CS655: Introduction to Linear Programming
Reading Project Assessment -1

Ananya Agrawal (200117) (ananya20@iitk.ac.in)
Anushka Panda (200174) (apanda20@iitk.ac.in)

September 2023

The reading for our team is from the second and the third chapter from the thesis on "Efficient Algorithms Using The Multiplicative Weights Update Method", by Satyen Kale et. al. .

1 The Problem and a glimpse into the Algorithm

The problem addressed can be thought of as following: say we are trying to predict the performance of a stock in the market, and make our decisions. For the simplest version of the problem, let us say we are just trying to determine a binary result: whether the stock moves up or down. In case the stock performs contrary to our prediction, say we lose 1 dollar. We are making predictions for the stock for T points of time in the future, and we are allowed to decide (and thus, change if necessary) our course of action (i.e. our prediction for the performance of the stock) at every point from $t = 1 \dots T$.
In order to make a good prediction, we are allowed to watch the predictions of n "experts" at every point in time. The ultimate aim is to make the best possible prediction at each $t = 1 \dots T$. That can be achieved by following the predictions of the "best performing experts". The catch is that the best expert cannot be decided upon at the beginning of time, since it depends on whether the predictions of the expert actually prove to be correct.
The multiplicative weights algorithm is based on the general version of the above problem. The latter involves costs ranging from $[-1, 1]$ for multiple kinds of events, which may be infinite in number. We take a probability distribution (say, $\mathbf{p}(t)$ for point t in time) over the n experts and try and minimise the overall expected cost for time over $t = 1 \dots T$. That is, we must bring the total expected cost as close as possible to the case where we had known all costs before hand, and had picked the best expert at each time t. At every point in time, we must pick an expert and then bear with the cost of their course of action, which nature will reveal. The costs for all the experts for a particular event's occuring will be stored in $\mathbf{m}(t)$, and thus, for time t, the expected cost would be $\mathbf{p}(t) \cdot \mathbf{m}(t)$.
What the multiplicative weights algorithm does is that:

- It assigns a constant weight (say 1) to all the experts at the beginning of the iterations.
- For every t from $1 \dots T$, it updates the weights of the experts such that, the costly experts lose by a weight varying exponentially with t, and the cheap experts gain a weight varying exponentially with t.
- In short, if the cost for an expert i at time t is $m_i(t)$, and the weight for expert i at time t is $w_i(t)$, then for an $\epsilon > 0$,
  - if $m_i(t) \geq 0$ then $w_i(t+1) = w_i(t)(1-\epsilon)^{m_i(t)}$
  - if $m_i(t) < 0$ then $w_i(t+1) = w_i(t)(1+\epsilon)^{-m_i(t)}$

for every expert, at every t, it updates the weight Finally, the thesis makes the problem even more general, by using vectors $\mathbf{v}$ from the unit sphere in $\mathbb{R}^n$, i.e. $\mathbb{S}^{n-1}$ in order to represent experts. The probability distribution $\mathcal{D}$ over the sphere, is now given by a probability density matrix, $\mathbf{P}^{(t)}$, which is nothing but $\mathbb{E}_{\mathbf{v} \in \mathcal{D}}[\mathbf{v}\mathbf{v}^\top]$, and the cost function is now a cost matrix, given by the matrix $\mathbf{M}^{(t)} \in \mathbb{R}^{n \times n}$. What we have to minimise is the overall $\mathbb{E}_{\mathbf{v} \in \mathcal{D}}[\mathbf{v}^\top \mathbf{M}^t \mathbf{v}$, which condenses to the quantity $\mathbf{M}^{(t)} \bullet \mathbf{P}^{(t)}$ for a given t.

NOTE: As of now, the meaning of the $\bullet$ symbol is not entirely clear to us in the context of this reading. Another paper by Kale et al. that is based on Fast Algorithms for Approximate Semidefinite Programming, refers to the $\bullet$ symbol as an "inner product" representation, hence we suspect it to be the same.

2 What was known before that time?

The first problem discussed in the previous section, is nothing but the problem that is solved by Littlestone and Warmuths' **Weighted Majority Algorithm**, 1994. The Multiplicative Weights algorithm in the thesis expanded the weighted majority algorithm to multiple events instead of binary, and also the cost for each event by an expert was made continuous. The algorithm also takes inspiration from the Freund and Schapire's *Hedge* algorithm.

However, the problem can itself be traced back to very old and famous works on economic game theory, specifically, to Brown's original "Fictitious Play" problem, 1950, wherein a player continuously updates their strategies in an online gaming after each move by the opponent; the most trivial case of that being that the player assumes that the opponent is picking their strategies from an empirical distribution based on the played strategies' frequencies. It is also mentioned in the thesis that the 1995 paper by Grigoriadis and Khachiyan actually showed a randomized variation of the "Fictitious Play", which can now be attributed to the same multiplicative weights algorithm the reading is based on.
In Machine Learning, the problem and the algorithm, are also quite popular, although they had been derived from independent methods. Examples include

- The Winnow Algorithm by Nick Littlestone that involves using weights updation in order to build a classifier, a generalised version of which is the above mentioned Weighted Majority algorithm.
- Cover, Foster and Vohra, and Blums' surveys, that discuss concepts like online regret games, a generalised Weighted majority Algorithm, etc. .
- Many other works correlating different kinds of programming with the online prediction problem have also been discussed in the thesis.

A paper by Tsuda, Rätsch and Warmuth had independently mentioned a matrix based exponential weight updation method in 2005. Also, some of the proofs, for instance the proof of the time complexity to the solution of zero-sum games via the Multiplicative Weights algorithm from the thesis, had also been independently proven before by Freund and Schapire, albeit by the KL Divergence method.

3 Relevance and Impact

As is evident from the above section, a lot of work had already been done for the above problem as well as algorithm, whether it be economic game theory, or machine learning, or solving programming problems (linear, semidefinite etc.). However, the problems had not been generalised enough to be expressed such that, just an updation of the variables and parameters would generate each of the above mentioned problems.
Some of the impacts of the above generalisation include:

- **Approximate solution of zero-sum games**: The trick is to put the row-player as the experts, and the column player as the events. Let $\mathbf{A}(i, \mathbf{q}) := \mathbb{E}_{j \in \mathbf{q}}[A(i, j)$, where $\mathbf{q}$ is a probability distribution on the columns. $\mathbf{A}(\mathbf{p}, j)$ is the equivalent for the rows. With von Neumann's minimax theorem and the above description of the problem, we get to find a $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{q}}$ for the above setting, such that they have an error parameter, say $\delta > 0$ with respect to the common solution from the minimax theorem. The complexity of the algorithm comes out to be $O(\frac{\log n}{\delta^2})$.
- **Linear Feasibility programs on Convex domains**: For a linear program to find an $\mathbf{x} \in \mathcal{P}$ st. $\mathbf{Ax} \geq \mathbf{b}$, where $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{x} \in \mathbb{R}^n$ and $\mathcal{P}$ a convex set in dimension n; given a probability distribution $\mathbf{p}$ over the m constraints, we find the solution to the following linear program:

  $$x \in \mathcal{P}, \mathbf{p}^\top \mathbf{Ax} \geq \mathbf{p}^\top \mathbf{b}$$

  In case the algorithm is unable to find an appropriate solution to the above problem, the problem is declared infeasible. Taking the probability vector over the constraints as the distribution over the experts, and the cost vector to be $\frac{1}{\rho}(\mathbf{Ax} - \mathbf{b})$ with the "width" factors ρ, l (more in the thesis), the algorithm solves the problem in with a runtime complexity of $O(\frac{l\rho \log m}{\delta^2})$. The same concept has been demonstrated to solve the problems for
  - Concave constraints (a generalisation of linear constraints)
  - Fractional Packing problems
  - Fractional Covering problems

4 Future research

In the future we would like to

- Compare the runtime of the various algorithms independently derived but mentioned in the thesis along with the multiplicative weights algorithm.
- Deep dive into the further applications of the Matrix Multiplicative Weights algorithm. The chapters covering those applications could not be covered in the current report due to resource constraints.