**COS 511: Theoretical Machine Learning**

Lecturer: Rob Schapire
Scribe: Madhuvanthi Jayakumar

Lecture # 17
April 08, 2013

# 1 Review of Winnow Algorithm

**Algorithm 1** Winnow.

**procedure** WINNOW
  $\forall i : w_{1,i} = \frac{1}{N}$
  **for** $t = 1, \ldots, T$ **do**
    Get $\mathbf{x}_t \in \mathbb{R}^N$
    Predict $\hat{y}_t = sign(\mathbf{w}_t \cdot \mathbf{x}_t)$
    Get $y_t \in \{-1, +1\}$
    Update, if mistake: $w_{t+1,i} = \frac{w_{t,i}}{Z_t} e^{\eta y_t x_{t,i}}$

**Comments about algorithm:**

- $\mathbf{w}_t$ is the weight vector. We can view it as a probability distribution (non-negative values, and sums up to 1). Initially it is uniformly distributed.
- $\mathbf{x}_t$ can be thought of as a point in space or as an N-dimensional vector
- If there is no mistake, the weight is carried over to the next iteration: $w_{t+1} = w_t$

# 2 Upperbounding Number of Mistakes for Winnow

**Assumptions:**

1. A mistake is made on every round.
2. $\forall t : \|\mathbf{x}_t\|_\infty \leq 1$
3. $\exists \delta, \mathbf{u}, \forall t : y_t(\mathbf{u} \cdot \mathbf{x}_t) \geq \delta > 0$
4. $\forall t : \|\mathbf{u}\|_1 = 1$
5. $\forall i : u_i \geq 0$

**Theorem 2.1.** *Under the assumptions made above, we can bound the maximum number of*

*mistakes that can be made by the algorithm to* $\frac{2\ln N}{\delta^2}$ *if* $\eta = \frac{1}{2}\ln\frac{1+\delta}{1-\delta}$

$$\begin{aligned} \#\ mistakes &\leq \frac{\ln N}{\eta\delta + \ln(\frac{2}{e^{\eta}+e^{-\eta}})} \\ &\leq \frac{2\ln N}{\delta^2} \\ &if \quad \eta = \frac{1}{2}\ln(\frac{1+\delta}{1-\delta}) \end{aligned} \tag{1}$$

*Proof.* Last time, we showed:

$$\Phi_t = RE(\mathbf{u}||\mathbf{w}) \geq 0 \tag{2}$$

$$\Phi_{t+1} - \Phi_t \leq -c \tag{3}$$

$$\text{where } c = \eta\delta + \ln\frac{2}{e^{\eta}+e^{-\eta}}$$

Today, we will bound the number of mistakes by bounding the total change in potential. We first upperbound $\Phi_1$, the potential on the first round.

$$\begin{aligned} \Phi_1 &= RE(\mathbf{u}||\mathbf{w}) \\ &= \sum u_i \ln(u_i N) \\ &\leq \sum u_i \ln(N) \\ &= \ln N \end{aligned} \tag{4}$$

At this point we have upper bounded the potential for the first round. We also know the following two things:

1. The minimum change in each potential is $c$, and since there are $T$ rounds
   - This imiplies that the minimum total change in potential has to be at least $cT$.
2. The potential can never be negative (and $\Phi_1 \leq \ln N$).
   - This implies that the maximum total change in potential can be at most $\ln N$.

Hence, we can upperbound the maximum number of iterations of the algorithm in terms of $N$ and $c$ in the following way.

$$\begin{gathered} cT \leq \text{total change} \leq \ln N \\ cT \leq \ln N \\ T \leq \frac{\ln N}{c} \end{gathered}$$

Since our first assumption is that there is a mistake on every round, the number of mistakes is also bounded by this value. We have thus proved Theorem 2.1 to bound the number of mistakes in the algorithm which does not depend on probability assumptions.

**Minimizing bound:** We would like to set the value of $c$ such that the bound is minimized. To do this, we get the value of $\eta$ that will result in the minimum bound by taking the derivative and setting it equal to 0. Solving for $\eta$ we get:

$$\eta = \frac{1}{2} \ln \frac{1+\delta}{1-\delta}$$

Plugging this back into our equation for c, we get that:

$$c = RE(\frac{1}{2} - \frac{\delta}{2} || \frac{1}{2})$$

Since we are taking the Relative Entropy between two bernoulli items, we can reduce it (to 2 times the difference between the two bernoulli items) to get:

$$c \geq 2 \cdot (\frac{\delta}{2})^2 = \frac{\delta^2}{2}$$

We have thus proved the bound in the theorem:

$$T \leq \frac{\ln N}{c} \leq \frac{2 \ln N}{\delta^2}$$

□

# 3 Balanced Winnow

Now we would like to remove the fifth assumption that $\forall i : u_i \geq 0$
The quick and dirty way of doing this is to modify the two vectors, $\mathbf{x}$ and $\mathbf{u}$. Say we have vectors x and u as given below. We modify $\mathbf{x}$ to get $\mathbf{x'}$ by creating two copies of $\mathbf{x}$ and modifying the second copy such that their signs are negated. We modify $\mathbf{u}$ to get $\mathbf{u'}$ by creating two copies of u and modifying the second copy such that all the negative components are zero.
$\mathbf{x} = (1 \quad 0.7 \quad -0.4) \quad \rightarrow \quad \mathbf{x'} = (1 \quad .7 \quad -.4 \quad | \quad -1 \quad -.7 \quad .4)$
$\mathbf{u} = (0.5 \quad -0.2 \quad 0.3) \quad \rightarrow \quad \mathbf{u'} = (0.5 \quad 0 \quad .3 \quad | \quad 0 \quad .2 \quad 0)$
We've doubled the number of components but the other three previously made assumptions still hold, as shown:

**Assumptions:**

1. A mistake is made on every round: yes.
2. $\forall t : \|\mathbf{x'}_t\|_\infty \leq 1$
   - $\|\mathbf{x'}_t\|_\infty$ is also unchanged since we haven't changed the maximum absolute value.
3. $\exists \delta, \mathbf{u'}, \forall t : y_t(\mathbf{u'} \cdot \mathbf{x'}_t) \geq \delta > 0$
   - Inner product is unchanged: $\mathbf{x'} \cdot \mathbf{u'} = \mathbf{x} \cdot \mathbf{u}$. The dot product on non-negative portion stays the same. The dot product of the negative portion cancels out because it is positive but taking the inner product with negative of $\mathbf{x}_t$.
4. $\forall t : \|\mathbf{u'}\|_1 = 1$
   - $\|\mathbf{u'}\|_1$ is also unchanged since we take the absolute value of each component exactly once.

# 4 Compare Perceptron and Winnow/WMA

| Perceptron | Winnow/WMA |
|---|---|
| Additive Update | Multiplicative update |
| $\|\mathbf{x}_t\|_2 \leq 1$ | $\|\mathbf{x}_t\|_\infty \leq 1$ |
| $\|\mathbf{u}\|_2 = 1$ | $\|\mathbf{u}\|_1 \leq 1$ |
| SVM | Adaboost |

# 5 Regression

Until now, we have been learning to classify objects as labels 0/1 or -1/1. Our goal was to minimize the number of mistakes made by the algorithm or minimze the probability of making mistakes. We talked about how PAC outputs a hypothesis whose probability of making a mistake we wanted to be low. With online learning algorithms, we wanted the number of mistakes made by the algorithm to be low. We were able to evaluate these by looking at the labels. Now we want to switch focus and have a different goal which is not just to get the correct label.

## 5.1 Example

We will introduce this topic with an example. Say we are looking to hire Alice or Bob to predict the weather. We ask each of them to predict the probability that it rains. Alice says the probability of it raining is 70% and Bob says it is 80%. We see the outcome (that it rains) but we don't know the underlying probability, so we can't say whether Alice or Bob was closer to the actual probability. In the following sections, we explore how to come up with a percentage that is close to the true probabilities even though we can never observe true probabilities.

First we will formally state the problem. We will then create a model for scoring the prediction and state and prove a theorem that shows why the model works.

## 5.2 Formal Statement of Problem

We are given the weather conditions $\mathbf{x}$, and we would like to predict the value of $y$, which is 1 if it rains, and 0 otherwise. We obtain both $\mathbf{x}$ and $y$ from distribution $D$, $(\mathbf{x}, y) \sim D$. Our goal is to estimate $Pr[y = 1|\mathbf{x}]$. We never get to observe this, we only get the outcome $y$. We define $p(\mathbf{x}) = Pr[y = 1|\mathbf{x}] = \mathbb{E}[y|\mathbf{x}]$. We use expectation to have a more general problem statement, that allows $y$ to be any real number and not necessarily restrict it to 0 or 1. This problem is called regression. We may define $h_A(\mathbf{x})$ as Alice's estimate of the porbability of rain given x, and $h_B(\mathbf{x})$ as Bob's estimate of the probability of rain given $\mathbf{x}$.

## 5.3 Model

We define **square/quadratic loss** as $(h(x) - y)^2$ and we use this to score how good the prediction is. Taking the difference of the hypothesis and outcome is a natural way to characterize this and squaring gives nicer mathematical properties (differentiable, etc). We define risk to be the expected loss, $\mathbb{E}[(h(x)-y)^2]$, and choose $h$ that minimizes the expected value over $x, y$. We define the risk with respect to the true distribution $D$ as the true risk. We will show how to estimate this from samples, by looking at a set of predictions and outcomes and taking the average loss. Theorem 5.1 and its associated proof shows why

minimizing this expectation leads to a good prediction.

First we fix $\mathbf{x}$ and define the following:

$$h = h(\mathbf{x}) \text{ and } p = p(\mathbf{x}) = Pr[y = 1|\mathbf{x}]$$

Then we have that:

$$\mathbb{E}[(h - y)^2] = p(h - 1)^2 + (1 - p)h^2.$$

This comes from the definition of expectation: Pr[y=1] · resulting loss + P[y=0] · resulting loss.
To minimize the expectation, we take the derivative with respect to $h$ and set it equal to 0.

$$\frac{d\mathbb{E}}{dh} = 2(h - p) = 0, \text{ resulting in } h = p.$$

This implies that the loss is minimized when we set $h = p$. We cannot directly observe $p$, but this result shows that minimizing the loss will lead us to choose $h$ which is equal to $p$. Now we will state a theorem that is more general and stronger that applies to any value of $\mathbf{x}$.

**Theorem 5.1.** $\mathbb{E}_x[(h(\mathbf{x}) - p(\mathbf{x}))^2] = \mathbb{E}_{\mathbf{x},y}[(h(x) - y)^2] - \mathbb{E}_{x,y}[(p(\mathbf{x}) - y)^2]$

Note:

- The first term is the loss/risk we can measure to estimate from data.
- The second term measures the intrinsic noise of y.
- Our goal is to minimize $\mathbb{E}[(h(\mathbf{x}) - p(\mathbf{x}))^2]$.
  - Since the variance of y does not depend on h, we have that $\mathbb{E}[(h(\mathbf{x}) - p(\mathbf{x}))^2] = \mathbb{E}[(h(\mathbf{x}) - y)^2]-$constant
  - Therefore minimizing $\mathbb{E}[(h(\mathbf{x}) - y)^2]$ also minimizes $\mathbb{E}[(h(\mathbf{x}) - p(\mathbf{x}))^2]$ and this justifies the use of square loss to get the best prediciton.

*Proof.* We will prove Theorem 5.1 for fixed $\mathbf{x}$. Then we can use linearity of expectations to show that it holds for any expectation of $\mathbf{x}$. We define $h = h(\mathbf{x})$, and $p = p(\mathbf{x}) = \mathbb{E}[y|\mathbf{x}] = \mathbb{E}[y]$. We will modify the LHS and RHS of the equations to show that they are both equal to each other.

$$\text{LHS} = (h - p)^2.$$

Explanation: we can remove the expectation because $x$ is fixed.

$$\begin{aligned}
\text{RHS} &= \mathbb{E}[(h - y)^2] - \mathbb{E}[(p - y)^2] \\
&= \mathbb{E}[(h^2 - 2hy + y^2) - (p^2 - 2py + y^2)] \\
&= h^2 - 2h\mathbb{E}[y] - p^2 + 2p\mathbb{E}[y] \\
&= h^2 - 2hp + p^2 \\
&= (h - p)^2 \\
&= \text{LHS}
\end{aligned} \tag{5}$$

□

# 6 Linear Regression

We can estimate the smallest value of $\mathbb{E}[(h(x)-y)^2]$ by looking at the empirical average of the given sample. Given $m$ samples, $(x_1, y_1)...(x_m, y_m) \sim D$:

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

This is the empirical risk that can be measured using training data. We define

$$L_h(x, y) = (h(x)-y)^2$$

Now we do the following two things:

1. Prove/argue w.h.p. $\forall h \in H$, $\hat{\mathbb{E}}[L_h] \approx \mathbb{E}[L_h]$. This is a uniform convergence problem.
2. Minimize $\hat{\mathbb{E}}[L_h]$. This is a computational problem.

## 6.1 Example

Suppose we are given $m$ examples, $(x_1, y_1)...(x_m, y_m)$ with $x_i \in \mathbb{R}^n$ and labels $y_i \in R$
Our hypothesis is of the form, $h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$, which is linear for some fixed vector $\mathbf{w}$
We then have:

$$\begin{aligned} \hat{\mathbb{E}}[L_h] &= \frac{1}{m}\sum_{i=1}^{m}(h(\mathbf{x}_i) - y_i)^2 \\ &= \frac{1}{m}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y)^2 \end{aligned} \tag{6}$$

We would like to find the $\mathbf{w}$ that minimizes this quantity. This is called linear regression. We will work with this in matrix form, defining $M$ to be the matrix of vectors $\mathbf{x}$.

$$\begin{pmatrix} \leftarrow & \mathbf{x}_1^T & \rightarrow \\ \leftarrow & \mathbf{x}_2^T & \rightarrow \\ & \dots & \\ \leftarrow & \mathbf{x}_m^T & \rightarrow \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \dots \\ w_n \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix}$$

$$M\mathbf{w} - \mathbf{b}$$

$$\left\| \begin{matrix} \mathbf{w}\cdot\mathbf{x}_1 - y_1 \\ \mathbf{w}\cdot\mathbf{x}_2 - y_2 \\ \dots \\ \mathbf{w}\cdot\mathbf{x}_m - y_m \end{matrix} \right\|_2^2 = ||M\mathbf{w} - \mathbf{b}||_2^2$$

The Euclidean length squared will give us the sum of squared errors, $\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2$
Now we find $\mathbf{w}$ that minimizes this. To do this, we compute the gradient and set it equal to 0 and solve for $\mathbf{w}$.

$$\nabla\Phi = 2M^T(M\mathbf{w} - \mathbf{b}) = 0$$
$$\mathbf{w} = (M^TM)^{-1}M^T\mathbf{b}$$

$(M^TM)^{-1}M^T$is known as the pseudo inverse of $M^T$.

# 7 Combining Regression with Online Learning

Now we take a look at what regression looks like in an online learning setting.

**Algorithm 2** Regression with Online Learning

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

Our goal is to minimize loss, $L_A = \sum_{t=1}^{T} (\hat{y}_t - y_t)^2$

We are interested in updating the weight $\mathbf{w}_t$ and using it in a linear way to make predictions without making any randomness assumptions. We analyze the loss suffered using one particular weight vector $u \in \mathbb{R}^N$:

- Predict: $\mathbf{u} \cdot \mathbf{x}_t$
- Loss: $(\mathbf{u} \cdot \mathbf{x}_t - \mathbf{y}_t)^2$
- Cumulative loss $\sum_{t=1}^{T} (\mathbf{u} \cdot \mathbf{x}_t - y_t)^2$

We would like to achieve the result:

$$L_A \leq min L_u + \text{small amount: "regret"}$$

$$\text{where } L_u = \sum_{t=0}^{T} (\mathbf{u} \cdot \mathbf{x}_t - y_t)^2$$

$L_u$ is the loss of a linear predictor $\mathbf{u}$. The above inequality for $L_A$ says that if there exists any predictor that gives good predictions, then our algorithm performs close to that predictor.

# 8 Widrow-Hoff (WH) Algorithm

Towards the end of class, we introduced the Widrow Hoff Algorithm, which is a particular kind of online regression algorithm. This algorithm uses the weight vector in the following way:

- intialize: $\mathbf{w}_1 = 0$
- update: $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta(\mathbf{w}_t \cdot \mathbf{x}_t - y_t)x_t$

There are two main motivations for this algorithm. We discuss the first one here and will continue the rest during the next lecture.
We define our loss function as:

$$L(\mathbf{w}, \mathbf{x}, y) = (\mathbf{w} \cdot \mathbf{x} - y)^2$$

The gradient descent of this is:

$$\nabla_{\mathbf{w}} L = 2(\mathbf{w} \cdot \mathbf{x} - y) \cdot \mathbf{x}$$

We have $\mathbf{w_t}$ and we use $\mathbf{x_t}$ and $y_t$ to improve $\mathbf{w}_{t+1}$ so that loss will be slightly smaller on the example that we just observed. The equation below moves $\mathbf{w}$ in the direction of the gradient, which minimizes the loss function.

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \tfrac{1}{2}\eta \nabla_{\mathbf{w}} L(\mathbf{w}_t, \mathbf{x}_t, y_t)$$