CS 598: Theoretical Machine Learning

Lecturer: Pranjal Awasthi
Scribe: Pritish Sahu

Lecture #12

---

1 Online Learning with Infinite Action Space

Consider a simple scenario where the algorithm on day t has to predict(y_t^*) between a real value [0,1]. The real value is revealed to the algorithm after the prediction and it can compute the loss. We can write the quadratic loss function as:

$$l_t(y_t) = (y_t - y_t^*)^2$$

The objective is to obtain a no regret algorithm.

$$\text{loss of the any algorithm} = \frac{1}{T}\sum_{t=0}^{T} l_t(y_t)$$

$$\text{loss of the best expert} = \min_{y\in[0,1]} \frac{1}{T}\sum_{t=0}^{T} l_t(y_t)$$

We covered "Follow The Leader(FTL)" algorithm in last class which follows the expert who has performed best so far. The algorithms predicts on day 't',

$$y_t = \underset{y\in[0,1]}{argmin} \frac{1}{t-1}\sum_{i=1}^{t-1} l_i(y)$$

For the case of quadratic loss, it is

$$y_t = \frac{1}{t-1}\sum_{i=1}^{t-1} y_i^*$$

The theorem we proved was, Regret(FTL) $\leq \mathcal{O}\big(\frac{\log T}{T}\big)$. At every time step, one just needs to compute the average and this can be implemented in polynomial time. The algorithm achieves regret zero even though the action space is infinite. We looked the quadratic loss function using this theorem. Next, we will try to provide guarantees for general loss functions.

Let's say we want to predict(y_t) on day t, where $y_t \in k$(action space and $k \subseteq \mathbb{R}^d$) and the loss function be $l_t(y_t)$. We can assume the loss function to be a convex function. For instance, loss function for one dimension is $l_t(y_t) = (y_t - y_t^*)^2$, but in general it could be any loss function. Now instead of looking at action space, y $\in$[0,1], new action space can be the $y \in k$ and the algorithm can fetch expert for time 't' which minimizes the loss function till time 't-1'. Since these are convex functions, we can minimize it. We put a bound on quadratic loss which is $\mathcal{O}\big(\frac{\log \mathcal{T}}{\mathcal{T}}\big)$ and our objective now is to compute the regret for general loss functions. Our claim is, FTL does not work for arbitrary convex function, it needs some modifications. Let's see some example to support our claim.

Let's consider the action space $k \in [-1, 1]$. And the loss functions are provided by the adversary till time step 'T',
$l_1(x) = \frac{1}{2}x$
$l_2(x) = -x$
$l_3(x) = x$
$l_4(x) = -x$
.
.
.
$l_t(x) = (-1)^{t+1}x$, t > 1
.
.
.
$l_T(x) = (-1)^{T+1}x$, T is odd
If you know the loss functions in advance,then you will choose an action that will minimize the sum of all these loses. And the sum is just $\frac{1}{2}$ as the rest gets cancelled out. So, we just need to minimize $\frac{x}{2}$, and x = -1 minimizes the loss. This gives us loss of the best expert as $-\frac{1}{2}$.
Below table provide the result of FTL for few steps,

t	FTL	loss	
1	0	0	(at t = 1, FTL does not have any prior information and it predicts randomly. Let's say it predicts 0, which means the loss is also 0 according the the loss function at t=1)
2	-1	1	(at t = 2, FTL will optimize previsous losses and it has only one. It will predict -1 and the actual loss is 1.)
3	1	1	
4	-1	1	

For each time steps the loss of the algorithm is one and in 'T' steps the loss of the best expert is $\frac{1}{2}$. It is clear that this is not a no regret algorithm as the average loss is a linear function of 't'. This algorithm is very sensitive to its past. The remedy is to make the algorithm less sensitive to the past. Let's look at two different action space and find an algorithm to minimize the regret:

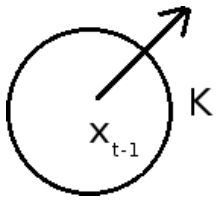

Figure 1: Predicting outside the set k

1. When the action space k is a convex set and k ∈ [-1,1]. Also, known as online gradient descent(OGD). The action space(k) we are optimizing over is $\subseteq \mathbb{R}^d$ which is a convex set and the loss function is convex too.

Online gradient descent(OGD) starte by predicting x_1 randomly in k. After that on day t, OGD looks at the loss in last time step to incorporate that to its current step in a smooth fashion. At every time step it moves opposite to the gradient with step size η. In this process, sometimes the prediction might go out of the set k[**??**] which can be rectified by projecting it back i.e find a point in set k which is near.

$$\begin{aligned} y_t &= x_{t-1} - \eta \nabla l_{t-1}(x_{t-1}) \\ x_t &= proj_k(y_t) = \underset{x \in k}{argmin} ||x - y_t||^2 \end{aligned}$$

Theorem 1. *For set k if diameter(k) = D and loss at any time t, $||\nabla l_t(x)|| \leq G$, $\forall x \in k$ and $\eta = \frac{D}{G\sqrt{T}}$ then,*

$$Regret(OGD) \leq \mathcal{O}(\frac{DG}{\sqrt{T}})$$

Proof. Let, $l_1(), ..., l_T()$ be the loss functions which are convex and their gradients are bounded. Let, $x^* \in k$ be the best expert.

$$x^* = \underset{x \in k}{argmin} \sum_{t=1}^{T} l_t(x)$$

We can write the regret of this algorithm as,

$$\begin{aligned} Regret(OGD) &= \frac{1}{T}\sum_{t=1}^{T} l_t(x_t) - \frac{1}{T}\sum_{t=1}^{T} l_t(x^*) \\ &= \frac{1}{T}\sum_{t=1}^{T} l_t(x_t) - l_t(x^*) \\ &\leq \frac{1}{T}\sum_{t=1}^{T} \nabla l_t(x_t)(x_t - x^*) \qquad \text{(refer to the convex property mentioned below)} \end{aligned}$$

Convex property, if f is convex then $\forall x, y$:

$$\begin{aligned} & f(x) - f(y) \leq \nabla f(x)(x - y) \\ \implies & f(y) \geq f(x) + \nabla f(x)(y - x) \end{aligned}$$

Let's look at x_{t+1} and y_{t+1} [**??**].

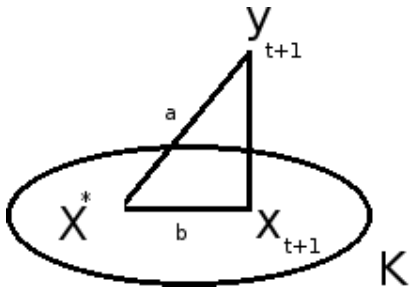

Figure 2: Convex Function Property

$$
\begin{aligned}
||x_{t+1} - x^*||^2 &\leq ||y_{t+1} - x^*||^2 \\
&\leq ||x_t - \eta \nabla l_t(x_t) - x^*||^2 \\
&\leq ||x_t - x^*||^2 + \eta^2 ||\nabla l_t(x_t)||^2 - 2\eta \nabla(x_t)(x_t - x^*) \\
&\leq ||x_t - x^*||^2 + \eta^2 G^2 - 2\eta \nabla l_t(x_t)(x_t - x^*) \\
\implies \nabla l_t(x_t)(x_t - x^*) &\leq ||x_t - x^*||^2 - ||x_{t+1} - x^*||^2 + \eta^2 G^2 \\
\implies \frac{1}{T}\sum_{t=1}^{T} l_t(x_t)(x_t - x^*) &\leq \frac{1}{T}\sum_{t=1} T||x_t - x^*||^2 - ||x_{t+1} - x^*||^2 + \eta^2 G^2 \\
&\leq \frac{1}{T}\big[\frac{||x_1 - x^*||^2}{2\eta} - \frac{||x_{T+1} - x^*||^2}{2\eta}\big] + \frac{\eta G^2}{2} \\
&\leq \frac{D^2}{2\eta T} + \frac{\eta G^2}{2} \\
&\leq \frac{DG}{\sqrt{T}} \qquad \big(\text{ where } \eta = \frac{D}{G\sqrt{T}} \big)
\end{aligned}
$$

□

2. When k is arbitrary may be not convex, $k \subseteq \mathbb{R}^d$. This is known as "Follow The Perturbed Leader". Let the loss, l_t is some linear combination (c_t.x).

   For time steps, t=1,2,...,T, FTL used to predict on day t,

$$x_t = \underset{x \in k}{argmin}(c_1 + c_2 + ... + c_{t-1})x$$

   But, Follow The Perturbed Leader(FTPL) pretends there is a day 0 and pick $c_0 \sim [-\frac{1}{\epsilon}, \frac{1}{\epsilon}]^d$. Now apply FTL, on day t and predict:

$$x_t = \underset{x \in k}{argmin}(c_0 + c_1 + c_2 + ... + c_{t-1})x$$

   Theorem 2. *If L-1 norm of diameter $||k||_1$=D and L-1 norm of cost vectors $||G||_1 \leq G$ then,*

$$E[Regret(FTPL)] \leq \mathcal{O}(\frac{D\sqrt{G}}{\sqrt{T}})$$

Proof. We saw that running FTL was overfitting as it gives emphasis on the past and this can be avoided by adding regularization. The new algorithm with regularization is called "Follow The Regularized Leader". Let's start by picking a convex function $R(x)$. And on day t,

$$x_t = \underset{x \in k}{argmin} \Big[\frac{1}{t-1} \sum_{i=1}^{t-1} l_i(x) + \eta R(x) \Big]$$

R(x)	Remark
L-2 norm : $\|\|x - x_{t-1}\|\|$	we get online gradient descent
$c_0.x$	c_0 is a random vector, we get FTPL
-H(x)(entropy)	we get Randomized multiplicative weights

The regularized function mentioned above are fixed, but these can change over time or it can also be learned. Recently research has been done to adaptively learn regularization function, which is referred to as Adagrad.

Let consider an algorithm A,

$$x_t = \underset{x \in k}{argmin} (c_0 + c_1 + ... + c_{t-1})x$$

Let consider an algorithm B which knows loss at t^{th} time step,

$$x_t = \underset{x \in k}{argmin} (c_0 + c_1 + ... + c_t)x$$

(a) Let's view the process as last time from c_0 to c_t. But, the actual loss we are looking at is from c_1 to c_t and c_0 is a random vector we introduced.

$$Regret(B) \leq 0 + \max_{x \in k} c_0.x \leq \frac{d}{\epsilon T}$$

(b) For any time step 't', A is trying to optimize, $V_1 = c_1 + ... + C_{t-1}$ and similarly B is trying to optimize $V_2 = c_1 + ... + C_t$. For picking c_0, we can say algorithm A is trying to pick a point randomly from cube centered at V_1[**??**] and algorithm B is doing the same from cube centered at V_2[**??**].

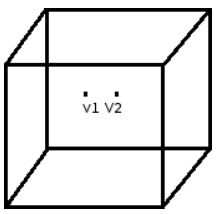

Figure 3: Cube with side length $\frac{2}{\epsilon}$

We are going to ϵ such that $\frac{2}{\epsilon} > G$ and if the previous condition holds, these two cubes wil overlap a lot. This means w.p. $\geq 1 - \frac{\epsilon}{2}$ they will behave same i.e. choose

the same point. They will differ only when they choose random vectors to optimize. And w.p. $\leq \frac{\epsilon}{2}$, $< loss_t(A), loss_t(B) > \leq \frac{\epsilon}{2} DG$
Finally,

$$E[Regret(B) - Regret(A)] \leq \frac{\epsilon}{2} DG$$

$$Regret \leq \mathcal{O}(\frac{D\sqrt{G}}{\sqrt{T}}) \qquad (\text{ where } \epsilon = \sqrt{\frac{2}{GT}}\ )$$

□