1 Winnow

review from last time

\[ \eta > 0 \quad \text{← learning rate} \]
\[ \vec{w}_{1,i} = 1/N \quad \text{← Initial distribution} \]
for \( t = 1, 2, \ldots T \) \( \text{← T steps} \)
get \( \vec{x}_t \in \mathbb{R}^n \)
predict \( \hat{y}_t = sign(\vec{w}_t \cdot \vec{x}_t) \)
Make prediction for the current step
observe \( y_t \in \{-1, 1\} \)

(update:)
if \( y_t = \hat{y}_t \) then \( \vec{w}_{t+1} = \vec{w}_t \)
We got it right, so we don’t do any updating
else

\[ w_{t+1,i} = \frac{w_{t,i} e^{\eta y_t x_{t,i}}}{Z_t} \]

Equation 1 has the property that if the sign of \( y_t x_{t,i} \) is positive, then it will increase \( w_{t+1,i} \), and if the sign is negative, it will decrease it.

1.1 Analysis

Assume \( ||\vec{x}_t||_\infty \leq 1 \) 
note: \( L_\infty \) norm is the maximum absolute value of any component
\[ \exists \delta > 0, \vec{u} \in \mathbb{R}^n \text{ st} \]
\[ \forall t \ y_t(\vec{u} \cdot \vec{x}_t) \geq \delta \quad \text{← for all examples, margin is at least } \delta \]
\[ ||\vec{u}||_1 = 1 \quad \text{← Sum of the absolute value of all components of } u \text{ is } 1. \]
\[ u_i \geq 0 \]
Thm:

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

Solving for minimum value for Equation 2, we get:

\[ \# \text{ mistakes} \leq \frac{2 \ln N}{\delta^2} \quad \text{if } \eta = \frac{1}{2} \ln \left( \frac{1 + \delta}{1 - \delta} \right) \]
\subsection{Proof}

Measure of progress - how close $\vec{w}_t$ (predicted weights) is to $\vec{u}$ (actual weights).

$\Phi$ = Potential function of measure of progress

Since both $\vec{u}$ and $\vec{w}_t$ are probability distributions, we use Relative Entropy (RE):

$$\Phi_t = RE(\vec{u} \| \vec{w}_t) = \sum_p p_i \ln \frac{p_i}{q_i}$$

(4)

try to prove every time makes a mistake $\Phi$ drops by some amount. Since RE always $\geq 0$, this gives a bound on the total number of mistakes.

Since nothing happens when the algorithm does not make a mistake, we assume that it makes a mistake on every round.

$$\Phi_{t+1} - \Phi_t = \sum_i u_i \ln \frac{u_i}{w_{t+1,i}} - \sum_i u_i \ln \frac{u_i}{w_{t,i}}$$

(5)

$$\ln\left(\frac{u_i}{w_{t+1,i}}\right) = \ln u_i - \ln w_{t+1,i} \quad \text{and} \quad \ln\left(\frac{u_i}{w_{t+1,i}}\right) = \ln u_i - \ln w_{t,i}$$

(6)

Given Equation 5 and 6, you get 7:

$$\Phi_{t+1} - \Phi_t = \sum_i u_i \ln \frac{w_{t,i}}{w_{t+1,i}} = \sum_i u_i \ln \frac{Z_t}{e^{\eta y_t x_{t,i}}}$$

(7)

$$= \sum_i u_i \ln Z_t - \sum_i u_i \eta y_t x_{t,i}$$

(8)

$$= \ln Z_t - \eta y_t (\vec{u} \cdot \vec{x}_t)$$

(9)

We know that $y_t(\vec{u} \cdot \vec{x}_t) \geq \delta$ and that

$$Z_t = \sum_i w_i e^{\eta y_t x_{t,i}}$$

(10)

So how do we upper bound an exponential?

We upperbound the exponential by a linear as shown in Figure 1.

The new equation using the linear bound is:

$$Z_t \leq \sum_i w_i \left[ \left(1 + \frac{y x_i}{2}\right) e^{\eta} + \left(1 - \frac{y x_i}{2}\right) e^{-\eta} \right]$$

(11)

$$\leq \left(\frac{e^{\eta} + e^{-\eta}}{2}\right) \sum_i w_i + \left(\frac{e^{\eta} + e^{-\eta}}{2}\right) \sum_i w_i y x_i$$

(12)
Since: \( \sum w_i = 1 \), \( \frac{e^\eta + e^{-\eta}}{2} > 0 \) and since we made a mistake, \( y_t(\vec{w}_t \cdot \vec{x}_t) \leq 0 \), we can conclude that right half is always negative, and hence the bound from Equation 12 is:

\[ Z_t \leq \frac{e^\eta + e^{-\eta}}{2} \]  

Thus:

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

We define: 
\[ c = \ln \left( \frac{e^\eta + e^{-\eta}}{2} \right) - \eta \delta \]

Continuing:

\[ \Phi_1 = \text{RE}(\vec{u} || \vec{w}) = \sum_i u_i \ln(u_i N) \leq \sum_i u_i \ln N = \ln N \]  

Thus the first round: \( \Phi_1 \) has an upperbound of \( \ln N \), and each additional round this value must drop by \( c \), as shown by Equation 14.

Hence, the maximum number of mistakes is: \( \leq \frac{\ln N}{c} \).

If \( \eta = \frac{1}{2} \ln(\frac{1+\delta}{1-\delta}) \) then: \( c = \text{RE}(\frac{1}{2} \cdot \frac{\delta}{2} || \frac{1}{2}) \) which \( \geq 2(\frac{\delta}{2})^2 = \frac{\delta^2}{2} \)

Summary:

For perceptron: \( \frac{1}{\delta^2} \to Nk \) mistakes for \( k \) experts.

For Winnow: \( \rightarrow 2k^2 \ln N \), which is better when \( k << N \).

1.3 What about the constraint \( u_i \geq 0 \)

Until now, we assumed that \( \vec{u} \) is all positive, so how do we permit components of \( \vec{u} \) to be negative, or to correspond with negative values, without causing math problems later?

The solution is to duplicate the components of \( \vec{u} \), but make the right half (the duplicates) negative, and to have \( \vec{u} \) broken into two halves, one for the positive components, and one for the negative components.

For example: let’s say we wanted the following:
\[ \vec{x} = (1, .7, -.4) \quad \vec{u} = (.5, 2, -.3) \]

We would duplicate and invert the sign of the components of \( \vec{x} \), so:
\[ \vec{x} = (1, .7, -.4) \rightarrow (1, .7, -4, -1, -7, .4) \]

For \( \vec{u} \) we zero out the negative components on the left, and zero out the positive components on the right as shown:
\[ \vec{u} = (.5, 2, -.3) \rightarrow (.5, 2, 0, 0, .3) \]

This results in the same dot product as if you used your original values for \( \vec{u} \) and \( \vec{x} \). The resulting algorithm is called the “balanced winnow” algorithm, and is accomplished by doubling the number of weights as described above.

2 Estimating Probabilities of Predictions

Previous classification learning problems the goal was to minimize the probability of making a mistake. The question is how do we estimate the probability of a given prediction.

For example:

- \( x \) is the current weather conditions, and \( y \) is the prediction for tomorrow.

\[
y = \begin{cases} 
1 & \text{if rain tomorrow} \\
0 & \text{otherwise}
\end{cases}
\]

This problem is a distribution of pairs \((x, y) \sim D\). The goal is to learn to estimate a distribution:
\[
p(x) = \Pr[y = 1|x].
\]

This is equal to the expectation or \( E[y|x] \). In this case \( y \) is binary, although in other problems, \( y \) might be a real. For example, \( y \) could be the amount of rain on a given day.

We define \( h(x) \) as an estimate of \( p(x) \) from a given expert. We want \( h(x) \approx p(x) \), but we never see \( p(x) \), we only see the \( x \) values. In otherwords, there might be a 80% chance of rain, although it might not actually rain. All we know is that it didn’t rain, not that there was an 80% chance of it.

The method is to penalize \( h \) on \((x, y)\) as follows:
\[
(h(x) - y)^2
\]

is a loss function, also called a cost function, in this case, square loss, quadratic loss or Breir score.

We have a set of predictions and \((x_1, y_1), \ldots, (x_m, y_m)\) and the actual events. We wish to choose \( h \) that minimizes the loss function, as in Equation 16:

\[
\sum_i (h(x_i) - y_i)^2 \tag{16}
\]

If \( h \) is unrestricted, when is the expected loss \( E[(h(x) - y)^2] \) minimized? Fix \( x \). Let \( p = p(x) = \Pr[y = 1|x], h = h(x) \). Then
\[
E[(h - y)^2] = p(h - 1)^2 + (1 - p)h^2 \tag{17}
\]

We now minimize over \( h \) by taking the derivative with respect to \( h \), and set equal to 0:
\[
\frac{d}{dh} = 2p(h - 1) + 2(1 - p)h = 2(h - p) \tag{18}
\]

Equation 18 has a minimum when \( h = p \). Hence, the loss function is minimized when \( h=p \).
Continuing:

\[ E_x[(h(x) - p(x))^2] = E_{x,y}[(h(x) - y)^2] - E_{x,y}[(p(x) - y)^2] \]  
(19)

Note: the expectation is over both \( x, y \), since it is constant in terms of \( h \). Also, the \( p(x) \) is the intrinsic randomness, or the variance avg over all \( x \)'s.

Prove for a single \( x \) then average over all \( x \)'s.

Claim:

\[ E_x[h(x) - p(x)]^2 = E_{x,y}[(h(x) - y)]^2 - E_{x,y}[(p(x) - y)]^2 \]  
(20)

\[ (h - p)^2 = E[(h - y)^2] - E[(p - y)^2] \]  
(21)

\[ (h - p)^2 = E[h^2 - 2hy + y^2] - E[p^2 - 2py + y^2] \]  
(22)

\[ (h - p)^2 = h^2 - 2h E_y y - p^2 + 2p E_y y = h^2 - 2hp + p^2 \]  
(23)

\[ (h - p)^2 = (h - p)^2 \]  
(24)

Hence, we prove the claim from Equation 20 for a fixed \( x \). To get the more general statement, we only need to average over random \( x \)'s since

\[ E_{x,y}[\text{ANY}] = E_x[E_y[\text{ANY}|x]] \]  
(25)

3 Estimate \( E[(h(x) - y)^2] \)

We estimate \( E[(h(x) - y)^2] \) by empirical average:

\[ \hat{E}[(h(x) - y)^2] = \frac{1}{m} \sum (h(x_i) - y_i)^2 \]  
(26)

\[ L_h(x, y) = (h(x) - y)^2 \]  
(27)

We want \( E[L_h] \approx \hat{E}[L_h] \) for all \( h \in \mathcal{H} \)

Chernoff bounds, union bound, VC-dim, growth function can all be generalized.

Q: How to minimize loss function for training set?

One answer: Perform a linear fit as shown in Figure 2.

Given \((x_1, y_1), \ldots, (x_m, y_m)\)
Figure 2: Fit data from $h(x)$ with a line.

$$\min : \sum_i (wx_i - y_i)^2$$

To minimize Equation 28 we set the derivative $\frac{d}{dw} = 2 \sum_i (wx_i - y_i)x_i$ to 0 and get Equation 29:

$$w = \frac{\sum y_ix_i}{\sum x_i^2}$$

4 Generalize to more than one dimension

given $(\vec{x}_1, y_1), \ldots, (\vec{x}_m, y_m), \vec{x}_i \in \mathcal{R}_n, y_i \in \mathcal{R}

\vec{w}$ using prediction rule $h(\vec{x}) = \vec{w} \cdot \vec{x}$

loss $(h) = \sum_i (\vec{w} \cdot \vec{x} - y_i)^2$

minimize:

$$\min \left\| \begin{pmatrix} \vec{x}_1^T \\ \vec{x}_2^T \\ \vdots \\ \vec{x}_m^T \end{pmatrix} - \begin{pmatrix} w_1 \\ w_2 \\ \cdots \\ w_m \end{pmatrix} \right\| \begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_m \end{pmatrix}^2_{2}$$

This can be solved by linear regression (next time).