COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Tim Alberdingk Thijm

Lecture #17
April 8, 2019

1 Introduction

Last time, we looked at the perceptron and winnow algorithms. Note that these algorithms are quite simple while not making stochastic assumptions!

Between the two algorithms, we noted that winnow is better than perceptron in certain cases, for instance, when N (the number of experts) is large and we assume the label is a majority vote of some "subcommittee" of k experts. For this lecture, we'll start by analyzing winnow.

2 Analysis of the Winnow Algorithm

All we need to do going from our analysis of perceptron to winnow is change the vector norms. This parallels the differences in norms between boosting and SVMs.

2.1 Assumptions for Winnow

- We make a mistake every round.
- $\forall t: \ \|\mathbf{x}_t\|_\infty \leq 1$
- $\exists \delta, \mathbf{u} \in \mathbb{R}^N : \|\mathbf{u}\|_1 = 1, \forall i: \ u_i \geq 0, \forall t: \ y_t(\mathbf{u} \cdot \mathbf{x}_t) \geq \delta > 0$ (note that we will later get rid of the assumption that the elements of $\mathbf{u}$ are non-negative)

2.2 Bounding winnow mistakes

Given these assumptions, we can prove the following bound on the number of mistakes made by winnow.

Theorem 2.1. *The number of mistakes for winnow is at most:*

$$\frac{\ln N}{\eta\delta + \ln\left(\frac{2}{e^\eta + e^{-\eta}}\right)}$$

When $\eta = \frac{1}{2}\ln(\frac{1+\delta}{1-\delta})$*, the number of mistakes is at most:*

$$\frac{2\ln N}{\delta^2}$$

As an example, let y_t be the majority vote of k out of N experts. Let $\mathbf{x}_t$ be some vector whose elements are all $+1$ or -1, here representing the predictions of N experts. Then $\|\mathbf{x}_t\|_\infty \leq 1$. Let $\mathbf{u}$ be some vector representing the majority vote of k of the experts, that is, its elements are all 0 or $\frac{1}{k}$, where there are k elements equal to $\frac{1}{k}$ so that $\|\mathbf{u}\|_1 = 1$. For all t, $y_t(\mathbf{u} \cdot \mathbf{x}_t) \geq \frac{1}{k}$. Then $\delta = \frac{1}{k}$.

Then the number of mistakes winnow makes is at most $2k^2 \ln N$.

Proof. The basic idea behind this type of proof is to pick a quantity to focus on and derive upper and lower bounds for it. As with perceptron, we will use a potential function Φ_t.

Let $\Phi_t = RE(\mathbf{u} \parallel \mathbf{w}_t)$. This is so we can measure the distance between vectors like the distance between probability distributions. Recall that:

$$RE(\mathbf{p} \parallel \mathbf{q}) = \sum_i p_i \ln\left(\frac{p_i}{q_i}\right)$$

We want to show that Φ_t always decreases *significantly* and never becomes negative, which will give us a bound on the number of mistakes.

Let's take a look at $\Phi_{t+1} - \Phi_t$.

$$\begin{aligned}
\Phi_{t+1} - \Phi_t &= \sum_i u_i \ln\left(\frac{u_i}{w_{t+1,i}}\right) - \sum_i u_i \ln\left(\frac{u_i}{w_{t,i}}\right) \\
&= \sum_i u_i \ln(u_i) - \sum_i u_i \ln(w_{t+1,i}) - \sum_i u_i \ln(u_i) + \sum_i u_i \ln(w_{t,i}) \\
&= \sum_i u_i \ln\left(\frac{w_{t,i}}{w_{t+1,i}}\right)
\end{aligned}$$

Now we have an expression which is related to the update that got us from $w_{t,i}$ to $w_{t+1,i}$.

$$\begin{aligned}
\Phi_{t+1} - \Phi_t &= \sum_i u_i \ln\left(\frac{Z_t}{\exp(\eta y_t x_{t,i})}\right) \\
&= \sum_i u_i \ln(Z_t) - \sum_i u_i \ln(\exp(\eta y_t x_{t,i})) \\
&= \ln(Z_t) - \sum_i u_i(\eta y_t x_{t,i}) && \textit{since } \sum_i u_i = 1 \\
&= \ln(Z_t) - \eta y_t(\mathbf{u} \cdot \mathbf{x}_t) \\
&\leq \ln(Z_t) - \eta\delta && y_t(\mathbf{u} \cdot \mathbf{x}_t) \geq \delta
\end{aligned}$$

We pause here to compute Z_t, the normalization constant of our w_i variables. For readability, as we are only looking at a particular t, we omit the t subscripts.

$$Z = \sum_i w_i e^{\eta y x_i}$$

Note that $y \in \{-1, +1\}$ and $-1 \leq x_i \leq 1$, so their product z is also between -1 and 1. We can then upper bound $e^{\eta z}$ using a linear equation $\frac{e^\eta + e^{-\eta}}{2} + \frac{e^\eta - e^{-\eta}}{2} z$ (see Figure 1).

$$\begin{aligned}
Z &\leq \sum_i w_i \left[\frac{e^\eta + e^{-\eta}}{2} + \frac{e^\eta - e^{-\eta}}{2} y x_i\right] \\
&= \frac{e^\eta + e^{-\eta}}{2} \sum_i w_i + \frac{e^\eta - e^{-\eta}}{2} y \sum_i (w_i x_i) && \textit{distribute inner terms} \\
&= \frac{e^\eta + e^{-\eta}}{2} + \frac{e^\eta - e^{-\eta}}{2} y(\mathbf{w} \cdot \mathbf{x}) && \textit{since } \sum_i w_i = 1 \\
&\leq \frac{e^\eta + e^{-\eta}}{2} && \textit{see explanation } (*) \textit{ below}
\end{aligned}$$

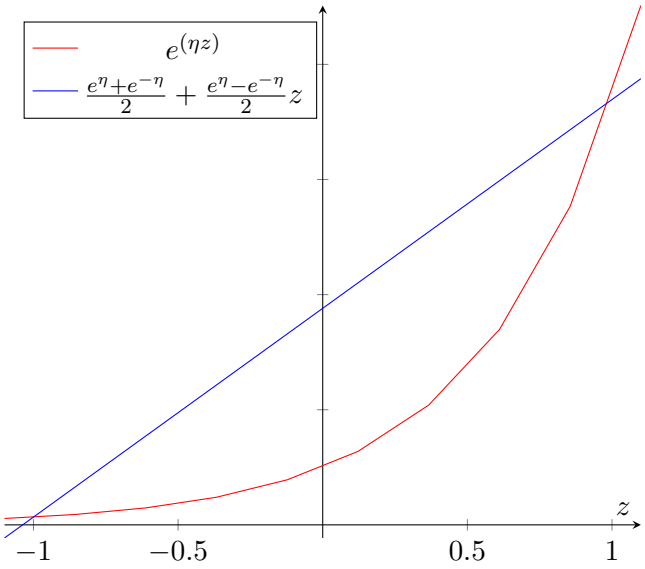

Figure 1: Graph of $e^{\eta z}$ and its bounding linear function

$(*)$: We've assumed a mistake every round, so $y(\mathbf{w} \cdot \mathbf{x}) \leq 0$ while $\frac{e^{\eta}-e^{-\eta}}{2} \geq 0$. Hence, their product is non-positive.

Returning to $\Phi_{t+1} - \Phi_t$:

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

Our potential is then going down by C each round.

Now we note that $\Phi_1 = RE\left(\mathbf{u} \parallel \mathbf{w}_1\right) = \sum_i u_i \ln\left(u_i N\right) \leq \sum_i u_i \ln N = \ln N$.

From the initial potential $\Phi_1 \leq \ln N$, the potential then decreases by at least C every round, and, being a relative entropy, is never negative. Therefore $T \leq \frac{\ln N}{C}$. Minimizing the bound over η, we obtain $\eta = \frac{1}{2}\ln\left(\frac{1+\delta}{1-\delta}\right)$ which gives:

$$\mathcal{C} = RE\left(\frac{1}{2} - \frac{\delta}{2} \parallel \frac{1}{2}\right) \geq 2\left(\frac{\delta}{2}\right)^2$$

Plugging this into T gives the bound stated in the theorem.

□

We can get rid of the assumption that all of our u_i elements are non-negative by mapping our N-dimensional vectors to $2N$ dimensions. We construct a vector $\mathbf{x}'_t$ where elements $0, \ldots, N$ correspond to $\mathbf{x}$ and elements $N+1, \ldots, 2N$ correspond to $-\mathbf{x}$. We construct a vector $\mathbf{u}'$ from $\mathbf{u}$ as follows:

1. If element $u_i \geq 0$, $u'_i = u_i$ and $u'_{i+N} = 0$.
2. If element $u_i < 0$, $u'_i = 0$ and $u'_{i+N} = -u_i$.

This ensures that we have the following properties:

- $\mathbf{x}' \cdot \mathbf{u}' = \mathbf{u} \cdot \mathbf{x}$, which means now we can assume $\forall i: \; u'_i \geq 0$

- $\|\mathbf{x}'\|_\infty = \|\mathbf{x}\|_\infty$
- $\|\mathbf{u}'\|_1 = \|\mathbf{u}\|_1$

While N has been replaced by $2N$, this increase is negligible since our bounds are logarithmic in N.

3 Regression

We've been focussing in the classes so far on classification. For these problems, our goal has been minimizing the number of mistakes our algorithms make.

Now we want to look at a range of learning problems beyond classification.

3.1 Predicting the Weather

For example, what if we wanted to predict the weather? Let's say we have a few meteorological websites which give us predictions (our experts, but now giving percentages!). How would we figure out which of these experts made the better prediction that it will rain? Suppose the National Weather Service predicts a 50% chance of rain, and AccuWeather predicts a 70% chance. If it did in fact rain, who made the better prediction? To answer that question, we'd need to determine the actual probability of it raining, which we can never observe directly.

Let x be today's weather conditions. Let $y = 1$ if it will rain tomorrow, and $y = 0$ otherwise. Then let's say that (x, y) are sampled from some distribution D, ignoring the fact that weather is not at all i.i.d.

In that case, we want to estimate $\Pr[y = 1|x] = p(x) = \mathbb{E}[y|x]$. Using expectation here is somewhat more general: for instance, we can use the same set up when the problem is to predict the expected inches of rain, where y now represents inches of rainfall.

Problems of this form are called *regression* problems.

3.2 Risk Minimization

Let's say we have our 2 experts who give us hypotheses $h_1(x)$ and $h_2(x)$ respectively. We want to find out who is closer to $p(x)$ given only observations of y.

To do so, we use a **loss function**, which can be used in general to measure how well a prediction fits an observed outcome. We were actually using a loss function during classification of the form $\mathbb{1}\{h(x) \neq y\}$. Here, we are instead using square or quadratic loss: $(h(x) - y)^2$. We want to use the loss function to assign a score to every prediction and pick the best one.

We generally are interested in minimizing the expected loss, which we call the **risk or true risk**, and define it as:

$$\mathbb{E}_{(x,y)\sim D}[(h(x) - y)^2]$$

Our goal is to pick h which has the lowest risk.

Which predictor h minimizes the square loss? Let's say we fix a particular x and have $y \in \{0, 1\}$. Then for brevity we write p for $p(x)$ and h for $h(x)$. Then $\mathbb{E}_y[(h - y)^2] = p(h-1)^2 + (1-p)h^2$. Taking the derivative of this term with respect to h and setting it to 0 gives us $h = p$. This tells us that the risk is minimized when $h = p$, and gives us estimates over p.

More generally, we can prove the following theorem.

Theorem 3.1. *For all h,*

$$\mathbb{E}_x[(h(x) - p(x))^2] = \mathbb{E}_{(x,y)\sim D}[(h(x) - y)^2] - \mathbb{E}_{(x,y)\sim D}[(p(x) - y)^2]$$

The middle term is the risk, and the term on the left is a kind of measure of how close $h(x)$ is on average to $p(x)$, something we can never measure. Since the term on the right is constant with respect to h, this theorem shows that minimizing the risk, which is something that can be done approximately from data, is equivalent to minimizing the term on the left, even though it involves quantities $p(x)$ that can never be observed directly. Additionally, it is a kind of expected variance.

Proof. Recall our earlier use of marginalization to state:

$$\mathbb{E}_{x,y}[\ldots] = \mathbb{E}_x[\mathbb{E}_{y|x}[\ldots]]$$

This means that it is sufficient to prove the theorem with x fixed, and then to simply take expectation on both sides with respect to x. Fix x and write h for $h(x)$, p for $p(x)$. Note that $p = \mathbb{E}[y]$.

The left-hand side is $\mathbb{E}[(h-p)^2] = (h-p)^2$, since h and p are constant.

The right-hand side can be expanded as follows:

$$\begin{aligned}
\mathbb{E}[(h-y)^2] - \mathbb{E}[(p-y)^2] &= \mathbb{E}[h^2 - 2hy + y^2] - \mathbb{E}[p^2 - 2py + y^2] \\
&= h^2 - 2h\mathbb{E}[y] - p^2 + 2p\mathbb{E}[y] \\
&= h^2 - 2hp + p^2 && \mathbb{E}[y] = p \\
&= (h-p)^2
\end{aligned}$$

Both sides match and the theorem holds. □

To approximately minimize the true risk $\mathbb{E}_{x,y\sim D}[(h(x) - y)^2]$, if we have a sample $(x_1, y_1), \ldots, (x_m, y_m)$, then we can simply minimize the average risk on the training set, called the empirical risk:

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

This is called empirical risk minimization.

Write $L_h(x, y) = (h(x) - y)^2$ for our loss function. To prove that this works, we would need to prove that $\mathbb{E}[L_h] \approx \hat{\mathbb{E}}[L_h]\ \forall h \in \mathcal{H}$ (cf. uniform convergence results).

One approach to obtain these bounds on empirical versus true risk is to take what we've worked on using Chernoff bounds, VC-dimension, growth functions and so on, and generalize it to regression. We're instead going to focus on using online learning algorithms.

3.3 Linear Regression

Let's start by asking what specific form $h(x)$ can take. One typical case is where we are working with $\mathbf{x} \in \mathbb{R}^N$ and $h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$. Then we want to find $\mathbf{w}$ that minimizes the sum of squared errors $\frac{1}{m}\sum_i(\mathbf{w} \cdot \mathbf{x}_i - y_i)^2$.

This is known as linear regression.

We'll focus on the online version of linear regression.

Figure 2: Online linear regression template

initialize $\mathbf{w}_1$
for $t \leftarrow 1, \ldots, T$ **do**
    observe $\mathbf{x}_t \in \mathbb{R}^N$
    predict $\hat{y}_t = \mathbf{w}_t \cdot \mathbf{x}_t$
    observe $y_t \in \mathbb{R}$
    loss $\leftarrow (\hat{y}_t - y_t)^2$
    update $\mathbf{w}_t$
end for

Here we do not need to assume that the data is random and coming from a distribution D! Although intended for an online setting, such methods often are useful in batch settings as well.

Instead of a total number of mistakes (which was just a kind of loss), we will define the algorithm's total loss L_A for square loss as follows:

$$L_A = \sum_{t=1}^{T} (\hat{y}_t - y_t)^2$$

We also will want to know what loss would be suffered if we always used some particular $\mathbf{u}$:

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

Like before, we want:

$$L_A \leq \min_{\mathbf{u}} L_{\mathbf{u}} + (\textbf{some small regret})$$

We'll leave coming up with how to predict $\hat{y}_t$ and update $\mathbf{w}_t$ for next time.