mass of elements by suitably chosen factors based on feedback obatined by running another algorithm on the distribution.
- The algorithm can be used to solve a variety of problems, including online learning, game theory, and optimization.
- The MWU method has been shown to have strong theoretical guarantees, including a logarithmic regret bound, which ensures that the algorithm performs well even in the worst-case scenario.

HISTORY

- An algorithm similar in favor to the Multiplicative Weights algorithm were proposed in game theory in the early fifties . Following Brown , this algorithm was called \Fictitious Play"
- At each step each player observes actions taken by his opponent in previous stages, updates his beliefs about his opponents' strategies, and chooses myopic pure best responses against these beliefs.
- Grigoriadis and Khachiyan showed how a randomized variant of “Fictitious Play" (multiplicative weights algorithm.) can solve two player zero-sum games efficiently.
- when the player gives higher weight to the strategies which pay of better, and chooses her strategy using these weights rather than choosing the myopic best response strategy.
- In addition to machine learning and geometry, the MWU method has been used in game theory, Linear programming, convex programming and Heuristics in NP-Hard problems.

MULTIPLE WEIGHTS UPDATE ALGORITHM

ABOUT THE ALGORITHM

The proposed algorithm involves a basic model with the following elements

1. A decision maker is presented with n options and must repeatedly select a decision to receive a corresponding payoff.
2. The decision maker's overarching objective is to achieve a total payoff that is equivalent to the maximum possible profit with hindsight.
3. Initially, the best decision is unknown.
4. This objective can be accomplished by assigning weights to each decision and choosing a decision randomly based on the probabilities proportional to the weights. In each subsequent round, the weights are modified by multiplying them with factors that are dependent on the actual payoff received from the selected decision in that round.

WEIGHTED MAJORITY ALGORITHM

Weighted majority algorithm

Initialization: Fix an $\varepsilon \leq \frac{1}{2}$. For each expert i, associate the weight $w_i{}^{(1)} := 1$.
For $t = 1, 2, \ldots, T$:

1. Make the prediction that is the weighted majority of the experts' predictions based on the weights $w_1{}^{(t)}, \ldots, w_n{}^{(t)}$. That is, predict "up" or "down" depending on which prediction has a higher total weight of experts advising it (breaking ties arbitrarily).

2. For every expert i who predicts wrongly, decrease his weight for the next round by multiplying it by a factor of $(1 - \varepsilon)$:

$$w_i{}^{(t+1)} = (1 - \varepsilon) w_i{}^{(t)} \qquad \text{(update rule)}. \tag{2.1}$$

- **Consider the following problem. we are trying to invest in a certain stock. For simplicity, think of the price movements as increasing or decreasing only(up/down).Each day, we try to predict whether the price will go up or down.**

- **In our predictions, we are allowed to watch the predictions of n experts. The algorithm we present should limit the mistakes we make to roughly equal to the number of mistakes the best expert can do or less than it. We try to present the algorithm as explained in the previous slide by using weights and factors.**

Theorem 1. *After T steps, let $m_i^{(T)}$ be the number of mistakes of expert i and $m^{(T)}$ be the number of mistakes our algorithm has made. Then we have the following bound for every i:*

$$m^{(T)} \leq \frac{2 \ln n}{\varepsilon} + 2(1+\varepsilon) m_i^{(T)}.$$

In particular, this holds for i which is the best expert, i.e. having the least $m_i^{(T)}$.

PROOF: A simple induction shows that $w_i^{(t+1)} = (1-\varepsilon)^{m_i^{(t)}}$. Let $\Phi^{(t)} = \sum_i w_i^{(t)}$ ("the potential function"). Thus $\Phi^{(1)} = n$. Each time we make a mistake, the weighted majority

of experts also made a mistake, so at least half the total weight decreases by a factor $1-\varepsilon$. Thus, the potential function decreases by a factor of at least $(1-\varepsilon/2)$:

$$\Phi^{(t+1)} \leq \Phi^{(t)} \left(\frac{1}{2} + \frac{1}{2}(1-\varepsilon) \right) = \Phi^{(t)}(1-\varepsilon/2).$$

Thus another simple induction gives $\Phi^{(T+1)} \leq n(1-\varepsilon/2)^{m^{(T)}}$. Finally, since $\Phi_i^{(T+1)} \geq w_i^{(T+1)}$ for all i, the claimed bound follows by comparing the above two expressions and using the fact that $-\ln(1-\varepsilon) \leq \varepsilon + \varepsilon^2$ since $\varepsilon < \frac{1}{2}$. □

THE MULTIPLICATIVE WEIGHTS ALGORITHM

MULTIPLICATIVE WEIGHTS ALGORITHM

- *Suppose the set or outcomes may not necessarily be binary and could be infinite. Now we choose an expert and use his advice. We suffer the cost of action recommended by the expert we choose. Instead of a binary advice from the expert, they provide us with an course of action.*
- We can now set up some notation. Let t = 1,2,. .,T denote the current round and **i** be a generic expert. In each round t, we select a distribution $p^{(t)}$over the set of experts, and select an expert i randomly from it and use his course of action. At this point, the costs of all actions recommended by the expert are revealed in the form of vector $m^{(t)}$ such that expert i incurs cost $m_i^{(t)}$
- The expected cost of the algorithm for choosing the distribution is $p^{(t)}$

$$E_{i \epsilon p^{(t)}}\left[m_i^{(t)}\right] = \mathbf{m^{(t)}} \cdot \mathbf{p^{(t)}}$$

The total expected cost over all rounds is therefore $\sum_{t=1}^{T} \mathbf{m^{(t)}} \cdot \mathbf{p^{(t)}}$. Just as before our goal is to design an algorithm which achieves a total expected cost not too much more than the cost of the best expert i.e

$$min_i \sum_{t=1}^{T} m_i^{(t)}$$

Multiplicative Weights algorithm

Initialization: Fix an $\varepsilon \leq \frac{1}{2}$. For each expert i, associate the weight ${w_i}^{(t)} := 1$.
For $t = 1, 2, \ldots, T$:

1. Choose expert i with probability proportional to his weight ${w_i}^{(t)}$. I.e., use the distribution $\mathbf{p}^{(t)} = \{{w_1}^{(t)}/\Phi^{(t)}, \ldots, {w_n}^{(t)}/\Phi^{(t)}\}$ where $\Phi^{(t)} = \sum_i {w_i}^{(t)}$.
2. Observe the costs of the experts $\mathbf{m}^{(t)}$.
3. Penalize the costly experts by updating their weights as follows: for every expert i,

$$ {w_i}^{(t+1)} = \begin{cases} {w_i}^{(t)}(1-\varepsilon)^{{m_i}^{(t)}} & \text{if } {m_i}^{(t)} \geq 0 \\ {w_i}^{(t)}(1+\varepsilon)^{-{m_i}^{(t)}} & \text{if } {m_i}^{(t)} < 0 \end{cases} $$

★ Our goal of the algorithm is to achieve total cost not too much more than the cost of the best expert.

Theorem 2. *In the given setup, the Multiplicative Weights algorithm guarantees that after T rounds, for any expert i, we have*

$$\sum_{t=1}^{T} \mathbf{m}^{(t)} \cdot \mathbf{p}^{(t)} \leq \sum_{t=1}^{T} m_i{}^{(t)} + \varepsilon \sum_{t=1}^{T} |m_i{}^{(t)}| + \frac{\ln n}{\varepsilon}.$$

PROOF: We use the following facts, which follow immediately from the convexity of the exponential function:

$$(1-\varepsilon)^x \leq (1-\varepsilon x) \quad \text{if } x \in [0,1]$$
$$(1+\varepsilon)^{-x} \leq (1-\varepsilon x) \quad \text{if } x \in [-1,0]$$

The proof is along the lines of the earlier one, using the potential function $\Phi^{(t)} = \sum_i w_i{}^{(t)}$.

Since $m_i{}^{(t)} \in [-1,1]$, using the facts above we have,

$$\begin{aligned}
\Phi^{(t+1)} &= \sum_i w_i{}^{(t+1)} \\
&= \sum_{i:\ m_i{}^{(t)} \geq 0} w_i{}^{(t)}(1-\varepsilon)^{m_i{}^{(t)}} + \sum_{i:\ m_i{}^{(t)} < 0} w_i{}^{(t)}(1+\varepsilon)^{-m_i{}^{(t)}} \\
&\leq \sum_i w_i{}^{(t)}(1-\varepsilon m_i{}^{(t)}) \\
&= \Phi^{(t)} - \varepsilon \Phi^{(t)} \sum_i m_i{}^{(t)} p_i{}^{(t)}
\end{aligned}$$

$$
\begin{aligned}
&= \Phi^{(t)} - \varepsilon\Phi^{(t)} \sum_i {m_i}^{(t)} {p_i}^{(t)} \\
&= \Phi^{(t)}(1 - \varepsilon \mathbf{m}^{(t)} \cdot \mathbf{p}^{(t)}) \\
&\leq \Phi^{(t)} \exp(-\varepsilon \mathbf{m}^{(t)} \cdot \mathbf{p}^{(t)}).
\end{aligned}
$$

Here, we used the fact that ${p_i}^{(t)} = {w_i}^{(t)}/\Phi^{(t)}$. Thus, by induction, after T rounds, we have

$$
\Phi^{(T+1)} \leq \Phi^{(1)} \exp(-\varepsilon \sum_{t=1}^{T} \mathbf{m}^{(t)} \cdot \mathbf{p}^{(t)}) = n \cdot \exp(-\varepsilon \sum_{t=1}^{T} \mathbf{m}^{(t)} \cdot \mathbf{p}^{(t)}).
$$

Furthermore, for every expert i,

$$
\Phi^{(T+1)} \geq {w_i}^{(T+1)} = (1-\varepsilon)^{\sum_{\geq 0} {m_i}^{(t)}} \cdot (1+\varepsilon)^{-\sum_{<0} {m_i}^{(t)}},
$$

where the subscripts "≥ 0" and "< 0" in the summations refer to the rounds t where ${m_i}^{(t)}$ is ≥ 0 and < 0 respectively. Now we get the desired bound by taking logarithms and simplifying as before. We used the facts that $\ln(\frac{1}{1-\varepsilon}) \leq \varepsilon + \varepsilon^2$ and $\ln(1+\varepsilon) \geq \varepsilon - \varepsilon^2$ for $\varepsilon \leq \frac{1}{2}$. □

REMARK: From the proof, it can be seen that the following multiplicative update rule:

$$
{w_i}^{(t+1)} = {w_i}^{(t)}(1 - \varepsilon {m_i}^{(t)})
$$

regardless of the sign of ${m_i}^{(t)}$, would also give the same bounds. Such a rule may be easier to implement.

APPROXIMATING LINEAR FEASIBILITY PROGRAMS ON CONVEX DOMAINS

CHECKING FEASIBILITY OF LP

•Now Lets consider the classic problem of checking the feasibility of a convex domain. Which boils down to,

$$\exists ? x \epsilon P : Ax \geq b \quad \dashrightarrow \quad (1)$$

- Here P denotes the convex domain where easy constraints are satisfied , like non negativity. A represents hard constraints to satisfy.
- We wish to design an algorithm that given an error parameter δ > 0, either solves the problem to an additive error of δ, i.e., finds an x ∈ P such that $A_i x > B_i - \delta$, or failing that, proves that the system is infeasible. Here, A_i is the i^th row of A.
- We assume the existence of an algorithm, called Oracle, which, given a probability vector p on the m constraints, solves the following feasibility problem:

$$\exists ? x \epsilon P : p^T Ax \geq p^T b \quad \dashrightarrow \quad (2)$$

WHAT ARE THE ADVANTAGES OF CONVERTING ?

- Well, we can call this an optimization's procedure because instead of checking all the inequalities we are only checking just a single inequality.
- We can guarantee the feasibility of equation 1 if there is a solution for equation 2,but we can also ay that if equation 2 is not feasible then equation 1 is also not feasible.
- We assume that the Oracle satisfies the following technical condition, which is necessary for deriving run time bound.
- An (l, ρ)-bounded Oracle, for parameters 0 ≤ l ≤ ρ, is an algorithm which given a probability vector p over the constraints, solves the feasibility equation (1). Furthermore, there is a fixed subset I ⊆ [m] of constraints such that whenever the Oracle manages to find a point x ∈ P satisfying (2), the following holds

$$\forall i \in I : A_i x - b_i \in [-l, \rho]$$

$$\forall i \in [m] \setminus I : A_i x - b_i \in [-\rho, l]$$

HOW TO VIEW THIS PROBLEM IN TERMS OF MULTIPLICATIVE WEIGHTS ALGORITHM?

- Typically, the Multiplicative Weights method is applied in the following manner. We let an expert represent each constraint in the problem, and the events correspond to points in the domain of interest (P). The penalty of the expert is made proportional to how well the corresponding constraint is satisfied on the point represented by an event.
- To map our general framework to this situation, we have an expert representing each of the m constraints. Events correspond to vectors x ∈ P. The loss of the expert corresponding to constraint *i* for event x is $\frac{1}{\rho}[A_i x - B_i]$ (so that the costs lie in the range [−1, 1]).
- In each round t, given a distribution over the experts (i.e. the constraints) $p^{(t)}$, we run the **Oracle** with $p^{(t)}$ .If the Oracle declares that there is no x ∈ P such that $p^{(t)}Ax \geq p^{(t)}b$, then we stop, because now $p^{(t)}$ is proof that the problem (1) is infeasible

HOW IS IT OPTIMISING?

- Let δ > 0 be a given error parameter. Suppose there exists an (l, ρ)-bounded Oracle for the feasibility problem (1). Assume that l ≤δ/2,Then there is an algorithm which either solves the problem up to an additive error of δ, or correctly concludes that the system is infeasible, making only $\mathcal{O}\left(\frac{l\rho\log(m)}{\delta^2}\right)$ to the Oracle which takes O(m) per call

PROOF:

Weights algorithm is specified to be $\mathbf{m}^{(t)} := \frac{1}{\rho}[\mathbf{A}\mathbf{x}^{(t)} - \mathbf{b}]$, we conclude that the expected cost in each round is non-negative:

$$\mathbf{m}^{(t)} \cdot \mathbf{p}^{(t)} = \frac{1}{\rho}[\mathbf{A}\mathbf{x}^{(t)} - \mathbf{b}] \cdot \mathbf{p}^{(t)} = \frac{1}{\rho}[{\mathbf{p}^{(t)}}^{\top}\mathbf{A}\mathbf{x} - {\mathbf{p}^{(t)}}^{\top}\mathbf{b}] \geq 0.$$

Let $i \in I$. Then Theorem 2 tells us that after T rounds,

$$\begin{aligned} 0 &\leq \sum_{t=1}^{T}\frac{1}{\rho}[\mathbf{A}_i\mathbf{x}^{(t)} - b_i] + \varepsilon\sum_{t=1}^{T}\frac{1}{\rho}|\mathbf{A}_i\mathbf{x}^{(t)} - b_i| + \frac{\ln m}{\varepsilon} \\ &= (1+\varepsilon)\sum_{t=1}^{T}\frac{1}{\rho}[\mathbf{A}_i\mathbf{x}^{(t)} - b_i] + 2\varepsilon\sum_{<0}\frac{1}{\rho}|\mathbf{A}_i\mathbf{x}^{(t)} - b_i| + \frac{\ln m}{\varepsilon} \\ &\leq (1+\varepsilon)\sum_{t=1}^{T}\frac{1}{\rho}[\mathbf{A}_i\mathbf{x}^{(t)} - b_i] + \frac{2\varepsilon\ell}{\rho}T + \frac{\ln m}{\varepsilon} \end{aligned}$$

Here, the subscript "< 0" refers to the rounds t when $\mathbf{A}_i\mathbf{x}^{(t)} - b_i < 0$. The last inequality follows because if $\mathbf{A}_i\mathbf{x}^{(t)} - b_i < 0$, then $|\mathbf{A}_i\mathbf{x}^{(t)} - b_i| \leq \ell$. Dividing by T, multiplying by ρ, and letting $\bar{\mathbf{x}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{x}^{(t)}$ (note that $\bar{\mathbf{x}} \in \mathcal{P}$ since $\mathcal{P}$ is a convex set), we get that

$$0 \;\leq\; (1+\varepsilon)[\mathbf{A}_i\bar{\mathbf{x}} - b_i] + 2\varepsilon\ell + \frac{\rho \ln(m)}{\varepsilon T}.$$

Now, if we choose $\varepsilon = \frac{\delta}{4\ell}$ (note that $\varepsilon \leq \frac{1}{2}$ since $\ell \geq \frac{\delta}{2}$), and $T = \lceil \frac{8\ell\rho\ln(m)}{\delta^2} \rceil$, we get that

$$0 \;\leq\; (1+\varepsilon)[\mathbf{A}_i\bar{\mathbf{x}} - b_i] + \delta \quad\Longrightarrow\quad \mathbf{A}_i\bar{\mathbf{x}} \;\geq\; b_i - \delta.$$

Reasoning similarly for $i \notin I$, we get the same inequality. Putting both together, we conclude that $\bar{\mathbf{x}}$ satisfies the feasibility problem (2.6) up to an additive δ factor, as desired. □

IN CONCAVE DOMAINS?

- Yes, we can use this algorithm in concave domains also

$$\exists x \in P : \forall i \in [m] : f_i(x) \geq 0 \quad \dashrightarrow \quad (3)$$

- Everything remains the same except for the definition of f_i : P →R We wish to satisfy this system approximately, up to an additive error of δ. Again, we assume the existence of an Oracle, which, when given a probability distribution p

$$\exists \mathbf{x} \in \mathcal{P} : \forall i \in [m] : \sum p_i f_i(\mathbf{x}) \geq 0 \quad \dashrightarrow \quad (4)$$

- Even the definition of (l, ρ)-bounded Oracle doesn’t change ,We can use the theorem which is similar to the theorem in convex domains, the way we prove this theorem is also similar

APPROXIMATE ORACLES

- Define a δ-approximate Oracle for the feasibility problem (1) to be one that solves the feasibility problem (2) up to an additive error of δ. That is, given a probability vector p on the constraints, either it finds an x ∈ P such that $p^{(t)}Ax \geq p^{(t)}b - \delta$ or it declares correctly that (2) is infeasible.

- This approximation still makes $\mathcal{O}\left(\frac{l\rho\log(m)}{\delta^2}\right)$ calls to the oracle(proof is similar)

.

- Can we optimize further?

Fractional Covering Problems and Fractional Packaging problems

FRACTIONAL COVERING PROBLEMS AND FRACTIONAL PACKAGING PROBLEMS

Fractional Covering Problems and Fractional Packaging problems

- In fractional covering problems, the framework is the same as above, with the crucial difference that the coefficient matrix A is such that Ax ≥ 0 for all x ∈ P, and b > 0. A δ-approximation solution to this system is an x ∈ P such that Ax ≥ (1 − δ)b.
- If we scale all the b_i's to 1,then we can assume the fractional covering problem can be solved by **(1, ρ)-bounded Oracle** using the run time complexity from the above approximation algorithm gives $O(\frac{\rho \log(m)}{\delta^2})$
- In fractional packaging problems, the framework is the same as above, with the crucial difference that the coefficient matrix A is such that Ax ≥ 0 for all x ∈ P, and b > 0. A δ-approximation solution to this system is an x ∈ P such that Ax ≥ (1 + δ)b.
- Similarly, fractional packaging problems can also be solved in $O(\frac{\rho \log(m)}{\delta^2})$.