COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Stefan Keselj

Lecture #16
April 04, 2018

1 Old model (individuals)

1.1 Recap

Recall the online learning model we have looked at the last two lectures:

- $N = \#$ experts
- For $t = 1, 2, \ldots T$
  - expert i predicts $\xi_i \in \{0, 1\}$
  - learner predicts $\hat{y} \in \{0, 1\}$
  - observe outcome $y \in \{0, 1\}$

We denote the numbers of mistakes of each expert and the algorithm as a whole as:

- $L_i = \#$ mistakes of expert i
- $L_A = \mathrm{E}[\ \#$ mistakes of learner $]$

Last class we showed that if $\min_i L_i \leq K$, a variant of the randomized weighted majority algorithm will achieve:

$$L_A \leq \min_i L_i + \sqrt{K \ln(N)} + \frac{\lg(N)}{2}$$

1.2 Regret lower bound

We usually have that $\min_i L_i \leq T/2$. All we have to do to guarantee this is take any expert and create a counter expert that always predicts the opposite of the first expert; one of them has to have error less than or equal to $T/2$.

Plugging this into the bound above yields:

$$L_A \leq \min_i L_i + \sqrt{\frac{T \ln(N)}{2}} + \frac{\lg(N)}{2}$$

$$\frac{L_A}{T} \leq \min_i \frac{L_i}{T} + \sqrt{\frac{\ln(N)}{2T}} + \frac{\lg(N)}{2T}$$

As $t \to \infty$, the rate at which the learner is making mistakes will approach the corresponding rate for the best expert.

It would be nice if we could show that any algorithm must have regret at least $\sqrt{\frac{T\ln(N)}{2}}$. Then we would know our algorithm has optimal regret with respect to the best expert, with even the constants matching exactly.

Let us consider the worst case adversarial setting. This is the setting where none of the experts give any information about the outcomes. Imagine that every round, for all i:

$$\xi_i = \begin{cases} 1, \text{ with probability } 1/2 \\ 0, \text{ else} \end{cases} \quad \text{and } y = \begin{cases} 1, \text{ with probability } 1/2 \\ 0, \text{ else} \end{cases}$$

Every learner will make mistakes half the time, so $\forall i$, $E[L_i] = \frac{T}{2}$.

No matter what algorithm we use, it will make mistakes half the time, so $E[L_A] = \frac{T}{2}$.

Although it might seem that $E[\min_i L_i] = \frac{T}{2}$ by similar logic, it will actually be less. This is because out of many experts, it is likely the best one got lucky on the random distribution and achieved better than even error. The value of this expectation is known and we won't take time to prove it. It is: $E[\min_i L_i] \approx \frac{T}{2} - \sqrt{\frac{T\ln(N)}{2}}$.

Therefore, $E[L_A] \gtrsim E[\min_i L_i] + \sqrt{\frac{T\ln(N)}{2}}$ and we have achieved the desired bound in expectation. This shows that the regret bound for this algorithm is the very best possible. One reason this is significant is that it demonstrates that in this model we do just as well in a stochastic setting as we do in an adversarial one.

2 New model (committees)

2.1 Setup

Thus far we have been judging the performance of algorithms against the performance of the best expert. But what if there is no single expert that performs well, but there is a subcommittee that performs well when they vote together?

This scenario can be formalized with the following model. Here we will interpret each component i of input vector $\mathbf{x}_t$ as an expert. Note that we are using the labels $\{-1, +1\}$ instead of $\{0, 1\}$.

- $N = \#$ experts
- For $t = 1, 2, \ldots T$
  - get $\mathbf{x}_t \in \{-1, +1\}^N$
  - learner predicts $\hat{y}_t \in \{-1, +1\}$
  - observe outcome $y_t \in \{-1, +1\}$
- Assume there is a vector $\mathbf{u} \in \mathbb{R}^n$ such that $y_t = \text{sign}(\mathbf{u} \cdot \mathbf{x}_t)$ for all t. This vector $\mathbf{u}$ defines a subcommittee that will predict every outcome correctly under a weighted vote.

There are many potential ways to output $\hat{y}_t$ at each time step. We will focus on algorithms that maintain a weight vector $\mathbf{w}_t$ and use it to compute a weighted vote, or linear threshold function: $\hat{y}_t = \text{sign}(\mathbf{w}_t \cdot \mathbf{x}_t)$. An important observation we should make is that the problem of learning a subcommittee is just an example. More generally, the examples $\mathbf{x}_t$ are really just points in $\mathbb{R}^N$, and we are supposing that there is a linear threshold function defined by a vector $\mathbf{u}$ that correctly classifies all of the examples.

- initialize $\mathbf{w}_1 = \mathbf{0}$
- For $t = 1, 2, \ldots T$
  - get $\mathbf{x}_t \in \mathbb{R}^N$
  - learner predicts $\hat{y}_t = \text{sign}(\mathbf{w}_t \cdot \mathbf{x}_t) \in \{-1, +1\}$
  - observe outcome $y_t \in \{-1, +1\}$
  - update: $\mathbf{w}_{t+1} = F(\mathbf{w}_t, \mathbf{x}_t, y_t)$

2.2 Perceptron

The first such algorithm we will look at is called the Perceptron algorithm. It's been around for a while, since the 1950's. This is how it works:

- initialize $\mathbf{w}_1 = \mathbf{0}$
- For $t = 1, 2, \ldots T$

  if (learner made mistake) [which is equivalent to $(\hat{y}_t \neq y_t)$ or $(y_t(\mathbf{w}_t \cdot \mathbf{x}_t) \leq 0)$]

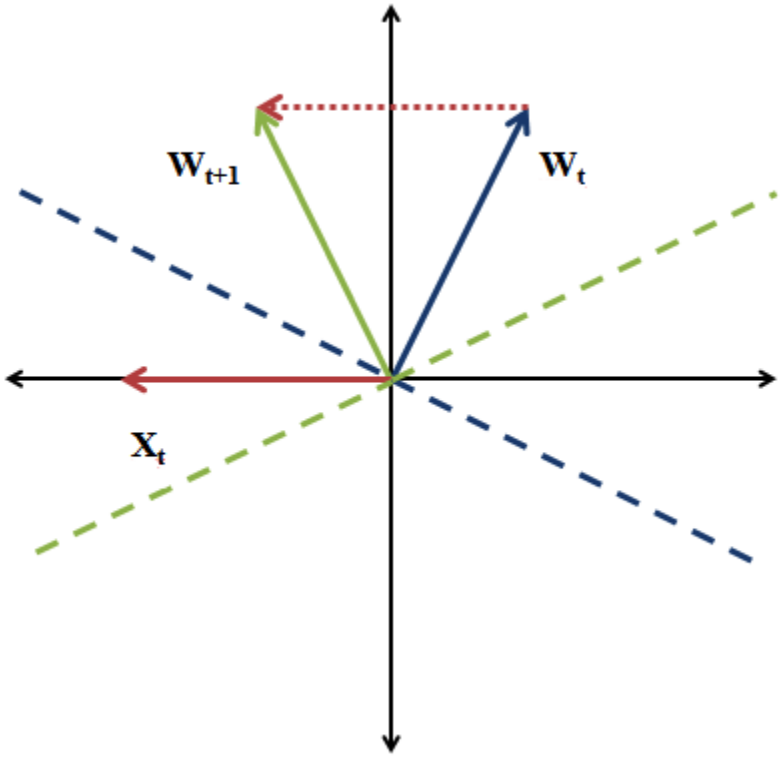
  – $\mathbf{w}_{t+1} = \mathbf{w}_t + y_t \mathbf{x}_t$

  else

  – $\mathbf{w}_{t+1} = \mathbf{w}_t$

In the case where an example is classified correctly, the algorithm doesn't change anything about its state $\mathbf{w}_t$. Intuitively this makes sense; as the adage goes, "if it ain't broke, don't fix it".

In the case where an example is classified incorrectly, the algorithm pushes the hyperplane towards the point if it is positive or away from it if it's negative. This makes sense because it helps the algorithm not repeat similar mistakes. To see why this is the case, let us consult a simple example. Say $\mathbf{w}_t = [\frac{1}{2}, \frac{\sqrt{3}}{2}]$ and example $\mathbf{x}_t = [-1, 0]$ is labelled positively. The new weight will be $\mathbf{w}_{t+1} = \mathbf{w}_t + y_t \mathbf{x}_t = [\frac{1}{2}, \frac{\sqrt{3}}{2}] + [-1, 0] = [-\frac{1}{2}, \frac{\sqrt{3}}{2}]$. This new weight would classify $\mathbf{x}_t$ correctly. This example is illustrated in the figure below.



A natural next step is for us to try to analyze this algorithm. To say anything interesting we need to make some assumptions:

1. The learner makes a mistake at every round. This implies T is equal to the number of mistakes. We can assume this without loss of generality because the algorithm does nothing on rounds on which the learner makes no mistake.

2. $\forall t : ||\mathbf{x}_t||_2 \leq 1$

3. $\exists\, \mathbf{u}, \delta$ such that $||\mathbf{u}||_2 = 1$ and $\forall t$: $y_t(\mathbf{u} \cdot \mathbf{x}_t) \geq \delta > 0$. This means that every example is correctly classified by a margin of at least δ.

Theorem. Under the assumptions above, the Perceptron algorithm makes at most $1/\delta^2$ mistakes.

Proof.

The driver of our proof will be a function Φ_t called the potential function. In physics, a potential function describes how much potential there is for stuff to happen at each time t, which is usually tied to how much energy there is left in a system at time t.

In our case, the potential function is $\Phi_t = \mathbf{w}_t \cdot \mathbf{u}/||\mathbf{w}_t||_2 \leq 1$. It is the cosine similarity between our vector $\mathbf{w}_t$ and the desired vector $\mathbf{u}$. We will show that with every iteration, our cosine similarity grows, and since it can be at most 1, the number of iterations is bounded by some number. Since there is exactly one mistake every round, this number will also

bound the number of mistakes.

Step 1. $\mathbf{w}_{T+1} \cdot \mathbf{u} \geq T\delta$

Proof.

$$\begin{aligned}\mathbf{w}_{t+1} \cdot \mathbf{u} &= (\mathbf{w}_t + y_t \mathbf{x}_t) \cdot \mathbf{u} \\ &= \mathbf{w}_t \cdot \mathbf{u} + y_t(\mathbf{u} \cdot \mathbf{x}_t)\end{aligned}$$

By assumption 3 $y_t(\mathbf{u} \cdot \mathbf{x}_t) \geq \delta$, so with every iteration $\mathbf{w}_t \cdot \mathbf{u}$ increases by at least δ. Since we start at $\mathbf{w}_1 = \mathbf{0}$, by the time the Tth trial has finished we will have $\mathbf{w}_{T+1} \cdot \mathbf{u} \geq T\delta$.

Step 2. $||\mathbf{w}_{T+1}||_2^2 \leq T$

Proof.

$$\begin{aligned}||\mathbf{w}_{T+1}||_2^2 &= \mathbf{w}_{t+1} \cdot \mathbf{w}_{t+1} \\ &= (\mathbf{w}_t + y_t \mathbf{x}_t) \cdot (\mathbf{w}_t + y_t \mathbf{x}_t) \\ &= \mathbf{w}_t \cdot \mathbf{w}_t + 2y_t(\mathbf{w}_t \cdot \mathbf{x}_t) + y_t^2 \mathbf{x}_t \cdot \mathbf{x}_t\end{aligned}$$

Now there's a lot of cleanup we can do here. The first term is $||\mathbf{w}_t||_2^2$. The second term is less than or equal to 0 because of assumption 1 (we make a mistake at each round). The third term is $||\mathbf{x}_t||_2^2 \leq 1$ because no matter what y_t is, $y_t^2 = 1$. Therefore, $||\mathbf{w}_{t+1}||_2^2 \leq ||\mathbf{w}_t||_2^2 + 1$. In similar fashion to step 1, since we know $||\mathbf{w}_1|| = 0$ and $||\mathbf{w}_t||^2$ increases by at most 1 every iteration, we know $||\mathbf{w}_{T+1}||^2 \leq T$.

Putting steps 1 and 2 together yields: $1 \geq \Phi_{T+1} \geq \frac{T\delta}{\sqrt{T}} = \delta\sqrt{T} \implies T \leq 1/\delta^2$. Since there is one mistake every round: $T = \#$ mistakes $\leq 1/\delta^2$, as desired.

An interesting consequence of this is begotten by the fact that an algorithm must make at least as many mistakes as the VC-dimension of its hypothesis class. If we let $\mathcal{H}$ be the hypothesis class of the perceptron, then:

$$\text{VC-dim}(\mathcal{H}) \leq \#\text{ mistakes} \leq 1/\delta^2$$

So the VC-dimension of the algorithm's hypothesis class can be at most $1/\delta^2$.

Let's take a moment to appreciate the significance of this statement. The hypothesis space of the perceptron is the hypothesis space of linear threshold functions with margin at least δ, exactly the same class we looked at for SVM's. At that time, it was claimed without proof that the VC-dimension of this class is at most $1/\delta^2$. Now we see that this bound follows immediately as a corollary from the analysis of the perceptron algorithm.

This new model was motivated by the scenario where there is a good subcommittee and we want to select it. An implicit assumption here was that the subcommittee does not consist of all the experts. Thus far we have not considered the explicit exclusion of certain experts.

Let us consider this case now, it will look something like this:

$\mathbf{x}_t = \frac{1}{\sqrt{N}}\Big[+1, -1, -1, +1, \dots\Big]$, all the experts report their predictions

$\mathbf{u} = \frac{1}{\sqrt{k}}\Big[0, 0, 1, 1, 0, 1, \dots\Big]$, k experts form the subcommittee

The input vector and subcommittee selection vector are normalized to have 2-norm 1. Let it be that the subcommittee that is selected always gets the right answer. This means $y_t(\mathbf{u} \cdot \mathbf{x}_t) \geq 1/\sqrt{Nk}$, because this is the smallest positive value the $\mathbf{u} \cdot \mathbf{x}_t$ can take.

This is a margin that satisfies assumption 3 in our prior scenario, where $\delta = 1/\sqrt{Nk}$. The theorem we just proved then implies that the number of mistakes mistakes is at most Nk. This is a useful bound when N and k are not too large, but it is not useful when N gets very large. In many applications this can be the case, because N corresponds to the number of features in our inputs.

It is for this reason that the Winnow algorithm was invented. It is similar to both the Perceptron and AdaBoost, and we will overview it now. A disclaimer: Winnow refers to a family of algorithms, and we are looking at just one version.

2.3 Winnow

The algorithm is as follows:

- initialize $\forall i$ $\mathbf{w}_{1,i} = 1/N$
- For $t = 1, 2, \dots T$

  if (learner made mistake)

  – $\forall i$: $\mathbf{w}_{t+1,i} = \mathbf{w}_{t,i}\, e^{\eta y_i x_{t,i}} / \mathcal{Z}_t$ [$\eta > 0$ is a parameter]

  else

  – $\mathbf{w}_{t+1} = \mathbf{w}_t$
- Note that $\mathcal{Z}_t$ is a normalization factor to make sure each $\mathbf{w}_t$ sums to 1.

We can prove a bound on the number of mistakes made by this algorithm, under a similar set of assumptions as before. In this class we are running out of time so the theorem will only be stated, then in the next class we will actually prove it.

The assumptions are:

1. The learner makes a mistake at every round. This implies $T = \#$ mistakes.
2. $\forall t : ||\mathbf{x}_t||_\infty \leq 1$
3. $\exists\, \mathbf{u}, \delta$ such that $||\mathbf{u}||_1 = 1$, $\forall i$: $u_i \geq 0$, and $\forall t$: $y_t(\mathbf{u} \cdot \mathbf{x}_t) \geq \delta > 0$

Theorem.

Under the assumptions above, the Winnow algorithm has # mistakes $\leq \dfrac{\ln(N)}{\eta\delta + \ln(2/(e^\eta + e^{-\eta}))}$

If we set $\eta = \frac{1}{2}\ln((1+\delta)/(1-\delta))$ then the bound is at most $2\ln(N)/\delta^2$.

Also, going back to the example with the subcommittee, we now normalize the vectors differently: $\mathbf{x}_t = [+1, -1, -1, \dots]$, $\mathbf{u} = \frac{1}{k}[0, 0, 1, \dots]$. This gives $\delta = \frac{1}{k}$ so the mistake bound we get in this case is $2k^2 \ln(N)$.