

Dual update algorithms to solve certain LPs approximately¹

- In this lecture, we look at how a particular “primal-dual” algorithm can be used to solve certain LPs approximately. This methodology, also called the *multiplicative weight update* method, underlies many different optimization problems, and has applications in other areas such as learning and portfolio management. We focus on the *set cover* LP for illustrative purposes.
- Let’s recall the LP for set cover.

$$\text{lp} := \min \sum_{j=1}^m \mathbf{c}_j \mathbf{x}_j \quad (\text{Set Cover LP})$$

$$\sum_{j: e \in S_j} \mathbf{x}_j \geq 1, \quad \forall e \in U \quad (1)$$

$$1 \geq \mathbf{x}_j \geq 0, \quad \forall j = 1, \dots, m \quad (2)$$

Given an $\varepsilon \in (0, 1)$, our goal is to find a *feasible* $\mathbf{x} \in [0, 1]^m$ such that $\sum_{j=1}^m c_j \mathbf{x}_j \leq (1 + \varepsilon) \text{lp}$.

- The main idea behind this algorithm is to select dual variables y_e for each $e \in U$. However, the function of this dual variables in this algorithm is **not** to generate a large dual objective. Rather, the function of the dual variables is to *aggregate* all the $n = |U|$ constraints of the form (1) into one single constraint, and then solve the primal LP with only a single constraint. This is similar to the Lagrangean function idea which we saw long ago; there one actually moved this “single constraint” also to the objective.
- Fix $y_e \in [0, 1]$ variables for every $e \in U$. We call this vector $\mathbf{y} \in [0, 1]^n$. Then, consider the following “single constraint” LP

$$\text{lp}(\mathbf{y}) := \min \sum_{j=1}^m \mathbf{c}_j \mathbf{x}_j \quad (\text{Aggregated Set Cover LP})$$

$$\sum_{e \in U} y_e \left(\sum_{j: e \in S_j} \mathbf{x}_j \right) \geq \sum_{e \in U} y_e, \quad (3)$$

$$1 \geq \mathbf{x}_j \geq 0, \quad \forall j = 1, \dots, m \quad (4)$$

Since any feasible solution $\mathbf{x}$ to (Set Cover LP) is also a feasible solution to (Aggregated Set Cover LP), we get the following observation.

Observation 1. For any $\mathbf{y} \in [0, 1]^n$, if $\mathbf{x} \leftarrow \mathcal{O}(\mathbf{y})$, then we have $\mathbf{c}^\top \mathbf{x} \leq \text{lp}$.

¹Lecture notes by Deeparnab Chakrabarty. Last modified : 14th Feb, 2022
These have not gone through scrutiny and may contain errors. If you find any, or have any other comments, please email me at deeparnab@dartmouth.edu. Highly appreciated!

- **The Main Idea, qualitatively.** Given a dual vector $\mathbf{y} \in [0, 1]^n$, we first solve (Aggregated Set Cover LP) to obtain a solution $\mathbf{x}$ which has objective value $\leq \text{lp}$, but may not be feasible for (Set Cover LP). In the next bullet point we show how to do this, and we denote this algorithm as an *oracle* $\mathcal{O}$, and use the notation $\mathbf{x} \leftarrow \mathcal{O}(\mathbf{y})$. Next, we modify the dual vector $\mathbf{y}$ in a natural way : for elements e which are violated, we bump up the y_e value with the intuition that (Aggregated Set Cover LP) would tend to satisfy the e th constraint. Indeed, the “bump” would be a function of the violation. For elements e which are not violated, we bump down their y_e ’s since they seem safe. Once we do this, we again call the oracle to get a new primal solution $\mathbf{x}$. We then repeat the above steps. After T such rounds, we have many $\mathbf{x}_t$ ’s, each of which have LP objective value at most lp , and yet individually they may not be feasible for (Set Cover LP). What is quite interesting is that there is a systematic way for “bumping up/down” such that after a reasonable number of rounds, the *average* of all these $\mathbf{x}_t$ ’s are “close” to being feasible. And then if we scale up the average, then we get a truly feasible solution to (Set Cover LP) whose objective value is at most $(1 + \varepsilon)\text{lp}$.
- **Oracle.** Before we move on, let’s note that solving (Aggregated Set Cover LP) is quite easy. In particular, given $\mathbf{y}$, we can rewrite (Aggregated Set Cover LP) as

$$\min \sum_{j=1}^m \mathbf{c}_j \mathbf{x}_j \quad : \quad \sum_{j=1}^m \mathbf{w}_j \mathbf{x}_j \geq \beta, \quad \mathbf{x}_j \in [0, 1] \quad (5)$$

where $\beta := \sum_{e \in U} y_e$ and $\mathbf{w}_j := \sum_{e \in S_j} y_e$.

Now we observe that (5) is easy to solve. Rename the sets such that $\frac{c_1}{w_1} \leq \frac{c_2}{w_2} \leq \dots \leq \frac{c_m}{w_m}$. The optimum solution to (5) is obtained by setting $\mathbf{x}_j = 1$ for $1 \leq j \leq k$ where k is largest entry with $\sum_{j=1}^k \mathbf{w}_j \leq \beta$. We set $\mathbf{x}_{k+1} := \frac{1}{w_{k+1}} \cdot \left(\beta - \sum_{j=1}^k \mathbf{w}_j \right)$. The remaining $\mathbf{x}_j = 0$ for $j > k + 1$.

Exercise: 🍷 Prove that the vector $\mathbf{x} \in [0, 1]^m$ is the optimum solution to (5).

- **Feasibility Vector and Multiplicative Weight Update (MWU).** We are now ready to give details about the main idea. The algorithm proceeds in rounds. At the beginning of round t , we specify the dual variables $\mathbf{y}^{(t)} \in [0, 1]^n$ for each element in U . We then apply the oracle $\mathcal{O}(\mathbf{y}^{(t)})$ to obtain a solution $\mathbf{x}^{(t)} \in [0, 1]^m$. We then use $\mathbf{x}^{(t)}$ to obtain $\mathbf{y}^{(t+1)}$.

To describe the latter process, we need to define a “satisfiability” vector $\text{sat}^{(t)} \in [-1, +1]^n$ which indicates “how satisfied” element e is with respect to the current primal solution $\mathbf{x}^{(t)}$. More precisely,

$$\forall e \in U : \quad \text{sat}^{(t)}(e) := \frac{1}{d} \cdot \left(\sum_{j: e \in S_j} \mathbf{x}_j^{(t)} - 1 \right) \quad (6)$$

Here, d is the maximum *number* of sets an element e can be in. The reason for doing this is to make sure that the range of $\text{sat}^{(t)}(e) \in [-\frac{1}{d}, +1]$. In fact, we state this as an observation since it is going to be crucial.

Observation 2. For any e and any $\mathbf{x}^{(t)} \in [0, 1]^m$, the corresponding $\text{sat}^{(t)}(e)$ lies in $[-\frac{1}{d}, 1]$.

Proof. When $\mathbf{x}^{(t)} \equiv \mathbf{0}$, then the value of $\text{sat}^{(t)}(e) = -1/d$, and when $\mathbf{x}^{(t)} \equiv \mathbf{1}$, then the value of $\text{sat}_e^{(t)} = d_e/d \leq 1$. $\square$

We make another observation about the $\text{sat}^{(t)}$ vector which in plain English states that the $\mathbf{y}$ -linear combination for the satisfiabilities is non-negative.

Observation 3. For any t , $\sum_{e \in U} \mathbf{y}_e^{(t)} \text{sat}^{(t)}(e) \geq 0$.

Proof. The LHS is simply $\frac{1}{d} \cdot \left(\sum_{e \in U} \mathbf{y}_e^{(t)} \cdot \sum_{j: e \in S_j} \mathbf{x}_j^{(t)} - \sum_{e \in U} \mathbf{y}_e^{(t)} \right)$. Since $\mathbf{x}^{(t)} \leftarrow \mathcal{O}(\mathbf{y}^{(t)})$, we know that $\mathbf{x}^{(t)}$ satisfies (3). And thus, the parenthesized term is ≥ 0 . $\square$

Now we are ready to state the algorithm.

```

1: procedure MWU SET COVER LP SOLVER( $U, S_1, \dots, S_m$ ):
2:   Initialize  $\text{wt}^{(1)}(e) := 1$  for all  $e \in U$ .
3:    $\Phi^{(1)} := \sum_{e \in U} \text{wt}^{(1)}(e) = n$ ;  $\mathbf{y}_e^{(1)} = \frac{\text{wt}^{(1)}(e)}{\Phi^{(1)}}$ 
4:   for  $t = 1$  to  $T$  do:  $\triangleright$  The value of  $T$  will be set later.
5:     Obtain  $\mathbf{x}^{(t)} \leftarrow \mathcal{O}(\mathbf{y}^{(t)})$ .
6:     Obtain  $\text{sat}^{(t)}(e)$  for all  $e$  using (6).
7:      $\triangleright$  Update the wt and y vector as follows;  $\eta < 1$  is a parameter which will be set later.
8:     For all  $e \in U$ ,  $\text{wt}^{(t+1)}(e) := \text{wt}^{(t)}(e) \cdot (1 - \eta \cdot \text{sat}^{(t)}(e))$ 
9:      $\Phi^{(t+1)} := \sum_{e \in U} \text{wt}^{(t+1)}(e)$ 
10:     $\mathbf{y}_e^{(t+1)} = \frac{\text{wt}^{(t+1)}(e)}{\Phi^{(t+1)}}$ 
11:    Let  $\bar{\mathbf{x}} := \frac{1}{T} \sum_{t=1}^T \mathbf{x}^{(t)}$ .
12:    Return  $\mathbf{x}_{\text{alg}} := (1 + \varepsilon) \cdot \bar{\mathbf{x}}$ .

```

As you can see, the $\mathbf{y}$ -vector is in fact a *probability* distribution generated by “weights” on each element. If element e has low $\text{sat}^{(t)}(e)$, and in particular negative $\text{sat}^{(t)}(e)$ indicating the constraint for e is violated, then its weight is “bumped up”. Alternately, if element e has high $\text{sat}^{(t)}(e)$, then its weight is “bumped down”. Since $\eta < 1$ and $\text{sat}^{(t)}(e) \leq 1$, the weights always remain positive. This is important.

After running for T rounds, the final answer is a $(1 + \varepsilon)$ -multiplicative scaling of the *average* of the T different $\mathbf{x}^{(t)}$ ’s. One thing is immediate from [Observation 1](#).

Observation 4. $\mathbf{c}^\top \bar{\mathbf{x}} = \frac{1}{T} \left(\sum_{t=1}^T \mathbf{c}^\top \mathbf{x}^{(t)} \right) \leq \text{lp}$. Therefore, $\mathbf{c}^\top \mathbf{x}_{\text{alg}} \leq (1 + \varepsilon) \text{lp}$.

- **Analysis.** The crux of the analysis is in showing that $\mathbf{x}_{\text{alg}}$ is indeed feasible if η and T are set carefully. This in turn proceeds by showing that $\bar{\mathbf{x}}$ is “almost feasible”. Here is the main lemma. Note this immediately implies for $\varepsilon < 1$, which implies that $(1 + \varepsilon)(1 - \varepsilon/2) > 1$, the scaled version $\mathbf{x}_{\text{alg}}$ is feasible.

Lemma 1. Suppose $\eta := \frac{\varepsilon}{4}$ and $T := \frac{8d \ln n}{\varepsilon^2}$. Fix an element $e \in U$. Then, $\sum_{j:e \in S_j} \bar{x}_j \geq 1 - \frac{\varepsilon}{2}$.

Proof. The proof is a really slick argument. First note from the definition of $\bar{x}$ and $\text{sat}^{(t)}(e)$ that

$$\left(\sum_{j:e \in S_j} \bar{x}_j - 1 \right) = \frac{d}{T} \cdot \sum_{t=1}^T \text{sat}^{(t)}(e) \quad (7)$$

So we need to lower bound the RHS. In order to do so, we look at the potential function $\Phi^{(t)}$.

Claim 1. For any t , $\Phi^{(t+1)} \leq \Phi^{(t)}$. Thus, $\Phi^{(T+1)} \leq \Phi^{(1)} = n$.

Proof. By definition, $\text{wt}^{(t+1)}(e) = \text{wt}^{(t)}(e) \cdot (1 - \eta \cdot \text{sat}^{(t)}(e))$. Using the fact that $\mathbf{y}^{(t)}(e) = \Phi^{(t)} \cdot \text{wt}^{(t)}(e)$, we get

$$\text{wt}^{(t+1)}(e) = \text{wt}^{(t)}(e) \cdot (1 - \eta \mathbf{y}_e^{(t)} \text{sat}^{(t)}(e))$$

Adding over all $e \in U$ and using [Observation 3](#), we get the claim. $\square$

The above claim says that the $\Phi^{(T+1)}$ at the end of the algorithm is “small”. However, $\Phi^{(T+1)}$ is the *sum* of the $\text{wt}^{(T+1)}(e)$ over all $e \in U$, and these weights are positive. Therefore, $\Phi^{(T+1)}$ is strictly greater than the final weight of this particular element e under consideration. However, if e was violated by “too many” $\mathbf{x}^{(t)}$ ’s, then its weight would have been bumped pretty high. Since this weight is not too high ($\leq n$), we can argue that “most” $\mathbf{x}^{(t)}$ ’s satisfied e , and thus the average $\bar{x}$ also almost satisfies it. To make this rigorous, we need some analytic gimmickry, but the idea is precisely this. Let’s get to the details.

First, let us figure out $\text{wt}^{(T+1)}(e)$ at the end of the T for-loops. By definition this is

$$\text{wt}^{(T+1)}(e) = \prod_{t=1}^T (1 - \eta \cdot \text{sat}^{(t)}(e)) \quad (8)$$

Now we are going to use the following inequalities which is readily checked

$$\text{For } 0 \leq x \leq 1, (1 - \eta x) \geq (1 - \eta)^x; \quad \text{For } -1 \leq x \leq 0, (1 - \eta x) \geq (1 + \eta)^{-x} \quad (9)$$

Now, let $P \subseteq \{1, 2, \dots, T\}$ denote the t ’s with $\text{sat}^{(t)}(e) \geq 0$, and N denote the t ’s with $\text{sat}^{(t)}(e) < 0$. Then, substituting (9) in (8)

$$\text{wt}^{(T+1)}(e) \geq \prod_{t \in P} (1 - \eta)^{\text{sat}^{(t)}(e)} \cdot \prod_{t \in N} (1 + \eta)^{-\text{sat}^{(t)}(e)} \quad (10)$$

Since the LHS above is $< \Phi^{(T+1)} \leq n$, we get the RHS above is $\leq n$. Now, we take log to the base e on both sides to get

$$\ln n \geq \left(\sum_{t \in P} \text{sat}^{(t)}(e) \right) \ln(1 - \eta) - \left(\sum_{t \in N} \text{sat}^{(t)}(e) \right) \ln(1 + \eta) \quad (11)$$

At this point, suppose η was “very tiny” and suppose we could approximate $\ln(1 - \eta) \approx -\eta$ and $\ln(1 + \eta) \approx \eta$, then we would get $\ln n \geq -\eta \left(\sum_{t=1}^n \text{sat}^{(t)}(e) \right)$. Rearranging, we would get that $\frac{d}{T} \left(\sum_{t=1}^n \text{sat}^{(t)}(e) \right) \geq \frac{-d \ln n}{\eta T}$. So, for this “very tiny” value of η , if we chose $T = \frac{2d \ln n}{\eta \varepsilon}$, then using (7) we would obtain the proof of the lemma.

But how “very tiny” does this η need to be? Turns out, that η needs to be $< \varepsilon$ so that the errors in the above approximation do not dominate. And thus, the dependence of T on ε is inverse quadratic.

Let us now make the above precise. For that we state another helpful inequality which is easily verified using calculus (or visualized using Wolfram alpha).

$$\text{For } 0 < \eta < 1/2, \quad \ln(1 - \eta) \geq -(\eta + \eta^2); \quad \ln(1 + \eta) \geq \eta - \eta^2 \quad (12)$$

Substituting (12) in (11), we get

$$\begin{aligned} \ln n &\geq -(\eta + \eta^2) \left(\sum_{t \in P} \text{sat}^{(t)}(e) \right) - (\eta - \eta^2) \left(\sum_{t \in N} \text{sat}^{(t)}(e) \right) \\ &\geq -\eta(1 + \eta) \left(\sum_{t=1}^T \text{sat}^{(t)}(e) \right) - 2\eta^2 \sum_{t \in N} |\text{sat}^{(t)}(e)| \\ &\geq -2\eta \left(\sum_{t=1}^T \text{sat}^{(t)}(e) \right) - \frac{2\eta^2 T}{d} \end{aligned}$$

where in the last inequality we used $\eta \leq 1$, and [Observation 2](#). Rearranging, and using (7), we get

$$\left(\sum_{j: e \in S_j} \bar{x}_j - 1 \right) = \frac{d}{T} \cdot \sum_{t=1}^T \text{sat}^{(t)}(e) \geq -\frac{d \ln n}{2\eta T} - \eta$$

Substituting $\eta := \frac{\varepsilon}{4}$ and $T := \frac{8d \ln n}{\varepsilon^2}$, we get that $\sum_{j: e \in S_j} \bar{x}_j \geq 1 - \frac{\varepsilon}{2}$, proving the lemma. $\square$

- Therefore, we see that the set-cover LP can be solved to ε -accuracy in $O(d \ln n / \varepsilon^2)$ oracle calls. Each oracle call standalone may take $O(m)$ time, but one can be clever and amortize this cost to get an $O((n + m) \ln n / \varepsilon^2)$ running time. For constant ε , this is a “near linear” running time. Which is pretty good.

Notes

The idea described here is originally from the paper [2] by Plotkin, Shmoys, and Tardos. The exposition in this note heavily borrows from the beautiful survey [1] on the multiplicative weight update method by Arora, Hazan, and Kale. As can be expected there is nothing special about set-cover LP, and the above technique holds for a much general class of LPs. We refer the reader to the above two papers.

References

- [1] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.
- [2] S. Plotkin, D. Shmoys, and E. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Math. Oper. Res.*, 20:257–301, 1995.