Contents

1 Last Class and Brief Review

Solutions from the last homeworks will be posted on Blackboard. A new exercise is up.

Remark 1.1. Let us address one problem from the homeworks. Recall $X = N_H - N_T$, and the goal was to prove $E[|X|] = \Omega(\sqrt{T})$. Then $P\{X = 0\} \sim \frac{1}{\sqrt{T}}$. If you look at the distribution from $-T$ to $T$, the maximum is obtained at $0$. Any other outcome is less likely. Therefore, let us compute the probability that we are in some band around the origin. It’s less than $\frac{1}{2\sqrt{T}}$. So you could bound appropriately to get the desired result. There’s an easy route to all the problems I give. Each problem can be solved in half a page, one page at most.

Next week, a prominent professor will give a talk - it will be a surprise! You guys have a treat.

Today I want to do two things. One of them is relating regret minimization to statistical learning. We will show that one is stronger than the other. The second topic is how to learn with partial information.

In summary, today’s topics for lecture will be:

1. Relate statistical and online learning. Let us recall the definition of statistical learning (from the first lecture or so): there are

   (a) a distribution $D$ on $\mathcal{X} \times \mathcal{Y} \in [0, 1]$ (many times we suppose binary labels)
   
   (b) a set of hypotheses $h \in \mathcal{H}, h : \mathcal{X} \to \mathcal{Y}$
   
   (c) a sample $S \subset \mathcal{X} \times \mathcal{Y}$, assuming i.i.d. samples

   (d) $\text{error}(h) = E_{(x,y) \sim D}[h(x) \neq y]$, in the case $y \in \{0, 1\}$. We can be more general than $0 - 1$, and instead define a loss $\mathcal{X} \times \mathcal{Y} \to [0, 1]$. We can write $l(\hat{y}, y) = 1$ if $\hat{y} \neq y$, 0 otherwise. We also have the error with respect to the sample $S$. We only saw one statistical learning algorithm: Empirical Risk Minimization (see notes for lectures 1 and 2 for a reminder). We had a fundamental theorem that said if we take $h^* = \arg\min_{h \in \mathcal{H}} \text{err}_S(h)$, then $\text{err}(h^*) \leq \epsilon$ if $|S| \geq \frac{VC(\mathcal{H})}{\epsilon^2} \log(\frac{1}{\epsilon})$. We actually saw the agnostic version, which is a bit stronger: we can show $\text{err} h^* \leq \min_h \text{err}(h) + \epsilon.$

2. For the past few lectures, we talked about online learning. There is no distribution, the setting is adversarial, kind of game-theoretic. We have

   (a) a decision set $\mathcal{K}$

   (b) a decision maker $DM$ that chooses $x_t \in \mathcal{K}$, which gives

   (c) $f_t(x_t), f_t$ an adversarially chosen loss function

   (d) Regret is given by $\sum_{t=1}^{T} f_t(x_t) - \min_{x^* \in \mathcal{K}} \sum f_t(x^*)$
2 Statistical Learning and Regret Minimization

The goal is to relate these two settings. What would the decision set $K$ correspond to in statistical learning versus online learning? Informally, online learning implies statistical learning but not the other way around. Online learning is strictly stronger than statistical learning which is why we have been studying online learning.

We will reduce statistical learning to regret minimization.

Let $A$ be a low-regret algorithm over $H$. $K$ can capture all finite hypothesis classes, and most interesting infinite classes.

What does a reduction mean? From this algorithm $A$, we want to construct a hypothesis with low error given the ability to sample from some distribution. How can this be done? Let us apply the following trick:

For $t = 1, \ldots, T$, for some big number $T$ which we will choose in a minute, we

1. Sample $(x_t, y_t)$ from $D$. We do this one by one, since we assume we can sample from this distribution.
   If you only have a set $S$, then you just take the next example in $S$, which is perfectly fine since we just sampled $S$ from $D$.

2. Now we define a loss function. We choose $f_t(h) = l(h(x_t), y_t)$ - this is a loss function over hypothesis, since the online algorithm knows how to work with the hypothesis class, not examples.

3. $h_{t+1} = \text{Alg}(f_1, \ldots, f_t)$ (We do one more step of the cost function with a new cost function at each step.)

Then at the end, we return $\bar{h} = \frac{1}{T} \sum_{t=1}^{T} h_t$. So what does it mean to take the average of hypotheses? If $H$ is convex, then the average is well-defined. Otherwise, we can think of $H$ as `the convex hull of $H$'. We can always convexify this way.

**Example 2.1.** Suppose that $H = \{h_1, \ldots, h_n\}$ is a discrete set of hypotheses. Then what is the best algorithm to take for a discrete set. We have used multiplicative weights before for discrete sets, so let us use that here. We basically reduce statistical learning to an experts problem. Also, let us say that loss is $0 - 1$. Every hypothesis either makes a correct prediction or it does not. Then we can apply multiplicative updates over all previous functions. Each sample gives a $0 - 1$ vector for whether the $i^{th}$ hypothesis made an error or not. Our algorithm runs multiplicative updates over these cost vectors. We are returning the average of all the experts - a distribution over hypotheses rather than a specific hypothesis.

**Example 2.2.** Hyperplanes are a classical example in statistical learning. Let us say $H = \{w \in \mathbb{R}^d; |w|_2 \leq 1\}$, $X = \{x \in \mathbb{R}^d\}$, $y \in \mathbb{R}$. Recall the VC-dimension of this hypothesis class is $D$. We will let $l(x, y) = (w^T x - y)^2$, and we can choose online gradient descent for our algorithm. Basically we keep updating our hyperplane, and return the average, which is well-defined since this is a convex set. Note that this is not best in hindsight, it’s a different algorithm. We do the update: $w_{t+1} := \prod (w_t \cdot \eta \nabla l_{x_t, y_t}(w_t))$.

**Theorem 2.3.** Suppose that $\text{Reg}_t(\text{Alg}) \leq R_T$. Then with probability $1 - \delta$, $\text{err}_D(\bar{h}) \leq \min_{h \in H} \text{err}_D(h) + \frac{R_T}{T} + \sqrt{\frac{10 \log(\frac{1}{\delta})}{T}}$.

Why don’t we get any dependence on VC-dimension? It might come in with $R_T$. $R_T$ will be like a square-root number of iterations. So we might get something like $\sim \frac{GD + \log(\frac{1}{\delta})}{\sqrt{T}}$, if for example we are using OGD.

For **Example 2.1.**, we have $R_T \sim \sqrt{T \log(|H|)}$, and we can easily get $\text{err}_D(\bar{h}) - \text{err}_D(h^*) \leq \frac{H}{e^2} - \frac{\log(\frac{1}{\delta})}{e^2}$. For **Example 2.2.**, we have $R_T \leq \sqrt{T}$, and we will get some bound like $\frac{\log(\frac{1}{\delta})}{e^2}$.

Before we prove this, let us think of whether the converse is also true. Is it true that statistical learning implies low regret? Suppose I have an algorithm for statistical learning, can I get low-regret? The answer is no. Let us prove **Theorem 2.3.** We will use martingales and the Azuma Inequality.
Definition 2.4. Martingales.
\[ X_1, \ldots, X_T, E[x_t|x_1, \ldots, x_{t-1}] = x_{t-1} \]

Definition 2.5. Azuma’s Inequality for Martingales.
For \( \{X_i|i = 1, \ldots, T\} \) is a martingale, and \( |X_t - X_{t-1}| \leq 1 \), then \( \mathbb{P}\{|X_T - X_0| > c\} \leq 2e^{-\frac{c^2}{2T}}. \)

**Proof.** We will assume \( |l(\hat{y}, y)| \leq 1 \). If this is not the case, then we can still make this work with larger constants. We will proceed with some martingale analysis.

The outline of the proof is given by:

1. The intuition will be \( \text{err}_S(\hat{h}) \leq \text{err}_S(h^*) + \frac{R_T}{T}. \)
2. By concentration, \( \text{err}_S(\hat{h}) \sim \text{err}_D(\hat{h}). \)
3. By concentration \( \text{err}_S(h^*) \sim \text{err}_D(h^*). \)

\[ Z_t := \text{err}_D(h_t) - I_{x,y \sim D}(h_t(x_t), y_t). \] Now, \( \mathbb{E}[Z_t] = 0 \), since \( \text{err}_D(h_t) = \mathbb{E}_{(x,y) \sim D}[l(h_t(x_t), y_t)]. \) So \( Z_t \)
depends on \( Z_{t-1} \), since \( h_t \) depends on \( h_{t-1} \). But the expectation of this is 0, and we use a martingale to get rid of this dependency. In addition, let us define a martingale. Let \( X_t := \sum_{i=1}^{T} Z_i. \)

Now we can write \( |X_{t+1} - X_t| \leq 1 \). \( \mathbb{E}[X_{t+1}|X_1, \ldots, X_t] = X_{t+1}. \) By Azuma, we can write \( \mathbb{P}\{|X_{T+1} - X_0| \leq c\} = 2e^{-\frac{c^2}{2T}}. \) This implies that \( \mathbb{P}\{|\frac{1}{T} \sum_{t=1}^{T} \text{err}_D(h_t) - \frac{1}{T} \sum_{t=1}^{T} l(h_t(x_t), y_t)| > \sqrt{\frac{2\log(T)}{T}} \} < \frac{\delta}{2}. \) Therefore, \( \text{err}_S(\hat{h}) \sim \text{err}_D(\hat{h}). \) We can similarly define \( X_t, Z_t \) for \( h^* \), and apply the same thing (or even Hoeffding, since \( Z_t \)'s are no longer related.) We have \( \mathbb{P}\{|\frac{1}{T} \sum_{t=1}^{T} \text{err}_D(h_t) - \frac{1}{T} \sum_{t=1}^{T} l(h_t(x_t), y_t)| > \sqrt{\frac{2\log(T)}{T}} \} < \frac{\delta}{2}. \) Thus we have \( \text{err}_S(h^*) \sim \text{err}_D(h^*). \)

So with probability \( 1 - \delta \), \( \text{err}_D(\hat{h}) - \text{err}_D(h^*) \leq \frac{1}{T} \sum_t \text{loss}(h_t(x_t), y_t) - \text{err}_D(h^*) + \sqrt{\frac{\log(T)}{T}}. \)

By the second martingale, this is \( \leq \text{err}_S(h) - \text{err}_S(h^*) - \sqrt{\frac{4\log(T)}{T}} = \sum_t \text{loss}(h_t(x_t), y_t) - \sum_t \text{loss}(h^*(x_t), y_t) \leq \frac{1}{T} \sum_t \text{loss}(h_t(x_t), y_t) - \sum_t \text{loss}(h^*(x_t), y_t). \) The idea is that this is bounded by \( \frac{\text{Reg}(A)}{T}. \) This works because by Jensen, the average of the errors is at most the error of the average.

We now assert that this does not work the other way - statistical learning does not imply OCO.

One particular statistical learner is the ERM algorithm, which does the best in hindsight. Can we use this for online learning? We have seen a simple setting where it doesn’t work (where you flip \(+/- \) ones) - there is a deep question about computational equivalence. Can we use the ERM (as an optimization oracle) potentially we could find the hypothesis in one shot. Regret implies optimization, if you can do it efficiently, then you can optimize efficiently. The other direction is not clear at all, it is a matter of research. We will talk about it in the last lecture of this course.

3 Learning with Partial Information

**Example 3.1.** Online routing \( G = (V, E) \). You may have only very partial observations. It is very reasonable to assume that you don’t know the individual edges of a path, that you only know the total length, for instance. We could model a path like a vector that lies in the flow polytope, and there are weights (length of the edges). \( W_t : E \to [0, 1] \), and \( l_t(x_t) = \sum_{e \in x_t} W_t(w) \). If you assume there is a distribution over congestions, then average path that we choose has its average length approach that of the best chosen path in hindsight. But this means we need to know all the weights. So we will introduce another issue into our observation model.
3.1 Bandit Convex Optimization

We have so far assumed we can see the gradient, which was justified for portfolio selection. But sometimes, you can only see the actual loss. Let us assume that is all we get to see. Can we still optimize?

Definition 3.2. Bandit Convex Optimization

We have the OCO model from before, except we observe only \( f_t(x_t) \in \mathbb{R} \). \( K \), \( f_t \), and regret are defined as before.

Let us start with a special case, called the Multi-Armed Bandit problem.

Definition 3.3. Multi-Armed Bandit (MAB)

\( N \) experts/bandits, at time \( t \) you pick one \( i_t \), and get to see a loss which is \( l_t(i_t) \). We would like to minimize \( \frac{1}{T} ( \sum_t l_t(i_t) - \min_i \sum_t l_t(i^*) ) \). Let \( l_t \in \{0, 1\}^N \). The only difference from the experts problem is that we don’t know how well EVERY expert did - we only know how well the expert we chose did. We would like the regret to \( \to 0 \) as \( t \to \infty \).

Suppose we have \( l_t(i) \sim \text{i.i.d distribution} \). But we don’t know the distribution. Every iteration, we draw from the same distribution. This is very much like statistical learning, but picking the best in hindsight is not well-defined.

The simplest thing we could do is estimate each arm of the bandit for time blocks, and in each time block, estimate with an average and pick the best. Then for the rest of the iterations, I will know how much error I have. This is a kind of silly thing to do, since we could just forget the bad guys.

We can reproduce the experts guarantee in the adversarial setting for this problem.

There is a fundamental difficulty called Exploration-Exploitation tradeoff in bandit problems - we need to explore to some extent to check that other arms are not good. On the other hand, we need to exploit the best one. We need to explore all the time to be sure something else does not change. Intuitively, exploring gives us information about the matrix, and exploiting gives us good regret.

We will separate exploration and exploitation.

Definition 3.4. Simple MAR Algorithm

for \( t = 1, \cdots, T \):

- with probability \( q \) pick \( i_t \sim [N] \) uniformly at random
- create \( \hat{l}_t(j) = N \ast l_t(i_t) \ast \frac{1}{q} \) for \( j = i_t \), 0 otherwise.
- and \( x_{t+1} := MW(x_t, \hat{l}_t) \). \( x_{t+1}(i) := x_t(i)e^{\hat{l}_t(i)} + \text{scaling} \)
  after these \( T \), play \( x_{T+1} = x_T \).

Two things come into play: we have multiplicative weights on a ”good” subset of the data (depends on \( q \)), and we have unbiased estimators for each of the data points. Note that \( \mathbb{E}[\hat{l}_t] = l_t \ast \frac{1}{q} = \sum_j \mathbb{P}\{i_t = j\} \left( \frac{1}{q} \ast (N \ast l_t(i)) \right) \) or 0.

We want to prove

Theorem 3.5. \( \text{Regret(simple)} = O(\sqrt{NT^2}). \)

We lost something (not as good as multiplicative weights), but we must expect to since we took away a fair portion of our assumptions.

Proof.

\[
\mathbb{E}[\text{Regret}] = \mathbb{E}\left[ \sum_t l_t(x_t) - l_t(x^*) \right] \\
\leq \mathbb{E}\left[ \sum_t \hat{l}_t(x_t) - \hat{l}_t(x^*) + 1_{\text{explore}} \right] \tag{1}
\]
This is because 1 is the worst case possible in the case that we don’t use our ‘good’ bandit arm, since we assume our losses are in $[0, 1]$. Also note that $E[\hat{l}_t] = l_t$ for the no-exploration case. Then

$$E[\text{Regret}] \leq \frac{n}{q} \sqrt{T \log(N)} + qT$$

since loss is either $\frac{N}{q}$ or 0 instead of 1 or 0

$$= O(\sqrt{NT^3 \log(N)})$$

choosing $q = \sqrt{N \cdot T^3}$

(2)

For online routing, this doesn’t work, since $N = 2^{|V|}$, which is too large. This bound is also not tight in general, we can do better than blindly choosing examples. We can combine explore-exploit.

**Definition 3.6.** EXP-3 algorithm

It will be identical to the simple MAR algorithm, except we will do explore and exploit on each step. Estimators are biased now, since our feedback is based on what we have seen so far. It is more complicated analysis, nevertheless it works.

for $t = 1, \cdots, T$:

choose $i_t \sim x_t \in \Delta_n$

create $\hat{l}_t = l_t(i_t) * \frac{1}{x_t(i_t)}$ for $j = i_t$, 0 otherwise.

and $x_{t+1}(i) := x_t(i)e^{\eta\hat{l}_t(i)} + \text{scaling}$

**Theorem 3.7** (Aure, CB, F, Schapire). $E[\text{Regret}(EXP3)] \leq \sqrt{TN \log(N)}$

It can be shown that any multi-armed bandits algorithm must suffer $\Omega(\sqrt{TN})$. This question will be extra-credit on the homework.

In this setting, there is a time complexity issue and a regret issue.

4 Summary

Today we learned the following facts:

1. We started by showing how the online learning model implies statistical learning.
2. Then we talked about multi-armed bandits and gave a simple algorithm to do it. Think about how to use the simple approach for Bandit Convex Optimization.