Multiplicative Weight Update Method

From Wikipedia, the free encyclopedia

Multiplicative weight update method is a meta-algorithm. It 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 obtained by running another algorithm on the distribution".[1] It was discovered repeatedly in very diverse fields such as machine learning (AdaBoost, Winnow, Hedge), optimization (solving LPs), theoretical computer science (devising fast algorithm for LPs and SDPs), and game theory.

Contents

Name

"Multiplicative weights" implies the iterative rule used in algorithms derived from the Multiplicative Weight Update Method.[2] It is given with different names in the different fields where it was discovered or rediscovered.

History and Background

The earliest known version of this technique was in an algorithm named "Fictitious Play" which was proposed in game theory in the early 1950s. Grigoriadis and Khachiyan[3] applied a randomized variant of "Fictitious Play" to solve two-player zero-sum games efficiently using the multiplicative weights algorithm. In this case, player allocates higher weight to the actions that had a better outcome and choose his strategy relying on these weights. In machine learning, Littlestone applied the earliest form of the multiplicative weights update rule in his famous Winnow Algorithm, which is similar to Minsky and Papert's earlier perceptron learning algorithm.Later, he generalized the Winnow Algorithm to Weighted Majority Algorithm. Freund and Schapire followed his steps and generalized the Winnow Algorithm in the form of Hedge Algorithm.

The Multiplicative weights algorithm is also widely applied in computational geometry such as Clarkson's algorithm for linear programming (LP) with a bounded number of variables in linear time.[4][5] Later, Bronnimann and Goodrich employed analogous

methods to find Set Covers for hypergraphs with small VC dimension.[6]

In operation research and on-line statistical decision making problem field, the weighted majority algorithm and its more complicated versions have been found independently.

In computer science field, some researchers have previously observed the close relationships between multiplicative update algorithms used in different contexts. Young discovered the similarities between fast LP algorithms and Raghavan's method of pessimistic estimators for derandomization of randomized rounding algorithms; Klivans and Servedio linked boosting algorithms in learning theory to proofs of Yao's XOR Lemma; Garg and Khandekar defined a common framework for convex optimization problems that contains Garg-Konemann and Plotkin-Shmoys-Tardos as subcases.[7]

General Setup

A binary decision needs to be made based on n experts' opinions to attain an associated payoff. In the first round, all experts' opinions have the same weight. The decision maker will make the first decision based on the majority of the experts' prediction. Then, in each successive round, the decision maker will repeatedly update the weight of each expert's opinion depending on the correctness of his prior predictions. Real life examples includes predicting if it is rainy tomorrow or if the stock market will go up or go down.

Algorithm Analysis

Halving Algorithm[2]

Given a sequential game played between an Adversary and an Aggregator who is advised by N experts. The goal is for aggregator to make as few mistakes as possible. Assume there is an expert among N experts always gives the correct prediction. In the halving algorithm, only the consistent experts are retained. Experts who make mistake will all be dismissed. With the remaining experts, take a majority vote. Therefore, every time the Aggregator makes a mistake, at least half of the remaining experts are dismissed. Aggregator makes at most $\log_2(N)$ mistakes.[2]

Weighted Majority Algorithm[7][8]

Unlike halving algorithm which dismisses experts who have made mistakes, weighted majority algorithm discounts their advice. Given the same "expert advice" setup, suppose we have n decisions, and we need to select one decision for each loop. In each loop, every decision incurs a cost. All costs will be revealed after making the choice. The cost is 0 if the expert is correct, and 1 otherwise. this algorithm's goal is to limit its cumulative losses to roughly the same as the best of experts. The very first algorithm that makes choice based on majority vote every iteration does not work since the majority of the experts can be wrong consistently every time. The weighted majority algorithm corrects above trivial algorithm by keeping a weight of experts instead of fixing the cost at either 1 or 0.[7] This would make fewer mistakes compared to halving algorithm.

Initialization:

Fix an $\eta \leq 1/2$. For each expert, 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 $\mathbf{w}_1^t, \ldots, \mathbf{w}_n^t$. That is, making a binary choice depending on which prediction has a higher total we
2. For every expert i who predicts wrongly, decrease his weight for the next round by multiplying it by a factor of (1-η):

$$w_i^{t+1} = (1-\eta) w_i^t \qquad \text{(update rule)}$$

If $\eta = 0$, the weight of the expert's advice will remain the same. When η increases, the weight of the expert's advice will decrease. Note that some researchers fix $\eta = 1/2$ in weighted majority algorithm.

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:

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

In particular, this holds for i which is the best expert. Since the best expert will have the least m_i^T, it will give the best bound on the number of mistakes made by the algorithm as a whole.

Randomized Weighted Majority Algorithm[2][9]

Given the same setup with N experts. Consider the special situation where the proportions of experts predicting positive and negative, counting the weights, are both close to 50%. Then, there might be a tie. Following the weight update rule in weighted majority algorithm, the predictions made by the algorithm would be randomized. The algorithm calculates the probabilities of experts predicting positive or negatives, and then makes a random decision based on the computed fraction:

predict

$$f(x) = \begin{cases} 1 & \text{with probability } \frac{q_1}{W} \\ 0 & \text{otherwise} \end{cases}$$

where

$$W = \sum_i w_i = q_0 + q_1 .$$

The number of mistakes made by the Randomized Weighted Majority Algorithm is bounded as:

$$E\left[\#\text{mistakes of the learner}\right] \leq \alpha_\beta \left(\#\text{ mistakes of the best expert}\right) + c_\beta \ln(N)$$

where $\alpha_\beta = \dfrac{\ln(\frac{1}{beta})}{1-\beta}$ and $c_\beta = \dfrac{1}{1-\beta}$.

Note that only the learning algorithm is randomized. The underlying assumption is that the examples and experts' predictions are not random. The only randomness is the randomness where the learner makes his own prediction. In this randomized algorithm, $\alpha_\beta \to 1$ if $\beta \to 1$. Compared to weighted algorithm, this randomness halved the number of mistakes the algorithm is going to make.[10] However, it is important to note that in some research, people define $\eta = 1/2$ in weighted majority algorithm and allow $0 \leq \eta \leq 1$ in randomized weighted majority algorithm.[2]

Applications

Multiplicative Weights method is usually used to solve a constrained optimization problem. Let each expert be the constraint in the problem, and the events represent the points in the area of interest. The punishment of the expert corresponds to how well its corresponding constraint is satisfied on the point represented by an event.[1]

Solving Zero-Sum Games Approximately (Oracle Algorithm):[1][7][10]

Suppose we were given the distribution P on experts. Let A = payoff matrix of a finite 2-player zero-sum game, with n rows.

When the row player p_r uses plan i and the column player p_c uses plan j, the payoff of player p_c is $A\left(i,j\right) := A_{ij}$, assuming $A\left(i,j\right) \in [0,1]$.

If player p_r chooses action i from a distribution P over the rows, then the expected result for player p_c selecting action j is $A\left(P,j\right) = E_{i \in P}\left[A\left(i,j\right)\right]$.

Hence, in order to maximize $A\left(P,j\right)$, player p_c is should choose plan j. Similarly, the expected payoff for player p_l is $A\left(i,P\right) = E_{j \in P}\left[A\left(i,j\right)\right]$. Choosing plan i would minimize this payoff. By John Von Neumann's Min-Max Theorem, we obtain:

$$\min_P \max_j A\left(P,j\right) = \max_Q \min_i A\left(i,Q\right)$$

where P and i changes over the distributions over rows, Q and j changes over the columns.

Then, let λ^* denote the common value of above quantities, also named as the "value of the game". Let $\delta > 0$ be an error parameter. To solve the zero-sum game bounded by additive error of δ,

$$\lambda^* - \delta \leq \min_i A\left(i,q\right)$$
$$\max_j A\left(p,j\right) \leq \lambda^* + \delta$$

Therefore, there is an algorithm solving zero-sum game up to an additive factor of δ using O($\log_2(n)/\boldsymbol{\delta^2}$) calls to ORACLE, with an additional processing time of O(n) per call [10]

Machine Learning

In Machine Learning, Littlestone and Warmuth generalized the Winnow algorithm to the Weighted Majority algorithm.[11] Later, Freund and Schapire generalized it in the form of Hedge algorithm.[12] AdaBoost Algorithm formulated by Yoav Freund and Robert Schapire also employed the Multiplicative Weight Update Method.[7]

Winnow Algorithm [7]

Based on current knowledge in algorithms, multiplicative weight update method was first used in Littlestone's Winnow Algorithm. It is utilized in machine learning to solve a linear program.

Given m labeled examples $\left(a_1, l_1\right), \ldots, \left(a_m, l_m\right)$ where $a_j \in \mathbb{R}^n$ are feature vectors, and $l_j \in \{-1, 1\}$ are their labels.

The aim is to find non-negative weights such that for all examples, the sign of the weighted combination of the features matches its

labels. That is, require that $l_j a_j x \geq 0$ for all j. Without loss of generality, assume the total weight is 1 so that they form a distribution. Thus, for notational convenience, redefine a_j to be $l_j a_j$, the problem reduces to finding a solution to the following LP:

$$\forall j = 1, 2, \ldots, m : a_j x \geq 0,$$
$$1 * x = 1,$$
$$\forall i : x_i \geq 0.$$

This is general form of LP.

Hedge Algorithm [2]

Hedge Algorithm is similar to Weighted Majority Algorithm. However, their exponential update rules are different.[2] It is generally used to solve the problem of binary allocation in which we need to allocate different portion of resources into N different options. The loss with every option is available at the end of every iteration. The goal is to reduce the total loss suffered for a particular allocation. The allocation for the following iteration is then revised, based on the total loss suffered in the current iteration using multiplicative update.[13]

Analysis

Assume $\epsilon \leq 1$ and for $t \in [T]$, p^t is picked by Hedge. Then for all experts i,

$$\sum_{t \leq T} p^t m^t \leq \sum_{t \leq T} m^t + \frac{\ln(N)}{\epsilon} + \epsilon T$$

Initialization: Fix an $\eta \leq 12$. For each expert, associate the weight w_i^1 :=1 For t=1,2,...,T:

1. Pick the distribution $p_j^t = \frac{w_1^t}{\Phi t}$ where $\Phi t = \sum_i w_i^t$.
2. Observe the cost of the decision m^t.
3. Set
$$w_i^{t+1} = w_i^t * \exp(-\epsilon * m_i^t).$$

AdaBoost Algorithm[12]

This algorithm maintains a set of weights w^t over the training examples. On every iteration t, a distribution p^t is computed by normalizing these weights. This distribution is fed to the weak learner WeakLearn which generates a hypothesis h_t that (hopefully) has small error with respect to the distribution. Using the new hypothesis h_t, AdaBoost generates the next weight vector w^{t+1}. The process repeats. After T such iterations, the final hypothesis h_f is the output. The hypothesis h_f combines the outputs of the T weak hypotheses using a weighted majority vote.[12]

Input:
- Sequence of N labeled examples $(x_1, y_1),\ldots,(x_N, y_N)$
- Distribution D over the N examples
- Weak learning algorithm "'WeakLearn"'
- Integer T specifying number of iterations

Initialize the weight vector: $w_i^1 = D(i)$ for $i = 1,\ldots, N$.

Do for $t = 1,\ldots, N$

1. Set $p^t = \frac{w^t}{\sum_{i=1}^N w_i^t}$.
2. Call WeakLearn, providing it with the distribution p^t; get back a hypothesis $h_t : X \to$ [0,1].
3. Calculate the error of $h_t : \epsilon_t = \sum_{i=1}^N p_i^t \, | \, h_t(x_i)$.
4. Set $\beta_t = \frac{\epsilon_t}{1 - \epsilon_t}$.
5. Set the new weight vector to be $w_i^{t+1} = w_i^t \beta_t^{1-|h_t(x_i)-y_i|}$.

Output the hypothesis:

$$f(x) = \begin{cases} 1 & \text{if} \sum_{t=1}^T \log(1/\beta_t) h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \log(1/\beta_t) \frac{q_1}{W} \\ 0 & \text{otherwise} \end{cases}$$

Solving Linear Programs Approximately[14]

Problem

Given a $m \times n$ matrix A and $b \in \mathbb{R}^n$, is there a x such that $Ax \geq b$?

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

Assumption

Using the oracle algorithm in solving zero-sum problem, with an error parameter $\epsilon > 0$, the output would either be a point x such that $Ax \geq b - \epsilon$ or a proof that x does not exist, i.e., there is no solution to this linear system of inequalities.

solution

Given vector $p \in \Delta_n$, solves the following relaxed problem

$$\exists ?x : p^{\mathsf{T}}Ax \geq p^{\mathsf{T}}b \quad (2)$$

If there exists a x satisfying (1), then x satisfies (2) for all $p \in \Delta_n$. The contrapositive of this statement is also true. Suppose if oracle returns a feasible solution for a p, the solution x it returns has bounded width $\max_i |(Ax)_i - b_i| \leq 1$. Therefore, if there is a solution to (1), then there is an algorithm that its output x satisfies the system (2) up to an additive error of 2ϵ. The algorithm makes at most $\frac{\ln(m)}{\epsilon^2}$ calls to a width-bounded oracle for the problem (2). The contrapositive stands true as well. The Multiplicative Updates is applied in the algorithm in this case.

Other Applications

Operations Research and On-line Statistical Decision Making [7]

In operations research and on-line statistical decision making problem field, the weighted majority algorithm and its more complicated versions have been found independently.

Computational Geometry [7]

Multiplicative weights algorithm is also widely applied in computational geometry such as Clarkson's algorithm for linear programming (LP) with a bounded number of variables in linear time.[4][5] Later, Bronnimann and Goodrich employed analogous methods to find Set Covers for hypergraphs with small VC dimension.[6]

Gradient Descent Method[1]

Matrix Multiplicative Weights Update[1]

Plotkin, Shmoys, Tardos Framework for Packing/Covering LPs [7]

Approximating Multi-commodity Flow Problems[7]

O (logn)- Approximation for many NP-hard problems[7]

Learning Theory and Boosting[7]

Hard-Core Sets and the XOR Lemma[7]

Hannan's Algorithm and Multiplicative weights[7]

Online Convex Optimization[7]

References

1. "Efficient Algorithms Using The Multiplicative Weights Update Method" (ftp://ftp.cs.princeton.edu/techreports/2007/804.pdf) (PDF). November 2007.
2. "The Multiplicative Weights Algorithm*" (https://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15859-f11/www/notes/lecture16.pdf) (PDF). Retrieved 2016-11-09.
3. "A sublinear-time randomized approximation algorithm for matrix games." (PDF). 1995.
4. KENNETH L. CLARKSON. *A Las Vegas algorithm for linear programming when the dimension is small.*, In Proc. 29th FOCS, pp. 452–456. IEEE Comp. Soc. Press, 1988.[doi:10.1109/SFCS.1988.21961] 123, 152.
5. KENNETH L. CLARKSON. *A Las Vegas algorithm for linear and integer programming when the dimension is small.*, J. ACM, 42:488–499, 1995. [doi:10.1145/201019.201036] 123, 152.
6. H. BRONNIMANN AND ¨ M.T. GOODRICH. *Almost optimal set covers in finite VC-dimension.*, Discrete Comput. Geom., 14:463–479, 1995. Preliminary version in 10th Ann. Symp. Comp. Geom. (SCG'94). [doi:10.1007/BF02570718] 123, 152

7. "The Multiplicative Weights Update Method: A Meta-Algorithm and Applications" (http://www.satyenkale.com/papers/mw-survey.pdf) (PDF). 2012.
8. "Lecture 8: Decision-making under total uncertainty: the multiplicative weight algorithm" (https://www.cs.princeton.edu/courses/archive/fall13/cos521/lecnotes/lec8.pdf) (PDF). 2013.
9. "COS 511: Foundations of Machine Learning" (http://www.cs.princeton.edu/courses/archive/spr06/cos511/scribe_notes/0330.pdf) (PDF). 2006-03-20.
10. "An Algorithmist's Toolkit" (https://ocw.mit.edu/courses/mathematics/18-409-topics-in-theoretical-computer-science-an-algorithmists-toolkit-fall-2009/lecture-notes/MIT18_409F09_scribe24.pdfformat=PDF) (PDF). 2009-12-08. Retrieved 2016-11-09.
11. DEAN P. FOSTER AND RAKESH VOHRA (1999). *Regret in the on-line decision problem*, p. 29. Games and Economic Behaviour
12. Yoav, Freund. Robert, E. Schapire (1996). *TA Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting**, p. 55. journal of computer and system sciences.
13. "Online Learning from Experts: Weighed Majority and Hedge" (http://www.shivani-agarwal.net/Teaching/E0370/Aug-2011/Lectures/20-scribe1.pdf) (PDF). Retrieved 2016-12-07.
14. "Fundamentals of Convex Optimization" (http://tcs.epfl.ch/files/content/sites/tcs/files/Lec2-Fall14-Ver2.pdf) (PDF). Retrieved 2016-11-09.

External links