COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Matthew Halbasch

Lecture # 17
April 09, 2018

---

1 Analysis of Winnow

In the last lecture, we studied two different online learning algorithms for learning Linear Threshold functions. We looked at the Perceptron algorithm, and proved a theorem about the number of mistakes it can make. At the end of lecture, we introduced Winnow, which we outlined as:

- Initialize $w_{1,i} = 1/N$ for every i.
- for $t = 1, \ldots, T$:
  - if learner makes mistake:
    - $w_{t+1,i} = \frac{1}{Z_t} w_{t,i} \exp(\eta y_i x_{t,i})\ \forall i$, where Z_t is a normalization factor.
  - else:
    - $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t$.

As usual, the prediction $\hat{y}_t$ of our algorithm on round t is $\text{sign}(\boldsymbol{w}_t \cdot \boldsymbol{x}_t)$.

We stated a theorem on the number of mistakes of Winnow, which we will prove now:

Theorem 1.1. *Assume that we run Winnow for T rounds, and assume that*

1. *Winnow makes a mistake on every round,*
2. *$\forall t$, $||\boldsymbol{x}_t||_\infty \leq 1$,*
3. *$\exists\, \delta > 0, \boldsymbol{u}$ such that $\forall t$, $y_t(\boldsymbol{u} \cdot \boldsymbol{x}_t) \geq \delta$, with $||\boldsymbol{u}||_1 \leq 1$ and $u_i \geq 0\ \forall i$.*

Then,

$$\begin{aligned}(\#\ \textit{mistakes of Winnow}) &\leq \frac{\ln(N)}{\eta\delta + \ln\left(\frac{2}{e^{\eta}+e^{-\eta}}\right)} \\ &\leq \frac{2\ln N}{\delta^2} \quad \textit{if } \eta = \frac{1}{2}\ln\left(\frac{1+\delta}{1-\delta}\right) .\end{aligned}$$

Proof. As with the analysis of Perceptron, we want to introduce a potential function Φ which gives useful upper and lower bounds. This time, we are comparing the vector $\boldsymbol{w_t}$ to some "true" weight vector $\boldsymbol{u}$, with the condition that both define probability distributions. One natural measure for the "distance" between probability distributions is the KL-divergence, or relative entropy. We will choose this to be our potential function:

$$\Phi_t = \mathrm{RE}\left(\boldsymbol{u} \parallel \boldsymbol{w}_t\right),$$

where the relative entropy between discrete distributions P and Q is given by

$$\mathrm{RE}\left(P \parallel Q\right) = \sum_i P_i \ln\left(\frac{P_i}{Q_i}\right).$$

The idea for this proof will be to look at how much Φ changes from round to round. Since the relative entropy is always nonnegative, this will provide a bound on the number of rounds the algorithm can run, and thus the number of mistakes it can make (since we make a mistake every round).

So, the first thing to look at is the difference between Φ_{t+1} and Φ_t:

$$\begin{aligned}
\Phi_{t+1} - \Phi_t &= \mathrm{RE}\left(\boldsymbol{u} \parallel \boldsymbol{w}_{t+1}\right) - \mathrm{RE}\left(\boldsymbol{u} \parallel \boldsymbol{w}_t\right) \\
&= \sum_i u_i \ln\left(\frac{u_i}{w_{t+1,i}}\right) - \sum_i u_i \ln\left(\frac{u_i}{w_{t,i}}\right) \\
&= \sum_i u_i \ln(u_i) - \sum_i u_i \ln(w_{t+1,i}) - \sum_i u_i \ln(u_i) + \sum_i u_i \ln(w_{t,i}) \\
&= \sum_i u_i \ln\left(\frac{w_{t,i}}{w_{t+1,i}}\right).
\end{aligned}$$

Conveniently, now, our update rule is multiplicative, so the ratio $w_{t,i}/w_{t+1,i}$ is easy to calculate. Explicitly, our update rule is

$$w_{t+1,i} = \frac{w_{t,i}\exp(\eta y_t x_{t,i})}{Z_t},$$

and the ratio $w_{t,i}/w_{t+1,i}$ is

$$\frac{w_{t,i}}{w_{t+1,i}} = Z_t \exp(-\eta y_t x_{t,i}).$$

This tells us that the difference $\Phi_{t+1} - \Phi_t$ is given by

$$\begin{aligned}\Phi_{t+1} - \Phi_t &= \sum_i u_i \ln(Z_t) - \sum_i u_i \eta y_t x_{t,i} \\ &= \ln(Z_t) \cdot 1 - \eta\, y_t \left(\sum_i u_i x_{t,i} \right) .\end{aligned}$$

The sum in parentheses is just $\boldsymbol{u} \cdot \boldsymbol{x}_t$, however, and we have assumed that $y_t(\boldsymbol{u} \cdot \boldsymbol{x}_t) \geq \delta$, so we can now bound this difference:

$$\Phi_{t+1} - \Phi_t \leq \ln(Z_t) - \eta\delta\,.$$

So, we need to find a bound on Z_t now. It is a normalization constant on round t, which is given by the sum of numerators:

$$Z_t = \sum_i w_{t,i} \exp(\eta y_t x_{t,i})\,.$$

We can bound each of these exponentials by using the fact that the function $e^{\eta z}$ is convex. The product $y_t x_{t,i}$ is between -1 and 1, so we want to bound $e^{\eta z}$ in the region $[-1, 1]$. See Figure 1 for a graphical depiction of this.

We find that we can bound the exponential by

$$\begin{aligned}e^{\eta z} &\leq \frac{e^{\eta} + e^{-\eta}}{2} + \frac{e^{\eta} - e^{-\eta}}{2} z \\ &= \cosh(\eta) + z \sinh(\eta)\,.\end{aligned}$$

Using $z = y_t x_{t,i}$, we can bound Z_t:

$$\begin{aligned}Z_t &\leq \sum_i \left(w_i \left[\cosh(\eta) + y_t x_{t,i} \sinh(\eta) \right] \right) \\ &= \cosh(\eta) \sum_i w_{t,i} + \sinh(\eta) y_t \left(\sum_i w_{t,i} x_{t,i} \right) .\end{aligned}$$

Now, the first sum is the sum of $w_{t,i}$ over all i, but since $\boldsymbol{w}_t$ defines a probability distribution, this is just 1. The second sum is $\boldsymbol{w}_t \cdot \boldsymbol{x}_t$, which has the same sign as $\hat{y}_t$ by definition. But since we make a mistake on every round $\hat{y}_t \neq y_t$, which means their product is nonpositive, and so $y_t(\boldsymbol{w}_t \cdot \boldsymbol{x}_t) \leq 0$. Finally, since $\eta \geq 0$, $\sinh(\eta) \geq 0$, and so the bound simply becomes

$$Z_t \leq \cosh(\eta) = \frac{e^{\eta} + e^{-\eta}}{2}\,.$$

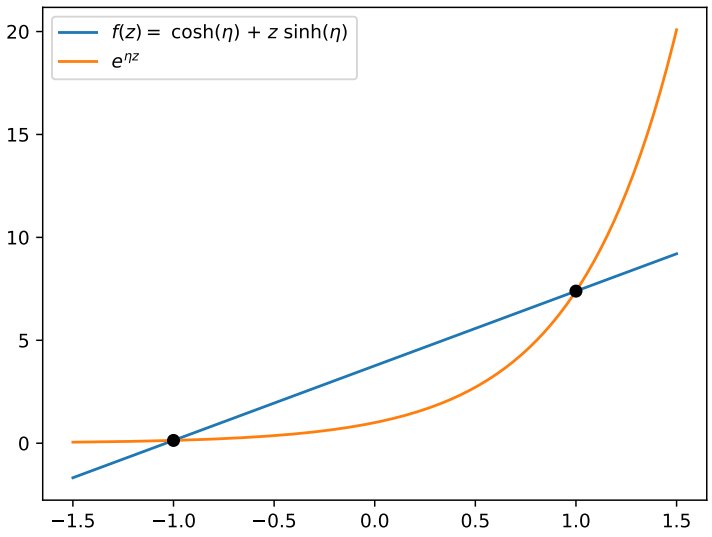

Figure 1: Depiction of a linear bound for $e^{\eta z}$ for $z \in [-1, 1]$. The bound is the line connecting $(-1, e^{-\eta})$ with $(1, e^{\eta})$, and is given by $f(z) = \cosh(\eta) + z\sinh(\eta)$.

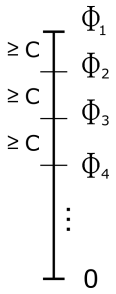

Figure 2: An illustration of how the potential drops on each round.

So, going back to Φ, we can obtain a lower bound for how much it must drop each round (which we will call C) in terms of just η and δ. Explicitly,

$$\Phi_{t+1} - \Phi_t \leq \ln\left(\frac{e^{\eta} + e^{-\eta}}{2}\right) - \eta\delta =: -C\,.$$

So, we know that Φ decreases by at least C every round, and so after T rounds it must decrease by $C \cdot T$ (see the illustration in Figure 2 for a visual). The number of rounds that the algorithm can run is then determined by Φ_1, which we can also bound, since we initialized $w_{1,i} = 1/N\ \forall i$ to be uniform:

$$\begin{aligned}
\Phi_1 &= \mathrm{RE}\left(\boldsymbol{u} \parallel \boldsymbol{w}_1\right) \\
&= \sum_i u_i \ln\left(\frac{u_i}{1/N}\right) \\
&= \sum_i u_i \ln(N u_i) \\
&\leq \sum_i u_i \ln N \\
&= \ln N\,,
\end{aligned}$$

where we have used the fact that $u_i \leq 1\,\forall i$ in the fourth line. So, as we have argued, this gives a bound on the number of rounds the algorithm can run:

$$T \leq \frac{\ln N}{C}\,.$$

Using the definition of C, and noting that the number of mistakes is equal to the number of rounds, we have

$$\#\text{ mistakes} \leq \frac{\ln N}{\eta\delta + \ln\left(\frac{2}{e^{\eta}+e^{-\eta}}\right)}\,,$$

which is the bound we wanted to prove. As a quick note, if we make an appropriate choice of η,

$$\eta = \frac{1}{2}\ln\left(\frac{1+\delta}{1-\delta}\right),$$

then the bound on the number of mistakes becomes

$$\#\text{ mistakes} \leq \frac{\ln N}{\mathrm{RE}\left(\frac{1}{2} - \frac{\delta}{2} \,||\, \frac{1}{2}\right)} \leq \frac{2\ln(N)}{\delta^2}\,,$$

which we can get by bounding the relative entropy. □

2 Extending Winnow

For this proof, we assumed that all $u_i \geq 0$. In general, we may want to relax this condition, and there is an easy way to do it. To describe it, consider an example where

$$\begin{aligned} \boldsymbol{u} &= (.5,\ -.2,\ .3)\,, \\ \boldsymbol{x} &= (1,\ .7,\ -.4)\,. \end{aligned}$$

We can map these to two 6 dimensional vectors $\boldsymbol{x}'$ and $\boldsymbol{u}'$ according to

$$\begin{aligned} \boldsymbol{u} \mapsto \boldsymbol{u}' &= (.5,\ 0,\ .3,\ \vdots\ 0,\ .2,\ 0)\,, \\ \boldsymbol{x} \mapsto \boldsymbol{x}' &= (1,\ .7,\ -.4,\ \vdots\ -1,\ -.7,\ .4)\,. \end{aligned}$$

That is, we place in the first half of $\boldsymbol{u}'$ the positive elements of $\boldsymbol{u}$, and the negative elements (without the negative sign) in the second half, and fill in zeros everywhere else. In $\boldsymbol{x}'$, we place $\boldsymbol{x}$ in the first half, and $-\boldsymbol{x}$ in the second half. This construction preserves the dot product:

$$\boldsymbol{u} \cdot \boldsymbol{x} = \boldsymbol{u}' \cdot \boldsymbol{x}'\,,$$

and we still have $||\boldsymbol{u}||_1 = ||\boldsymbol{u}'||_1$ and $||\boldsymbol{x}||_\infty = ||\boldsymbol{x}'||_\infty$. So, all the assumptions of our theorem are satified, and we can run Winnow on these new vectors. The resulting algorithm is called *balanced Winnow*.

The only added complexity from this is it changes $N \to 2N$, but since the number of mistakes is logarithmic in N, this is not a large change.

3 Introduction to Regression

Up to now, we have only worried about predicting a yes or no, and have looked at the probability of being wrong, or the number of times we are wrong. Often, however, predictions are *probabilistic* in nature, and we instead want to predict the *chance* of an outcome (e.g. the chance of rain).

How do we analyze these predictions? For example, if we have two forecasts, and one predicts a 70% chance of rain, and the other predicts an 80% chance, how do we compare them if it does rain?

To formalize this, we can define x to be the weather conditions, and $y = 1$ if it rains, and 0 otherwise. Assume that $(x, y) \sim D$ and assume that these are i.i.d. We want to estimate $p(x) = \Pr[y = 1|x] = E[y|x]$. (This last formulation is more general, and allows us to predict e.g. the number of inches of rain we expect). These kinds of problems are

called *regression* problems, as opposed to the classification problems we have considered so far in this class. We are now trying to estimate a real valued number.

In our example, let's assume that, say, AccuWeather uses $h_1(x)$ to estimate $p(x)$, and the National Weather Service uses $h_2(x)$. We want to see if $h_1(x)$ or $h_2(x)$ predicts $p(x)$ better, but of course we don't actually know $p(x)$ either. Instead, we have to rely on the observed outcomes alone, and find a way to "score" the predictions.

To do this, we consider a "loss" function, which provides a measure of how to "score" an error. In our first look, we will use a *square* or *quadratic* loss function, which is simply the square of the difference between a predicted probability or expectation and its realized value:

$$L = (h(x) - y)^2 .$$

We want to minimize the expected value of our loss (called the *risk*):

$$\mathrm{E}_{(x,y)\sim D}\left[(h(x) - y)^2\right] .$$

In the earlier parts of this course, our loss function was just the indicator variable $\mathbb{1}\{h(x) \neq y\}$.

So why does quadratic loss work? We will show that by minimizing the risk, we are actually forcing $h(x)$ to be as close as possible to $p(x)$. To prove this, consider a single x, and define $p = p(x) = E[y|x]$, and $h = h(x)$. Then, the loss is

$$\begin{aligned}\mathrm{E}[(h-y)^2] &= p(h-1)^2 + (1-p)(h-0)^2 \\ &= p(h-1)^2 + (1-p)h^2 .\end{aligned}$$

To minimize this, we take a derivative and set it equal to 0:

$$\begin{aligned}\frac{d}{dh}\mathrm{E}[(h-y)^2] &= 2p(h-1) + 2(1-p)h \\ &= 2ph - 2p + 2h - 2ph \\ &= 2(h-p) = 0 ,\end{aligned}$$

which is solved when $h = p$.

So, if we can minimize the loss exactly, we must have $h(x) = p(x)$. But what about a more general case, where we are just able to get close to $p(x)$? Here we have a theorem:

Theorem 3.1.

$$\mathrm{E}\left[(h(x) - p(x))^2\right] = \mathrm{E}\left[(h(x) - y)^2\right] - \mathrm{E}\left[(p(x) - y)^2\right] ,$$

where all expectations are over both x and y

Proof. To understand this theorem, the term on the left is what we want to make small. The first term on the right is something we can actually measure, since it depends only on y, not $p(x)$. The second term on the right is a measure of inherent randomness in y, and doesn't depend on h at all.

To actually prove this, we will first fix x, and take an expected value at the end. This is using marginalization, which says

$$\mathrm{E}_x\left[\mathrm{E}_y\left[\cdots|x\right]\right] = \mathrm{E}_{x,y}\left[\cdots\right].$$

So, if x is fixed, we again define $h = h(x)$, $p = p(x) = \mathrm{E}[y|x] = \mathrm{E}[y]$. Then, the left hand side is just

$$\mathrm{E}\left[(h-p)^2\right] = (h-p)^2,$$

since there is no y dependence here. On the right hand side, we have

$$\begin{aligned}
\text{RHS} &= \mathrm{E}\left[(h-y)^2\right] - \mathrm{E}\left[(p-y)^2\right] \\
&= \mathrm{E}\left[(h^2 - 2hy + y^2) - (p^2 - 2py + y^2)\right] \\
&= \mathrm{E}\left[h^2 - 2hy - p^2 + 2py\right] \\
&= h^2 - 2h\,\mathrm{E}[y] - p^2 + 2p\,\mathrm{E}[y] \\
&= h^2 - 2hp - p^2 + 2p^2 \\
&= (h-p)^2,
\end{aligned}$$

where in the second to last line we have used $\mathrm{E}[y] = p$. This is exactly what we got for the LHS, so the two are equal, which proves the theorem.

□

So, now we have justification for minimizing the risk according to this quadratic loss function. The idea from the first half of class would be to draw m samples $(x_1, y_1), \ldots, (x_m, y_m)$ from D, and minimize the *sample* loss,

$$\hat{\mathrm{E}}\left[(h(x)-y)^2\right] = \frac{1}{m}\sum_{i=1}^{m}(h(x_i)-y_i)^2.$$

Under this approach, we would consider a class of predictors $h \in \mathcal{H}$, and define the loss $L_h(x,y) = (h(x)-y)^2$, and estimate

$$\mathrm{E}\left[L_h\right] \approx \hat{\mathrm{E}}\left[L_h\right].$$

Then we would look for uniform convergence results, which can be found in a similar fashion to what we did earlier in the class. However, we will take a different approach here, and instead consider particular cases, as well as algorithms specific to those cases.

In particular, let's look at cases where we have *linear* functions of $\boldsymbol{x}$. That is, $\boldsymbol{x} \in \mathbb{R}^n$, and $h(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x}$. In this case, the empirical risk becomes

$$\frac{1}{m} \sum_{i=1}^{m} (\boldsymbol{w} \cdot \boldsymbol{x_i} - y_i)^2 \,,$$

and we want to find the vector $\boldsymbol{w}$ that minimizes this equation. This is just linear regression!

Linear regression is a long-studied problem, and we know of analytical solutions, but we'll take a different approach here, and consider how to use online learning to solve this problem.

In particular, let's use an online learning model, and follow an algorithm like

- Initialize $\boldsymbol{w}_1$
- for $t = 1, \ldots, T$:
  - get $\boldsymbol{x}_t$
  - predict $\hat{y}_t = \boldsymbol{w}_t \cdot \boldsymbol{x}_t \in \mathbb{R}$
  - observe $y_t \in \mathbb{R}$
  - Loss $= (\hat{y}_t - y_t)^2$
  - update $\boldsymbol{w}_t$
- $L_A = \sum_{t=1}^{T} (\hat{y}_t - y_t)^2$

In this algorithm outline, y_t may represent the probability of rain, or the expected number of inches of rain, etc. We will not make any statistical assumptions about the data here.

4 Conclusion

Next time, we will consider how to analyze these kinds of algorithms. We always need something to compare the performance of our algorithm to, and in this case we will compare it to $\boldsymbol{u}$, the best linear predictor. In particular, we will compare our loss to that of u:

$$L_{\mathbf{u}} = \sum_{t=1}^{T} (\boldsymbol{u} \cdot \boldsymbol{x}_t - y_t)^2 \,.$$

We will aim for a result of the form

$$L_A \leq \min_{\boldsymbol{u}} L_{\boldsymbol{u}} + \text{small amount} \,,$$

where the small amount is once again called the *regret*.