princeton univ. F'14 cos 521: Advanced Algorithm Design

Lecture 10: Applications of multiplicative weight updates: LP solving, Portfolio Management

Lecturer: *Sanjeev Arora* Scribe:

Today we see how to use the multiplicative weight update method to solve other problems. In many settings there is a natural way to make local improvements that "make sense."The multiplicative weight updates analysis from last time (via a simple potential function) allows us to understand and analyse the net effect of such sensible improvements. (Formally, what we are doing in many settings is analysing an algorithm called *gradient descent* which we'll encounter more formally later in the course.)

1 Solving systems of linear inequalities

We encountered systems of linear inequalities in Lecture 6. Today we study a version that seems slightly more restricted but is nevertheless as powerful as general linear programming. (Exercise!)

$$\begin{aligned} &\text{system 1} \\ a_1 \cdot x &\geq b_1 \\ a_2 \cdot x &\geq b_2 \\ &\vdots \\ a_m \cdot x &\geq b_m \\ x_i &\geq 0 \qquad \forall i = 1, 2, \ldots, n \\ \sum_i x_i &= 1. \end{aligned}$$

In your high school you learnt the "graphical"method to solve linear inequalities, and as we discussed in Lecture 6, those can take $m^{n/2}$ time. Here we design an algorithm that, given an error parameter $\varepsilon > 0$, runs in $O(mL/\varepsilon)$ time and either tells us that the original system is infeasible, or gives us a solution x satisfying the last two lines of the above system, and

$$a_j \cdot x \geq b_j - \varepsilon \qquad \forall j = 1, \ldots, m.$$

(Note that this allows the possibility that the system is infeasible *per se* and nevertheless the algorithm returns such an approximate solution. In that case we have to be happy with the approximate solution.) Here L is an instance-specific parameter that will be clarified below; roughly speaking it is the maximum absolute value of any coefficient. (Recall that the dependence would need to be poly($\log L$) to be considered polynomial time. We will study such a method later on in the course.)

What is a way to certify to somebody that the system is infeasible? The following is *sufficient*: Come up with a system of *nonnegative* weights $w_1, w_2, \ldots, w_m$, one per inequality, such that the following linear program has a negative value:

$$\begin{aligned} &\text{SYSTEM 2} \\ &\max \sum_j w_j(a_j \cdot x - b_j) \\ &x_i \geq 0 \qquad \forall i = 1, 2, \ldots, n \\ &\sum_i x_i = 1. \end{aligned}$$

Note: the w_j's are fixed constants. So this linear program has only two nontrivial constraints (not counting the constraints $x_i \geq 0$) so it is trivial to find a solution quickly, as we saw in class.

EXAMPLE 1 The system of inequalities $x_1 + x_2 \geq 1, x_1 - 5x_2 \geq 5$ is infeasible when combined with the constraints $x_1 + x_2 = 1, x_1 \geq 0, x_2 \geq 0$ since we can multiply the first inequality by 5 and the second by 1 and add to obtain $6x_1 \geq 10$. Note that $6x_1 - 10$ cannot take a positive value when $x_1 \leq 1$.

This method of certifying infeasibility is eminently sensible and the weighting of inequalities is highly reminiscent of the weighting of experts in the last lecture. So we can try to leverage it into a precise algorithm. It will have the following guarantee: (a) Either it finds a set of nonnegative weights certifying infeasibility or (b) It finds a solution $x^{(f)}$ that approximately satisfies the system, in that $a_j \cdot x - b_j \geq -\varepsilon$. Note that conditions (a) and (b) are not disjoint; if a system satisfies both conditions, the algorithm can do either (a) or (b).

We use the meta theorem on MW (Theorem 2) from Lecture 8, where experts have positive or negative costs (where negative costs can be seen as payoffs) and the algorithm seeks to minimize costs by adaptively decreasing the weights of experts with larger cost. The meta theorem says that the algorithm's payoff over many steps tracks —within $(1 + \varepsilon)$ multiplicative factor—the cost incurred by the best player, plus an additive term $O(\log n/\varepsilon)$.

We identify m "experts,"one per inequality. We maintain a weighting of experts, with ${w_1}^{(t)}, {w_2}^{(t)}, \ldots, {w_m}^{(t)}$ denoting the weights at step t. (At $t = 0$ all weights are 1.) Solve SYSTEM 2 using these weights. If it turns out to have a negative value, we have proved the infeasibility of SYSTEM 1 and can HALT right away. Otherwise take any solution, say $x^{(t)}$, and think of it as imposing a "cost "of ${m_j}^{(t)} = a_i \cdot x^{(t)} - b_i$ on the jth expert. (In particular, the first line of SYSTEM 2 is merely —up to scaling by the sum of weights— the expected cost for our MW algorithm, and it is positive.) Thus the MW update rule will update the experts' weights as:

$${w_j}^{(t+1)} \leftarrow {w_j}^{(t)}(1 - \eta\ {m_j}^{(t)}).$$

We continue thus for some number T of steps and if we never found a certificate of the infeasibility of SYSTEM 1 we output the solution $x^{(f)} = \frac{1}{T}(x^{(1)} + x^{(2)} + \cdot + x^{(T)})$, which is the

average of all the solution vectors found at various steps. Now let L denote the maximum possible absolute value of any $a_i \cdot x - b$ subject to the final two lines of SYSTEM 2.

CLAIM: *If $T > L^2 \log n/\varepsilon^2$ then $x^{(f)}$ satisfies $a_j \cdot x^{(f)} - b_j \geq -\varepsilon$ for all j.*

The proof involves the MW meta theorem which requires us to rescale (multiplying by $1/L$) so all costs lie in $[-1, 1]$ and setting $\varepsilon = \sqrt{\log n/T}$.

We wish to make T large enough so that the per-step additive error $\sqrt{\log n/T} < \varepsilon/L$, which implies $T > L^2 \log n/\varepsilon^2$.

Then we can reason as follows: (a) The expected per-step cost of the MW algorithm was positive (in fact it was positive in each step). (b) The quantity $a_j \cdot x^{(f)} - b_j$ is simply the average cost for expert j per step. (c) The total number of steps is large enough that our MW theorem says that (a) cannot be ε more than (b).

Here is another intuitive explanation that suggests why this algorithm makes sense independent of the experts idea. Vectors $x^{(1)}, x^{(2)}, \ldots, x^{(T)}$ represent simplistic attempts to find a solution to SYSTEM 1. If $a_i \cdot x^{(t)} - b_i$ is positive (resp., negative) this means that the jth constraint was satisfied (resp., unsatisfied) and thus designating it as a cost (resp., reward) ensures that the constraint is given less (resp., more) weight in the next round. Thus the multiplicative update rule is a reasonable way to search for a weighting of constraints that gives us the best shot at proving infeasibility.

Remarks: See the AHK survey on multiplicative weights for the history of this algorithm, which is actually a quantitative version of an older algorithm called *Lagrangian relaxation*.

1.1 Duality Theorem

The duality theorem for linear programming says that our method of showing infeasibility of SYSTEM 1 —namely, show for some weighting that SYSTEM 2 has negative value–is not just sufficient but also *necessary*.

This follows by imagining letting ε go to 0. If the system is infeasible, then there is some ε_0 (depending upon the number of constraints and the coefficient values) such that there is no ε-close solution with the claimed properties of $x^{(f)}$ for $\varepsilon < \varepsilon_0$. Hence at one of the steps we must have failed to find a positive solution for SYSTEM 2.

We'll further discuss LP duality in a later lecture.

2 Portfolio Management

Now we return to a a more realistic version of the stock-picking problem that motivated our MW algorithm. (You will study this further in a future homework.) There is a set of n stocks (e.g., the 500 stocks in S& P 500) and you wish to manage an investment portfolio using them. You wish to do at least as well as the best stock in hindsight, and also better than *index* funds, which keep a fixed proportion of wealth in each stock. Let ${c_i}^{(t)}$ be the price of stock i at the end of day t.

If you have ${P_i}^{(t)}$ fraction of your wealth invested in stock i then on the tth day your portfolio will rise in value by a multiplicative factor $\sum_i {P_i}^{(t)} {c_i}^{(t)} / {c_i}^{(t-1)}$. Looks familiar?

Let ${r_i}^{(t)}$ be shorthand for ${c_i}^{(t)} / {c_i}^{(t-1)}$.

If you invested all your money in stock i on day 0 then the rise in wealth at the end is

$$\frac{{c_i}^{(T)}}{{c_i}^{(0)}} = \prod_{t=0}^{T-1} {r_i}^{(t)}.$$

Since $\log ab = \log a + \log b$ this gives us the idea to set up the MW algorithm as follows. We run it by looking at n imagined experts, each corresponding to one of the stocks. The payoff for expert i on day t is $\log {r_i}^{(t)}$. Then as noted above, the total payoff for expert i over all days is $\sum_t \log {r_i}^{(t)} = \log({c_i}^{(T)}/{c_i}^{(0)})$. This is simply the log of the *multiplicative factor* by which our wealth would increase in T days if we had just invested all of it in stock i on the first day. (This is the jackpot we are shooting for: imagine the money we could have made if we'd put all our savings in Google stock on the day of its IPO.)

Our algorithm plays the canonical MW strategy from last lecture with a suitably small η and with the probability distribution $P^{(t)}$ on experts at time t being interpreted as follows: ${P_i}^{(t)}$ is the fraction of wealth invested in stock i at the start of day t. Thus we are no longer thinking of picking one expert to follow at each time step; the distribution on experts is the way of splitting our money into the n stocks. In particular on day t our portfolio increases in value by a factor $\sum_i {P_i}^{(t)} \cdot r^{(t)}$.

Note that we are playing the MW strategy that involves maximising payoffs, not minimizing costs. (That is, increase the weight of experts if they get positive payoff; and reduce weight in case of negative payoff.) The MW theorem says that the total payoff of the MW strategy, namely,
$\sum_t \sum_i {P_i}^{(t)} \cdot \log {r_i}^{(t)}$, is at least $(1-\varepsilon)$ times the payoff of the best expert provided T is large enough.

It only remains to make sense of the total payoff for the MW strategy, namely, $\sum_t \sum_i {P_i}^{(t)} \cdot \log {r_i}^{(t)}$, since thus far it is just an abstract quantity in a mental game that doesn't make sense *per se* in terms of actual money made.

Since the logarithm is a concave function (i.e. $\frac{1}{2}(\log x + \log y) \leq \log \frac{x+y}{2}$) and $\sum_i {P_i}^{(t)} = 1$, simple calculus shows that

$$\sum_i {P_i}^{(t)} \cdot \log r^{(t)} \leq \log(\sum_i {P_i}^{(t)} \cdot r^{(t)}).$$

The right hand side is exactly the logarithm of the rise in value of the portfolio of the MW strategy on day t. Thus we conclude that the total payoff over all days lower bounds the sum of the logarithms of these rises, which of course is the log of the ratio (final value of the portfolio)/(initial value).

All of this requires that the number of steps T should be large enough. Specifically, if $\left|\log {r_i}^{(t)}\right| \leq 1$ (i.e., no stock changes value by more than a factor 2 on a single day) then the total difference between the desired payoff and the actual payoff is $\sqrt{\log n/T}$ times $\max_i \sum_t \left|\log {r_i}^{(t)}\right|$, as noted in Lecture 8. This performance can be improved by other variations of the method (see the paper by Hazan and Kale). In practice this method doesn't work very well; we'll later explore a better algorithm.

REMARK: One limitation of this strategy is that we have ignored trading costs (ie dealer's commisions). As you can imagine, researchers have also incorporated trading costs

in this framework (see Blum and Kalai). Perhaps the bigger limitation of the MW strategy is that it assumes *nothing* about price movements whereas there is a lot known about the (random-like) behavior of the stock market. Traditional portfolio management theory assumes such stochastic models, and is more akin to the decision theory we studied two lectures ago. But stochastic models of the stock market fail sometimes (even catastrophically) and so ideally one wants to combine the stochastic models with the more pessimistic viewpoint taken in the MW method. See the paper by Hazan and Kale. See also a recent interesting paper by Abernathy et al. that suggests that the standard stochastic model arises from optimal actions of market players.

Thomas Cover was the originator of the notion of managing a portfolio against an adversarial market. His strategy is called *universal portfolio.*

BIBLIOGRAPHY

1. A. Blum and A. Kalai. *Efficient Algorithms for Universal Portfolios.* J. Machine Learning Research, 2002.

2. E. Hazan and S. Kale. *On Stochastic and Worst-case Models for Investing.* Proc. NIPS 2009.

3. J. Abernethy, R. Frongillo, A. Wibisono. *Minimax Option Pricing Meets Black-Scholes in the Limit.* Proc. ACM STOC 2012.