12 MWU Algorithm Revisit and Its Applications

12.1 MWU as Mirror Descent

Recall the (projected) mirror descent algorithm on the N dimensional simplex:

- **Feasible Set**: Probability simplex $\Delta_N : \{(p_1, \ldots, p_N) \mid \sum p_i = 1, \forall\, i, p_i \geq 0\}$.
- **Mirror Map**: Negative entropy $\psi(p) = \sum_{i=1}^{N} p_i \ln p_i$.
- **Bregman Divergence**: $D_\psi(p, q) = \sum_{i=1}^{N} p_i \ln\left(\frac{p_i}{q_i}\right)$.

Denote $\nabla f(p^t)$ by ℓ^t. At iteration t, mirror descent solves:

$$p^{t+1} = \arg\min_{p \in \Delta_N} \left\{\eta \langle \ell^t, p\rangle + D_\psi(p, p^t)\right\}.$$

Substitute the KL divergence D_ψ into the objective function:

$$\begin{aligned}
&\min_{p \in \Delta_N} \left\{\eta \sum_{i=1}^{N} \ell_i^t p_i + \sum_{i=1}^{N} p_i \ln\left(\frac{p_i}{p_i^t}\right)\right\} \\
&= \min_{p \in \Delta_N} \left\{\sum_{i=1}^{N} (\eta \ell_i^t p_i + p_i \ln p_i - p_i \ln p_i^t)\right\}
\end{aligned}$$

Now introduce Lagrange multiplier λ for the constraint $\sum_i p_i = 1$ (constraints $p_i \geq 0$ hold trivially):

$$\mathcal{L}(p, \lambda) = \sum_{i=1}^{N} (\eta \ell_i^t p_i + p_i \ln p_i - p_i \ln p_i^t) + \lambda\left(1 - \sum_{i=1}^{N} p_i\right)$$

For each p_i, set $\frac{\partial \mathcal{L}}{\partial p_i} = 0$, which gives

$$\eta \ell_i^t + \ln p_i + 1 - \ln p_i^t - \lambda = 0$$

Rearrange terms:

$$\ln p_i = \ln p_i^t - \eta \ell_i^t - 1 + \lambda$$

Exponentiate both sides:

$$p_i = p_i^t \exp(-\eta \ell_i^t - 1 + \lambda)$$

The multiplier λ enforces $\sum p_i = 0$. So we should normalize p_i. Let $W_t = \sum_{j=1}^{N} p_j^t \exp(-\eta \ell_j^t)$. We have

$$p_i = p_i^t \exp(-\eta \ell_i^t)/W_t\,.$$

Substitute back into the iterative rule of mirror descent:

$$p_i^{t+1} = p_i^t \exp(-\eta \ell_i^t) \cdot \frac{1}{W_t} = \frac{p_i^t \exp(-\eta \ell_i^t)}{\sum_{j=1}^{N} p_j^t \exp(-\eta \ell_j^t)}$$

This gives the MWU update rule with weights:

$$w_i^{t+1} = w_i^t \exp(-\eta \ell_i^t)$$

From the convergence analysis of mirror descent, we have

$$\frac{1}{T}\sum_{t=1}^{T}\langle \ell^t, p^t\rangle - \min_{p\in\Delta_N}\frac{1}{T}\sum_{t=1}^{T}\langle \ell^t, p\rangle \le \frac{R}{\eta T} + \frac{\eta L^2}{2},$$

where $L = \max_t \|\ell^t\|_\infty$ and

$$R = \max_q D_\psi\Big(q, \frac{1}{N}\mathbf{1}_N\Big) = \ln N.$$

For $\eta = \sqrt{\frac{2\ln N}{TL^2}}$ and $T = 2L^2 \ln N/\varepsilon^2$, the regret satisfies:

$$\frac{1}{T}\sum_{t=1}^{T}\langle \ell^t, p^t\rangle - \min_{p\in\Delta_N}\frac{1}{T}\sum_{t=1}^{T}\langle \ell^t, p\rangle \le \sqrt{2L^2\ln N/T} \le \varepsilon.$$

12.2 Solving LP by MWU: Plotkin-Shmoys-Tardos Algorithm

An interesting application of MWU is the Plotkin-Shmoys-Tardos Algorithm for solving linear programmings.

The PST algorithm solves packing/covering LPs of the form:

- **Covering LP**:
$$\min_{\mathbf{x}\ge 0} \mathbf{c}^\top \mathbf{x} \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} \ge \mathbf{b}$$
- **Packing LP**:
$$\max_{\mathbf{x}\ge 0} \mathbf{c}^\top \mathbf{x} \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} \le \mathbf{b}$$

Solve an LP is equivalent to finding a feasible solution in a polytope, namely, deciding whether

$$\exists x? \text{ such that } Ax \le b.$$

Intuitively, each constraint can be viewed as an expert, where the tightest constraint corresponds to the best expert. We do not know which one is the best, but the regret analysis of MWU algorithm provides a **lower bound** of the tightest constraint.

At round t, we relax the constraints $Ax \le b$ as $\langle p^t, Ax - b\rangle \le 0$. If this is infeasible, then clearly the there is no solution in the original polytope. Otherwise, if we can find solutions in T rounds, the inequality

$$\frac{1}{T}\sum_{t=1}^{T}\langle \ell^t, p^t\rangle - \min_{p\in\Delta_N}\frac{1}{T}\sum_{t=1}^{T}\langle \ell^t, p\rangle \le \varepsilon$$

tells us each constraint $A_i x \le b_i$ is approximately feasible, i.e., $A_i x \le b_i + \varepsilon$. Here ℓ^t is defined as

$$\ell_i^t = A_i x^t - b_i,$$

where x^t is a feasible solution at round t, namely, $\langle p^t, Ax^t - b\rangle \le 0$.

We now introduce an example: deciding whether a bipartite graph has a perfect matching. Given a graph $G = (V, E)$ on n vertices and m edges, the following linear programming reformulate the perfect matching problem:

$$\begin{aligned} \text{find} \quad & x \in \mathbb{R}^m \\ \text{subject to} \quad & \sum_{e \in E} x_e = n \\ & \sum_{e: v \in e} x_e \leq 1 && \forall\, v \in V \\ & x_e \geq 0 && \forall\, e \in E \end{aligned}$$

A solution to the above linear feasibility program is called a *fractional perfect matching*. The question which arises very naturally is whether the "fractional" problem is equivalent to the original one? It basically asserts that the fractional bipartite matching polytope (defined in the linear programming description above) has only integral vertices. So we assume G is a bipartite graph here.

Our goal is to produce approximate answers, that is, for an $\varepsilon > 0$, we would like to answer whether there exists $x \in \mathbb{R}^m$ such that

$$\begin{aligned} & \sum_{e \in E} x_e = n \\ & \sum_{e: v \in e} x_e \leq 1 + \varepsilon && \forall\, v \in V \\ & x_e \geq 0 && \forall\, e \in E \end{aligned}$$

In fact, from such an approximate fractional matching one can construct a matching in G of cardinality at least $(1-\varepsilon)n$ and, thus, also solve the perfect matching problem exactly by taking $\varepsilon < 1/n$.

The following algorithm based on MWU constructs an ε-approximate fractional perfect matchings.

Algorithm 1 Plotkin-Shmoys-Tardos Algorithm for Perfect Matchings

1: **Input**: A bipartite graph $G = (V, E)$, an $\eta > 0$, an integer $T > 0$
2: **Output**: An approximate fractional perfect matching $x \in [0, 1]^m$, or "no answer"
3: Initialize weights $w_v^1 = 1$ for all $v \in V$
4: **for** $t = 1$ to T **do**
5: Compute distribution $p_v^t = \frac{w_v^t}{\sum_{v \in V} w_v^t}$
6: Find a point $x^t \in \mathbb{R}^m_{\geq 0}$ satisfying

$$\sum_{v \in V} p_v^t \Big(\sum_{e: v \in e} x_e^t \Big) \leq 1 \quad \text{and} \quad \sum_{e \in E} x_e^t = n$$

7: If there is no such x^t, output "no answer"
8: Compute loss: $\ell_v^t = 1 - \sum_{e: v \in e} x_e^t$
9: Update weights multiplicatively:

$$w_v^{t+1} = w_v^t \cdot \exp\left(-\eta \ell_v^t\right)$$

10: **end for**
11: **Output**: $\bar{\mathbf{x}} = \frac{1}{T} \sum_{t=1}^{T} x^t$

Clearly once there is no x^t satisfying constraints, we know that the original problem has no solution. If we find x^t at all T rounds, then we have

$$\frac{1}{T} \sum_{t=1}^{T} \langle \ell^t, p^t \rangle - \min_{p \in \Delta_N} \frac{1}{T} \sum_{t=1}^{T} \langle \ell^t, p \rangle \leq \frac{R}{\eta T} + \frac{\eta L^2}{2},$$

where $L = \max_t \|\ell^t\|_\infty$ and $R = \ln n$. Note that for each t, $\langle \ell^t, p^t \rangle \geq 0$ by the constraints of x^t. By selecting $T = O(L^2 \ln n / \varepsilon^2)$, we obtain that for all $v \in V$,

$$0 \leq \frac{1}{T} \sum_{t=1}^{T} \langle \ell^t, p^t \rangle \leq \frac{1}{T} \sum_{t=1}^{T} \ell_v^t + \varepsilon = 1 + \varepsilon - \sum_{e: v \in e} \bar{x}_e^t \,,$$

which shows that $\bar{\mathbf{x}}$ is an ε-approximate solution.

The final ingredient is finding x^t efficiently. In fact, this is very easy. Let us rewrite the condition

$$\sum_{v \in V} p_v^t \Big(\sum_{e: v \in e} x_e^t \Big) \leq 1$$

in the form

$$\sum_{e \in E} q_e x_e^t \leq 1$$

where $q_e = \sum_{v \in e} p_v^t$. Let e^* be the edge such that $e^* = \arg\min_{e \in E} q_e$. Since $\sum x_e^t = n$, we have that

$$n q_{e^*} \leq \sum_{v \in V} q_e x_e^t.$$

Hence, setting $x_{e^*}^t = n$ and $x_e^t = 0$, $\forall\, e \neq e^*$ is always a valid solution (otherwise the original problem is infeasible).

Such a choice of x^t also guarantees that for every $v \in V$:

$$1 - n \leq 1 - \sum_{e: v \in e} x_e^t \leq 1$$

which in particular implies that $L = \max_t \|\ell^t\|_\infty \leq n - 1$. Thus our algorithm runs in time $O(\varepsilon^{-2} n^2 m \ln n)$ (iterate $T = O(\varepsilon^{-2} n^2 \ln n)$ rounds and cost $O(m)$ time at each round).