CS 7545: Machine Learning Theory **Fall 2018**

Lecture 7: Exponential Weights Algorithm and Perceptron

Lecturer: Jacob Abernethy *Scribes: David Betancourt and Roland Samuelson*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications.*

7.1 Introduction

In the previous lectures we have introduced the online learning problem of the experts model, in which the number of mistakes $M_T(\mathcal{A})$ of an algorithm $\mathcal{A}$ is to be minimized given the predictions of N experts. We analyzed the performance of the Weighted Majority Algorithm (WMA)

In this lecture, we further generalize this algorithm to the Exponential Weights Algorithm (EWA), which operates in a setting where predictions may be in the range $[0, 1]$, rather than just 0 or 1, and show a bound similar to that of WMA on its performance with regard to a convex loss function. We will also introduce the *Hedge* framework for online learning, which is equivalent to the expert framework in a natural manner. Finally, we will introduce the linear prediction framework, and the perceptron algorithm for linear prediction.

7.2 Online Learning Frameworks

In this section we first describe the generalization of the *prediction with expert advice* framework followed by the *Hedge* framework.

7.2.1 Prediction with Expert Advice

Assume we are given a *convex loss function* $\ell : [0, 1] \times [0, 1] \to [0, 1]$, such that $\ell(\cdot)$ is convex in its first argument.

For a convex loss function $\ell(\cdot)$, a pool of N experts, and an algorithm $\mathcal{A}$, **prediction with expert advice** is the following online learning framework.

Algorithm 1 Prediction with Expert Advice

1: **for** $t = 1$ to T **do**
2: Expert i predicts $x_i^t \in [0, 1]$ for $i \in \{1, \ldots, N\}$
3: Algorithm $\mathcal{A}$ predicts $\hat{y}^t \in [0, 1]$
4: Nature reveals $y^t \in [0, 1]$
5: **end for**

Definition 7.1 (algorithm loss) *The **loss** $L_T(\mathcal{A})$ of an algorithm $\mathcal{A}$ at time T is the sum of the loss values $\ell(\hat{y}^t, y^t)$ from time $t = 1$ to T.*

$$L_T(\mathcal{A}) := \sum_{t=1}^{T} \ell(\hat{y}^t, y^t).$$

Definition 7.2 (expert loss) *The **loss** $L_T(i)$ of expert i at time T is the sum of the loss values $\ell(x_i^t, y^t)$ from time $t = 1$ to T.*

$$L_T(i) := \sum_{t=1}^{T} \ell(x_i^t, y^t).$$

Definition 7.3 (regret) *The **regret** $R_T(\mathcal{A})$ of an algorithm $\mathcal{A}$ at time T is the difference between its loss at time T and the loss of the best expert at time T.*

$$R_T(\mathcal{A}) := L_T(\mathcal{A}) - \min_{i \in [N]} L_T(i).$$

7.2.2 Hedge Framework

We now describe the Hedge framework, in which at each timestep, instead of N experts, there are N possible actions from which to choose, and a loss associated with each action at that time step. As we will see, these frameworks are equivalent in a certain sense.

For N actions, and an algorithm $\mathcal{A}$, the **Hedge framework** is the following online learning framework.

Algorithm 2 HEDGE FRAMEWORK

1: **for** $t = 1$ to T **do**
2: Algorithm $\mathcal{A}$ selects a distribution $\vec{p^t} \in \Delta_N$[1]
3: Nature reveals action costs $\ell_i^t \in [0, 1]$ for $i \in \{1, \ldots, N\}$
4: Algorithm pays cost $\vec{p^t} \cdot \vec{\ell^t} = \mathbb{E}_{I_t \sim \vec{p^t}}[\ell_{I_t}^t]$
5: **end for**

Definition 7.4 (algorithm loss) *The **loss** $L_T(\mathcal{A})$ of an algorithm $\mathcal{A}$ at time T is the sum of the costs $\vec{p^t} \cdot \vec{\ell^t}$ from time $t = 1$ to T.*

$$L_T(\mathcal{A}) := \sum_{t=1}^{T} \vec{p^t} \cdot \vec{\ell^t}$$

Definition 7.5 (expert loss) *The **loss** $L_T(i)$ of an action i at time T is the sum of the costs ℓ_i^t from time $t = 1$ to T.*

$$L_T(i) := \sum_{t=1}^{T} \ell_i^t$$

Definition 7.6 (regret) *The **regret** $R_T(\mathcal{A})$ of an algorithm $\mathcal{A}$ at time T is the difference between its loss at time T and the loss of the best action at time T.*

$$R_T(\mathcal{A}) := L_T(\mathcal{A}) - \min_{i \in [N]} L_T(i)$$

HEDGE algorithm can be viewed as choosing an expert $I_t \in \{1, \ldots, N\}$ in round t, where $I_t \sim \vec{p^t}$. Then, the algorithm suffers loss $\ell_{I_t}^t$, which is element I_t of the loss vector $\vec{\ell^t}$. Note that each element of the loss vector $\vec{\ell^t}$ can be viewed as the cost of choosing the corresponding expert in round t. For the ease of analysis, we consider the expected cost, $\vec{p^t} \cdot \vec{\ell^t}$. The HEDGE setting is equivalent to the PREDICTION WITH EXPERT ADVISE framework in the following sense: if we choose the loss values of the HEDGE framework to be $\ell_i^t \equiv \ell(x_i^t, y^t)$ (where the ℓ on the right denotes the loss function of the EXPERT framework), then the loss of each expert/action will be equivalent.

7.3 Exponential Weights Algorithm

We describe the EXPONENTIAL WEIGHTS ALGORITHM (EWA) in both the experts and Hedge frameworks.

We now show that the EWA guarantees a regret (in the experts framework) that is similar to the mistake bound of the WMA, in that it is linear in the loss of the best expert times the parameter η, plus a factor logarithmic in N.

[1] $\Delta_N := \{p \in \mathbb{R}^N \mid \forall_i p_i \in [0, 1], \sum_{i=1}^N p_i = 1\}$ is the set of discrete probability distributions on N choices

Algorithm 3 EXPONENTIAL WEIGHTS ALGORITHM (Expert Framework)

Parameter: $\eta \in (0,1)$

Initialize weights $w_i^1 = 1$ for $i \in \{1,\dots,N\}$
for $t = 1$ to T **do**
  Expert i predicts $x_i^t \in [0,1]$ for $i \in \{1,\dots,N\}$
  Predict $\hat{y}^t = \frac{\sum_{i=1}^N w_i^t x_i^t}{\sum_{j=1}^N w_j^t}$
  Nature reveals $y^t \in [0,1]$
  Update weights according to $w_i^{t+1} = w_i^t \exp(-\eta \ell(x_i^t, y^t))$
end for

Algorithm 4 EXPONENTIAL WEIGHTS ALGORITHM (Hedge Framework)

Parameter: $\eta \in (0,1)$

Initialize weights $w_i^1 = 1$ for $i \in \{1,\dots,N\}$
for $t = 1$ to T **do**
  Select $p_i^t = \frac{w_i^t}{\sum_{j=1}^N w_j^t}$
  Nature reveals $\ell_i^t \in [0,1]$ for $i \in \{1,\dots,N\}$
  Algorithm pays cost $p^t \cdot \ell^t = \mathbb{E}_{i\sim p^t}[\ell_i^t]$
  Update weights according to $w_i^{t+1} = w_i^t \exp(-\eta \ell_i^t)$
end for

Theorem 7.7 *EWA(η) guarantees, for any expert $i \in \{1,\dots,N\}$, that*

$$L_T(EWA(\eta)) \le \frac{\log(N) + \eta L_T(i)}{1 - e^{-\eta}}$$

Corollary 7.8 *For appropriately tuned $\eta > 0$,*

$$L_T(EWA(\eta)) - L_T(i^*) \le \log(N) + \sqrt{2 L_T(i^*) \log(N)}$$

Where $i^* = \arg\min_i L_T(i)$.

(This corollary will be proven as a homework exercise)

Lemma 7.9 *For any r.v. $X \in [0,1]$ and any $s \in \mathbb{R}$,*

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

(This lemma was proven in the previous lecture)

Proof: Similarly to the proof of the mistake bound for WMA, we will use a potential argument. We use a potential function Φ_t where

$$\Phi_t := -\log(\sum_{i=1}^N w_i^t)$$

For each t, let X_t be a random variable which takes the value $\ell(x_i^t, y^t)$ with probability $\frac{w_i^t}{\sum_{j=1}^N w_j^t}$. We find a lower bound for the difference $\Phi_{t+1} - \Phi_t$.

$$\Phi_{t+1} - \Phi_t = -\log(\frac{\sum_{i=1}^N w_i^{t+1}}{\sum_{j=1}^N w_j^t}) = -\log(\frac{\sum_{i=1}^N w_i^t \exp(-\eta \ell(x_i^t, y^t))}{\sum_{j=1}^N w_j^t}) = -\log(\mathbb{E}[e^{-\eta X_t}])$$

Using Lemma 7.9, we apply this inequality with $X = X_t$ and $s = -\eta$ to see that

$$\Phi_{t+1} - \Phi_t \geq (1 - e^{-\eta})\mathbb{E}[X_t] = (1 - e^{-\eta}) \sum_{i=1}^{N} \frac{w_i^t}{\sum_{j=1}^{N} w_j^t} \ell(x_i^t, y^t)$$

We apply Jensen's inequality, using the convexity of the first argument of ℓ.

$$\Phi_{t+1} - \Phi_t \geq (1 - e^{-\eta})\ell(\sum_{i=1}^{N} \frac{w_i^t}{\sum_{j=1}^{N} w_j^t}, y^t) = (1 - e^{-\eta})\ell(\hat{y}^t, y^t)$$

Hence,

$$(1 - e^{-\eta})L_T(\text{EWA}(\eta)) = (1 - e^{-\eta}) \sum_{t=1}^{T} \ell(\hat{y}^t, y^t) \leq \sum_{t=1}^{T} (\Phi_{t+1} - \Phi_t) = \Phi_{T+1} - \Phi_1$$

We note that

- $\Phi_1 = -\log(N)$
- $\Phi_{T+1} \leq -\log(w_i^{T+1}) = \eta \sum_{t=1}^{T} \ell(x_i^t, y^t) = \eta L_T(i)$ for any $i \in \{1, \ldots, N\}$

To see that

$$L_T(\text{EWA}(\eta)) \leq \frac{\log(N) + \eta L_T(i)}{1 - e^{-\eta}}$$

■

7.4 Linear Prediction

7.4.1 Linear Prediction Framework

In the *linear prediction* framework, on each round, an algorithm will make a prediction about some weight vector $\vec{w} \in \mathbb{R}^d$, nature will select some vector $\vec{x} \in R^d$, and the accuracy of the chosen weight vector will be determined based on whether it correctly classifies $\vec{x}$ based on a linear decision boundary created by $\vec{w}$.

For an algorithm $\mathcal{A}$, the **linear prediction framework** is the following online learning framework.

Algorithm 5 Linear Prediction

1: **for** $t = 1, 2, \ldots$ **do**
2: Algorithm $\mathcal{A}$ selects $\vec{w^t} \in \mathbb{R}^d$
3: Nature selects $\vec{x^t} \in \mathbb{R}^d$
4: Algorithm predicts $\hat{y}^t = \text{sign}(\vec{w^t} \cdot \vec{x^t}) \in \{-1, 1\}$
5: Nature reveals $y^t \in \{-1, 1\}$
6: **end for**

In Line 4 of the algorithm, the $\text{sign}(x)$ function is

$$\text{sign}(x) := \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}$$

For linear prediction, we frequently make the assumption of the existence of a perfect expert in a similar fashion to the perfect expert assumption in the Prediction with Expert Advice framework.

Definition 7.10 (perfect expert) *Say a vector $\vec{w^*} \in \mathbb{R}^d$ is a **perfect expert** if $\left\|\vec{w^*}\right\|_2 \leq 1$ and there is some $\gamma > 0$ such that $(\vec{w^*} \cdot \vec{x^t})y^t > \gamma$ for any t. Equivalently, $\left\|\vec{w^*}\right\|_2 \leq \frac{1}{\gamma}$ and $(\vec{w^*} \cdot \vec{x^t})y^t > 1$.*

Algorithm 6 PERCEPTRON ALGORITHM

Initialize $\vec{w^1} = \vec{0} \in \mathbb{R}^d$
for $t = 1$ to T **do**
  Select $\vec{w^t}$
  Nature selects $\vec{x^t}$
  Predict $\hat{y}^t = \text{sign}(\vec{w^t} \cdot \vec{x^t})$
  Nature reveals y^t
  Update weights according to $\vec{w^{t+1}} = \begin{cases} \vec{w^t} & (\vec{w^t} \cdot \vec{x^t})y^t > 0 \\ \vec{w^t} + y^t\vec{x^t} & (\vec{w^t} \cdot \vec{x^t})y^t \leq 0 \end{cases}$
end for

7.4.2 Perceptron Algorithm

Theorem 7.11 *If there is a perfect expert $\vec{w^*}$ as in Def. 7.10, then the* PERCEPTRON *algorithm guarantees*

$$M_T(\text{PERCEPTRON}) \leq \frac{1}{\gamma^2}$$

Proof idea: Create a potential $\Phi_t := \left\|\vec{w^*} - \vec{w^t}\right\|_2^2$.