[CS3958: Lecture 8] Learning with Expert Advice, Online Stochastic Gradient Descent

Instructor: Chihao Zhang, Scribed by Yulin Wang

November 25, 2022

Today, we talk about applications of the online gradient descent (OGD) algorithm. These problems, at the first glance, are not convex optimization problems. However, we show that the OGD algorithm still applies after certain transformations.

1 Learning with Expert Advice

In this problem, we can treat each member of $[n]$ as an expert and in each round, the player needs to pick one expert's advice to follow. Then the adversary reveals the loss of each expert. So the game lasts for T rounds, and for each $t = 0, 1, \dots, T-1$,

- The player picks some $x_t \in [n]$;
- The adversary picks some $\ell_t : [n] \to \mathbb{R}$;
- ℓ_t is revealed, and the player pays the cost $\ell_t(x_t)$.

Here ℓ_t can be viewed as a vector in $[0,1]^{[n]}$. The regret is therefore the gap between the accumulative loss of the player and the best expert, i.e. $\sum_{t=0}^{T-1} \ell_t(x_t) - \ell_t(x^*)$.

Clearly $V = [n]$ is not convex and therefore all previous algorithms based on convex optimization approach do not apply. In fact, any *deterministic* strategy suffers linear regret. To see this, assume $n = 2$ and whenever the player pick $X_t = i \in \{1, 2\}$, then the adversary pick $\ell_t(j) = \begin{cases} 1, & \text{if } j = i \\ 0, & \text{otherwise} \end{cases}$. After T rounds, the cumulative loss of the player is T and the best expert suffers at most $\frac{T}{2}$.

Therefore, a *randomized* strategy is necessary. That is, in t-th round,

- The player picks some distribution $x_t \in \Delta_{n-1}$ where $\Delta_{n-1} = \left\{x \in [0,1]^n : \sum_{i=1}^n x_i = 1\right\}$ is the probability simplex;
- The adversary picks some $\ell_t : [n] \to \mathbb{R}$;
- The player plays $A_t \sim x_t$;
- ℓ_t is revealed, and the player pays the cost $\ell_t(A_t)$.

The *expected regret* of this strategy with respect on a fixed expert $j \in [n]$ is

$$\mathbf{E}\left[\sum_{t=1}^{T} \ell_t(A_t) - \ell_t(j)\right] = \sum_{t=1}^{T} \langle \ell_t, x_t - \mathbf{e}_j \rangle,$$

where $\mathbf{e}_j$ is the j-th standard base of $\mathbb{R}^n$.

Surprisingly, the problem of learning expert advice reduces to a convex online optimization problem on Δ_n. We apply the online gradient descent algorithm to this problem:

- Let $x_1 = (\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n})$.
- In round $t = 1, 2, \dots, T$:
  - The adversary picks $\ell_t : [n] \to \mathbb{R}$;
  - The player plays $A_t \sim x_t$;
  - The adversary reveals ℓ_t, and the player pays the loss $\ell_t(A_t)$;
  - $x_{t+1} = \Pi_{\Delta_{n-1}}(x_t - \eta \ell_t)$.

Assuming notations in previous sections, we obtain the upper bound on the expected regret of the algorithm

$$\frac{\mathrm{Diam}(\Delta_n)}{2\eta} + \frac{\eta T n}{2} = \sqrt{nT}$$

by choosing $\eta = \frac{1}{\sqrt{nT}}$.

2 *Online Stochastic Gradient Descent*

Let us come back to the multi-armed bandits problem! The problem we met before is similar to the learning with expert advice except:

- After each round, the player can only observe $\ell_t(A_t)$ instead of the whole ℓ_t vector;
- The adversary picks each ℓ_t by sampling from a fixed distribution.

We relax the second point by allowing the adversary to arbitrarily pick ℓ_t. This is called *adversarial multi-armed bandit* problem. We are going to solve the problem using online learning approach we learnt, so we have to overcome the difficulty of not knowing the complete ℓ_t vector.

The idea is to guess ℓ_t using the knowledge of $\ell_t(A_t)$. We construct $\hat{\ell}_t$ as an estimate of ℓ_t in each round satisfying $\mathbf{E}\left[\hat{\ell}_t\right] = \ell_t$ and feed this $\hat{\ell}_t$ into the gradient descent algorithm. One natural choice of $\hat{\ell}_t$ is that

$$\forall i \in [n],\ \hat{\ell}_t(i) = \frac{\mathbf{1}_{A_t = i}}{x_t(i)} \ell_t(i).$$

Then clearly $\mathbf{E}\left[\hat{\ell}_t\right] = \ell_t$.

Now consider the following algorithm:

- Let $x_1 = (\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n})$.
- In round $t = 1, 2, \dots, T$:

- The player samples and plays $A_t \sim x_t$;
- The adversary chooses ℓ_t, and the player pays the loss $\ell_t(A_t)$;
- Compute $\hat{\ell}_t$ with the knowledge of $\ell_t(A_t)$;
- $x_{t+1} = \Pi_{\Delta_n}(x_t - \eta \cdot \hat{\ell}_t)$.

This is called the *online stochastic gradient descent (OSGD)* algorithm.

For every $j \in [n]$, we want to estimate

$$\mathbf{E}\left[\sum_{t=1}^{T}\langle \ell_t, x_t - \mathbf{e}_j\rangle\right] = \sum_{t=1}^{T}\mathbf{E}\left[\langle \ell_t, x_t - \mathbf{e}_j\rangle\right].$$

If we use $\mathcal{F}_t$ denote the σ-algebra containing information in the first t rounds, then

- x_t is $\mathcal{F}_{t-1}$-measurable;
- ℓ_t is $\mathcal{F}_{t-1}$ measurable, and $\ell_t = \mathbf{E}\left[\hat{\ell}_t \,\middle|\, \mathcal{F}_{t-1}\right]$.

So for every $t \in [T]$, we have

$$\begin{aligned}\mathbf{E}\left[\langle \ell_t, x_t - \mathbf{e}_j\rangle\right] &= \mathbf{E}\left[\langle \mathbf{E}\left[\hat{\ell}_t \,\middle|\, \mathcal{F}_{t-1}\right], x_t - \mathbf{e}_j\rangle\right] \\ &= \mathbf{E}\left[\mathbf{E}\left[\langle \hat{\ell}_t, x_t - \mathbf{e}_j\rangle \,\middle|\, \mathcal{F}_{t-1}\right]\right] \\ &= \mathbf{E}\left[\langle \hat{\ell}_t, x_t - \mathbf{e}_j\rangle\right].\end{aligned}$$

Surprisingly, we can directly apply our previous analysis of the gradient descent algorithm to the OSGD as if the estimator $\hat{\ell}_t$ is the true loss function!

Following previous bounds, we obtain

$$\mathbf{E}\left[\sum_{t=1}^{T}\langle \ell_t, x_t - \mathbf{e}_j\rangle\right] \le \frac{1}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\mathbf{E}\left[\|\hat{\ell}_t\|^2\right].$$

Obvserve that

$$\begin{aligned}\mathbf{E}\left[\|\hat{\ell}_t\|^2\right] &= \mathbf{E}\left[\mathbf{E}\left[\|\hat{\ell}_t\|^2 \,\middle|\, \mathcal{F}_{t-1}\right]\right] \\ &= \mathbf{E}\left[\sum_{i=1}^{n} x_t(i) \cdot \left(\frac{\ell_t(i)}{x_t(i)}\right)^2\right] \\ &\le \mathbf{E}\left[\sum_{i=1}^{n}\frac{1}{x_t(i)}\right].\end{aligned}$$

The problem here is that we have no control of x_t and it is possible that some $x_t(i) = 0$ and thus the *variance* of $\|\hat{\ell}_t\|$ is unbouded. In other words, if the probability that some arm i is pulled is very close to zero, then our estimate to $\ell_t(i)$ oscilates much.

To overcome this, we have to modify our algorithm so that each arm is pulled with some non-zero probabilty. We define $\tilde{x}_t = (1-\alpha)\cdot x_t + \alpha \cdot \mathbf{u}$ where $\mathbf{u} = \left(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right)$ is the uniform distribution in each round for some parameter $\alpha \in [0,1]$ and play A_t following $\tilde{x}_t$.

- Let $x_1 = (\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n})$.
- In round $t = 1, 2, \dots, T$:
  - Compute $\tilde{x}_t = (1-\alpha) \cdot x_t + \alpha \cdot \mathbf{u}$;
  - The player samples and plays $A_t \sim \tilde{x}_t$;
  - The adversary chooses ℓ_t, and the player pays the loss $\ell_t(A_t)$;
  - Compute $\hat{\ell}_t$ (using $\tilde{x}_t$ instead of x_t) with the knowledge of $\ell_t(A_t)$;
  - $x_{t+1} = \Pi_{\Delta_n}(x_t - \eta \cdot \hat{\ell}_t)$.

It remains to bound $\mathbf{E}\left[\sum_{t=1}^T \langle \hat{\ell}_t, x_t - \mathbf{e}_j \rangle\right]$ for any $j \in [n]$. We have

$$\begin{aligned}
\mathbf{E}\left[\sum_{t=1}^T \langle \hat{\ell}_t, x_t - \mathbf{e}_j \rangle\right] &= \mathbf{E}\left[\sum_{t=1}^T \langle \hat{\ell}_t, (1-\alpha) \cdot x_t + \alpha \cdot \mathbf{u} - \mathbf{e}_j \rangle\right] \\
&\le \mathbf{E}\left[\sum_{t=1}^T \langle \hat{\ell}_t, x_t - \mathbf{e}_j \rangle\right] + \alpha \cdot \mathbf{E}\left[\sum_{t=1}^T \langle \hat{\ell}_t, \mathbf{u} \rangle\right] \\
&\le \frac{1}{2\eta} + \frac{\eta}{2} \sum_{t=1}^T \mathbf{E}\left[\sum_{i=1}^n \tilde{x}_t(i) \cdot \left(\frac{\ell_t(i)}{\tilde{x}_t(i)}\right)^2\right] + \alpha \cdot T \\
&\le \frac{1}{2\eta} + \frac{\eta n^2 T}{2\alpha} + \alpha \cdot T \\
&= 3 \cdot 4^{-\frac{1}{3}} \cdot (nT)^{\frac{2}{3}}
\end{aligned}$$

by choosing $\eta = \frac{4^{\frac{1}{3}}}{2}(nT)^{-\frac{2}{3}}$ and $\alpha = \left(\frac{n^2}{4T}\right)^{\frac{1}{3}}$.

The dependency of the bound on T is $T^{\frac{2}{3}}$, which is worse than the full information case where the dependency is $T^{\frac{1}{2}}$.

In fact, using a more sophisticated gradient based algorithm, we can match the $T^{\frac{1}{2}}$ bound in the bandit case. We will introduce *online stochastic mirror descent* in the next lecture.

3 *Online Shortest Paths*

Given a directed graph $G = (V, E)$ and two vertices $u, v \in V$, consider the following T-rounds game: In round $t = 11, \dots, T$,

- The player picks a path $P \in \mathcal{P}_{u,v}$, where $\mathcal{P}_{u,v}$ is the set of all (simple) paths from u to v;
- The adversary picks a weight function $w_t : E \longrightarrow \mathbb{R}$;
- The player pays $\ell_t(P) \triangleq \sum_{e \in P} w_(e)$.

The regret is $\sum_{t=1}^T \ell_t(P) - \ell_t(P^*)$ where $P^* = \arg\min_{P \in \mathcal{P}_{u,v}} \sum_{t=1}^T \ell_t(P)$.

One can treat each path in $\mathcal{P}_{u,v}$ as an expert and reduce the problem of *learning with expert advice* we studied before. However, the number of paths in $\mathcal{P}_{u,v}$ can be exponential in the size of the graphs and thus the reduction is computational infeasible.

Since our strategies in learning with expert advice are distributions over experts, one can use the following more efficient way to encode distributions on $\mathcal{P}_{uv}$.

The distribution is encoded by a *probability flow* from u to v. Let $\mathscr{P}$ denote the collection of all probability flow (which is convex). Then the expected cost at each round is $\sum_{e\in E} p(e) \cdot w_t(e)$. We can now apply OSGD.

A probability flow from u to v is a function $p : E \to 0, 1]$ such that for any $x \in V \setminus \{u, v\}$,

$$\sum_{y:(x,y)\in E} p(x, y) = \sum_{y:(y,x)\in E} p(y, x)$$

and

$$\sum_{y:(u,y)\in E} p(u, y) = \sum_{y:(y,v)\in E} p(y, v) = 1.$$

For each vertex x, it induces a distribution $\hat{p}(x, \cdot)$ over its out-neighbours in the way that

$$\forall (x, y) \in E\colon \hat{p}(x, y) = \frac{p(x, y)}{\sum_{z:(x,z)\in E} p(x, z)}.$$

This further induce a distribution $\mathcal{P}_{u,v}$ such that $\mathbf{Pr}\left[P\right] = \prod_{e\in P} \hat{p}(e)$.