EE6550 Machine Learning

Lecture Six – On-Line Learning

Chung-Chin Lu

Department of Electrical Engineering

National Tsing Hua University

April 17, 2017

Motivations

- PAC learning:
  - Distribution is fixed but unknown over time (training and testing).
  - IID assumption for samples.
- On-line learning:
  - No distributional assumption.
  - Worst-case analysis (adversarial).
  - Mixed training and testing.
  - Performance measure: mistake model, regret.

The Contents of This Lecture

- Prediction with expert advice.
- Linear classification.
- On-line to batch conversion.

General On-Line Setting

- For $t = 1$ to T do
  - receive instance $\omega_t \in \mathscr{I}$;
  - predict $\hat{y}_t \in \mathscr{Y}'$;
  - receive label $c(\omega_t) \in \mathscr{Y}$ of ω_t;
  - incur loss $L(\hat{y}, c(\omega_t))$, where $L : \mathscr{Y}' \times \mathscr{Y} \to \mathbb{R}^+$ is the loss function.
- Classification: $\mathscr{Y}' = \mathscr{Y} = \{0, 1\}$ and $L(y', y) = 1_{y' \neq y}$.
- Regression: $\mathscr{Y}' = \mathscr{Y} = \mathbb{R}$ and $L(y', y) = (y' - y)^2$.
- Objective: minimizing the total loss $\sum_{t=1}^{T} L(\hat{y}_t, c(\omega_t))$.

Prediction with Expert Advice

- For $t = 1$ to T do
  - receive instance $\omega_t \in \mathscr{I}$ and advice $\hat{y}_{t,i} \in \mathscr{Y}$, $i \in [1, N]$;

wunderground.com
bbc.com
weather.com
cnn.com
?
algorithm

  - predict $\hat{y}_t \in \mathscr{Y}'$;
  - receive label $c(\omega_t) \in \mathscr{Y}$ of ω_t;
  - incur loss $L(\hat{y}, c(\omega_t))$.
- Objective: minimizing the (external) regret R_T, i.e., difference of total loss incurred and that of best expert,

$$R_T \triangleq \sum_{t=1}^{T} L(\hat{y}_t, c(\omega_t)) - \min_{i \in [1,N]} \sum_{t=1}^{T} L(\hat{y}_{t,i}, c(\omega_t)).$$

On-line Learning a Realizable Concept

- $\mathcal{H}$: the hypothesis set used by an on-line learning algorithm $\mathbb{A}$.
- C : a concept class to learn by $\mathbb{A}$.
- Assumption : $C \subseteq \mathcal{H}$.
- Question : How many mistakes before we learn a particular concept $c \in C$?
  - Since we are in the realizable case, i.e., we have assumed $C \subseteq \mathcal{H}$, after some number of rounds T, we will learn the concept and no longer make mistakes in subsequent rounds.

Mistake Bound Model for Realizable On-line Learning

- C : a concept class to learn which is assumed to be a subset of the hypothesis set $\mathcal{H}$.
- $S = (\omega_1, \ldots, \omega_T)$: a sample of T items with labels $(c(\omega_1), \ldots, c(\omega_T))$ from a fixed but unknown concept c.
- $\mathbb{A}$: an on-line learning algorithm which will output the target concept c from the hypothesis set $\mathcal{H}$ after T rounds based on the sample S.
- $\text{mistakes}(\mathbb{A}, c; S) = \{\omega_t, t \in [1, T] \mid \hat{y}_t \neq c(\omega_t)\}$: the set of all items in S for which the on-line algorithm $\mathbb{A}$ makes mistake before the target concept c is learned.

- $M_{\mathbb{A}}(c)$: the maximum number of mistakes the on-line learning algorithm $\mathbb{A}$ makes to learn a particular concept c,

$$M_{\mathbb{A}}(c) \triangleq \max_{S} |\text{mistakes}(\mathbb{A}, c; S)|.$$

- $M_{\mathbb{A}}(C)$: the maximum number of mistakes the on-line learning algorithm $\mathbb{A}$ makes to learn an arbitrary concept in the concept class C,

$$M_{\mathbb{A}}(C) \triangleq \max_{C} M_{\mathbb{A}}(c).$$

- A mistake bound of an on-line learning algorithm $\mathbb{A}$ for a concept class C is a bound for $M_{\mathbb{A}}(C)$.

The Halving Algorithm

```
HALVING($\mathcal{H}$)    ▷ $\mathcal{H}$ is the hypothesis set of the learning algorithm
 1. $\mathcal{H}_1 \leftarrow \mathcal{H}$
 2. for $t \leftarrow 1$ to $T$ do
 3.       RECEIVE($\omega_t$)
 4.       $\hat{y}_t \leftarrow$ MAJORITYVOTE($\mathcal{H}_t, \omega_t$)
 5.       RECEIVE($c(\omega_t)$)
 6.       if ($\hat{y}_t \neq c(\omega_t)$) then
 7.             $\mathcal{H}_{t+1} \leftarrow \{h \in \mathcal{H}_t \mid h(\omega_i) = c(\omega_i)\}$
 8.       else
 9.             $\mathcal{H}_{t+1} \leftarrow \mathcal{H}_t$
10. return $\mathcal{H}_{T+1}$
```

Finite Hypothesis Set – Realizable Concepts

Theorem 7.1: Let

- $\mathbb{A} = \text{Halving}$: the Halving algorithm is the on-line learning algorithm;
- $\mathcal{H}$: the hypothesis set of the on-line learning algorithm which is finite;
- $C = \mathcal{H}$.

Then

$$M_{\text{Halving}}(\mathcal{H}) \leq \log_2 |\mathcal{H}|.$$

Proof. It is clear from the Halving algorithm. $\square$

Finite Hypothesis Set – Realizable Concepts

Theorem 7.2: Let

- $\mathcal{H}$: the hypothesis set of the on-line learning algorithm which is finite;
- $C = \mathcal{H}$;
- $\mathrm{opt}(\mathcal{H})$: the optimal mistake bound for $\mathcal{H}$.

Then,

$$\mathrm{VCdim}(\mathcal{H}) \leq \mathrm{opt}(\mathcal{H}) \leq M_{\mathrm{Halving}}(\mathcal{H}) \leq \log_2 |\mathcal{H}|.$$

Proof. To prove the first inequality, we let $d = \mathrm{VCdim}(\mathcal{H})$.

- There exists a fully shattered set of d items by the hypothesis set $\mathcal{H}$.
- With such a fully shattered set of d items, we can form a

complete binary tree of expert advice with height d.

- Since in the worst case, there is no consensus among experts at any node of the complete binary tree of advice, we may choose labels at each round of learning so that d mistakes are made in d rounds.

Note that this adversarial argument is valid since the on-line setting makes no statistical assumptions about the data. $\square$

On-line Learning a Non-Realizable Concept

- N : the number of experts used by an on-line learning algorithm $\mathbb{A}$.
  - These N experts form the hypothesis set $\mathcal{H}$ of the on-line algorithm $\mathbb{A}$.
- C : a concept class to learn by $\mathbb{A}$.
  - In general, $C \not\subseteq \mathcal{H}$.
- Questions : How many mistakes m_T after T rounds of on-line learning ? What is the regret R_T ?
  - Since we are in the non-realizable case, we will continue to make mistakes in subsequent rounds.

The Weighted Majority Algorithm

```
WEIGHTED-MAJORITY($N$)
 1. **for** $i \leftarrow 1$ **to** $N$ **do**
 2.     $w_{1,i} \leftarrow 1$
 3. **for** $t \leftarrow 1$ **to** $T$ **do**
 4.     RECEIVE($\omega_t$)
 5.     **if** $\sum_{i:\hat{y}_{t,i}=1} w_{t,i} \geq \sum_{i:\hat{y}_{t,i}=0} w_{t,i}$ **then**
 6.         $\hat{y}_t \leftarrow 1$
 7.     **else**
 8.         $\hat{y}_t \leftarrow 0$
 9.     RECEIVE($c(\omega_t)$)
10.     **if** $(\hat{y}_t \neq c(\omega_t))$ **then**
```

```
11.         for $i \leftarrow 1$ to $N$ do
12.             if $(\hat{y}_{t,i} \neq c(\omega_t))$ then
13.                 $w_{t+1,i} \leftarrow \beta w_{t,i}$
14.             else
15.                 $w_{t+1,i} \leftarrow w_{t,i}$
16.     else
17.         for $i \leftarrow 1$ to $N$ do
18.             $w_{t+1,i} \leftarrow w_{t,i}$
19. return $\mathbf{w}_{T+1}$
```

Remarks

- The weighted majority (WM) algorithm weights the importance of experts as a function of their mistake rate.
  - The WM algorithm begins with uniform weights over all N experts.
- At each round, the WM algorithm generates predictions using a weighted majority vote.
- After receiving the true label, the algorithm then reduces the weight of each incorrect expert by a factor of $\beta \in [0, 1)$.
  - When $\beta = 0$, the weighted majority algorithm becomes to the halving algorithm.
- m_T : the number of mistakes made by the WM algorithm after T rounds of on-line learning.

- m_T^* : the number of mistakes made by the best expert in hindsight, i.e., after T rounds of on-line learning.
- The next theorem gives a bound for m_T as a function of m_T^*.

Weighted Majority - Bound for Non-Realizable Concepts

Theorem 7.3: Let

- N : the number of experts;
- $\beta \in (0, 1)$: the weight reduction factor.

Then

$$m_T \le \frac{\ln N + m_T^* \ln \frac{1}{\beta}}{\ln \frac{2}{1+\beta}}.$$

Proof.

- $W_t \triangleq \sum_{i=1}^{N} w_{t,i}, t \in [1, T]$: the potential function.
  - $W_1 = N$.
- A mistake is made at the t-th round (if and) only if

$$\sum_{i:\hat{y}_{t,i} \neq c(\omega_t)} w_{t,i} \ge \sum_{i:\hat{y}_{t,i} = c(\omega_t)} w_{t,i}.$$

Since $W_t = \sum_{i:\hat{y}_{t,i}\neq c(\omega_t)} w_{t,i} + \sum_{i:\hat{y}_{t,i}=c(\omega_t)} w_{t,i}$, we have

$$\sum_{i:\hat{y}_{t,i}\neq c(\omega_t)} w_{t,i} = \frac{W_t}{2} + \alpha \text{ and } \sum_{i:\hat{y}_{t,i}=c(\omega_t)} w_{t,i} = \frac{W_t}{2} - \alpha$$

for some $\alpha \geq 0$. Thus we have

$$\begin{aligned} W_{t+1} &= \sum_{i:\hat{y}_{t,i}\neq c(\omega_t)} \beta w_{t,i} + \sum_{i:\hat{y}_{t,i}=c(\omega_t)} w_{t,i} \\ &= \beta\left(\frac{W_t}{2} + \alpha\right) + \frac{W_t}{2} - \alpha \\ &= \left(\frac{1+\beta}{2}\right) W_t - (1-\beta)\alpha \\ &\leq \left(\frac{1+\beta}{2}\right) W_t. \end{aligned}$$

- Since there are m_T mistakes after T rounds, we have

$$W_{T+1} \leq \left(\frac{1+\beta}{2}\right)^{m_T} W_1.$$

- $m_{T,i}, i \in [1, N]$: the number of mistakes made by expert i after T rounds.
- $W_{T+1} = \sum_{j=1}^{N} w_{T+1,j} \geq w_{T+1,i} = \beta^{m_{T,i}}$ for all $i \in [1, N]$.

Now we have

$$\beta^{m_T^*} \leq W_{T+1} \leq \left(\frac{1+\beta}{2}\right)^{m_T} N$$

and then

$$m_T \leq \frac{\ln N + m_T^* \ln \frac{1}{\beta}}{\ln \frac{2}{1+\beta}}.$$

$\square$

Remarks

- The WM algorithm guarantees a bound for m_T,

$$m_T \leq O(\ln N) + \text{ constant } \cdot |\text{mistakes of the best expert}|.$$

- This bound requires no assumption about the sequence of items and labels generated.
- In the realizable case where $m_T^* = 0$, the bound reduces to $m_T \leq O(\ln N)$ as for the Halving algorithm.

Drawback of Deterministic Algorithms

- In the case of zero-one loss, no deterministic algorithm can achieve a regret $R_T = o(T)$ over all sequences of items.
- For any deterministic algorithm $\mathbb{A}$ and any $t \in [1, T]$, we can adversarially select $c(\omega_t)$ to be 1 if the algorithm predicts 0, and choose it to be 0 otherwise. Thus, $\mathbb{A}$ makes a mistake at every item of such a sequence and its cumulative mistake is $m_T = T$.
- The number m_T^* of mistakes made by the best expert is no greater than $T/2$.
  - Assume for example that $N = 2$ and that one expert always predicts 0, the other one always 1. The error of the best expert over that sequence (and in fact any sequence of that length) is then at most $m_T^* \leq T/2$.

- Thus for any deterministic algorithm, we have

$$R_T = m_T - m_T^* \geq T/2$$

  which shows that $R_T = o(T)$ cannot be achieved in general.
- This leads us to consider randomized algorithms.

The Randomized Weighted Majority Algorithm

```
RANDOMIZED-WEIGHTED-MAJORITY($N$)
 1. for $i \leftarrow 1$ to $N$ do
 2.     $w_{1,i} \leftarrow 1$
 3.     $p_{1,i} \leftarrow 1/N$
 4. for $t \leftarrow 1$ to $T$ do
 5.     for $i \leftarrow 1$ to $N$ do
 6.         if $(l_{t,i} = 1)$ then
 7.             $w_{t+1,i} \leftarrow \beta w_{t,i}$
 8.         else
 9.             $w_{t+1,i} \leftarrow w_{t,i}$
10.     $W_{t+1} \leftarrow \sum_{i=1}^{N} w_{t+1,i}$
```

```
11.        **for** $i \leftarrow 1$ **to** $N$ **do**
12.            $p_{t+1,i} \leftarrow w_{t+1,i}/W_{t+1}$
13. **return** $\mathbf{w}_{T+1}$
```

Randomized Scenario of On-Line Learning

- $Q = \{1, \ldots, N\}$: a set of N available actions.
  - For example, action i corresponds to the advice made by expert i.
- $\mathbf{p}_t$: a distribution over the set Q of N actions used by an on-line algorithm $\mathbb{A}$ to select action i to make a prediction $\hat{y}_t$ for the label $c(\omega_t)$ of the item ω_t at each round $t \in [1, T]$.
- $\mathbf{l}_t$: a loss vector received by the on-line algorithm $\mathbb{A}$, whose ith component $l_{t,i} \in \{0, 1\}$ is the zero-one loss associated with action i.
- $L_t = \sum_{i=1}^{N} p_{t,i} l_{t,i}$: the incurred expected loss at the tth round.
  - $L_t \leq 1$.
- $\mathcal{L}_T = \sum_{t=1}^{T} L_t$: the total loss incurred by the on-line algorithm $\mathbb{A}$ over T rounds.

- $\mathcal{L}_{T,i} = \sum_{t=1}^{T} l_{t,i}$: the total loss associated to action i.
- $\mathcal{L}_T^{\min} = \min_{i \in Q} \mathcal{L}_{T,i}$: the minimal loss of a single action.
- The regret R_T of the on-line algorithm after T rounds is also defined by the difference of the total loss of the algorithm and that of the best single action:

$$R_T = \mathcal{L}_T - \mathcal{L}_T^{\min}.$$

- The following theorem gives a strong guarantee on the regret R_T of the randomized weighted majority (RWM) algorithm, showing that it is in $O(\sqrt{T \ln N})$.

Randomized Weighted Majority - Bound for Non-Realizable Concepts

Theorem 7.4: Let

- N : the number of actions;
- $\beta \in [1/2, 1)$: the weight reduction factor.

Then the total loss of the RWM algorithm on any sequence of T items is

$$\mathcal{L}_T \leq \frac{\ln N}{1-\beta} + (2-\beta)\mathcal{L}_T^{\min}.$$

In particular, for $\beta = \max(1/2, 1 - \sqrt{(\ln N)/T})$, the total loss is

$$\mathcal{L}_T \leq \mathcal{L}_T^{\min} + \begin{cases} 2\sqrt{T \ln N}, & \text{if } \sqrt{(\ln N)/T} \leq 1/2, \\ 2\ln N + \frac{T}{2}, & \text{otherwise.} \end{cases}$$

Thus for sufficient large T such that $\sqrt{(\ln N)/T} \leq 1/2$, we have

$$\mathcal{L}_T \leq \mathcal{L}_T^{\min} + 2\sqrt{T \ln N}.$$

Proof.

- $W_t \triangleq \sum_{i=1}^N w_{t,i}, t \in [1,T]$: the potential function.
  - $W_1 = N$.
- By the RWM algorithm, for each $t \in [1,T]$, we have

$$\begin{aligned} W_{t+1} &= \sum_{i:l_{t,i}=1} \beta w_{t,i} + \sum_{i:l_{t,i}=0} w_{t,i} \\ &= W_t + (\beta - 1) \sum_{i:l_{t,i}=1} w_{t,i} \\ &= W_t + (\beta - 1) W_t \sum_{i:l_{t,i}=1} p_{t,i} \qquad \text{since } p_{t,i} = \frac{w_{t,i}}{W_t} \\ &= W_t + (\beta - 1) W_t L_t = W_t(1 - (1-\beta)L_t). \end{aligned}$$

- Since $W_1 = N$, we have

$$W_{T+1} = N \prod_{t=1}^{T} (1 - (1-\beta) L_t).$$

- $W_{T+1} = \sum_{j=1}^{N} w_{T+1,j} \geq \max_{i \in Q} w_{T+1,i} = \beta^{\mathcal{L}_T^{\min}}$.

Now we have

$$\beta^{\mathcal{L}_T^{\min}} \leq W_{T+1} = N \prod_{t=1}^{T} (1 - (1-\beta) L_t)$$

and then

$$
\begin{aligned}
& \mathcal{L}_T^{\min} \ln \beta \leq \ln N + \sum_{t=1}^{T} \ln(1-(1-\beta)L_t) \\
\Rightarrow\quad & \mathcal{L}_T^{\min} \ln \beta \leq \ln N - (1-\beta)\sum_{t=1}^{T} L_t \text{ since } \ln(1-x) \leq -x \;\forall\; x < 1 \\
\Rightarrow\quad & \mathcal{L}_T^{\min} \ln \beta \leq \ln N - (1-\beta)\mathcal{L}_T \\
\Rightarrow\quad & \mathcal{L}_T \leq \frac{\ln N}{1-\beta} - \frac{\ln \beta}{1-\beta}\mathcal{L}_T^{\min} \\
\Rightarrow\quad & \mathcal{L}_T \leq \frac{\ln N}{1-\beta} - \frac{\ln(1-(1-\beta))}{1-\beta}\mathcal{L}_T^{\min}.
\end{aligned}
$$

Since $-\ln(1-x) \leq x + x^2$ for all $x \in [0, 1/2]$ and $\beta \in [1/2, 1)$, we have

$$
-\ln(1-(1-\beta)) \leq (1-\beta) + (1-\beta)^2 = (1-\beta)(2-\beta)
$$

and then

$$\mathcal{L}_T \le \frac{\ln N}{1-\beta} + (2-\beta)\mathcal{L}_T^{\min}.$$

Since $\mathcal{L}_T^{\min} \le T$, we have

$$\mathcal{L}_T \le \mathcal{L}_T^{\min} + \frac{\ln N}{1-\beta} + (1-\beta)T.$$

The upper bound in above is minimized with
$\beta = \beta_0 = 1 - \sqrt{(\ln N)/T}$ if $\sqrt{(\ln N)/T} \le 1/2$ and with
$\beta = \beta_0 = 1/2$ if $\sqrt{(\ln N)/T} > 1/2$. With $\beta = \beta_0$, we have

$$\mathcal{L}_T \le \mathcal{L}_T^{\min} + \begin{cases} 2\sqrt{T\ln N}, & \text{if } \sqrt{(\ln N)/T} \le 1/2, \\ 2\ln N + \frac{T}{2}, & \text{otherwise.} \end{cases}$$

$\square$

Remarks

- The RWM algorithm has the average regret or regret per round R_T/T decreases as $O(1/\sqrt{T})$.
- These results are optimal since for $T \geq N$, a lower bound of $R_T = \Omega(\sqrt{T \ln N})$ can be proven for any algorithm or from a result shown in the following theorem.

Khintchine-Kahane Inequality

Let

- $\sigma = (\sigma_1, \ldots, \sigma_m)$: Rademacher variables, i.e., i.i.d. random variables taking values in $\{-1, +1\}$ with equal probability.
- $\mathbf{x}_i, \ldots, \mathbf{x}_m$: vectors in a normed vector space $(\mathbb{H}, \|\cdot\|)$.

Then we have

$$\frac{1}{2} \underset{\sigma}{E}\left[\left\|\sum_{i=1}^{m} \sigma_i \mathbf{x}_i\right\|^2\right] \leq \left(\underset{\sigma}{E}\left[\left\|\sum_{i=1}^{m} \sigma_i \mathbf{x}_i\right\|\right]\right)^2 \leq \underset{\sigma}{E}\left[\left\|\sum_{i=1}^{m} \sigma_i \mathbf{x}_i\right\|^2\right].$$

A Lower Bound of Regret for Randomized Algorithms

Theorem 7.5: Let

- $N = 2$.

There exists a stochastic sequence of losses for which the regret of any random on-line learning algorithm has $E[R_T] \geq \sqrt{T/8}$.

Proof.

- For each $t \in [1, T]$, the loss vector $\mathbf{l}_t$ takes the values $\mathbf{l}_{01} = (0,1)^T$ and $\mathbf{l}_{10} = (1,0)^T$ with equal probability.
- The expected total loss of any random on-line algorithm is

$$E[\mathcal{L}_T] = E\left[\sum_{t=1}^{T} \mathbf{p}_t \cdot \mathbf{l}_t\right] = \sum_{t=1}^{T} \mathbf{p}_t \cdot E[\mathbf{l}_t] = \sum_{t=1}^{T} \frac{1}{2} p_{t,1} + \frac{1}{2}(1 - p_{t,1}) = \frac{T}{2},$$

where $\mathbf{p}_t$ is the distribution selected by the random on-line algorithm at round t.

- $\mathcal{L}_{T,1} + \mathcal{L}_{T,2} = T$.
- $\mathcal{L}_T^{\min} = \min(\mathcal{L}_{T,1}, \mathcal{L}_{T,2}) = \frac{1}{2}(\mathcal{L}_{T,1} + \mathcal{L}_{T,2} - |\mathcal{L}_{T,1} - \mathcal{L}_{T,2}|) = \frac{T}{2} - |\mathcal{L}_{T,1} - \frac{T}{2}|$.
- The expected regret of the random on-line algorithm is
$$E[R_T] = E[\mathcal{L}_T] - E[\mathcal{L}_T^{\min}] = E[|\mathcal{L}_{T,1} - \frac{T}{2}|].$$
- $\sigma_t, t \in [1,T]$: Rademacher variables taking values in $\{-1,+1\}$ with equal probability.
- $\mathcal{L}_{T,1} = \sum_{t=1}^T \frac{1+\sigma_t}{2} = \frac{T}{2} + \frac{1}{2}\sum_{t+1}^T \sigma_t$.
- With $\alpha_t = 1/2$ for all $t \in [1,T]$, we have
$$E[R_T] = E\left[\left|\sum_{t=1}^T \sigma_t \alpha_t\right|\right] \geq \sqrt{\frac{1}{2}\left|\sum_{t=1}^T \sigma_t\alpha_t\right|^2} = \sqrt{\frac{1}{2}\sum_{t=1}^T \alpha_t^2} = \sqrt{\frac{T}{8}}$$

by the Khintchine-Kahane inequality and the fact that $E[\sigma_i\sigma_j] = 0$ for all $i \neq j$. $\square$

The Exponential Weighted Average Algorithm

```
Exponential-Weighted-Average($N$)
1. for $i \leftarrow 1$ to $N$ do
2.     $w_{1,i} \leftarrow 1$
3. for $t \leftarrow 1$ to $T$ do
4.     Receive($\omega_t$)
5.     $\hat{y}_t \leftarrow \frac{\sum_{i=1}^N w_{t,i}\hat{y}_{t,i}}{\sum_{i=1}^N w_{t,i}}$
6.     Receive($c(\omega_t)$)
7.     for $i \leftarrow 1$ to $N$ do
8.         $w_{t+1,i} \leftarrow w_{t,i} e^{-\eta L(\hat{y}_{t,i}, c(\omega_t))}$
9. return $\mathbf{w}_{T+1}$
```

Remarks

- The EWA algorithm can be extended to other loss functions L taking values in $[0, 1]$.
- For the exponential weighted average (EWA) algorithm presented here, we assume that the loss function $L(y', y)$ is convex in its first argument y'.
- Although the EWA algorithm is deterministic, it admits a very favorable regret guarantee, as shown in the next theorem.
- The EWA algorithm's prediction is the weighted average of the advice of N experts,

$$\hat{y}_t = \frac{\sum_{i=1}^{N} w_{t,i} \hat{y}_{t,i}}{\sum_{i=1}^{N} w_{t,i}}.$$

- The EWA algorithm updates the weights at the end of round t according to the following rule:

$$w_{t+1,i} = w_{t,i} e^{-\eta L(\hat{y}_{t,i}, c(\omega_t))} = e^{-\eta L_{t,i}},$$

where $L_{t,i} = \sum_{s=1}^{t} L(\hat{y}_{s,i}, c(\omega_s))$ is the total loss incurred by expert i after t rounds.

Hoeffding's Lemma

Let

- X : a r.v. with zero mean, i.e., $E[X] = 0$.
- $a \leq X \leq b$ with $b > a$.

Then for any $t > 0$, we have

$$E[e^{tX}] \leq e^{\frac{t^2(b-a)^2}{8}}.$$

A Regret Bound for the EWA Algorithm

Theorem 7.6: Let

- $\{1, 2, \ldots, N\}$: the set of N experts.
- $L(y', y)$: a loss function which is convex in the first argument y' and takes values in $[0, 1]$;
- $\eta > 0$: a weight update parameter.

Then, for any label sequence $c(\omega_1), \ldots, c(\omega_T)$, the regret of the exponential weighted average (EWA) algorithm after T rounds has

$$R_T \leq \frac{\ln N}{\eta} + \frac{\eta T}{8}.$$

In particular, for $\eta = \sqrt{8(\ln N)/T}$, the regret is bounded as

$$R_T \leq \sqrt{(T/2) \ln N}.$$

Proof.

- $\Phi_t \triangleq \ln\left(\sum_{i=1}^N w_{t,i}\right), t \in [1,T]$: the potential function.
  - $\Phi_1 = \ln N$.
- $\mathbf{p}_t, t \in [1,T]$: the distribution over $\{1,2,\ldots,N\}$ with $p_{t,i} = \frac{w_{t,i}}{\sum_{i=1}^N w_{t,i}}$.
- By the EWA algorithm, the difference of two consecutive potential values is

$$\Phi_{t+1} - \Phi_t = \ln \frac{\sum_{i=1}^N w_{t,i} e^{-\eta L(\hat{y}_{t,i}, c(\omega_t))}}{\sum_{i=1}^N w_{t,i}} = \ln \underset{\mathbf{p}_t}{E}[e^{\eta X}],$$

  where $X(i) = -L(\hat{y}_{t,i}, c(\omega_t)) \in [-1,0]$ is a r.v. on $[1,N]$ with distribution $\mathbf{p}_t$.
- By applying Hoeffding's lemma to the centered r.v. $X - \underset{\mathbf{p}_t}{E}[X]$,

we have

$$
\begin{aligned}
\Phi_{t+1} - \Phi_t &= \ln \underset{\mathbf{p}_t}{E}[e^{\eta(X - \underset{\mathbf{p}_t}{E}[X]) + \eta \underset{\mathbf{p}_t}{E}[X]}] \\
&\leq \frac{\eta^2}{8} + \eta \underset{\mathbf{p}_t}{E}[X] \\
&= \frac{\eta^2}{8} - \eta \underset{\mathbf{p}_t}{E}[L(\hat{y}_{t,i}, c(\omega_t))] \\
&\leq \frac{\eta^2}{8} - \eta L(\underset{\mathbf{p}_t}{E}[\hat{y}_{t,i}], c(\omega_t)) \\
&\quad \text{by the convexity of } L \text{ in the first argument} \\
&= \frac{\eta^2}{8} - \eta L(\hat{y}_t, c(\omega_t)).
\end{aligned}
$$

- Summing up for $t \in [1, T]$, we have an upper bound of the potential function

$$\Phi_{T+1} - \Phi_1 \leq \frac{\eta^2 T}{8} - \eta \sum_{t=1}^{T} L(\hat{y}_t, c(\omega_t)).$$

- A lower bound can be obtained as

$$\begin{aligned} \Phi_{T+1} - \Phi_1 &= \ln \sum_{i=1}^{N} e^{-\eta L_{T,i}} - \ln N \\ &\geq \ln \max_{i \in [1,N]} e^{-\eta L_{T,i}} - \ln N \\ &= -\eta \min_{i \in [1,N]} L_{T,i} - \ln N. \end{aligned}$$

Now we have

$$-\eta \min_{i \in [1,N]} L_{T,i} - \ln N \leq \frac{\eta^2 T}{8} - \eta \sum_{t=1}^{T} L(\hat{y}_t, c(\omega_t))$$

which implies that

$$R_T = \sum_{t=1}^{T} L(\hat{y}_t, c(\omega_t)) - \min_{i \in [1,N]} L_{T,i} \leq \frac{\ln N}{\eta} + \frac{\eta T}{8}.$$

The upper bound in above is minimized with $\eta = \sqrt{8(\ln N)/T}$ and we have

$$R_T \leq \sqrt{(T/2) \ln N}.$$

$\square$

Remarks

- The optimal choice of η in Theorem 7.6 requires knowledge of the horizon T, which is an apparent disadvantage of this analysis.
- However, we can use a standard doubling trick to eliminate this requirement, at the price of a small constant factor.
- The standard doubling trick : dividing time into periods $[2^k, 2^{k+1} - 1]$ of length 2^k with $k = 0, \ldots, n$.
- Assume that $2^n \le T \le 2^{n+1} - 1$.
- Choose $\eta_k = \sqrt{\frac{8 \ln N}{2^k}}$ in each period $[2^k, 2^{k+1} - 1]$ of length 2^k with $k = 0, \ldots, n$.
  - Reset $w_{t,i} \leftarrow 1/N$ at each $t = 2^k$ for $k = 0, \ldots, n$.

A Regret Bound for the EWA Algorithm with Doubling Trick

Theorem 7.7: Let

- $\{1, 2, \ldots, N\}$: the set of N experts.
- $L(y', y)$: a loss function which is convex in the first argument y' and takes values in $[0, 1]$.

Then, for any label sequence $c(\omega_1), \ldots, c(\omega_T)$, the regret of the exponential weighted average (EWA) algorithm after T rounds has

$$R_T \leq \frac{\sqrt{2}}{\sqrt{2}-1}\sqrt{(T/2)\ln N} - \frac{1}{\sqrt{2}-1}\sqrt{(\ln N)/2}.$$

Proof.

- $n \triangleq \lfloor \log_2 T \rfloor$.
- $I_k \triangleq [2^k, 2^{k+1}-1], k \in [0, n]$.
- $\mathcal{L}_{I_k}$: the total loss incurred in the interval I_k.
- Since we reset $w_{t,i} \leftarrow 1/N$ at each $t = 2^k$ for $k \in [0, n]$ and choose $\eta = \eta_k = \sqrt{8 \ln N / 2^k}$ for $t \in I_k = [2^k, 2^{k+1}-1]$, we have

$$\begin{aligned}
\mathcal{L}_{I_k} - \min_{i \in [1,N]} \mathcal{L}_{I_k,i} &\leq \sqrt{(2^k/2) \ln N},\ \forall\ k \in [0, n-1], \\
\mathcal{L}_{[2^n,L]} - \min_{i \in [1,N]} \mathcal{L}_{[2^n,L],i} &\leq \sqrt{((T+1-2^n)/2) \ln N} \\
&\leq \sqrt{(2^n/2) \ln N}
\end{aligned}$$

by Theorem 7.6, where $\mathcal{L}_{[2^n,L]} = \sum_{t=2^n}^{T} L(\hat{y}_t, c(\omega_t))$ and $\mathcal{L}_{[2^n,T],i} = \sum_{t=2^n}^{T} L(\hat{y}_{t,i}, c(\omega_t))$.

Thus we have

$$
\begin{aligned}
\mathcal{L}_T &= \sum_{k=0}^{n-1} \mathcal{L}_{I_k} + \mathcal{L}_{[2^n,L]} \\
&\le \sum_{k=0}^{n-1} \min_{i\in[1,N]} \mathcal{L}_{I_k,i} + \min_{i\in[1,N]} \mathcal{L}_{[2^n,L],i} + \sum_{k=0}^{n} \sqrt{2^k(\ln N)/2} \\
&\le \min_{i\in[1,N]} \left(\sum_{k=0}^{n} \mathcal{L}_{I_k,i} + \mathcal{L}_{[2^n,L],i} \right) + \sqrt{(\ln N)/2} \sum_{k=0}^{n} 2^{k/2} \\
&= \min_{i\in[1,N]} \mathcal{L}_{T,i} + \sqrt{(\ln N)/2}\, \frac{2^{(n+1)/2}-1}{\sqrt{2}-1}.
\end{aligned}
$$

Since $2^n \le T$, we have

$$
\frac{2^{(n+1)/2}-1}{\sqrt{2}-1} \le \frac{\sqrt{2T}-1}{\sqrt{2}-1} \le \frac{\sqrt{2}}{\sqrt{2}-1}\sqrt{T} - \frac{1}{\sqrt{2}-1}
$$

so that

$$R_T = \mathcal{L}_T - \min_{i \in [1,N]} \mathcal{L}_{T,i} \leq \frac{\sqrt{2}}{\sqrt{2}-1}\sqrt{(T/2)\ln N} - \frac{1}{\sqrt{2}-1}\sqrt{(\ln N)/2}.$$

$\square$

The Contents of This Lecture

- Prediction with expert advice.
- Linear classification.
- On-line to batch conversion.

The Perceptron Algorithm

- The perceptron algorithm learns a linear hypothesis (without an offset), i.e., a separating hyperplane (through the origin), by processing training points one at a time.
- The algorithm maintains a weight vector $\mathbf{w}_t \in \mathbb{R}^N$ defining the hyperplane learned, starting with an arbitrary vector $\mathbf{w}_0$.
- At each round $t \in [1, T]$, it predicts the label of the point $\mathbf{x}_t \in \mathbb{R}^N$ received, using the current vector $\mathbf{w}_t$.
- When the prediction made does not match the correct label, it updates $\mathbf{w}_t$ by adding $c(\mathbf{x}_t)\mathbf{x}_t$.
- More generally, when a learning rate $\eta > 0$ is used, the vector added is $\eta c(\mathbf{x}_t)\mathbf{x}_t$.

- This update can be partially motivated by examining the inner products of the current and the updated weight vector with $c(\mathbf{x}_t)\mathbf{x}_t$:
  - Just before an update, $\mathbf{x}_t$ is misclassified and thus the inner product $c(\mathbf{x}_t)\mathbf{w}_t \cdot \mathbf{x}_t$ is negative;
  - Afterward, the inner product becomes to $c(\mathbf{x}_t)\mathbf{w}_{t+1} \cdot \mathbf{x}_t = c(\mathbf{x}_t)\mathbf{w}_t \cdot \mathbf{x}_t + \eta\|\mathbf{x}_t\|^2$;
  - Thus, the update corrects the weight vector in the direction of making the inner product positive by augmenting it with this quantity $\eta\|\mathbf{x}_t\|^2$.

The Perceptron Algorithm

```
PERCEPTRON($\mathbf{w}_0$)
 1. $\mathbf{w}_1 \leftarrow \mathbf{w}_0$        $\triangleright$ typically $\mathbf{w}_0 = \mathbf{0}$
 2. for $t \leftarrow 1$ to $T$ do
 3.     RECEIVE($\mathbf{x}_t$)
 4.     $\hat{y}_t \leftarrow \text{sgn}(\mathbf{w}_t \cdot \mathbf{x}_t)$
 5.     RECEIVE($c(\mathbf{x}_t)$)
 6.     if $(\hat{y}_t \neq c(\mathbf{x}_t))$ then
 7.         $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + c(\mathbf{x}_t)\mathbf{x}_t$   $\triangleright$ more generally $\eta c(\mathbf{x}_t)\mathbf{x}_t, \eta > 0$
 8.     else
 9.         $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t$
10. return $\mathbf{w}_{T+1}$
```

An Interpretation of the Perceptron Algorithm

- The Perceptron algorithm can be seen to seek a weight vector $\mathbf{w}$ minimizing the following objective function

$$F(\mathbf{w}) \triangleq \frac{1}{T}\sum_{t=1}^{T} \max(0, -c(\mathbf{x}_t)(\mathbf{w}\cdot\mathbf{x}_t)) = \underset{\mathbf{x}\sim\hat{D}}{E}[\tilde{F}(\mathbf{w},\mathbf{x})],$$

  where
  - $\tilde{F}(\mathbf{w},\mathbf{x}) \triangleq \max(0, -c(\mathbf{x})(\mathbf{w}\cdot\mathbf{x}))$;
  - $\hat{D}$: the empirical distribution associated with the sequence $\mathbf{x}_1,\ldots,\mathbf{x}_T$ of T points, which is uniform over the T points.

- For any $t \in [1,T]$, $\mathbf{w} \mapsto -c(\mathbf{x}_t)(\mathbf{w}\cdot\mathbf{x}_t)$ is linear and then convex on $\mathbf{w}$.

- Since the max operator preserves convexity, this shows that $F(\mathbf{w})$ is convex. However, F is not differentiable.

Stochastic (On-line) Gradient Descent Technique

- This technique processes a point $\mathbf{x}_t$ at each round t, $t \in [1, T]$.
- $\eta > 0$: a learning rate parameter.
- $\tilde{F}(\mathbf{w}, \mathbf{x}_t)$: the object function to minimize at round t.
- The update rule of the weight vector $\mathbf{w}$ at round t is

$$\mathbf{w}_{t+1} \leftarrow \begin{cases} \mathbf{w}_t - \eta \nabla_{\mathbf{w}} \tilde{F}(\mathbf{w}_t, \mathbf{x}_t), & \text{if } \mathbf{w} \mapsto \tilde{F}(\mathbf{w}, \mathbf{x}_t) \text{ is} \\ & \text{differentiable at } \mathbf{w}_t, \\ \mathbf{w}_t, & \text{otherwise.} \end{cases}$$

- For the perceptron algorithm, the object function is

$$\tilde{F}(\mathbf{w}, \mathbf{x}_t) = \max(0, -c(\mathbf{x}_t)(\mathbf{w} \cdot \mathbf{x}_t))$$

and

- when $c(\mathbf{x}_t)(\mathbf{w}_t \cdot \mathbf{x}_t) > 0$, $\tilde{F}(\mathbf{w}, \mathbf{x}_t) = 0$ is differentiable at $\mathbf{w}_t$ and

$$\nabla_{\mathbf{w}} \tilde{F}(\mathbf{w}_t, \mathbf{x}_t) = \mathbf{0}.$$

- when $c(\mathbf{x}_t)(\mathbf{w}_t \cdot \mathbf{x}_t) < 0$, $\tilde{F}(\mathbf{w}, \mathbf{x}_t) = -c(\mathbf{x}_t)(\mathbf{w} \cdot \mathbf{x}_t)$ is also differentiable at $\mathbf{w}_t$ and

$$\nabla_{\mathbf{w}} \tilde{F}(\mathbf{w}_t, \mathbf{x}_t) = -c(\mathbf{x}_t)\mathbf{x}_t.$$

Thus the stochastic gradient descent update rule becomes

$$\mathbf{w}_{t+1} \leftarrow \begin{cases} \mathbf{w}_t + \eta c(\mathbf{x}_t)\mathbf{x}_t, & \text{if } c(\mathbf{x}_t)(\mathbf{w} \cdot \mathbf{x}_t) < 0, \\ \mathbf{w}_t, & \text{if } c(\mathbf{x}_t)(\mathbf{w} \cdot \mathbf{x}_t) > 0, \\ \mathbf{w}_t, & \text{otherwise,} \end{cases}$$

which coincides exactly with the update rule of the perceptron algorithm.

On-Line Learning Interpretation of the Perceptron Algorithm

- The Perceptron algorithm is closely related to the weighted majority (WM) algorithm as follows.
- When $x_{t,i} \in \{-1, +1\}$, N is interpreted as the number of experts and $\mathbf{x}_t$ is the feature vector of the item with $x_{t,i}$ regarded as the advice $\hat{y}_{t,i}$ of expert i.
- The classification rule

$$\operatorname{sgn}(\mathbf{w}_t \cdot \mathbf{x}_t) = \operatorname{sgn}\left(\sum_{t:x_{t,i}=1} w_{t,i} - \sum_{t:x_{t,i}=-1} w_{t,i} \right)$$

coincides with the weighted majority vote.

- When a mistake occurs in the t-th round, adding the weight $w_{t,i}$ by $\eta c(\mathbf{x}_t)x_{t,i}$ is equivalent to adding the weight $w_{t,i}$ of the correct (i.e., $c(\mathbf{x}_t) = x_{t,i}$) or the incorrect (i.e., $c(\mathbf{x}_t) = -x_{t,i}$) experts by $\eta|x_{t,i}|$ or $-\eta|x_{t,i}|$ respectively.
- The weight update rule of the Perceptron algorithm emphasizes the correct experts and deemphasizes the incorrect experts.

A Margin-Based Upper Bound on the Number of Mistakes Made by the Perceptron Algorithm – Linearly Separable Case

Theorem 7.8: Let

- $\mathbf{x}_1, \ldots, \mathbf{x}_T$: a sequence of T points in $\mathbb{R}^N$ with $\|\mathbf{x}_t\| \leq r$ for all $t \in [1, T]$, for some $r > 0$.

Assume that there exist $\rho > 0$ and $\mathbf{v} \in \mathbb{R}^N$ such that for all $t \in [1, T]$,

$$\rho \leq \frac{c(\mathbf{x}_t)(\mathbf{v} \cdot \mathbf{x}_t)}{\|\mathbf{v}\|}.$$

- The T points are linearly separable with a margin at least ρ.

Then, the number of updates made by the Perceptron algorithm when processing $\mathbf{x}_1, \ldots, \mathbf{x}_T$ is bounded by r^2/ρ^2.

Proof. Let

- I : the subset of T rounds at which the perceptron algorithm makes an update.
- $M = |I|$: the number of updates the perceptron algorithm makes during T rounds.

By the margin ρ, we have

$$\begin{aligned} M\rho &\leq \frac{\mathbf{v} \cdot \sum_{t \in I} c(\mathbf{x}_t)\mathbf{x}_t}{\|\mathbf{v}\|} \\ &\leq \left\| \sum_{t \in I} c(\mathbf{x}_t)\mathbf{x}_t \right\| \text{ by Cauchy-Schwartz inequality} \\ &= \frac{1}{\eta}\|\mathbf{w}_{T+1}\| \end{aligned}$$

But

$$\begin{aligned}
\|\mathbf{w}_{T+1}\| &= \sqrt{\sum_{t\in I}(\|\mathbf{w}_{t+1}\|^2 - \|\mathbf{w}_t\|^2)} \text{ note that } \mathbf{w}_0 = \mathbf{0} \\
&= \sqrt{\sum_{t\in I}(\|\mathbf{w}_t + \eta c(\mathbf{x}_t)\mathbf{x}_t\|^2 - \|\mathbf{w}_t\|^2)} \\
&= \sqrt{\sum_{t\in I}(2\eta c(\mathbf{x}_t)\mathbf{w}_t \cdot \mathbf{x}_t + \eta^2\|\mathbf{x}_t\|^2)} \\
&\leq \eta\sqrt{\sum_{t\in I}\|\mathbf{x}_t\|^2} \leq \eta\sqrt{Mr^2}
\end{aligned}$$

since $\eta c(\mathbf{x}_t)\mathbf{w}_t \cdot \mathbf{x}_t \leq 0$ for each $t \in I$. This shows that $M \leq r^2/\rho^2$.
$\square$

Remarks

- The final weight vector $\mathbf{w}_{T+1}$ of the perceptron algorithm after T rounds is a linear combination of the vectors $\mathbf{x}_t$ at which an update was made:

$$\mathbf{w}_{T+1} = \sum_{t \in I} \eta c(\mathbf{x}_t)\mathbf{x}_t.$$

  Thus, as in the case of SVMs, these vectors can be referred to as support vectors for the perceptron algorithm.

- The bound of Theorem 7.8 is remarkable, since it depends only on the normalized margin ρ/r and not on the dimension N of the space.

- This bound can be shown to be tight, that is, the number of updates can be equal to r^2/ρ^2 in some sequences of points (see Exercise 7.3 to show the upper bound is tight).

- Theorem 7.8 requires no assumption about the sequence of points $\mathbf{x}_1, \ldots, \mathbf{x}_T$.
- For a sample $S = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$ of size $m < r^2/\rho^2$, the perceptron algorithm will make a number of passes over these m points so that the result of Theorem 7.8 implies that when the sample S is linearly separable, the perceptron algorithm will converge after a number of passes over these m points.
  - For a small margin ρ, the convergence of the algorithm can be quite slow.
  - In fact, for some samples S, regardless of the order in which the points in S are processed, the number of updates made by the perceptron algorithm is in $\Omega(2^N)$ (see Exercise 7.1).

- Of course, if S is not linearly separable, the perceptron algorithm does not converge.
  - In practice, without knowing whether S is linearly separable, the perceptron algorithm stops after some number of passes over S.

Remarks

- There are many variants of the standard Perceptron algorithm which are used in practice and have been theoretically analyzed.
- One notable example is the voted Perceptron algorithm, which returns the hypothesis

$$\hat{y} = \operatorname{sgn}\left(\left(\frac{1}{T}\sum_{t=1}^{T}\mathbf{w}_{t+1}\right)\cdot\mathbf{x}\right),$$

where $\mathbf{w}_t, t \in [1, T]$ is the weight vector created in the standard Perceptron algorithm. The final weight vector of the voted Perceptron algorithm is

$$\frac{1}{T}\sum_{t=1}^{T}\mathbf{w}_{t+1} = \sum_{s\in I} p_{s+1}\mathbf{w}_{s+1},$$

where p_{s+1} is the proportion of the number of iterations that $\mathbf{w}_{s+1}$ survives in T rounds.

Definition: Leave-One-Out Error

- c : a fixed by unknown concept to learn.
- $\mathbb{A}$: a learning algorithm with a hypothesis set $\mathcal{H}$.
- $S = (\omega_1, \ldots, \omega_m)$: a sample of size m with labels $(c(\omega_1), \ldots, c(\omega_m))$.
- $h_{S \setminus \{\omega_i\}} = \mathbb{A}(S \setminus \{\omega_i\}; c, \mathcal{H})$: the learned hypothesis by $\mathbb{A}$ with the reduced sample $S \setminus \{\omega_i\}$ of size $m - 1$.
- $\hat{R}_{S,LOO}(\mathbb{A}) = \frac{1}{m} \sum_{i=1}^{m} 1_{h_{S \setminus \{\omega_i\}}(\omega_i) \neq c(\omega_i)}$: the leave-one-out error of $\mathbb{A}$ on the sample S, which is the arithmetic average of the errors of the returned hypotheses $h_{S \setminus \{\omega_i\}}$ by $\mathbb{A}$ with the training set $S \setminus \{\omega_i\}$ on the item ω_i for $i \in [1, m]$.
  - The leave-one-out error $\hat{R}_{S,LOO}(\mathbb{A})$ of $\mathbb{A}$ on the sample S is just the m-fold cross-validation error of the learning algorithm $\mathbb{A}$ on the training set S.

A Property of Leave-One-Out Error

Lemma 4.1: The leave-one-out error $\hat{R}_{S',LOO}(\mathbb{A})$ of a learning algorithm $\mathbb{A}$ on a random sample S' of size $m+1$ drawn i.i.d. from an input space $\mathscr{I}$ according to a distribution D is an unbiased estimate of the expectation of the generalization error $R(h_S)$ of the learned hypothesis h_S by $\mathbb{A}$ with a random training set S of size m drawn i.i.d. form the input space $\mathscr{I}$ according to the distribution D,

$$\underset{S'\sim D^{m+1}}{E}[\hat{R}_{S',LOO}(\mathbb{A})] = \underset{S\sim D^m}{E}[R(h_S)].$$

Proof.

$$
\begin{aligned}
& \underset{S' \sim D^{m+1}}{E}[\hat{R}_{S',LOO}(\mathbb{A})] \\
= \quad & \underset{S' \sim D^{m+1}}{E}\left[\frac{1}{m+1}\sum_{i=1}^{m+1} 1_{h_{S' \setminus \{\omega_i\}}(\omega_i) \neq c(\omega_i)}\right] \\
= \quad & \frac{1}{m+1}\sum_{i=1}^{m+1} \underset{S' \sim D^{m+1}}{E}\left[1_{h_{S' \setminus \{\omega_i\}}(\omega_i) \neq c(\omega_i)}\right] \\
= \quad & \underset{S \sim D^m, \omega \sim D}{E}\left[1_{h_S(\omega) \neq c(\omega)}\right] \text{ by symmetry} \\
= \quad & \underset{S \sim D^m}{E}\left[\underset{\omega \sim D}{E}\left[1_{h_S(\omega) \neq c(\omega)} | S\right]\right] \text{ by conditional expectation} \\
= \quad & \underset{S \sim D^m}{E}\left[\underset{\omega \sim D}{E}\left[1_{h_S(\omega) \neq c(\omega)}\right]\right] \\
& \text{ since } \omega \text{ is drawn statistically independent of drawing } S \\
= \quad & \underset{S \sim D^m}{E}[R(h_S)].
\end{aligned}
$$

□

A Margin-Based Upper Bound on the Expected Generalization Error of Hypothesis Returned by the Perceptron Algorithm when Processing a Linearly Separated Data Sequence

Theorem 7.9: Assume that

- the data is linearly separable.
  - This implies that any random sample $S = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$ of size m drawn i.i.d. from an input space $\mathscr{I} \subseteq \mathbb{R}^N$ according to a distribution D and a fixed but unknown concept c is linearly separable.
  - ρ_S : the largest margin of a separating hyperplane for S.
  - r_S : the radius of the smallest sphere containing all points in S.
  - h_S^{Percpt} : the hypothesis returned by the perceptron

algorithm after trained via a number of passes till convergence over S.

Then, the expected generalization error of h_S^{Percpt} is bounded above by

$$\underset{S \sim D^m}{E}[R(h_S^{Percpt})] \leq \underset{S' \sim D^{m+1}}{E}\left[\frac{r_{S'}^2/\rho_{S'}^2}{m+1}\right].$$

Proof. Let

- S' : a linearly separable random sample of size $m+1$ drawn i.i.d. from an input space $\mathscr{I} \subseteq \mathbb{R}^N$ according to the distribution D and the fixed but unknown concept c.
- $\mathbf{x}_i$: a point in S'.
- $M(S')$: the number of updates made by the perceptron algorithm after trained via a number of passes till convergence over S'.

If $\mathbf{x}_i$ is not a support vector for $h_{S'}^{Percpt}$, then $h_{S' \setminus \{\mathbf{x}_i\}}^{Percpt}$ will classify

$\mathbf{x}_i$ correctly. Equivalently, if $h^{Percpt}_{S'\setminus\{\mathbf{x}_i\}}$ misclassifies $\mathbf{x}_i$, then $\mathbf{x}_i$ must be a support vector for $h^{Percpt}_{S'}$. Thus the leave-one-out error of the perceptron algorithm on the sample S' is

$$\hat{R}_{S',LOO}(\text{Perceptron}) \leq \frac{M(S')}{m+1} \leq \frac{r^2_{S'}/\rho^2_{S'}}{m+1}.$$

where the last inequality is from Theorem 7.8. By Lemma 4.1, we have

$$\begin{aligned} \underset{S\sim D^m}{E}[R(h^{Percpt}_S)] &= \underset{S'\sim D^{m+1}}{E}[\hat{R}_{S',LOO}(\text{Perceptron})] \\ &\leq \underset{S'\sim D^{m+1}}{E}\left[\frac{r^2_{S'}/\rho^2_{S'}}{m+1}\right]. \end{aligned}$$

□

A Margin-Based L_2-Upper Bound on the Number of Mistakes Made by the Perceptron Algorithm – Non-Linearly Separable Case

Theorem 7.11: Let

- $\mathbf{x}_1, \ldots, \mathbf{x}_T$: a sequence of T points in $\mathbb{R}^N$ with labels $c(\mathbf{x}_1), \ldots, c(\mathbf{x}_T)$, not necessarily linearly separable;
- I : the subset of T rounds at which the perceptron algorithm makes an update;
- $M = |I|$: the number of updates the perceptron algorithm makes during T rounds.

Then we have

$$M \quad \leq \quad \inf_{\rho>0, \|\mathbf{u}\|_2 \leq 1} \left(\frac{\|\mathbf{L}_\rho(\mathbf{u})\|_2}{2} + \sqrt{\frac{\|\mathbf{L}_\rho(\mathbf{u})\|_2^2}{4} + \frac{\sqrt{\sum_{t \in I} \|\mathbf{x}_t\|_2^2}}{\rho}} \right)^2,$$

where

$$\mathbf{L}_\rho(\mathbf{u}) \triangleq \left(\left(1 - \frac{c(\mathbf{x}_1)(\mathbf{u} \cdot \mathbf{x}_1)}{\rho}\right)_+, \ldots, \left(1 - \frac{c(\mathbf{x}_T)(\mathbf{u} \cdot \mathbf{x}_T)}{\rho}\right)_+ \right).$$

Further assume

- $\|\mathbf{x}_t\|_2 \leq r$ for all $t \in [1, T]$, for some $r > 0$.

Then we have

$$M \leq \inf_{\rho>0, \|\mathbf{u}\|_2 \leq 1} \left(\frac{r}{\rho} + \sqrt{\|\mathbf{L}_\rho(\mathbf{u})\|_2} \right)^2.$$

Proof. Fix a $\rho > 0$ and a vector $\mathbf{u} \in \mathbb{R}^N$ with $\|\mathbf{u}\|_2 \leq 1$. We first reduce the problem to the linearly separable case by mapping each input vector $\mathbf{x}_t \in \mathbb{R}^N$ to a vector $\mathbf{x}'_t \in \mathbb{R}^{N+T}$ as follows:

$$\mathbf{x}_t \mapsto \mathbf{x}'_t = (\mathbf{x}_t^T, \Delta\mathbf{e}_t^T)^T,$$

where $\mathbf{e}_t$ is the standard unit vector in $\mathbb{R}^T$ with the nonzero component 1 in the t-th coordinate and $\Delta > 0$ is a parameter to be determined. Let the deviation of the t-th point $\mathbf{x}_t, t \in [1, T]$, be

$$d_t \triangleq \left(1 - \frac{c(\mathbf{x}_t)(\mathbf{u} \cdot \mathbf{x}_t)}{\rho}\right)_+ .$$

The vector $\mathbf{u} \in \mathbb{R}^N$ with $\|\mathbf{u}\|_2 \leq 1$ is mapped to a vector $\mathbf{u}' \in \mathbb{R}^{N+T}$ with $\|\mathbf{u}'\|_2 = 1$ as follows:

$$\mathbf{u} = (u_1, \ldots, u_N)^T \mapsto \mathbf{u}' = \left(\frac{u_1}{Z}, \ldots, \frac{u_N}{Z}, \frac{\rho c(\mathbf{x}_1) d_1}{Z\Delta}, \ldots, \frac{\rho c(\mathbf{x}_T) d_T}{Z\Delta}\right)^T .$$

Since we require

$$1 = \|\mathbf{u}'\|_2^2 = \frac{\|\mathbf{u}\|_2^2}{Z^2} + \frac{\rho^2}{\Delta^2 Z^2} \sum_{t=1}^{T} d_t^2,$$

we will set

$$Z = \sqrt{\|\mathbf{u}\|_2^2 + \frac{\rho^2}{\Delta^2} \sum_{t=1}^{T} d_t^2} = \sqrt{\|\mathbf{u}\|_2^2 + \frac{\rho^2}{\Delta^2} \|\mathbf{L}_\rho(\mathbf{u})\|_2^2}.$$

Let $\mathbf{w}'_t, t \in [1, T]$, be the weight vectors when the perceptron algorithm processes the sequence $\{\mathbf{x}'_t, t \in [1, T]\}$ of points in $\mathbb{R}^{N+T}$. Since the weight vector $\mathbf{w}'_t$ has zero component in the $(N+s)$-th coordinate for all $t \le s \le T$ and the first N coordinates of $\mathbf{w}'_t$ are the same as those of $\mathbf{w}_t$, we have

$$\mathbf{w}'_t \cdot \mathbf{x}'_t = \mathbf{w}_t \cdot \mathbf{x}_t$$

which implies that the predictions made by the Perceptron algorithm for $\mathbf{x}'_t, t \in [1, T]$, coincide with those made in the original space for $\mathbf{x}_t, t \in [1, T]$. Furthermore, by the definition of $\mathbf{u}'$ and $\mathbf{x}'_t$, we can write for any $t \in [1, T]$:

$$\begin{aligned} c(\mathbf{x}_t)(\mathbf{u}' \cdot \mathbf{x}'_t) &= c(\mathbf{x}_t)\left(\frac{\mathbf{u} \cdot \mathbf{x}_t}{Z} + \Delta \frac{\rho c(\mathbf{x}_t) d_t}{Z\Delta}\right) \\ &\geq \frac{c(\mathbf{x}_t)(\mathbf{u} \cdot \mathbf{x}_t)}{Z} + \frac{\rho - c(\mathbf{x}_t)(\mathbf{u} \cdot \mathbf{x}_t)}{Z} = \frac{\rho}{Z}, \end{aligned}$$

which shows that the sequence $\{\mathbf{x}'_t, t \in [1, T]\}$ of points in $\mathbb{R}^{N+T}$ are linearly separable with a margin at least $\frac{\rho}{Z}$. Summing up for all $t \in I$ and following the proof of Theorem 7.8, we have

$$\frac{\rho}{Z} M \leq \sum_{t \in I} c(\mathbf{x}_t)(\mathbf{u}' \cdot \mathbf{x}'_t) \leq \sqrt{\sum_{t \in I} \|\mathbf{x}'_t\|_2^2}$$

which implies that

$$\begin{aligned} M^2 &\leq \left(\frac{\|\mathbf{u}\|_2^2 + \frac{\rho^2}{\Delta^2}\|\mathbf{L}_\rho(\mathbf{u})\|_2^2}{\rho^2} \right) \sum_{t\in I} \|\mathbf{x}'_t\|_2^2 \\ &\leq \left(\frac{1}{\rho^2} + \frac{\|\mathbf{L}_\rho(\mathbf{u})\|_2^2}{\Delta^2} \right) \left(M\Delta^2 + \sum_{t\in I} \|\mathbf{x}_t\|_2^2 \right) \\ &= \frac{\sum_{t\in I}\|\mathbf{x}_t\|_2^2}{\rho^2} + \frac{\|\mathbf{L}_\rho(\mathbf{u})\|_2^2 \sum_{t\in I}\|\mathbf{x}_t\|_2^2}{\Delta^2} + \frac{M\Delta^2}{\rho^2} + M\|\mathbf{L}_\rho(\mathbf{u})\|_2^2 \end{aligned}$$

since $\|\mathbf{u}\|_2 \leq 1$. Selecting

$$\Delta^2 = \frac{\rho\|\mathbf{L}_\rho(\mathbf{u})\|_2 \sqrt{\sum_{t\in I}\|\mathbf{x}_t\|_2^2}}{\sqrt{M}}$$

to minimize the upper bound, we have

$$\begin{aligned} M^2 &\leq \frac{\sum_{t\in I}\|\mathbf{x}_t\|_2^2}{\rho^2} + 2\frac{\sqrt{M}\|\mathbf{L}_\rho(\mathbf{u})\|_2\sqrt{\sum_{t\in I}\|\mathbf{x}_t\|_2^2}}{\rho} + M\|\mathbf{L}_\rho(\mathbf{u})\|_2^2 \\ &= \left(\frac{\sqrt{\sum_{t\in I}\|\mathbf{x}_t\|_2^2}}{\rho} + \sqrt{M}\|\mathbf{L}_\rho(\mathbf{u})\|_2\right)^2, \end{aligned}$$

which implies that

$$
\begin{aligned}
& M \leq \frac{\sqrt{\sum_{t\in I}\|\mathbf{x}_t\|_2^2}}{\rho} + \sqrt{M}\|\mathbf{L}_\rho(\mathbf{u})\|_2 \\
\Rightarrow\quad & \left(\sqrt{M} - \frac{1}{2}\|\mathbf{L}_\rho(\mathbf{u})\|_2\right)^2 \leq \frac{\|\mathbf{L}_\rho(\mathbf{u})\|_2^2}{4} + \frac{\sqrt{\sum_{t\in I}\|\mathbf{x}_t\|_2^2}}{\rho} \\
\Rightarrow\quad & \sqrt{M} \leq \frac{1}{2}\|\mathbf{L}_\rho(\mathbf{u})\|_2 + \sqrt{\frac{\|\mathbf{L}_\rho(\mathbf{u})\|_2^2}{4} + \frac{\sqrt{\sum_{t\in I}\|\mathbf{x}_t\|_2^2}}{\rho}} \\
\Rightarrow\quad & M \leq \left(\frac{1}{2}\|\mathbf{L}_\rho(\mathbf{u})\|_2 + \sqrt{\frac{\|\mathbf{L}_\rho(\mathbf{u})\|_2^2}{4} + \frac{\sqrt{\sum_{t\in I}\|\mathbf{x}_t\|_2^2}}{\rho}}\right)^2 .
\end{aligned}
$$

We now yield the first inequality. When $\|\mathbf{x}_t\|_2 \leq r$ for all $t \in [1, T]$, for some $r > 0$, we have

$$\sqrt{\sum_{t \in I} \|\mathbf{x}_t\|_2^2} \leq \sqrt{M} r$$

and then

$$\begin{aligned} & M \leq \frac{\sqrt{M} r}{\rho} + \sqrt{M} \|\mathbf{L}_\rho(\mathbf{u})\|_2 \\ \Rightarrow \quad & M \leq \sqrt{M} \left(\frac{r}{\rho} + \|\mathbf{L}_\rho(\mathbf{u})\|_2 \right) \\ \Rightarrow \quad & \sqrt{M} \leq \left(\frac{r}{\rho} + \|\mathbf{L}_\rho(\mathbf{u})\|_2 \right) \\ \Rightarrow \quad & M \leq \left(\frac{r}{\rho} + \|\mathbf{L}_\rho(\mathbf{u})\|_2 \right)^2, \end{aligned}$$

which yields the 2nd inequality. $\square$

Remarks

- The main idea behind the proof of the above theorem is to map input points which may not be linearly separable to a higher-dimensional space where linear separation is possible, which coincides with the idea of kernel methods.
- The particular kernel used in the proof is close to a straightforward one with a feature mapping that maps each data point to a distinct dimension.

A Margin-Based L_1-Upper Bound on the Number of Mistakes Made by the Perceptron Algorithm – Non-Linearly Separable Case

Theorem 7.11A:[a] Let

- $\mathbf{x}_1, \ldots, \mathbf{x}_T$: a sequence of T points in $\mathbb{R}^N$ with labels $c(\mathbf{x}_1), \ldots, c(\mathbf{x}_T)$, not necessarily linearly separable;
- I : the subset of T rounds at which the perceptron algorithm makes an update;
- $M = |I|$: the number of updates the perceptron algorithm makes during T rounds.

[a] M. Mohri and A. Rostamizadeh, "Perceptron mistake bounds," arXiv:1305.0208, 2013.

Then we have

$$M \leq \inf_{\rho>0, \|\mathbf{u}\|_2 \leq 1} \left(\sum_{t \in I} \left(1 - \frac{c(\mathbf{x}_t)(\mathbf{u} \cdot \mathbf{x}_t)}{\rho}\right)_+ + \frac{\sqrt{\sum_{t \in I} \|\mathbf{x}_t\|_2^2}}{\rho} \right).$$

Further assume

- $\|\mathbf{x}_t\|_2 \leq r$ for all $t \in [1, T]$, for some $r > 0$.

Then we have

$$M \leq \inf_{\rho>0, \|\mathbf{u}\|_2 \leq 1} \left(\frac{r}{\rho} + \sqrt{\|\mathbf{L}_\rho(\mathbf{u})\|_1} \right)^2.$$

where

$$\mathbf{L}_\rho(\mathbf{u}) \triangleq \left(\left(1 - \frac{c(\mathbf{x}_1)(\mathbf{u} \cdot \mathbf{x}_1)}{\rho}\right)_+, \ldots, \left(1 - \frac{c(\mathbf{x}_T)(\mathbf{u} \cdot \mathbf{x}_T)}{\rho}\right)_+ \right).$$

Proof. Fix a $\rho > 0$ and a vector $\mathbf{u} \in \mathbb{R}^N$ with $\|\mathbf{u}\|_2 \leq 1$. For any $t \in [1, T]$,

$$1 - \frac{c(\mathbf{x}_t)(\mathbf{u} \cdot \mathbf{x}_t)}{\rho} \leq \left(1 - \frac{c(\mathbf{x}_t)(\mathbf{u} \cdot \mathbf{x}_t)}{\rho}\right)_+$$

and summing up for all $t \in I$, we have

$$M \leq \sum_{t \in I} \left(1 - \frac{c(\mathbf{x}_t)(\mathbf{u} \cdot \mathbf{x}_t)}{\rho}\right)_+ + \sum_{t \in I} \frac{c(\mathbf{x}_t)(\mathbf{u} \cdot \mathbf{x}_t)}{\rho}.$$

But

$$\begin{aligned}
\sum_{t \in I} c(\mathbf{x}_t)(\mathbf{u} \cdot \mathbf{x}_t) &= \mathbf{u} \cdot \sum_{t \in I} c(\mathbf{x}_t)\mathbf{x}_t \\
&\leq \quad \|\mathbf{u}\|_2 \left\| \sum_{t \in I} c(\mathbf{x}_t)\mathbf{x}_t \right\|_2 \quad \text{by Cauchy-Schwartz inequality} \\
&\leq \quad \frac{1}{\eta}\|\mathbf{w}_{T+1}\|_2
\end{aligned}$$

But

$$\begin{aligned}
\|\mathbf{w}_{T+1}\|_2 &= \sqrt{\sum_{t\in I}(\|\mathbf{w}_{t+1}\|_2^2 - \|\mathbf{w}_t\|_2^2)} \text{ note that } \mathbf{w}_0 = \mathbf{0} \\
&= \sqrt{\sum_{t\in I}(\|\mathbf{w}_t + \eta c(\mathbf{x}_t)\mathbf{x}_t\|_2^2 - \|\mathbf{w}_t\|_2^2)} \\
&= \sqrt{\sum_{t\in I}(2\eta c(\mathbf{x}_t)\mathbf{w}_t \cdot \mathbf{x}_t + \eta^2\|\mathbf{x}_t\|_2^2)} \\
&\leq \eta\sqrt{\sum_{t\in I}\|\mathbf{x}_t\|_2^2}
\end{aligned}$$

since $\eta c(\mathbf{x}_t)\mathbf{w}_t \cdot \mathbf{x}_t \leq 0$ for each $t \in I$. This shows the first inequality. When $\|\mathbf{x}_t\|_2 \leq r$ for all $t \in [1, T]$, for some $r > 0$, we have

$$\sqrt{\sum_{t\in I}\|\mathbf{x}_t\|_2^2} \leq \sqrt{M}r$$

so that

$$\begin{aligned} M &\le \sum_{t\in I}\left(1-\frac{c(\mathbf{x}_t)(\mathbf{u}\cdot\mathbf{x}_t)}{\rho}\right)_+ + \sqrt{M}\frac{r}{\rho} \\ &\le \sum_{t=1}^{T}\left(1-\frac{c(\mathbf{x}_t)(\mathbf{u}\cdot\mathbf{x}_t)}{\rho}\right)_+ + \sqrt{M}\frac{r}{\rho} \\ &= \|\mathbf{L}_\rho(\mathbf{u})\|_1 + \sqrt{M}\frac{r}{\rho}. \end{aligned}$$

Thus we have

$$\begin{aligned} \left(\sqrt{M}-\frac{1}{2}\frac{r}{\rho}\right)^2 &\le \frac{(r/\rho)^2+4\|\mathbf{L}_\rho(\mathbf{u})\|_1}{4} \\ \Rightarrow \quad \sqrt{M} &\le \frac{(r/\rho)+\sqrt{(r/\rho)^2+(2\sqrt{\|\mathbf{L}_\rho(\mathbf{u})\|_1})^2}}{2} \le \frac{r}{\rho}+\sqrt{\|\mathbf{L}_\rho(\mathbf{u})\|_1} \end{aligned}$$

since $\sqrt{x^2+y^2} \le x+y$ for all $x, y \ge 0$, which yields the 2nd inequality. $\square$

Remarks

- The Perceptron algorithm can be generalized, as in the case of SVMs, to define a linear separation in a high-dimensional space.
- First we present an equivalent dual form of the Perceptron algorithm, the dual Perceptron algorithm.
- The dual Perceptron algorithm maintains a vector $\alpha \in \mathbb{R}^T$ of coefficients assigned to each point $\mathbf{x}_t, t \in [1, T]$.
- The label of a point $\mathbf{x}_t$ is predicted according to the rule $\operatorname{sgn}(\mathbf{w} \cdot \mathbf{x}_t)$, where

$$\mathbf{w} = \sum_{s=1}^{T} \alpha_s c(\mathbf{x}_s)\mathbf{x}_s.$$

- The coefficient α_t is incremented by $\eta > 0$ when this prediction does not match the correct label. Thus, an update for $\mathbf{x}_t$ is equivalent to augmenting the weight vector $\mathbf{w}$ with $\eta c(\mathbf{x}_s)\mathbf{x}_s$, which shows that the dual algorithm matches exactly the standard Perceptron algorithm.
- The dual Perceptron algorithm can be written solely in terms of inner products between training instances. Thus, as in the case of SVMs, instead of the inner product between points in the input space, an arbitrary PDS kernel can be used, which leads to the kernel Perceptron algorithm.
- The kernel Perceptron algorithm and its voted version are commonly used algorithms in a variety of applications.

The Dual Perceptron Algorithm

```
DualPerceptron($\alpha_0$)
1. $\alpha \leftarrow \alpha_0$        $\triangleright$ typically $\alpha_0 = \mathbf{0} \in \mathbb{R}^T$
2. for $t \leftarrow 1$ to $T$ do
3.     Receive($\mathbf{x}_t$)
4.     $\hat{y}_t \leftarrow \mathrm{sgn}\left(\sum_{s=1}^{T} \alpha_s c(\mathbf{x}_s)(\mathbf{x}_s \cdot \mathbf{x}_t)\right)$
5.     Receive($c(\mathbf{x}_t)$)
6.     if ($\hat{y}_t \neq c(\mathbf{x}_t)$) then
7.         $\alpha_t \leftarrow \alpha_t + \eta$
8. return $\alpha$
```

The Kernel Perceptron Algorithm for PDS Kernel K

```
KernelPerceptron($\alpha_0$)
1. $\alpha \leftarrow \alpha_0$        ▷ typically $\alpha_0 = \mathbf{0} \in \mathbb{R}^T$
2. for $t \leftarrow 1$ to $T$ do
3.     Receive($\mathbf{x}_t$)
4.     $\hat{y}_t \leftarrow \operatorname{sgn}\left(\sum_{s=1}^{T} \alpha_s c(\mathbf{x}_s) K(\mathbf{x}_s, \mathbf{x}_t)\right)$
5.     Receive($c(\mathbf{x}_t)$)
6.     if ($\hat{y}_t \neq c(\mathbf{x}_t)$) then
7.         $\alpha_t \leftarrow \alpha_t + \eta$
8. return $\alpha$
```

The Winnow Algorithm

- The Winnow algorithm learns a weight vector defining a separating hyperplane (through the origin) by sequentially processing the training points.
- Unlike the additive update of the weight vector in the Perceptron algorithm, Winnow's update is multiplicative.
- $\eta > 0$: an input learning parameter of the Winnow algorithm
- The Winnow algorithm maintains a non-negative weight vector $\mathbf{w}_t$ with components summing to one, i.e., $\sum_{i=1}^{N} w_{t,i} = 1$ starting with the uniform weight vector.
  - N is the number of attributes of an item.
  - In on-line learning, N is interpreted as the number of experts and $\mathbf{x}_t$ is the feature vector of the item in the t-th round with $x_{t,i}$ the advice of the i-th expert.

- At each round $t \in [1, T]$, if the prediction does not match the correct label, each component $w_{t,i}$, $i \in [1, N]$, is updated by multiplying it by $\exp(\eta c(\mathbf{x}_t) x_{t,i})$ and dividing by the normalization factor $Z_t = \sum_{i=1}^{N} w_{t,i} e^{\eta c(\mathbf{x}_t) x_{t,i}}$ to ensure that the weights sum to one.
  - If the label $c(\mathbf{x}_t)$ and the attribute $x_{t,i}$ of $\mathbf{x}_t$ share the same sign, then $w_{t,i}$ is increased, while, in the opposite case, it is significantly decreased.
- As suggested by the name, the algorithm is particularly well suited to cases where a relatively small number of attributes or experts can be used to define an accurate weight vector.
- Many of the other attributes or experts may then be irrelevant.

The Winnow Algorithm with $c(\mathbf{x}_t) \in \{-1, +1\}\ \forall\ t \in [1, T]$

```
Winnow($\eta$)
1. $\mathbf{w}_1 \leftarrow \mathbf{1}/N$
2. for $t \leftarrow 1$ to $T$ do
3.     Receive($\mathbf{x}_t$)
4.     $\hat{y}_t \leftarrow \operatorname{sgn}(\mathbf{w}_t \cdot \mathbf{x}_t)$
5.     Receive($c(\mathbf{x}_t)$)
6.     if $(\hat{y}_t \neq c(\mathbf{x}_t))$ then
7.         $Z_t \leftarrow \sum_{i=1}^{N} w_{t,i} e^{\eta c(\mathbf{x}_t) x_{t,i}}$
8.         for $i \leftarrow 1$ to $N$ do
9.             $w_{t+1,i} \leftarrow \frac{w_{t,i} e^{\eta c(\mathbf{x}_t) x_{t,i}}}{Z_t}$
```

```
10.      else
11.          $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t$
12.  return $\mathbf{w}_{T+1}$
```

Remarks

- The Winnow algorithm is closely related to the weighted majority (WM) algorithm :
  - When $x_{t,i} \in \{-1, +1\}$ can be regarded as the advice $\hat{y}_{t,i}$ of expert i, the rule

$$\mathrm{sgn}(\mathbf{w}_t \cdot \mathbf{x}_t) = \mathrm{sgn}\left(\sum_{t:\hat{y}_{t,i}=1} w_{t,i} - \sum_{t:\hat{y}_{t,i}=-1} w_{t,i} \right)$$

    coincides with the weighted majority vote.
  - Also multiplying the weight $w_{t,i}$ of correct (i.e., $c(\mathbf{x}_t) = x_{t,i}$) or incorrect (i.e., $c(\mathbf{x}_t) = -x_{t,i}$) experts by e^{η} or $e^{-\eta}$ is equivalent to multiplying the weight $w_{t,i}$ of incorrect ones by $\beta = e^{-2\eta} < 1$.

- The multiplicative update rule of Winnow is of course similar to that of AdaBoost.

A Margin-Based Upper Bound on the Number of Mistakes Made by the Winnow Algorithm – Linearly Separable Case

Theorem 7.12: Let

- $\mathbf{x}_1, \ldots, \mathbf{x}_T$: a sequence of T points in $\mathbb{R}^N$ with $\|\mathbf{x}_t\|_\infty \triangleq \max_{i \in [1,N]}\{|x_{t,i}|\} \leq r_\infty$ for all $t \in [1,T]$, for some $r_\infty > 0$.

Assume that there exist $\rho_\infty > 0$ and $\mathbf{v} \in \mathbb{R}^N \setminus \{\mathbf{0}\}$ with $\mathbf{v} \geq \mathbf{0}$ such that for all $t \in [1,T]$,

$$\rho_\infty \leq \frac{c(\mathbf{x}_t)(\mathbf{v} \cdot \mathbf{x}_t)}{\|\mathbf{v}\|_1}.$$

- The T points $\mathbf{x}_t$ are linearly separable with a geometric margin at least $\rho_\infty \frac{\|\mathbf{v}\|_1}{\|\mathbf{v}\|_2}$.

Then, the number of updates made by the Winnow algorithm when processing $\mathbf{x}_1, \ldots, \mathbf{x}_T$ is bounded by $2r_\infty^2/\rho_\infty^2 \ln N$.

Proof. Let

- I : the subset of T rounds at which the Winnow algorithm makes an update.
- $M = |I|$: the number of updates the Winnow algorithm makes during T rounds.
- $\Phi_t \triangleq \sum_{i=1}^{N} \frac{v_i}{\|\mathbf{v}\|_1} \ln \frac{v_i/\|\mathbf{v}\|_1}{w_{t,i}}$: the potential function at round t, which is the relative entropy of the distribution $\{v_i/\|\mathbf{v}\|_1\}_{i=1}^{N}$ to the distribution $\{w_{t,i}\}_{i=1}^{N}$.

To derive an upper bound on Φ_t, we analyze the difference of the potential functions at two consecutive rounds. For all $t \in I$, this difference can be expressed and bounded as follows:

$$
\begin{aligned}
& \Phi_{t+1} - \Phi_t \\
= \quad & \sum_{i=1}^{N} \frac{v_i}{\|\mathbf{v}\|_1} \ln \frac{w_{t,i}}{w_{t+1,i}} = \sum_{i=1}^{N} \frac{v_i}{\|\mathbf{v}\|_1} \ln \frac{Z_t}{\exp(\eta c(\mathbf{x}_t) x_{t,i})} \\
= \quad & \ln Z_t - \eta \sum_{i=1}^{N} \frac{v_i c(\mathbf{x}_t) x_{t,i}}{\|\mathbf{v}\|_1} = \ln Z_t - \eta \frac{c(\mathbf{x}_t)(\mathbf{v} \cdot \mathbf{x}_t)}{\|\mathbf{v}\|_1} \\
\leq \quad & \ln \sum_{i=1}^{N} w_{t,i} e^{\eta c(\mathbf{x}_t) x_{t,i}} - \eta \rho_\infty \\
= \quad & \ln \underset{\mathbf{w}_t}{E}[\exp(\eta c(\mathbf{x}_t) X_t)] - \eta \rho_\infty \\
= \quad & \ln \underset{\mathbf{w}_t}{E}[\exp(\eta c(\mathbf{x}_t)(X_t - \underset{\mathbf{w}_t}{E}[X_t]))] + \eta c(\mathbf{x}_t) \underset{\mathbf{w}_t}{E}[X_t] - \eta \rho_\infty \\
= \quad & \ln \underset{\mathbf{w}_t}{E}[\exp(\eta c(\mathbf{x}_t)(X_t - \underset{\mathbf{w}_t}{E}[X_t]))] + \eta c(\mathbf{x}_t)(\mathbf{w}_t \cdot \mathbf{x}_t) - \eta \rho_\infty \\
\leq \quad & \ln \underset{\mathbf{w}_t}{E}[\exp(\eta c(\mathbf{x}_t)(X_t - \underset{\mathbf{w}_t}{E}[X_t]))] - \eta \rho_\infty .
\end{aligned}
$$

since $\eta c(\mathbf{x}_t)(\mathbf{w}_t \cdot \mathbf{x}_t) \le 0$ for each $t \in I$. By Hoeffding's Lemma, we have

$$\mathop{E}_{\mathbf{w}_t}[\exp(\eta c(\mathbf{x}_t)(X_t - \mathop{E}_{\mathbf{w}_t}[X_t]))] \le e^{\eta^2 (2r_\infty)^2/8}$$

since $-r_\infty \le X_t \le r_\infty$ for all $i \in [1, N]$. Thus, for all $t \in I$,

$$\Phi_{t+1} - \Phi_t \le \eta^2 r_\infty^2/2 - \eta\rho_\infty.$$

Summing up for all $t \in I$, we have

$$\Phi_{T+1} - \Phi_1 \le M(\eta^2 r_\infty^2/2 - \eta\rho_\infty).$$

Next we derive a lower bound by noting that

$$\Phi_1 = \sum_{i=1}^{N} \frac{v_i}{\|\mathbf{v}\|_1} \ln \frac{v_i/\|\mathbf{v}\|_1}{1/N} = \ln N + \sum_{i=1}^{N} \frac{v_i}{\|\mathbf{v}\|_1} \ln \frac{v_i}{\|\mathbf{v}\|_1} \le \ln N.$$

Additionally, since the relative entropy is always non-negative, we have $\Phi_{T+1} \ge 0$ and then

$$\Phi_{T+1} - \Phi_1 \ge 0 - \ln N = -\ln N.$$

Combining the upper and lower bounds, we have

$$-\ln N \leq M(\eta^2 r_\infty^2/2 - \eta\rho_\infty) \text{ and then } M \leq \frac{\ln N}{\eta\rho_\infty - \eta^2 r_\infty^2/2}$$

By selecting

$$\eta = \frac{\rho_\infty}{r_\infty^2}$$

to maximize $\eta\rho_\infty - \eta^2 r_\infty^2/2$, we have

$$M \leq 2(r_\infty^2/\rho_\infty^2)\ln N.$$

$\square$

Remarks

- The bound for Winnow algorithm is favorable when a sparse set of the experts $i \in [1, N]$ can predict well.
  - For example, if $\mathbf{v} = \mathbf{e}_1$ where $\mathbf{e}_1$ is the unit vector along the first axis in $\mathbb{R}^N$ and if $\mathbf{x}_t \in \{-1, +1\}^N$ for all $t \in [1, T]$ such that
    $$c(\mathbf{x}_t) = x_{t,1} \ \forall \ t \in [1, T],$$
    which means that expert 1 is perfectly accurate, so that
    $$\frac{c(\mathbf{x}_t)(\mathbf{v} \cdot \mathbf{x}_t)}{\|\mathbf{v}\|_1} = c(\mathbf{x}_t) x_{t,1} = 1 \ \forall \ t \in [1, T],$$

  then $\rho_\infty = 1$ and $r_\infty = 1$ and then the upper bound on the number of mistakes given for Winnow algorithm by Theorem 7.12 is only
  $$2 \ln N,$$

while the upper bound of Theorem 7.8 for the Perceptron algorithm is

$$\frac{r_2^2}{\rho_2^2} = N$$

since

$$\rho_2 = \frac{c(\mathbf{x}_t)(\mathbf{v} \cdot \mathbf{x}_t)}{\|\mathbf{v}\|_2} = 1 \text{ and } r_2 = \|\mathbf{x}_t\| = \sqrt{N}.$$

- The guarantee for the Perceptron algorithm is more favorable in the opposite situation, where sparse solutions are not effective.

The Contents of This Lecture

- Prediction with expert advice.
- Linear classification.
- On-line to batch conversion.

Description of an On-Line Algorithm $\mathbb{A}$

- $\mathcal{H}$: the hypothesis set of the one-line algorithm $\mathbb{A}$, which is a set of functions from the input space $\mathscr{I}$ to the output space $\mathscr{Y}'$.
- $L : \mathscr{Y}' \times \mathscr{Y} \to \mathbb{R}^+$: a bounded loss function, i.e., $L(y', y) \leq M$ for all $y' \in \mathscr{Y}'$, $y \in \mathscr{Y}$ for some $M \geq 1$.
- $S = ((\omega_1, c(\omega_1)), \ldots, (\omega_T, c(\omega_T))) \in (\mathscr{I} \times \mathscr{Y})^T$: a labeled sample of size T drawn i.i.d. according to some fixed but unknown distribution D.
  - The sample S is sequentially processed by the one-line algorithm $\mathbb{A}$.
- $h_1 \in \mathcal{H}$: the initial hypothesis for the one-line algorithm $\mathbb{A}$.
- $h_{t+1} \in \mathcal{H}, t \in [1, T]$: a new hypothesis generated by the one-line algorithm $\mathbb{A}$ after processing the pair $(\omega_t, c(\omega_t))$ at round t.

- $R_T = \sum_{t=1}^{T} L(h_t(\omega_t), c(\omega_t)) - \min_{h \in \mathcal{H}} \sum_{t=1}^{T} L(h(\omega_t), c(\omega_t))$: the regret of the algorithm $\mathbb{A}$.
- $R(h) = \underset{\omega \sim D}{E}[L(h(\omega), c(\omega))]$: the generalization error of the hypothesis h.

Azuma's Inequality

Let $V_1, V_2, \ldots$ be a martingale difference sequence with respect to the r.v.'s $X_1, X_2, \ldots$, i.e., for all $t \geq 1$, V_t is a function of $X_1, \ldots, X_t$ and

$$E[V_{t+1} | X_1, X_2, \ldots, X_t] = 0.$$

Assume that for all $t \geq 1$, there are a constant c_t and a r.v. Z_t, which is a function of $X_1, \ldots, X_{t-1}$, such that

$$Z_t \leq V_t \leq Z_t + c_t.$$

Then for all $\epsilon > 0$ and $T \geq 1$, we have

$$\begin{aligned} P\left(\sum_{t=1}^{T} V_t \geq \epsilon\right) &\leq \exp\left(\frac{-2\epsilon^2}{\sum_{t=1}^{T} c_t^2}\right), \\ P\left(\sum_{t=1}^{T} V_t \leq -\epsilon\right) &\leq \exp\left(\frac{-2\epsilon^2}{\sum_{t=1}^{T} c_t^2}\right). \end{aligned}$$

A Bound of the Average of the Generalization Errors of the Hypotheses Generated by an On-Line Algorithm $\mathbb{A}$

Lemma 7.1: For any $\delta > 0$, with probability at least $1 - \delta$, the following holds:

$$\frac{1}{T}\sum_{t=1}^{T} R(h_t) \leq \frac{1}{T}\sum_{t=1}^{T} L(h_t(\omega_t), c(\omega_t)) + M\sqrt{\frac{2\ln\frac{1}{\delta}}{T}}.$$

Proof. For each $t \in [1, T]$, let V_t be the random variable defined by

$$V_t = R(h_t) - L(h_t(\omega_t), c(\omega_t)).$$

Observe that for any $t \in [1, T]$,

$$\begin{aligned} E[V_t \mid \omega_1, \ldots, \omega_{t-1}] &= R(h_t) - E[L(h_t(\omega_t), c(\omega_t)) \mid h_t] \\ &= R(h_t) - R(h_t) = 0. \end{aligned}$$

Since the loss is bounded by $M \geq 1$, V_t takes values in the interval

$[-M, +M]$ for all $t \in [1, T]$. Thus, by Azuma's inequality, we have

$$P\left(\frac{1}{T}\sum_{t=1}^{T} V_t \geq \epsilon\right) \leq \exp\left(\frac{-2T\epsilon^2}{(2M)^2}\right).$$

By letting the right-hand side to be equal to δ, we prove the lemma. $\square$

A Bound on the Generalization Error of the Average of the Hypotheses Generated by an On-Line Algorithm $\mathbb{A}$

Theorem 7.13: Further assume that the loss function $L(y', y)$ is convex with respective to its first argument. For any $\delta > 0$, with probability at least $1 - \delta$, each of the following holds:

$$R\left(\frac{1}{T}\sum_{t=1}^{T} h_t\right) \leq \frac{1}{T}\sum_{t=1}^{T} L(h_t(\omega_t), c(\omega_t)) + M\sqrt{\frac{2\ln\frac{1}{\delta}}{T}}$$

$$R\left(\frac{1}{T}\sum_{t=1}^{T} h_t\right) \leq \inf_{h\in\mathcal{H}} R(h) + \frac{R_T}{T} + 2M\sqrt{\frac{2\ln\frac{2}{\delta}}{T}}.$$

Proof. By the convexity of $L(y', y)$ with respective to its first argument, we have

$$L\left(\frac{1}{T}\sum_{t=1}^{T}h_t(\omega), c(\omega)\right) \leq \frac{1}{T}\sum_{t=1}^{T}L(h_t h_t(\omega), c(\omega)).$$

Taking the expectation w.r.t. $\omega \sim D$, we have

$$R\left(\frac{1}{T}\sum_{t=1}^{T}h_t\right) \leq \frac{1}{T}\sum_{t=1}^{T}R(h_t).$$

The first inequality follows from Lemma 7.1. Now by the definition of the regret R_T, for any $\delta > 0$, the following holds with probability at least $1-\delta/2$:

$$\begin{aligned} R\left(\frac{1}{T}\sum_{t=1}^{T}h_t\right) &\leq \frac{1}{T}\sum_{t=1}^{T}L(h_t(\omega_t), c(\omega_t)) + M\sqrt{\frac{2\ln\frac{2}{\delta}}{T}} \\ &\leq \min_{h\in\mathcal{H}}\frac{1}{T}\sum_{t=1}^{T}L(h(\omega_t), c(\omega_t)) + \frac{R_T}{T} + M\sqrt{\frac{2\ln\frac{2}{\delta}}{T}}. \end{aligned}$$

By the definition of $\inf_{h \in \mathcal{H}} R(h)$, for any $\epsilon > 0$, there exists an $h^* \in \mathcal{H}$ with

$$R(h^*) \leq \inf_{h \in \mathcal{H}} R(h) + \epsilon.$$

By Hoeffding's inequality, for any $\delta > 0$, with probability at least $1 - \delta/2$,

$$\frac{1}{T} \sum_{t=1}^{T} L(h^*(\omega_t), c(\omega_t)) \leq R(h^*) + M \sqrt{\frac{2 \ln \frac{2}{\delta}}{T}}$$

Thus for any $\epsilon > 0$, by the union bound, the following holds with probability at least $1 - \delta$:

$$
\begin{aligned}
R\left(\frac{1}{T}\sum_{t=1}^{T} h_t\right) &\leq \frac{1}{T}\sum_{t=1}^{T} L(h^*(\omega_t), c(\omega_t)) + \frac{R_T}{T} + M\sqrt{\frac{2\ln\frac{2}{\delta}}{T}} \\
&\leq R(h^*) + M\sqrt{\frac{2\ln\frac{2}{\delta}}{T}} + \frac{R_T}{T} + M\sqrt{\frac{2\ln\frac{2}{\delta}}{T}} \\
&= R(h^*) + \frac{R_T}{T} + 2M\sqrt{\frac{2\ln\frac{2}{\delta}}{T}} \\
&\leq \inf_{h\in\mathcal{H}} R(h) + \epsilon + \frac{R_T}{T} + 2M\sqrt{\frac{2\ln\frac{2}{\delta}}{T}}
\end{aligned}
$$

for all $\epsilon > 0$. By letting $\epsilon \to 0$, the 2nd inequality is proved. $\square$

Remarks

- Theorem 7.13 can be applied to a variety of on-line regret minimization algorithms, for example when $R_T/T = O(1/\sqrt{T})$.
- In particular, we can apply the theorem to the exponential weighted average algorithm. Assuming that the loss function L is bounded by $M = 1$ and that the number of rounds T is known to the algorithm, we can use the regret bound of Theorem 7.6. The doubling trick (used in Theorem 7.7) can be used to derive a similar bound if T is not known in advance. Thus, for any $\delta > 0$, with probability at least $1 - \delta$, the following holds for the generalization error of the average of the hypotheses generated by exponential weighted average

algorithm:

$$R\left(\frac{1}{T}\sum_{t=1}^{T} h_t\right) \le \inf_{h\in\mathcal{H}} R(h) + \sqrt{\frac{\ln N}{T}} + 2M\sqrt{\frac{2\ln\frac{2}{\delta}}{T}}$$

where N is the number of experts, or the dimension of the weight vectors.