EECS 598-005: Theoretical Foundations of Machine Learning **Fall 2015**

Lecture 16: Perceptron and Exponential Weights Algorithm

Lecturer: Jacob Abernethy *Scribes: Yue Wang, Editors: Weiqing Yu and Andrew Melfi*

16.1 Review: the Halving Algorithm

16.1.1 Problem Setting

Last lecture we started our discussion of *online learning*, and more specifically, *prediction with expert advice*. In this problem setting, time proceeds as a sequence of rounds, $t = 1, \cdots, T$. At each round t, the "environment" reveals an outcome $y_t \in \{-1, 1\}$. We also have a pool of "experts", $i = 1, \cdots, N$. At each round t, the i-th expert makes a prediction on the outcome, $f_i^t \in \{-1, 1\}$. Our goal is to design an algorithm[1] $\mathcal{A}$ that combines the experts' advice to predict the outcome with as few mistakes as possible. It can be described as the following process.

Initialize the counter $\#\text{mistakes}[\mathcal{A}] = 0$. For each round $t = 1, \cdots, T$:

(1) $\mathcal{A}$ observes the prediction made by N experts, $f_i^t \in \{-1, 1\}, i = 1, \cdots, N$;

(2) $\mathcal{A}$ makes prediction on $\hat{y}_t \in \{-1, 1\}$;

(3) The environment reveals an outcome $y_t \in \{-1, 1\}$;

(4) If $\hat{y}_t \neq y_t$, $\#\text{mistakes}[\mathcal{A}]$ += 1.

Here, the "environment" (or "nature") is a mechanism that generates the sequence of outcome. In online learning setting, this mechanism can be deterministic, stochastic, or even adversarial (adaptive to $\mathcal{A}$'s behavior). This is much more general than the "batch learning" setting, where the mechanism is assumed to be stochastic.

16.1.2 The Halving Algorithm

The **Halving Algorithm** takes the majority vote of consistent experts up to the current round. Formally, denote the set of consistent experts up to round t as C_t,

$$C_t = \{i : f_i^\tau = y_\tau, \forall \tau = 1, \cdots, t-1\} \ .$$

On round t, the Halving Algorithm makes its prediction as

$$\hat{y}_t = \begin{cases} 1, & \text{if } |\{i \in C_t : f_i^t = 1\}| \geq \frac{|C_t|}{2}; \\ -1, & \text{otherwise.} \end{cases}$$

Theorem 16.1. ***(Upper bound on #mistakes[Halving].)*** *If there exists a perfect expert i such that $f_i^t = y_t$ for all $t = 1, \cdots, T$, then*

$$\#\textit{mistakes[Halving]} \leq \log_2 N \ .$$

[1] Also called *predictor*, *forecaster*, or *master algorithm*.

Proof: We have $|C_1| = N$. By the assumption that there is always at least one perfect expert, $|C_T| \geq 1$. By the nature of Halving Algorithm, if $\hat{y}_t \neq y_t$, then $|C_{t+1}| \leq \frac{1}{2}|C_t|$. Together we have

$$1 \leq |C_T| \leq |C_1| \left(\frac{1}{2}\right)^{\#\text{mistakes[Halving]}} = N \left(\frac{1}{2}\right)^{\#\text{mistakes[Halving]}}.$$

This implies $\#\text{mistakes[Halving]} \leq \log_2 N$. ■

Note that the upper bound is also tight. To see this, imagine that the outcomes are generated by an adversary, who always wants to inflict mistake on $\mathcal{A}$. That is, when $\hat{y}_t = 1$, the adversary will play $y_t = -1$, and vice versa. Suppose we have $N = 2^n$ experts, only one of whom is perfect. At each round, half of them predict 1, the other half predict -1, which maximally slows down the "shrinking rate" of C_t. Then the Halving algorithm will make exactly $\log_2 N$ mistakes to discover the perfect expert, and makes no more mistakes from then on.

The Halving Algorithm works well if there exists a perfect expert, which we call a *noise-free* setting[2]. **What if the best expert makes a few mistakes?** This is a *noisy* setting, and we need a better algorithm than majority vote. The idea is to maintain weights on experts, and reduce an expert's weight when she makes mistakes. But before we move on to the noisy setting, let us study another classical algorithm for the noise-free setting, the Perceptron algorithm for online linear prediction.

16.2 Online Linear Prediction and Perceptron

16.2.1 Problem Setting

As before, we play in rounds $t = 1, \cdots, T$. At each round t, the environment reveals an outcome $y_t \in \{-1, 1\}$. In online linear prediction, algorithm $\mathcal{A}$ observes a *feature vector* $\mathbf{x} \in \mathbb{R}^d$ before it makes prediction, and its prediction has the form $\hat{y}_t = \text{sign}(\mathbf{w}^\top \mathbf{x})$, $\mathbf{w} \in \mathbb{R}^d$. [3] The prediction protocol can be described as following process.

Initialize the counter $\#\text{mistakes}[\mathcal{A}] = 0$. For each round $t = 1, \cdots, T$:

(1) $\mathcal{A}$ selects $\mathbf{w}_t \in \mathbb{R}^d$;

(2) $\mathcal{A}$ observes $\mathbf{x}_t \in \mathbb{R}^d$, $\|\mathbf{x}_t\| \leq 1$;

(3) $\mathcal{A}$ predicts $\hat{y}_t = \text{sign}(\mathbf{w}_t^\top \mathbf{x}_t)$;

(4) The environment reveals an outcome $y_t \in \{-1, 1\}$;

(5) If $\hat{y}_t \neq y_t$, $\#\text{mistakes}[\mathcal{A}]$ += 1.

Essentially, $\mathcal{A}$'s prediction $\hat{y}_t$ is determined by the hypothesis weight vector $\mathbf{w}_t$.

16.2.2 The Perceptron Algorithm

The **Perceptron Algorithm** initially predicts $\mathbf{w}_1 = \mathbf{0}$; at round $t = 1, \cdots, T$,

$$\mathbf{w}_{t+1} = \begin{cases} \mathbf{w}_t, & \text{if } y_t \left(\mathbf{w}_t^\top \mathbf{x}_t\right) > 0; \\ \mathbf{w}_t + y_t \mathbf{x}_t, & \text{otherwise.} \end{cases}$$

[2] The corresponding scenario in batch learning is to have a finite hypothesis class that contains the target hypothesis.

[3] $\text{sign}(\cdot)$ is the sign function:

$$\text{sign}(x) = \begin{cases} 1, & \text{if } x \geq 0; \\ -1, & \text{otherwise.} \end{cases}$$

Theorem 16.2. ***(Upper bound on #mistakes[Perceptron].)*** *Suppose there exists a perfect hypothesis* $\mathbf{w}_* \in \mathbb{R}^d$ *such that* $y_t(\mathbf{w}_*{}^\top \mathbf{x}_t) \geq 1$ *for all* $t = 1, \cdots, T$*, then*

$$\#mistakes[Perceptron] \leq \|\mathbf{w}_*\|^2 \ .$$

Proof: The key (trick) is to define the potential function

$$\Phi_t := \|\mathbf{w}_* - \mathbf{w}_{t+1}\|^2$$

and look at its change over time.

$$\begin{aligned}
\|\mathbf{w}_*\|^2 &\geq \|\mathbf{w}_* - \mathbf{0}\|^2 - \|\mathbf{w}_* - \mathbf{w}_{T+1}\| \\
&= \Phi_0 - \Phi_T \\
&= \sum_{t=1}^{T} \Phi_{t-1} - \Phi_t \\
&= \sum_{t=1}^{T} \|\mathbf{w}_* - \mathbf{w}_t\|^2 - \|\mathbf{w}_* - \mathbf{w}_{t+1}\|^2 \\
&= \sum_{t: y_t(\mathbf{w}_t^\top \mathbf{x}_t) < 0} \|\mathbf{w}_* - \mathbf{w}_t\|^2 - \|\mathbf{w}_* - (\mathbf{w}_t + y_t \mathbf{x}_t)\|^2 \\
&= \sum_{t: y_t(\mathbf{w}_t^\top \mathbf{x}_t) < 0} 2 \left(\underbrace{y_t(\mathbf{w}_*^\top \mathbf{x}_t)}_{\geq 1} \underbrace{- y_t(\mathbf{w}_t^\top \mathbf{x}_t)}_{\geq 0} \right) \underbrace{- y_t^2 \|\mathbf{x}_t\|^2}_{\geq -1} \\
&\geq \sum_{t: y_t(\mathbf{w}_t^\top \mathbf{x}_t) < 0} 1 \\
&= \#\text{mistakes[Perceptron]}
\end{aligned}$$

■

Additional notes:

(1) Since a perfect hypothesis $\mathbf{w}_*$ satisfies $y_t(\mathbf{w}_*{}^\top \mathbf{x}_t) > 0$ for all $t = 1, \cdots, T$, we can always scale it such that $y_t(\mathbf{w}_*{}^\top \mathbf{x}_t) \geq 1$ as stated in Theorem 16.2.

(2) In this setting, the hypothesis $\mathbf{w}_*$ defines a hyperplane which separates feature vectors into two classes based on their outcomes. We refer to $\frac{1}{\|\mathbf{w}_*\|} = \gamma$ as the *margin*, because the distance between any feature vector and the hyperplane is at least γ.

(3) Recall that we have similar upper bound in the support vector machine (SVM) batch learning setting, where the training examples are assumed *i.i.d.*. Here, we make no *i.i.d.* assumption on the sequence $((\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_T, y_T))$; the upper bound holds for any sequence.

16.3 The Exponential Weights Algorithm

The drawback of Halving Algorithm is the strong assumption that a perfect expert exists. How can we remove this assumption? What if even the best expert makes mistakes? The answer is to use weights for the experts, based on past performance. This leads to the Exponential Weights Algorithm (EWA) below.

16.3.1 Problem Setting

Consider the same problem setting as Section 16.1.1. We assume that the master algorithm $\mathcal{A}$ can make "mixed" predictions $\hat{y}_t \in \{-1, 1\}$. That is, $\hat{y}_t$ takes the form

$$\hat{y}_t = \frac{\sum_{i=1}^N w_i^t f_i^t}{\sum_{i=1}^N w_i^t} \ .$$

Essentially, $\mathcal{A}$'s prediction $\hat{y}_t$ is determined by the weights $\mathbf{w}_t = (w_1^t, \cdots, w_i^t \cdots, w_N^t) \in \mathbb{R}^N$.

In addition, assume that we are given a loss function $l(\hat{y}, y)$ that is convex in $\hat{y}$. Instead of absolute number of mistakes, we measure the performance of algorithm $\mathcal{A}$ by the cumulative loss it suffered after T rounds *relative to the best expert*:

$$R_T = \sum_{t=1}^{T-1} l(\hat{y}_t, y_t) - \min_{1 \le i \le N} \sum_{t=1}^{T-1} l(f_i^t, y_t) \ .$$

R_T is called the *regret*. Our goal is to design a master algorithm $\mathcal{A}$ with a small regret.

16.3.2 Exponential Weights Algorithm

To ease notation, let us denote the cumulative loss of expert i up to round t as $L_i^t = \sum_{s=1}^{t-1} l(f_i^s, y_s)$, and the cumulative loss of algorithm $\mathcal{A}$ up to round t as $L_{\mathcal{A}}^t = \sum_{s=1}^{t-1} l(\hat{y}_s, y_s)$. The regret of $\mathcal{A}$ after T rounds is then $L_{\mathcal{A}}^T - \min_i L_i^T$.

The **Exponential Weights Algorithm (EWA)** initially sets $\mathbf{w}_1 = \mathbf{1}$; at round $t = 1, \cdots, T$, it sets

$$w_i^t = \exp(-\eta L_i^t) \ ,$$

where $\eta > 0$ is a constant (hyperparameter). Intuitively, EWA says that the (unnormalized) weight of expert i decays exponentially with her cumulative loss.

Theorem 16.3. ***(Upper bound on L_{EWA}^T.)*** *Assume the loss function takes value in $[0, 1]$, then the T-round cumulative loss of Exponential Weights Algorithm satisfies that for all i,*

$$L_{EWA}^T \le \frac{\eta L_i^T + \log N}{1 - e^{-\eta}} \ .$$

Corollary 16.4. ***(Regret upper bound after tuning η.)*** *For some optimal $\eta > 0$,*

$$L_{EWA}^T \le L_{i^*}^T + \log N + \sqrt{2 L_{i^*}^T \log N} \ ,$$

where $i^* = \arg\min_{1 \le i \le N} L_i^T$ is the "best expert" in hindsight.

Lemma 16.5. *Let X be a random variable in $[0, 1]$. For all $s \in \mathbb{R}$,*

$$\log \mathbb{E}\left[e^{sX}\right] \le (e^s - 1)\, \mathbb{E}\left[X\right] \ .$$

Proof: First, for $x \in [0, 1]$,

$$e^{sx} \le 1 + (e^s - 1)\, x \ .$$

This can be seen by drawing the graph of convex function $f(x) = e^{sx}$ and line segment $g(x) = 1 + (e^s - 1)\, x = f(0) + \frac{f(1) - f(0)}{1 - 0} x$ on the interval $x \in [0, 1]$. In fact, $g(x)$ is a "chord" of $f(x)$ connecting $(0, f(0))$ and $(1, f(1))$. Since $f(x)$ is convex, its graph is always below its chord $g(x)$. This holds for all $s \in \mathbb{R}$.

Taking expectation on both sides when $x = X$ is a random variable in $[0, 1]$, we have:

$$\mathbb{E}\left[e^{sX}\right] \le 1 + (e^s - 1)\, \mathbb{E}\left[X\right] \ .$$

Since $\log(\cdot)$ is monotonic, taking log on both sides preserves the inequality sign (both sides are strictly positive):

$$\log \mathbb{E}\left[e^{sX}\right] \leq \log\left(1 + (e^s - 1)\,\mathbb{E}\left[X\right]\right) \ .$$

Finally, notice that $\forall x \in \mathbb{R}, \log(1+x) < x$, we have

$$\log \mathbb{E}\left[e^{sX}\right] \leq (e^s - 1)\,\mathbb{E}\left[X\right] \ .$$

■

Proof: (Theorem 16.3.) Let $w_t = \sum_{i=1}^N w_i^t$. Define the potential function $\Phi_t = -\log w_t$. Observe the change of the potential function over time:

$$\begin{aligned}\Phi_{t+1} - \Phi_t &= -\log \frac{w_{t+1}}{w_t} \\ &= -\log \frac{\sum_{i=1}^N w_i^t \exp\left(-\eta l(f_i^t, y_t)\right)}{\sum_{i=1}^N w_i^t} \ .\end{aligned}$$

By defining the random variable $X \in [0,1]$, $\Pr\left[X = l(f_i^t, y_t)\right] = \frac{w_i^t}{\sum_{i=1}^N w_i^t}$, $i = 1, \cdots, N$, we have

$$\begin{aligned}\Phi_{t+1} - \Phi_t &= -\log \mathbb{E}_X\left[-\eta X\right] \\ &\geq \left(1 - e^{-\eta}\right)\mathbb{E}_X\left[X\right] \quad \text{(Lemma 16.5)} \\ &= \left(1 - e^{-\eta}\right)\sum_{i=1}^N \frac{w_i^t \ l(f_i^t, y_t)}{\sum_{i=1}^N w_i^t} \\ &\geq \left(1 - e^{-\eta}\right) l\left(\frac{\sum_{i=1}^N w_i^t f_i^t}{\sum_{i=1}^N w_i^t}, y_t\right) \quad (l \text{ convex + Jensen's inequality}) \\ &= \left(1 - e^{-\eta}\right) l(\hat{y}_t, y_t) \ .\end{aligned}$$

Summing from $t = 1$ to $T - 1$:

$$\begin{aligned}\Phi_T - \Phi_1 &= \sum_{t=1}^{T-1} \Phi_{t+1} - \Phi_t \\ &\geq \left(1 - e^{-\eta}\right)\sum_{t=1}^{T-1} l(\hat{y}_t, y_t) \\ &= \left(1 - e^{-\eta}\right) L_{EWA}^T \ . \end{aligned} \tag{16.1}$$

By nature of Exponential Weights Algorithm and the definition of the potential function Φ_t,

$$\Phi_1 = -\log \sum_{i=1}^N w_i^1 = -\log N \ ; \tag{16.2}$$

$$\Phi_T = -\log \sum_{i=1}^N w_i^T \leq -\log w_i^T = \eta L_i^T, \text{ for all } i \ . \tag{16.3}$$

Combining (16.1), (16.2), and (16.3), we have that for all i,

$$\eta L_i^T + \log N \geq \left(1 - e^{-\eta}\right) L_{EWA}^T \ .$$

Since $\eta > 0$, $1 - e^{-\eta} > 0$. Dividing both ends by $(1 - e^{-\eta})$ will reveal the inequality in the theorem. ■