# LECTURE #5



LECTURER: ELAD HAZAN SCRIBE: KIRAN VODRAHALLI

## Contents

## 1 Last Class and Review of Weighted Majority

hw3 due before spring break (about 10 days) notes covered in book linked on webpage

Where we are in the course: we covered statistical learning theory, now we will talk about the online model. (started last week). This will stay with us for 3, 4, maybe 5 lectures. At the end we will relate to statistical learning model, and we'll get to some advanced topics. So we are in the middle of the second portion.

Last week we talked about decision making, and the experts problem, with iterative decision making.

Repeated Decision Making $t = 1, 2, 3, \cdots, T$. number of experts = $N$. Iteratively, we have two decisions to take, and each expert gives advice and we have a loss function. We can think of a more general setting. Each expert has a loss function at time $t$ $l_t : \mathbb{Z}^+ \to [0, 1]$.

The decision at time $t$: decision maker chooses $i_t \in [N]$ and suffers loss $l_t(i_t)$. Here the losses can be continuous between 0 and 1 instead of just 0 or 1.

question about

**Theorem 1.1.** *Weighted Majority* $\sum_{t=1}^{T} l_t(i_t) - \min_{i^*} \sum_t l_t(i^*) \leq 2\epsilon \sum_t l_t(i^*) + 2\frac{\lg(N)}{\epsilon}$

We improve it a bit in the random setting.

**Theorem 1.2.** *Randomized Weighted Majority* $\boldsymbol{E}[\sum_{t=1}^{T} l_t(i_t) - \min_{i^*} \sum_t l_t(i^*)] \leq \epsilon \sum_t l_t(i^*) + \frac{\lg(N)}{\epsilon}$.

Now the question is: What is the optimal epsilon to choose? We could take derivative and set equal to 0 and all that (convex optimization).

Regret = $\mathbf{E}[\sum_{t=1}^{T} l_t(i_t) - \min_{i^*} \sum_t l_t(i^*)] \leq \sqrt{\sum_{t=1}^{T} l_t(i^*) \lg(N)} \leq \sqrt{T \lg(N)}$ if we have optimal $\epsilon$.

What's the problem with choosing epsilon this way? We would get after taking derivatives that $\epsilon = \sqrt{\frac{\lg(N)}{\sum_t l_t(i^*)}}$. We don't however know how well the best expert does! (also we don't know $t$ necessarily). So we cannot calculate the denominator. Thus this statement is too optimistic. Nevertheless it is possible to get within a factor of 4 (though it might actually be 2). Namely, we can get Regret $\leq 4 * \sqrt{T \lg(N)}$.

How is this possible? Lower bound the denominator (the performance of the best expert) and iteratively modify this bound. This is on the homework.

Some applications

1. Let us first consider the online learning of halfspaces.

Let us say we want to perform the same kind of technique, except not for experts, instead maybe on halfspaces.

So let $K = \{w | w \in \mathbb{R}^n, |w| \leq 1\}$. Recall linear classification setting where we have a bunch of examples $\{(x_i, y_i) | i \in [T]\}$. Suppose you don't have them advance, instead they arrive sequentially online. So we want some kind of adaptive guarantee on the best hyperplane. The loss we have considered thus far is loss$(w, (x_i, y_i)) = 0$ if sgn$(w^T x_i) = y_i$, or 1 otherwise. This is just $0 - 1$ loss.

In the online version, we have to iteratively predict the hyperplane from the set $K$, and we will suffer some loss 0 or 1. This is just the experts problem: the experts here come from the hyperplane. But this is an infinite class. We needed a finite number of experts.

A naive approach would be to discretize the set $K_\delta$, the set of all hyperplanes such that each coordinate times $\frac{1}{\delta}$ belongs to the integers. $K_\delta = \{w||w| \leq 1, w_i * \frac{1}{\delta} \in N\}$. $|K_\delta| = (\frac{1}{\delta})^N$. Need to show two things: $K_\delta$ approximates $K$: if there is a hyperplane in $K$, then there is a hyperplane in $K_\delta$ that is also good. Now it is perfectly legal to use weighted majority algorithm on $K_\delta$ since we have a finite number of experts. We can do not too bad in terms of bounding regret ($\sqrt{nT\log(\frac{1}{\delta})}$), but this is very inefficient and we can do better.

2. Another application to graphs: $G = (V, E); |V| = n, |E| = m$. Consider an online routing problem. $t = 1, 2, \cdots, T, \cdots$. Decision maker picks $p_t \in P_{st}$ and an adversary picks weights $W_t$ from the edges $E$ to $[0, 1]$, and $\forall p \in P_{st}$, loss$(p) = \sum_{(i,j)\in p} W_t(i, j)$.

We just want to do well compared to the best thing possible after adversary picks weights.

So Regret $= (\sum_t \text{loss}_t p_t - \min_p \sum \text{loss}_t(p)) \frac{1}{T} \to_{T\to\infty} 0$. We are looking at comparison to the average best performance.

What's the justification for this again? We just want to compare to average $\to$ law of large numbers approach.

We will have guarantee that regret will behave like $\frac{1}{T}\sqrt{T\text{lg}|P|} \leq \sqrt{\frac{n}{T}}$.

You can model many algorithms with this multiplicative weights scheme but these are inefficient and does not exploit the structure of the problem at all. Why should we treat each path as a totally different entity, for instance? We need to do something more clever than use randomized weighted majority. There is an efficiency issue here! Finitely many but exponentially many experts is not good...

# 2 Online Convex Optimization

Repeated Decision-Making. It is defined by the following objects:

1. Decision set $K$ a convex bounded set in $\mathbb{R}^n$. This is a bit different from what we did before. We will only use very basic facts of convex analysis. $K \subseteq \mathbb{R}^n$ is convex iff $x, y \in K \to (\alpha x + (1-\alpha)y) \in K$ for all $\alpha \in [0, 1]$.

   An example of a convex set is a ball $\{ws.t.|w|_2 \leq 1\}$. Also a cube $\{w, |w_i| \leq 1\}$. Also a polytope = convex hull of vertices $v_1, \cdots, v_k \in \mathbb{R}^d$. $u$ such that $u = \sum \alpha_i v_i$, such that $\sum alpha_i = 1$, $\alpha_i \geq 0$.

2. Player/ Decision Maker chooses point $x_t \in K$ at iteration $t$, and after the player chooses the point,

3. then the adversary chooses a convex cost function $f_t : K \to \mathbb{R}$.

4. player incurrs loss of $f_t(x_t)$.

Performance: Regret = $\sum_t f_t(x_t) - \min_{x^*\in K} \sum_t f_t(x^*)$.

Why can't we do something better than regret in this worst case? We will come back to this. We can make sure regret is always non-negative in the worst case.

**Definition 2.1.** Convex function $f : K \to \mathbb{R}$ is convex over $\mathbb{R}$ if its average over the endpoints is larger than the function at the average point.

1. continuous

2. $\forall x, y \in K$, the average of function evaluated on endpoints is larger than in the middle. For instance, $f(\frac{1}{2}x + \frac{1}{2}y) \leq \frac{1}{2}f(x) + \frac{1}{2}f(y)$. Of course this generalizes to functions of higher dimesion.

Example: $f(x) = w^T x$ is a linear convex function. $f(x) = \|x\|$ is convex for $L_1, L_2$. $f(x) = |x|$ in $1-d$. If $f$ and $g$ are convex, $h = \max_{f,g}$ is also convex (this maximum is a point-wise maximum).So it is very easy to see that $|x|$ is convex given that the maximum is convex.

After the break, we will look at how this setup is applied to learning hyperplanes and the like.

Recall the experts set-up. We have a discrete set of experts, and each expert has a loss between 0 and 1. How would we model the experts problem using this kind of formulation?

You have to model the distribution version of the experts.

## 2.1 Experts as OCO

1. $K \subseteq \mathbb{R}^N$: all distributions over $N$ experts, so the dimensionality is going to be $N$, $\{p| \sum_{i=1}^N p_i = 1, p_i \geq 0\}$, or the $N$-dimensional simplex $\triangle_N$. This set is convex: You have two non-negative sets $p_1, p_2 \in \triangle_N$, then $\frac{1}{2}p_1 + \frac{1}{2}p_2 \in \triangle_N$, since both are probability distributions (sum is 1). Note that this is a polytope. It is the convex hull. Just by this definition, it is the convex hull of $\{e_1, \cdots, e_n\}$, the standard basis vectors. Thus $\mathbf{p} = \sum_{i=1}^N \mathbf{e}_i \cdot \mathbf{p}_i$.

2. $x_t \in \triangle_N$

3. Let $\mathbf{l}_t \in [0,1]^N$. As for the cost functions, they are linear. $f_t(x) =$ expected loss if choose expert $i$ with probability $x_i = l_t^T \cdot x$.

Now we will do the hyperplane example.

**Example 2.2.** Learning hyperplanes.

$K = \{w | \|w\|_2 \leq 1\}$. Here we are restricted to convex loss functions, and $0-1$ loss is not convex! Because this is not convex, this is a hard problem to solve! Finding best hyperplane is NP-hard. So they approximate this with a convex function; we can use the hinge loss.

**Definition 2.3.** Hinge-loss.

$h_{x,y}(w) = \max\{0, 1 - y_i(\mathbf{w}^T x_i)\}$. If the inner product is larger than 1, then it is zero, otherwise, it gets closer to 1. In support vector machines, they replace $0-1$ by this kind of loss.

So we will let $f_t(w) = \text{hinge}_{x,y}(w)$.

**Example 2.4.** Online Routing as OCO

Let us try to represent the paths in low-dimension. We want to find the convex hull. Let us think of a path $p$ as a vector in $\mathbb{R}^m$. We are thinking of $s-t$ paths in $G(V,E)$.

Then the convex hull $CH(P_{st})$ is the $st$-flow-polytope over $G$. It is a convex set in only $m$ dimensions. $f \in F_{st}$ and thus $f \in \mathbb{R}^m$. Flows satisfy flow conservation, unit flow going from source. We have $\sum_{v \in \gamma(t)} f_{sv} = 1 = \sum_{v \in \gamma(t)} f_{vt}$, we also capacity constraints: have all flows are between $-1$ and 1, and we have $f_{uv} = f_{vu}$. Finally the total sum of flows is 0, or flow conservation.

Thus

1. $K$ = flows from $s$ to $t = \{f | \text{constraints}\} \subseteq \mathbb{R}^m$. Thus we can represent the shortest path problem very efficiently: in low dimension and with few constraints.

2. loss functions: $f_t(x) = \sum_{e \in X} w_t = w_t^T \cdot X$. Just a dot product of the weights of the path with the path vector.

# 3 Efficient Algorithms for OCO

However, we still haven't discussed any algorithms for this online convex optimization: we have only told how to model various problems.

## 3.1 Online Gradient Descent

**Definition 3.1.** Online Gradient Descent Algorithm

1. $x_1 \sim K$, arbitrary
2. for $t = 1, 2, \cdots$
   $x_{t+1} = \prod_k (x_t - \eta \nabla f_t(x_t))$ We move in the direction of the increase since that is a good idea. In many dimensions we use gradient. We update $x$ linearly - this might take us out of the domain! A bad thing. So we need to go back to the domain. Choose a point inside the convex domain after we step outside - the natural thing to do is to take the closest point in $K$: we project onto $K$. We have written the projection as $\prod_k$: $\prod_k(y) = \text{argmin}_{x \in K} |x - y|_2$, which is efficient to compute in general. Note that $\eta \in \mathbb{R}$, which is a step size.

**Theorem 3.2.** *Zinkevich '03*
*Regret of OGD for $\eta \sim \frac{1}{\sqrt{t}}$. We have $\sum f_t(x_t) = \min_{x^*} \sum_t f_t(x^*) = O(\sqrt{T})$. This gives us a good bound, and we are also good in the case of high dimension. Much more efficient algorithm.*

First some important bounds We are reasoning about regret. What happens if we scale up the function by a factor of 10? Then we have the regret also goes up by a factor of 10. So we need to bound the costs $f(x)$.

First, we say $K$ is bounded. Denote by $D$ the diameter of $K$: $\text{diam}(K) = \max_{x,y \in K} |x - y|$. We will also upper bound the norm of the gradient with $|\nabla f_t(x_t)|_2| \leq G$. This amounts to a Lipschitz restriction on the function.

Both of these are necessary.

*Proof.* Surprisingly easy.

Let $y_{t+1} = x_t - \eta \nabla f_t(x_t)$, and $x_{t+1} = \prod_k y_{t+1}$. The first observation is that $|x_{t+1} - x^*|_2^2 \leq |y_{t+1} - x^*|_2^2$. It is clear that the projection step can only bring us closer to $x^*$ by the fact that the hypotenuse is larger (Pythagoras theorem). Since it is a convex set (tangent contains convex set all on one side, tangent is perpendicular to surface), the angle of the triangle is obtuse and apply generalized Pythagoras theorem.

Then this equals $|x_t - \eta \nabla f_t(x_t) - x^*|^2 = |x_t - x^*|^2 - 2\eta \nabla f_t(x_t)(x_t - x^*) + \eta^2 |\nabla f_t(x_t)|^2$. Therefore taking the inequality in from before,

$$\begin{aligned}
\nabla f_t(x_t)(x^* - x_t) &\leq \frac{1}{2\eta} \left[ |x_t - x^*|^2 - |x_{t+1} - x^*|^2 \right] + \eta G^2 \\
f_t(x_t) - f_t(x^*) &\leq \nabla f_t(x_t)(x^* - x_t) \text{ by convexity} \\
\text{Regret } = \sum_t f_t(x_t) - f_t(x^*) &\leq \frac{1}{2\eta} \left[ |x_t - x^*|^2 - |x_{t+1} - x^*|^2 \right] + \eta T G^2 \\
&\leq \frac{1}{2\eta} D^2 + \eta T G^2 \\
&\leq 2DG\sqrt{T} \text{ where } \eta = \frac{D}{G\sqrt{T}}
\end{aligned} \tag{1}$$

□

These functions are changing adversarily, and you're moving in the direction of the previous function. Nevertheless, error is bounded by the squareroot of the number of iterations! The average regret approaches zero decreasing at $\frac{1}{\sqrt{T}}$.

## 3.2 Applying OGD to Various Examples

In general, the basic question of convex optimization is $\min_{x \in K} f(x)$. There are many ways to do this: Ellipsoid, Random-Walk, Newton's Method. If $K$ is not convex, this problem is NP-hard. Nevertheless, we can use OGD to solve this problem if $K$ is convex. We have $\frac{1}{T}\sum (f(x_t) - f(x^*)) \leq \frac{GD}{\sqrt{T}}$. Let us define $\bar{x} = \frac{1}{T}\sum_{t=1}^{T} x_t$, $f(\bar{x}) \leq f(\frac{1}{T}\sum_{t=1}^{T} x_t) \leq f(x^*) + O(\frac{1}{\sqrt{T}})$. We can say $T = \frac{1}{\epsilon}$ by simple convexity.

Here we apply it to experts.

Let $l_t(i)$ = loss of example $i$ at time $t$. Then $x_{t+1} < -\prod_{\triangle_N}(x_t - \eta l_t)$. Therefore, loss $\leq \sum_t l_t(i^*) + GD\sqrt{T}$. We have $D = \text{diam}(\triangle_N) \leq \sqrt{2}$. $G = \max|\nabla f_t(x_t)| = |l_t| \leq \sqrt{N}$. Thus we have Regret $\leq \sqrt{2NT}$.

Now for linear separators: $K = \{|w|_2 \leq 1\} \to D \leq 2$.

**Definition 3.3.** Subgradient
For any convex function that is continuous, we can define it. It is any $\nabla f(x_t)$ such that $f(x) + f(y) \leq \nabla f(x)(y - x)$. Generally it is a set of points - this is used where there are points that are not differentiable. We don't worry about this at all if function is differentiable.

We have that $G \leq 1$ in the setting with the Hinge Loss. Therefore we get Regret $\leq 2\sqrt{T}$ for linear classification. In fact this bound is tight.

**Example 3.4.** Online Routing

$K$ = s.t. flows. The vertices of the flow polytope are all the paths in the graph (each path is a vector). So How large is the diameter We have $\text{diam}(K) \leq 2\sqrt{m}$. You can probably give a better bound that is $\sim \sqrt{n}$, since the difference is size of path can be at most $n$ (for two of the vectors in $K$).

Then recall that $f_t(x) = w_t^T \cdot x$, and $w_t(e)$ = length of edge $e$ at time $t$, and therefore $G$ is bounded by $\sqrt{n}$ since the weights are between 0 and 1 and paths are of length $n$. Thus our regret bound is given by Regret $\leq \sqrt{mnT}$. (though it could be a bit better).

Let's think about what we're actually doing. We move through the flow space and want to sample from flow to get an actual path.

We have $f = \int_{p \sim P_{st}} w_p \cdot p$. We can do flow decomposition $f = \sum \alpha_p \mathbf{p}$, where $\sum \alpha_p = 1$, $\alpha_p \geq 0$. Only $m$ non-zeros, $\alpha$ values. The algorithm for doing Flow Decomposition: Find the minimum edge weight in a given flow, subtract it. Remove the path from the graph, left with one that is 0. We can remove flows $m$ times. ( Remember the thing from graph theory. )

Caratheodory's Theorem: We can decompose into convex combination of only $n$ vertices. LOOK THIS UP on wikipedia. Generalizes the previous flow decomposition.

So get a flow from the gradient, do flow decomposition, sample a path. Keep doing this. We get that the average length of the expected path is going to be the length of shortest path. This is a pretty cool result! The algorithm is efficient too.

**Remark 3.5.** Projection $f$ is convex, $k$ is convex, we take $\min_{x \in K} f(x)$. There exists a membership oracle. Finding $x \in K$ can be done in polynomial time. In general most efficient algorithm is Ellipsoid. Via ellipsoid, $O(n^3 \text{lg}(\frac{1}{\epsilon}))$. This is the number of points we need to ask the membership oracle.

**Remark 3.6.** Determinism So far, everything we have seen is deterministic! Not a stochastic environment. Maybe when drawing a path in the online routing example is some randomness, but aside from that everything is deterministic.

**Example 3.7.** Stochastic Optimization Let's relate things back to learning theory. Here $f \sim F$, a distribution. Learning theory is exactly stochastic optimization. If $\mathcal{H} = \{|w|_1 \leq 1\}$, disc over $(x, y)$ as $x \in \mathbb{R}^d, y \in \{-1, 1\}$. We have $f_{x,y} \sim D$, $f_{x,y}(w) = \max\{0, 1 - y(w^T x)\}$.

We have $\text{err}(w) = \mathbf{E}[f_{x,y}(w)]$. To apply OGD, we just do it on $f_k$ which is a sample of $f_{xy}$ as drawn from distribution. $K = \mathcal{H}$. We get Regret $= \frac{1}{T}\sum_{t=1}^{T} f_t(w_t) - \min_{w^*} \frac{1}{T}\sum_t f_t(w^*) \leq 2\frac{\sqrt{T}}{T}$. How do we get a good approximation? Take the average, $\bar{w} = \frac{1}{T}\sum_t w_t$. We have $\mathbf{E}[f_t] = \text{err}(\cdot)$. Our random variables are dependent, so we need to go through some concentration inequalities analysis. We have $\text{err}(\bar{w}) \leq \text{err}(w^*) + \frac{2}{\sqrt{T}}$. So we have essentially done agnostic learning with sample complexity that behaves as $\frac{1}{\epsilon^2}$ (i.e. $\epsilon = \frac{2}{\sqrt{T}}$). VC-dimension comes into this implicity through the dimension of the space we are living in.