Advanced Topics in ML and AGT **Fall Semester, 2018/9**

Lecture 2: October 22

Online Convex Programming

Lecturer: Yishay Mansour *Scribe: Gal Kashi, Ido Rozenberg, Guy Oren, Kathy Raz*

2.1 Online Convex Programming Model

Online convex programming is a framework of online optimization problems. The model is defined as below:

Input: A convex set S
At time $t \in [1, 2, ...]$:

1. Recieve input x_t
2. Select a vector $w_t \in S$
3. Receive a convex loss function $f_t : S \to \mathbb{R}^+$
4. Suffer loss $f_t(w_t)$

Total loss: $\sum_{t=1}^{T} f_t(w_t)$

2.1.1 Motivation

In online linear regression, we observe at time t the vector x_t. After choosing the weight vector w_t, we observe the result y_t. The loss we suffer, assuming square loss, is $(w_t^T x_t - y_t)^2$. So, we can think of the loss function as $f_t(w) = (w^T x_t - y_t)^2$, which is indeed a convex function.
The regret, as we defined it in Lecture 1, is:

$$Regret = \underbrace{\sum_{t=1}^{T} f_t(w_t)}_{\text{online loss}} - \underbrace{\min_{w \in S} \sum_{t=1}^{T} f_t(w)}_{\substack{\text{classic regression loss,} \\ \text{knowing the entire input in advance}}}$$

[0]*Based on the scribe of Assaf Yifrach and Ofek Kirzner, 2017.*

2.1.2 Regret Relative to Competing Vectors

Sometimes, we analyze the regret of an algorithm with respect to a different set than the input set S. For example, in linear regression we can compare to $U = \{w : ||w||_2 \leq 1\}$

Definition Regret with respect to a vector $u \in U$ at time T

$$\forall u \in U, \quad Regret_T(u) = \sum_{t=1}^{T} f_t(w_t) - \sum_{t=1}^{T} f_t(u).$$

Definition Regret with respect to a set U at time T

$$Regret_T(U) = \max_{u \in U} Regret_T(u).$$

2.2 Convexification

While it may be desirable to use convex optimization, the loss functions f_t might be non-convex, thus we introduce two techniques to adopt a non-convex loss function to the convex optimization model:

- Randomization.
- Surrogate Loss.

Given some online optimization problem, we solve it by translating it into an online convex optimization (OCO) problem and solving the OCO efficiently with some solver.

2.2.1 Convexification by Randomization

We explain the method using an example from Lecture 1: the experts problem, where there are d experts, each suffers some loss from $[0, 1]$ at every discrete time t. We wish to design an online algorithm that suffers a loss not much worse than the loss of the best expert.
Notations:
The experts $D = \{1, ..., d\}$.
The loss of expert i at time t: $y_t[i] \in [0, 1], y_t \in [0, 1]^d$

We can referred to the experts problem as a special case of classification, where expert i represents hypothesis h_i and the loss of an expert is the loss of its prediction; namely, $y_t[i] = l(h_i(x_t), y_t)$

If we limit the algorithm to choose a single expert in every step, the loss function will not be convex. In addition, as we saw in previous lecture, the impossibility result which guarantees that no algorithm can attain low regret for such scenario.
To address this issue, we let the online algorithm randomize the expert selection, without letting the adversary know the randomization's realization. At time t, the algorithm selects a distribution:

$$S = \{w \in \mathbb{R}^d : w \geq 0, ||w||_1 = 1\} = \Delta(D)$$

Choosing a single expert $p_t \in D$ can be implemented by using the selected distribution:

$$Pr[p_t = i] = w_t[i]$$

Now, the expected loss is:

$$E[y_t[p_t]] = \sum_{i=1}^{d} Pr[p_t = i] y_t[i] = w_t^T y_t$$

Note: We assume y_t is independent with p_t. While selecting y_t, the adversary might know the distribution w_t, but it does not know its realization p_t.

Now, define the loss function to be the expected loss $f_t(w) := w^T y_t$, which is convex. The function f_t depends on y_t, yet it is consistent with our model, since the loss f_t can depend on y_t (but not the realization p_t).
We would like to compare our algorithm to each single expert. That is, to the set of unit vectors: $U = \{(1, 0, ...0), (0, 1, 0, ...), ...\}$, i.e. the not-convex version of the problem.
In this specific example, the total loss of a constant strategy w is $\sum_{t=1}^{T} f_t(w)$, which is linear in w. Therefore the minimum is attained by some point $u \in U$, which implies that $Regret_T(U) = Regeret_T(S)$, i.e. $min_{w \in U} \Sigma_{t=1}^{T} f_t(w) = min_{w \in S} \Sigma_{t=1}^{T} f_t(w)$

2.2.2 Convexification with Surrogate Loss

Similarly to using a hinge-loss as an upper bound for $0 - 1$ loss, we upper bound the loss function with a convex function.
First, we define a convex set S of hypotheses. Then, we optimize a convex upper bound $\bar{l} : S \to \mathbb{R}$ over the original loss. Denote the original loss as l. We do not specify here in what sense $\bar{l}$ is an upper bound of l. We require:

1. $\bar{l} \geq l$ in some useful sense
2. $\bar{l}$ is convex
3. $\bar{l} = l$ for the optimal solution (or at least the difference is small)

Example: Online Classification with Finite Hypothesis Class

We demonstrate surrogate loss for the problem of online classification with finite hypothesis class which we solved in previous lecture using the halving algorithm. We assumed a realizable model, where the adversary chooses some hypothesis $h^* \in H$ and returns $y_t = h^*(x_t)$. Using the halving algorithm we a proved an upper bound on the number of errors of $log_2(|H|)$. We will get a similar result using convex programming with a surrogate loss ($O(log_2(|H|))$ - the bound difference will be constant).

The hypothesis class:

$$H = \{h_1, ..., h_d\}, \qquad h_i : X \to \{0, 1\}$$

The convex set S, is the distributions over H:

$$S = \{w \in \mathbb{R}^d : w \geq 0, ||w||_1 = 1\} = \Delta(H)$$

In time t, we observe the predictions of all hypothesis:

$$v_t = (h_1(x_t), ..., h_d(x_t)), \qquad v_t \in \{0, 1\}^d$$

The algorithm chooses $w_t \in S$ and predicts:

$$p_t = \begin{cases} 1 \text{ if } w_t^T v_t \geq \frac{1}{2} \\ 0 \text{ if } w_t^T v_t < \frac{1}{2} \end{cases}$$

Thus, one can think of p_t as weighted majority, and S as an alternative convex class of hypotheses.
We mark the error time stamps as $M = \{t : p_t \neq y_t\}$. Given the real outcomes $y_1, ..., y_t$ we define a convex loss function, with respect to w, that upper bounds the original loss $|y_t - p_t|$:

$$f_t(w) := \begin{cases} 2|w^T v_t - y_t| & \text{if } t \in M \\ 0 & \text{if } t \notin M \end{cases}$$

Note: The loss f_t depends on M and thus on $w_1, ..., w_t$. This is consistent with the online convex optimization model since f_t is given only after the algorithm chooses w_t.
We do not specify how the algorithm selects w_t at time t, there are various ways to do that, and we will address this issue later.

Properties of f_t :

1. $f_t(w)$ is *convex*:
   if $t \notin M$, $f_t = 0$ which is convex. Otherwise, f_t is convex since it is a composition of a convex function (absolute value) over an affine function $w^T v_t - y_t$.

2. f_t upper bounds the original loss - we show that $f_t(w_t) \geq |p_t - y_t|$:
   If $t \notin M$, $f_t(w_t) = 0 = |p_t - y_t|$. Otherwise, $t \in M$. Since $p_t \neq y_t$, then $|w_t^T v_t - y_t| \geq \frac{1}{2}$. So, $f_t(w_t) = 2|w_t^T v_t - y_t| \geq 1 = |p_t - y_t|$.

3. In the realizable case there exist $h^* \in H$ for which there exist a unit vector w^* so that $f_t(w^*) = 0$, and $y_t = h^*(x_t) = (w^*)^T v_t$.

Next, we bound the regret of our solution. For that purpose, we state what the unspecified online convex programming algorithm guarantees.

Preliminary definition - Lipschitz constant is a bound on how fast a function $f(w)$ can change.

Definition A function $f_t : S \to \mathbb{R}$ is L_t-Lipschitz with regard to some norm $(|| \cdot ||)$ if:

$$\forall w_1, w_2 \in S : |f_t(w_1) - f_t(w_2)| \leq L_t ||w_1 - w_2||$$

In the next section, we will show an algorithm for online convex programming which achieves the following bound:

$$\forall u \in S : \sum_{t=1}^{T} f_t(w_t) \leq \sum_{t=1}^{T} f_t(u) + \frac{log(d)}{\eta} + 2\eta \sum_{t=1}^{T} L_t$$

where $\eta > 0$ is a parameter, and L_t is the Lipschitz constant of f_t with regard to the L_1 norm.

In order to use this bound, we calculate a Lipschitz constant for f_t defined above:
If $t \notin M$, then $|f_t(w_1) - f_t(w_2)| = 0$, therefore $L_t = 0$.
Otherwise, $t \in M$, we show that $L_t = 1$:

$$\begin{aligned}
|f_t(w_1) - f_t(w_2)| &= |2|w_1^T v_t - y_t| - 2|w_2^T v_t - y_t|| \\
&\leq 2|(w_1 - w_2)^T v_t| && \text{triangle inequality} \\
&\leq 2 \sum_{i : w_1[i] \geq w_2[i]} w_1[i] - w_2[i] && \text{choosing } v_t \text{ which maximizes the expression} \\
&= 2\frac{1}{2}||w_1 - w_2||_1 && ||w_1||_1 = ||w_2||_1 = 1 \\
&= 1 \cdot ||w_1 - w_2||_1
\end{aligned}$$

Using $\eta = \frac{1}{4}$ we get:

$$\begin{aligned} |M| &\le \sum_{t=1}^{T} f_t(w_t) && \text{surrogate loss - upper bound true loss} \\ &\le \sum_{t=1}^{T} f_t(u) + 4log(d) + \frac{1}{2}\sum_{t=1}^{T} L_t \end{aligned}$$

Since $\sum_{t=1}^{T} L_t = |M|$, we have

$$\frac{1}{2}|M| \le \sum_{t=1}^{T} f_t(u) + 4log(d)$$

$$|M| \le 2\sum_{t=1}^{T} f_t(u) + 8log(d)$$

This is true for every $u \in S$, and since we are in the realizable case, there exists a unit vector $u \in S$ with $\forall t : u^T v_t = y_t$. Therefore $f_t(u) = 2|u^T v_t - y_t| = 0$ also for $t \in M$. So we get $\sum_{t=1}^{T} f_t(u) = 0$ and overall:

$$|M| \le 8log(d)$$

2.3 Follow the Leader (FTL)

We introduce a Greedy algorithm for online convex programming that minimizes the retrospective loss. We will examine all the functions from previous time slots that we have already seen, and choose a point that minimizes the cumulative loss function. Notice the only missing piece in the convex programming model, is the specification of how to choose w_t.

FTL

$$\forall t : w_t = \arg\min_{w \in S} \sum_{i=1}^{t-1} f_i(w) \text{ (break ties arbitrarily)}$$

Lemma 2.1 *Let $w_1, w_2, ...$ be the vectors produced by FTL, then $\forall u \in S$:*

$$Regret(u) = \sum_{t=1}^{T}(f_t(w_t) - f_t(u)) \le \sum_{t=1}^{T}(f_t(w_t) - f_t(w_{t+1}))$$

Proof: We will show that:

$$\sum_{t=1}^{T} f_t(w_{t+1}) \leq \sum_{t=1}^{T} f_t(u)$$

The proof is by induction on T:
The basis of the induction, $T = 1 : f_1(w_2) \leq f_1(u)$ from FTL definition $(w_2 = argmin_{w \in S}(f_1(w)))$.
For the induction step, assume the inequality holds for $T - 1$; i.e.,

$$\forall u : \sum_{t=1}^{T-1} f_t(w_{t+1}) \leq \sum_{t=1}^{T-1} f_t(u)$$

Adding $f_T(w_{T+1})$ to both sides,

$$\sum_{t=1}^{T-1} f_t(w_{t+1}) + f_T(w_{T+1}) \leq \sum_{t=1}^{T-1} f_t(u) + f_T(w_{T+1})$$

This is true for all u, specifically $u = w_{T+1}$:

$$\sum_{t=1}^{T} f_t(w_{t+1}) \leq \sum_{t=1}^{T} f_t(w_{T+1}) = \min_{w \in S} \sum_{t=1}^{T} f_t(w) \leq \sum_{t=1}^{T} f_t(u)$$

The equality follows from FTL definition. □

2.3.1 Online Quadratic Optimization

We will use FTL to solve online quadratic optimization, which is called online quadratic optimization, where $f_t(w) = \frac{1}{2}||w - z_t||_2^2$.

Assume $w \in S = \mathbb{R}^d$.

In this case, it is possible to derive steps of FTL in a closed form.
Let $F_t(w) = \sum_{i=1}^{t-1} f_t(w) = \frac{1}{2} \sum_{i=1}^{t-1} ||w - z_i||_2^2$
The function F_t is convex, so it enough to solve $\nabla F_t(w) = 0$:

$$\nabla F_t(w) = (t-1)w - \sum_{i=1}^{t-1} z_i = 0$$

Therefore $w_t = \frac{1}{t-1} \sum_{i=1}^{t-1} z_i$
To apply Lemma 2.1, we express w_{t+1} as a function of w_t.

$$w_{t+1} = \frac{1}{t}(w_t(t-1) + z_t) = (1 - \frac{1}{t})w_t + \frac{z_t}{t}$$

$$w_{t+1} - z_t = (1 - \frac{1}{t})(w_t - z_t)$$

To bound the regret, we need to analyze $f_t(w_t) - f_t(w_{t+1})$

$$f_t(w_t) - f_t(w_{t+1}) = \frac{1}{2}||w_t - z_t||_2^2 - \frac{1}{2}||w_{t+1} - z_t||_2^2 = \frac{1}{2}(1 - (1 - \frac{1}{t})^2)||w_t - z_t||_2^2 \le \frac{1}{t}||w_t - z_t||_2^2$$

Define $L = \max_i ||z_i||_2$, and since w_t is the average of $z_1, ...z_{t-1}$ then, $||w_t||_2 \le L$. From the triangle inequality: $||w_t - z_t||_2 \le 2L$

To summarize:

$$\sum_{t=1}^{T}(f_t(w_t) - f_t(w_{t+1})) \le \sum_{t=1}^{T}\frac{1}{t}||w_t - z_t||_2^2 \le 4L^2\sum_{t=1}^{T}\frac{1}{t} \le 4L^2(ln(T) + 1)$$

From Lemma 2.1, the regret is logarithmic in the number of time steps:

$$Regret \le 4L^2(ln(T) + 1)$$

2.3.2 Bad Example - FTL is unstable

Consider online linear optimization with the following setting:

The domain:

$$S = [-1, 1]$$

The loss:

$$f_t(w) = w^T z_t$$

For any $z \in \mathbb{R}$ we have:

$$argmin_{w \in S} w^T z = -Sign(z)$$

Consider the following sequence:

$$(z_t)_{t=1}^T = (-\frac{1}{2}, 1, -1, 1, -1, ...)$$

Note that $\sum_{i=1}^{2k-1} z_i = -\frac{1}{2}$ and $\sum_{i=1}^{2k} z_i = \frac{1}{2}$, and therefore, the cumulative loss at time t is:

$$F_t(w) = \sum_{i=1}^{t-1} f_t(w) = w^T\Big(\sum_{i=1}^{t-1} z_t\Big) = \begin{cases} \frac{1}{2}w \text{ if } t \text{ is odd} \\ -\frac{1}{2}w \text{ if } t \text{ is even} \end{cases}$$

In every step, w_t is determined by the sign of $\sum_{i=1}^{t-1} z_t$. Which means $w_1 = 0$, $w_2 = 1$, $w_3 = -1$, $w_4 = 1$ etc. So $\forall t > 1 : w_t^T z_t = 1$, and the total loss is $T - 1$.
However, for any constant strategy $u \in [-1, 1]$, the loss is at most $\frac{1}{2}|u| \le \frac{1}{2}$. So FTL incurs a regret of linear magnitude as function of $T - \frac{3}{2}$ with respect to any u.

2.4 Follow the Regularized Leader (FoReL)

As we have seen on the previous section, due to its instability FTL have a linear regret. The problem is that the FTL algorithm is unstable, it might performs dramatic change in each step.
We address this problem by adding regularization. Instead of just selecting the best vector (the one that minimizes the loss retrospectively), we add to minimize also the regularization. Recall the standard problem of linear regression. For an almost singular data matrix, the solution is unstable - for that case, ridge regression stabilizes the solution and stabilizes the bound over the loss.
Let $R : S \to \mathbb{R}_+$ be some (strong-)convex and non-negative regularization function.
Now, we minimize the retrospective loss, PLUS the regularization term.

FoReL

$$\forall t : w_t := argmin_{w \in S} \sum_{i=1}^{t-1} f_i(w) + R(w)$$

In the upcoming lessons, we will see that different regularization functions R provide different algorithms with different guarantees. For example, we will see that we can induce the multiplicative update rule that we have seen in *RMW* as a special case of *FoReL*.

2.4.1 FoReL Example - Online linear optimization with L_2-norm regularization

Consider $L_2 - norm$ regularization.

$$R(w) := \frac{1}{2\eta}||w||_2^2$$

Recall that the loss function of time t is

$$f_t(w) = w^T z_t$$

and

$$z \in [-1, 1].$$

The cumulative loss function to minimize at time t is

$$F_t(w) = \sum_{i=1}^{t-1} f_i(w) + \frac{1}{2\eta}||w||_2^2$$

Let's solve for w_t. The loss $f_i(w)$ is linear, and $R(w)$ is convex by definition, hence F_t is convex since it is a sum of convex functions, and therefore it is enough to solve $\nabla F_t(w) = 0$. We have,

$$\nabla F_t(w) = \sum_{i=1}^{t-1} z_i + \frac{1}{2\eta} 2w_t = 0,$$

which implies

$$w_t = -\eta \sum_{i=1}^{t-1} z_i$$

For future lessons, note that $w_t = w_{t-1} - \eta z_{t-1} = w_{t-1} - \eta \nabla f_{t-1}(w_{t-1})$. I.e. the update rule is practically online gradient descent.

2.4.2 Bounding the loss of FoReL

Next, we bound the loss of FoReL. For that purpose, we apply Lemma 2.1. In order to apply Lemma 2.1, start the time at $t = 0$ (instead of $t = 1$) and define the regularization as the loss of time 0, $f_0(w) := R(w)$.

$$\forall u \in S : \sum_{t=0}^{T} (f_t(w_t) - f_t(u)) \le \sum_{t=0}^{T} (f_t(w_t) - f_t(w_{t+1}))$$

$$f_0(w_0) - f_0(u) + \sum_{t=1}^{T} (f_t(w_t) - f_t(u)) \le f_0(w_0) - f_0(w_1) + \sum_{t=1}^{T} (f_t(w_t) - f_t(w_{t+1}))$$

$$\sum_{t=1}^{T} (f_t(w_t) - f_t(u)) \le f_0(u) - f_0(w_1) + \sum_{t=1}^{T} (f_t(w_t) - f_t(w_{t+1}))$$

Corollary 2.2 *FoReL with a convex regularization function R, guarantees $\forall u \in S$:*

$$Regret_T(u) \le R(u) - R(w_1) + \sum_{t=1}^{T} (f_t(w_t) - f_t(w_{t+1}))$$

Apply the corollary to online linear optimization with L_2-regularization to get the following theorem.

Theorem 2.3 *For online linear optimization by FoReL with L_2-norm regularization, that is*

$$f_t(w) = w^T z_t$$

$$S = \mathbb{R}^d$$
$$R(w) = \frac{1}{2\eta}||w||_2^2$$

then $\forall u \in \mathbb{R}^d$ *s.t.* $||u|| \leq B$*:*

$$Regret_T(u) \leq \frac{1}{2\eta}B^2 + \eta\sum_{i=1}^{T}||z_t||_2^2$$

Proof: From Corollary 2.2, we obtain:

$$Regret_T(u) \leq R(u) - R(w_1) + \sum_{t=1}^{T}(f_t(w_t) - f_t(w_{t+1}))$$
$$\leq R(u) + \sum_{t=1}^{T}(f_t(w_t) - f_t(w_{t+1})) \leq \frac{1}{2\eta}||u||_2^2 + \sum_{t=1}^{T}(w_t - w_{t+1})^T z_t$$

We have seen that $w_t - w_{t+1} = \eta z_t$ (by setting the $\nabla F_t(w) = 0$). So,

$$f_t(w_t) - f_t(w_{t+1}) = \eta z_t^T z_t = \eta||z_t||_2^2$$

And so,

$$Regret_T(u) \leq \frac{1}{2\eta}||u||_2^2 + \sum_{t=1}^{T}\eta||z_t||_2^2 \leq \frac{1}{2\eta}B^2 + \eta\sum_{t=1}^{T}||z_t||_2^2$$

□

In order to analyze the complexity of the bound above, we bound the right term with a parameter L and optimize η. Assume the average norm of z_t is bounded $\frac{1}{T}\sum_{t=1}^{T}||z_t||_2^2 \leq L^2$ and obtain

$$Regret_T(u) \leq \frac{1}{2\eta}B^2 + \eta T L^2$$

The optimal η is $\eta = \frac{B}{L\sqrt{2T}}$, and we get:

$$Regret_T(u) \leq \frac{L\sqrt{2T}}{2B}B^2 + \frac{B}{L\sqrt{2T}}L^2 = \frac{LB\sqrt{T}}{\sqrt{2}} + \frac{LB\sqrt{T}}{\sqrt{2}} = LB\sqrt{2T}$$

In terms of complexity, the regret grows as square root t.

2.4.3 The doubling trick - T is unknown

Notice, that the optimal η in the solution to the previous problem (online linear optimization with L_2-regularization) depends on T and is required for actually running the algorithm. However, T might be unknown ahead. This might happen generally in other algorithms as well.
A standard solution for that problem called the *doubling trick*. The idea is to split the time line into phases of lengths $1, 2, 4, 8, \ldots$. In the m^{th} phase, initialize the algorithm and run it for 2^m steps while assuming $T = 2^m$.
Assume we have an algorithm A that gives $Regret_T(u) \leq \alpha\sqrt{T}$, assuming T is known. If we run the doubling trick over phases for T steps we obtain:

$$\begin{aligned} Regret_T &= \sum_{t=1}^{T} f_t(w_t) - min_w \sum_{t=1}^{T} f_t(w) \\ &\geq \sum_{t=1}^{\tau_1} f_t(w_t) - min_w \sum_{t=1}^{\tau_1} f_t(w) + \sum_{t=\tau_1}^{T} f_t(w_t) - min_w \sum_{t=\tau_1}^{T} f_t(w) \end{aligned}$$

This implies that if we partition the time to intervals, the overall regret is bounded by the sum of the regrets in each interval. Therefore,

$$Regret_T(u) \leq \sum_{m=0}^{\lceil log_2(T) \rceil} Regret(m) \leq \sum_{m=0}^{\lceil log_2(T) \rceil} \alpha\sqrt{2^m} \approx \alpha \frac{\sqrt{2}^{log_2(T)} - 1}{\sqrt{2} - 1} = C\sqrt{T}$$

Where $Regret(m)$ is the regret in the m-th phase, which is of length 2^m. Thus, the regret grows like $\sqrt{T}$ as we wanted. However, this algorithm does not seem reasonable at all. In every phase transition we forget everything we learned so far. In future lessons, we will have a parameter η_t that depends rather smoothly on the current time t.

Note (about FoReL without the doubling trick): We run the algorithm with $\eta = \frac{D}{\sqrt{T}}$ (for some constant D). So during the first $\sqrt{T}$ steps, the regularization dominates the loss function $F_t(w) = \sum_{i=1}^{t-1} f_i(w) + \sqrt{T}||w||_2^2$ meaning, that we practically "ignore" the inputs during these steps.