COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Xingyuan Fang, Ethan

Lecture # 17
April 9th, 2013

1 Review of Winnow Algorithm

We have studied Winnow algorithm in Algorithm 1.

Algorithm 1 Winnow Algorithm

Initialize $w_{1,i} = \frac{1}{N}$
for $t = 1, ..., T$ **do**
Get $\mathbf{x}_t \in \mathbb{R}^N$
Predict $\widehat{y}_t = \text{sign}(\mathbf{w}_t \cdot \mathbf{x}_t)$
Observe $y_t \in \{-1, +1\}$
If $y_t = \widehat{y}_t$, $\mathbf{w}_{t+1} = \mathbf{w}_t$
Else $w_{t+1,i} = w_{t,i} e^{\eta y_t x_{t,i}} / Z_t$, where Z_t is a normalizing factor to make $\sum_i w_{t+1,i} = 1$.
end for

To analyze the performance of Winnow algorithm. We have made the following assumptions

- A mistake is made at every round.
- For all t, $\|\mathbf{x}_t\|_\infty = 1$.
- There exist $\delta, \mathbf{u}$ such that
  - For all t, $y_t(\mathbf{u} \cdot \mathbf{x}_t) \geq \delta > 0$
  - $\|\mathbf{u}\|_1 = 1$
  - For all i, $u_i \geq 0$

We have proved the following theorem:

Theorem 1.1. *Under the assumptions above, we have the following upper bound on the number of mistakes: that the number of mistakes of the Winnow algorithm is at most* $\frac{2 \ln N}{\delta^2}$.

In order to get rid of the assumption $u_i \geq 0$, we have introduce the *balanced Winnow algorithm*. We transform $\mathbf{x}$ and $\mathbf{u}$ by doubling the size of N and adding a complementary version. See (1.1) for an example.

$$\begin{aligned} \mathbf{x}_t = (1, -0.7, 0.32) &\rightarrow \mathbf{x}'_t = (1, -0.7, 0.32 | -1, 0.7, -0.32) \\ \mathbf{u} = (1, 0.2, -0.2) &\rightarrow \mathbf{u}' = (1, 0.2, 0 | 0, 0, 0.2). \end{aligned} \tag{1.1}$$

This transformation preserves the following properties: $\|\mathbf{x}_t\|_\infty = \|\mathbf{x}'_t\|_\infty$, $\|\mathbf{u}\|_1 = \|\mathbf{u}'\|_1$, and $\mathbf{u}' \cdot \mathbf{x}' = \mathbf{u} \cdot \mathbf{x}$. In addition, now $u_i \geq 0$ for all i. Let $\mathbf{w}^+$ and $\mathbf{w}^-$ denote the original and duplicated parts of $\mathbf{w}$. That is, in balanced Winnow algorithm, $\mathbf{w}' = (\mathbf{w}^+, \mathbf{w}^-)$ where $\mathbf{w}^+, \mathbf{w}^- \in \mathbb{R}^N$. We have the balanced Winnow algorithm in Algorithm 2.

Algorithm 2 Balanced Winnow Algorithm

Initialize : $w_{1,i}^{+} = w_{1,i}^{-} = \frac{1}{2N}$

for $t = 1, ..., T$ **do**

Predict $\widehat{y}_t = \text{sign}(\mathbf{w}_t^{+} \cdot \mathbf{x}_t - \mathbf{w}_t^{-} \cdot \mathbf{x}_t)$

If $\widehat{y}_t = y_t$ then $\mathbf{w}_{t+1}^{+} = \mathbf{w}_t^{+}$; $\mathbf{w}_{t+1}^{-} = \mathbf{w}_t^{-}$

Else $w_{t+1,i}^{+} = w_{t,i}^{+} e^{\eta y_t x_{t,i}} / Z_t$ and $w_{t+1,i}^{-} = w_{t,i}^{-} e^{-\eta y_t x_{t,i}} / Z_t$

end for

2 A Comparison Between Perceptron Algorithm and Winnow Algorithm

We list the difference between Perceptron algorithm and Winnow algorithm in the following table. The main difference is that Perceptron additively updates the weight, and Winnow multiplicatively updates the weight. In addition, they use different norms.

Table 1: A Comparison Between Perceptron Algorithm and Winnow Algorithm

Perceptron	Winnow/WMA
Additive update	Multiplicative update
$\|\mathbf{x}_t\|_2 = 1$ $\|\mathbf{u}\|_2 = 1$	$\|\mathbf{x}_t\|_\infty = 1$ $\|\mathbf{u}\|_1 = 1$
analogous to SVM	analogous to Boosting

3 Regression, Loss Function

Untill now, we have been always talking about classification problems. In these problems, we try to minimize the error rate of our classifier. Now, we begin to talk about another important problem which is regression.

Let's start with an example. Suppose that a TV station wants to hire a meteorologist to predict the weather. Two applicants, say Alice and Bob, made predictions on the next day's weather during the interview:

Alice said there is a 70% chance of raining tomorrow.

Bob said there is an 80% chance of raining tomorrow.

No matter if it rained or not on the second day. It is still difficult to decide which one we should hire. The reason is that we cannot observe the true probability of raining on that day.

Before going further, let us introduce some notation and formulate the problem mathematically.

Let $\boldsymbol{x}$ denote the current weather conditions, and let

$$y = \begin{cases} 1 & \text{if rain,} \\ 0 & \text{otherwise.} \end{cases}$$

We assume that $(\boldsymbol{x}, y)$ follows a joint distribution D that $(\boldsymbol{x}, y) \sim D$. Our goal is to estimate

$$p(\boldsymbol{x}) = \mathbb{P}(y = 1|\boldsymbol{x}) = \mathbb{E}(y|\boldsymbol{x}).$$

The difficulty here is that $p(\boldsymbol{x})$ is not observable for any $\boldsymbol{x}$. Alice gives $h_A(\boldsymbol{x})$, and Bob gives $h_B(\boldsymbol{x})$ to estimate $p(\boldsymbol{x})$ respectively. What we can observe is $\langle \boldsymbol{x}, h_A(\boldsymbol{x}), h_B(\boldsymbol{x}), y \rangle$. We wish to choose the one between h_A and h_B, which is closer to $p(\boldsymbol{x})$.

In a classification problem, we looked at how many mistakes a predictor h makes. Here, in order to judge the performance of a predictor $h(\boldsymbol{x})$, we look at the difference between $h(\boldsymbol{x})$ and y. We may use a *loss function* $L(h(\boldsymbol{x}), y)$ to measure the difference. For example, we may use quadratic loss $(h(\boldsymbol{x}) - y)^2$.

We justify why we use quadratic loss by the following proposition.

Proposition 3.1. $\mathbb{E}(h(\boldsymbol{x}) - y)^2$ *is minimized when* $h = p$.

Proof. Fix $\boldsymbol{x}$, then we write

$$p = p(\boldsymbol{x}) \text{ and } h = h(\boldsymbol{x}).$$

The quadratic loss function is:

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

As this is a convex function on h, to find the minimum, we may simply take the derivative and set it equal to zero:

$$\frac{dE}{dh} = 2(h - p) = 0,$$

which implies that

$$h = p.$$

□

This proposition tells us that h is equal to p if we minimize $\mathbb{E}(h(\mathbf{x}) - y)^2$ over all possible h. We next prove a theorem which shows that minimizing the observed squared difference between the prediction and actual outcomes is equivalent to minimizing the squared difference between the prediction and the true values.

Theorem 3.2.

$$\mathbb{E}(h(\mathbf{x}) - p(\mathbf{x}))^2 = \mathbb{E}(h(\mathbf{x}) - y)^2 - \mathbb{E}(p(\mathbf{x}) - y)^2, \tag{3.1}$$

where the first term is what we wish to minimize; the second term is the observation; and the last term captures the randomness and does not involve h.

Proof. Fix $\boldsymbol{x}$, we write

$$p = p(\boldsymbol{x}), \text{ and } h = h(\boldsymbol{x}).$$

Next, We compute the left and right-hand side of (3.1) respectively.

$$\begin{aligned} LHS &= \mathbb{E}(h - p)^2 = (h - p)^2 \\ RHS &= \mathbb{E}(h - y)^2 - \mathbb{E}(p - y)^2 \\ &= \mathbb{E}(h^2 - 2hy + y^2 - p^2 - y^2 + 2py) \\ &= h^2 - 2hp + p^2 \\ &= (h - p)^2. \end{aligned} \tag{3.2}$$

□

In classification problems, we try to minimize $\mathbb{P}\big(h(\mathbf{x}) \neq y\big)$. Now, we want to minimize $\mathbb{E}(h(\mathbf{x}) - y)^2$.

Given the data points $(\mathbf{x}_1, y_1), (\mathbf{x}_1, y_2), ..., (\mathbf{x}_m, y_m)$, we can estimate the expectation $\mathbb{E}(h(\mathbf{x}) - y)^2$ by looking at the empirical average:

$$\widehat{\mathbb{E}}(h(\mathbf{x}) - y)^2 = \frac{1}{m}\sum_{j=1}^{m}(h(\mathbf{x}_j) - y_j)^2.$$

If we define $L_h(\boldsymbol{x}, y) = (h(\boldsymbol{x}) - y)^2$, then we want

$$\mathbb{E}(L_h) \approx \widehat{\mathbb{E}}(L_h)$$

for all h in some class of functions $\mathcal{H}$. To show this, we can use Chernoff bounds. If $\mathcal{H}$ is finite then we can use the union bound to generalize the result. For the case that $\mathcal{H}$ is infinite, VC-style proofs can be used.

So far we have tried to justify the use of the quadratic loss function as a penalty function. However, we need to minimize this cost function in practice. We show how to solve this problem in the next section.

4 Linear Regression

Given $\mathbf{x} \in \mathbb{R}^n$, we want to predict y by some linear combination of the coordinates of $\mathbf{x}$, i.e., by $\mathbf{w} \cdot \mathbf{x}$. Our goal is to find $\mathbf{w}$.

Given m data points $(\mathbf{x}_1, y_1), (\mathbf{x}_1, y_2), ..., (\mathbf{x}_m, y_m)$, our problem is to find $\mathbf{w}$ to minimize

$$\Phi = \sum_{j=1}^{m}(\mathbf{w} \cdot \mathbf{x}_j - y_j)^2.$$

Φ can be written in matrix form in the following matrix form:

$$\begin{aligned}\Phi &= \left\| \begin{pmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_m^T \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \right\|_2^2 \\ &= \|\mathbf{X}^T\mathbf{w} - \mathbf{y}\|_2^2\end{aligned}$$

To minimize this function over $\mathbf{w}$, we take the gradients. We have

$$\bigtriangledown\Phi = 2\mathbf{X}(\mathbf{X}^T\mathbf{w} - \mathbf{y}) = 0.$$

When $\mathbf{X}\mathbf{X}^T$ is invertible, we have a unique solution of $\mathbf{w}$, which is

$$\mathbf{w} = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{y}.$$

The matrix $(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}$ is called the *pseudoinverse* of $\mathbf{X}^T$.

Algorithm 3 Online Linear Regression

Initialize $\mathbf{w}_1$

for $t = 1, ..., T$ **do**
  Get $\mathbf{x}_t \in \mathbb{R}^n$.
  Predict $\widehat{y}_t = \mathbf{w}_t \cdot \mathbf{x}_t$.
  Observe y_t.
  Compute the quadratic loss $(\widehat{y}_t - y_t)^2$.
  Update $\mathbf{w}_{t+1}$.
end for

5 Online Linear Regression

We next look at the online version of linear regression. Online linear regression can be considered as a modern version of linear regression. We formalize the online version of linear regression in Algorithm 3.

The goal is to minimize the cumulative loss of the learning algorithm A:

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

In particular, we want to achieve the following theoretical result:

$$L_A \leq \min_u L_u + (\text{a small number}),$$

where

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

L_u is the loss of a linear predictor $\mathbf{u}$. Our goal is to predict almost as well as the best static linear predictor.

6 Widrow-Hoff(WH) Algorithm

In this section, we introduce a particular online regression algorithm named *Widrow-Hoff(WH) algorithm*. WH algorithm initialize $\mathbf{w}_1 = \mathbf{0}$. Next, at each iteration t, we update $\mathbf{w}_{t+1}$ by

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta(\mathbf{w}_t \cdot \mathbf{x}_t - y_t)\mathbf{x}_t. \tag{6.1}$$

There are two motivations for the update rule in Widrow-Hoff. The first motivation is that our loss function is defined as:

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

To minimize the loss function, we take a step in the direction of steepest descent, i.e., in the direction of the negative gradient. In this case, we have:

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

which gives us

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

where η is a pre-specified step length.

The second motivation is that we have two goals:

- We want the loss on $(\mathbf{x}_t, y_t)$ to be small which means that we want to minimize $(\mathbf{w}_{t+1} \cdot \mathbf{x}_t - y_t)^2$.
- We do not want to be too far from $\mathbf{w}_t$. That is, we do not want $\|\mathbf{w}_t - \mathbf{w}_{t+1}\|$ to be too big.

Combining the above two goals, we compute $\mathbf{w}_{t+1}$ by solving the following optimization problem

$$\mathbf{w}_{t+1} = \operatorname{argmin} \eta(\mathbf{w}_{t+1} \cdot \mathbf{x}_t - y_t)^2 + \|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2,$$

where η is a pre-specified weighting factor.

Take the gradient of the above equation, and make it equal to zero. We have

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta(\mathbf{w}_{t+1} \cdot \mathbf{x}_t - y_t)\mathbf{x}_t.$$

Aprroximating $\mathbf{w}_{t+1}$ by $\mathbf{w}_t$ on the right-hand side gives the WH update defined in (6.1).

We will analyze the properties of WH algorithm in the next lecture.