Machine learning theory

On line learning

Hamid Beigy

Sharif university of technology

May 6, 2023

Table of contents

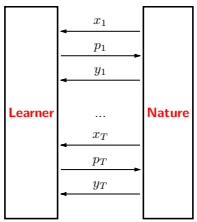
1. Introduction
2. Online classification in the realizable case
3. Online learnability
4. Online classification in the unrealizable case
5. Perceptron
6. Winnow algorithm
7. On-line to batch conversion
8. Summary

Introduction

1. We have analyzed some learning algorithms in the statistical setting,
   - We assume training and test data are both drawn i.i.d. from some distribution $\mathcal{D}$
   - Usually, we have two separate phases: training and test.
2. In this lecture,
   - We weaken the assumptions and assume that data can be generated completely adversarily.
   - We also move to the online setting where training and test are interleaved.
3. We make two shifts to the learning setup:
   - batch to online.
   - statistical to adversarial.

Considering the online learning framework for prediction.

1. We need to find a mapping $y = h(\mathbf{x})$, where $\mathbf{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$.
2. This setting can be thought of as a game between a learner and nature.
3. In each time stage $t = 1, 2, \ldots, T$,
   - Learner receives an input $\mathbf{x}_t \in \mathcal{X}$.
   - Learner outputs prediction $\hat{y}_t \in D$.
   - Learner receives true label $y_t \in \mathcal{Y}$.
   - Learner suffers loss $\ell(y_t, \hat{y}_t)$.
   - Learner updates model parameters.
4. Learning is **hopeless** if there is no correlation between past and present rounds.



Formally, learner is a function A that returns the current prediction given the full history

$$\hat{y}_{t+1} = A\left(\mathbf{x}_{1:t}, \hat{y}_{1:t}, y_{1:t}, \mathbf{x}_{t+1}\right)$$

Example (Online binary classification for spam filtering)

In online binary classification for spam filtering, we have

1. Inputs: $\mathcal{X} = \{0,1\}^n$ are boolean feature vectors (presence or absence of a word).
2. Outputs: $\mathcal{Y} = \{+1,-1\}$ whether a document is spam or not spam.
3. Loss: Zero-one loss $\ell(y_t, \hat{y}_t) = \mathbb{I}[y_t \neq \hat{y}_t]$ is whether the prediction was incorrect.

Remarks

1. The training phase and testing phase are **interleaved** in online learning.
2. The online learning setting leaves completely open the time and memory usage of the online algorithms.
3. In practice, online learning algorithms update parameters after each example, and hence tend to be faster than traditional batch optimization algorithms.
4. The real world is complex and constantly-changing, but online learning algorithms have the potential to **adapt**.
5. In some applications such as spam filtering, the inputs could be generated by an **adversary**, hence, we will make no assumptions about the input/output sequence.

1. How we measure the **quality** of an online learner A?
2. The learning algorithm is said to make a **mistake** in round t if $\hat{y}_t \neq y_t$.
3. The **goal** of the online learner is simply to make **few prediction mistakes**.
4. We encode prior knowledge on the problem using
   - Some representation of the instances and
   - Assuming that there is a class of hypotheses, $H = \{h : \mathcal{X} \mapsto \mathcal{Y}\}$, and on each online round the learner uses a hypothesis from H to make his prediction.
5. Sometimes the output of learner may be chosen from the set D, which is different from $\mathcal{Y}$.

Example (Online regression)

Inputs $\mathcal{X} = \mathbb{R}^d$ are set of measurements or features.

Outputs $D = \mathcal{Y} = \mathbb{R}$.

Loss $\ell(y_t, \hat{y}_t) = (y_t - \hat{y}_t)^2$ or $\ell(y_t, \hat{y}_t) = |y_t - \hat{y}_t|$.

Example (Prediction with expert advice)

In each round, the learner has to choose from advice of d experts. Then,

Inputs $\mathbf{x}_t \in \mathcal{X} \subset \mathbb{R}^d$, where $\mathbf{x}_t[i]$ is the advice from ith expert, i.e. $D = \{1, \ldots, d\}$.

Outputs $\mathbf{y}_t \in \mathcal{Y} = [0, 1]^d$ where $\mathbf{y}_t[i]$ is the cost of following ith expert.

Loss The loss is the cost of chosen expert, i.e. $\ell(y_t, \hat{y}_t) = y_t[\hat{y}_t]$.

Example (Online ranking)

On round t, the learner receives a query $\mathbf{x}_t \in \mathcal{X}$ and is required to order k elements according their relevance to the query.

Outputs D is the set of permutations of $\{1, 2, \ldots, k\}$.

Loss The loss $\ell(y_t, \hat{y}_t)$ is the prediction of y_t in the ranked list $\hat{y}_t$.

Online classification in the realizable case

1. Online learning is performed in a sequence of consecutive rounds, where at round t,
   1.1 Learner receives an input $\mathbf{x}_t \in \mathcal{X}$.
   1.2 Learner outputs prediction $\hat{y}_t \in \mathcal{Y}$.
   1.3 Learner receives true label $y_t \in \mathcal{Y}$.
2. In the realizable case, we assume that all labels are generated by some hypothesis, $h^* : \mathcal{X} \mapsto \mathcal{Y}$.
3. We also assume that h^* is taken from a hypothesis class H, which is known to the learner.
4. The learner should make as few mistakes as possible, assuming that both h^* and the sequence of instances can be chosen by an adversary.

Definition (Mistake bound)

For an online learning algorithm, A, we denote by $M_A(H)$ the maximal number of mistakes that the algorithm A might make on a sequence of examples which is labeled by some $h^* \in H$. A bound on $M_A(H)$ is called a **mistake-bound**.

We will study how to design algorithms for which $M_A(H)$ is minimal.

Definition (Mistake bounds, Online learnability)

Let H be a hypothesis class and let A be an online learning algorithm. Given any sequence $S = ((x_1, h^*(\mathbf{x}_1)), \ldots, (x_T, h^*(\mathbf{x}_T)))$, where T is any integer and $h^* \in H$, let $M_A(S)$ be the number of mistakes A makes on the sequence S. We denote by $M_A(H)$ the supremum of $M_A(S)$ over all sequences of the preceding form. A bound of the form $M_A(H) \leq B < \infty$ is called a **mistake bound**. We say that a hypothesis class H is **online learnable** if there exists an algorithm A for which $M_A(H) \leq B < \infty$.

Let $|H| < \infty$. A learning rule for online learning is to use any ERM hypothesis (any hypothesis which is consistent with all past examples).

Consistent algorithm

Let $V_1 = H$
for $t \leftarrow 1, 2, \ldots$ **do**
    Receive $\mathbf{x}_t$.
    Choose any $h \in V_t$ and predict $\hat{y}_t = h(\mathbf{x}_t)$.
    Receive true label $y_t = h^*(\mathbf{x}_t)$.
    Update $V_{t+1} = \{h \in V_t \mid h(\mathbf{x}_t) = y_t\}$.
end for

The Consistent algorithm maintains a set V_t, which is called **version space**.

Theorem (Mistakebound of Consistent algorithm)

Let H be a finite hypothesis class. Consistent algorithm has mistake bound $M_{Consistent}(H) \leq |H| - 1$.

Proof.

1. When Consistent makes a mistake, at least one hypothesis is removed from V_t .
2. Therefore, after making M mistakes we have $|V_t| \leq |H| - M$.
3. Since V_t is always nonempty (by the realizability assumption it contains h^*), we have $1 \leq |V_t| \leq |H| - M$.

□

We define a variant of Consistent which has much better mistake bound.

On each round, this algorithm choose a consistent hypothesis uniformly at random, as there is no reason to prefer one consistent hypothesis over another.

RandConsistent algorithm

Let $V_1 = H$
for $t \leftarrow 1, 2, \ldots$ **do**
    Receive $\mathbf{x}_t$.
    Choose some h from V_t uniformly at random.
    Predict $\hat{y}_t = h(\mathbf{x}_t)$.
    Receive true label $y_t = h^*(\mathbf{x}_t)$.
    Update $V_{t+1} = \{h \in V_t \mid h(\mathbf{x}_t) = y_t\}$.
end for

Theorem (Mistake bound of RandConsistent algorithm)

Let $|H| < \infty$, $h^* \in H$ and $S = ((\mathbf{x}_1, h^*(\mathbf{x}_1)), \ldots, (\mathbf{x}_T, h^*(\mathbf{x}_T)))$ be an arbitrary sequence of examples. Then, the expected number of mistakes the RandConsistent algorithm makes on this sequence is at most $\ln|H|$, where expectation is with respect to the algorithm's own randomization.

1. Consider round t and let α_t be the fraction of hypotheses in V_t, which are going to be correct on example $(\mathbf{x}_t, y_t)$.
   - If α_t is close to 1, we are likely to make a correct prediction.
   - If α_t is close to 0, we are likely to make a prediction error.
2. On the next round, after updating the set of consistent hypotheses, we will have $|V_{t+1}| = \alpha_t |V_t|$.
3. Since we now assume that α_t is small, we will have a much smaller set of consistent hypotheses in the next round.
4. If we are likely to have mistake on the current example, then we are going to learn a lot from this example as well, and therefore be more accurate in later rounds.

Proof of Mistake bound of RandConsistent algorithm.

1. For each round t, let $\alpha_t = \frac{|V_{t+1}|}{|V_t|}$. After T rounds we have $1 \leq |V_{T+1}| = |H| \prod_{t=1}^{T} \alpha_t$.
2. Using the inequality $b \leq e^{-(1-b)}$, which holds for all b, we get that

$$1 \leq |H| \prod_{t=1}^{T} e^{-(1-\alpha_t)} = |H| e^{-\sum_{t=1}^{T}(1-\alpha_t)} \Longrightarrow \sum_{t=1}^{T}(1-\alpha_t) \leq \ln|H|.$$

3. Since we predict $\hat{y}_t$ by choosing $h \in H$ uniformly, the probability to make a mistake on round t is

$$\mathbb{P}[\hat{y}_t \neq y_t] = \frac{|\{h \in V_t \mid h(\mathbf{x}_t) \neq y_t\}|}{|V_t|} = \frac{|V_t| - |V_t + 1|}{|V_t|} = (1-\alpha_t).$$

4. Therefore, the expected number of mistakes is

$$\sum_{t=1}^{T} \mathbb{E}[\mathbb{I}[\hat{y}_t \neq y_t]] = \sum_{t=1}^{T} \mathbb{P}[\hat{y}_t \neq y_t] = \sum_{t=1}^{T}(1-\alpha_t) \leq \ln(|H|).$$

□

1. It is interesting to compare the mistake bound of RandConsistent with the generalization bound of PAC model.
2. In PAC model, T equals to size of the training set.
3. PAC model implies that with probability of at least $(1 - \delta)$, the average error on new examples is guaranteed to be at most $\ln(|H|/\delta)/T$.
4. In contrast, the mistake bound of RandConsistent tells us a much stronger guarantee. We do not need to first train the model on T examples, in order to have error rate of $\ln(|H|)/T$.
5. We can have this same error rate immediately on the first T examples we observe.
6. Another important difference between these two models is that in online we don't assume that instances are sampled i.i.d. from some underlying distribution.

1. Removing the i.i.d. assumption is a big advantage.
2. In other hand, we only have a guarantee on $M_A(H)$ but we have no guarantee that after observing T examples we will identify h^*.
3. If we observe the same example on all the online rounds, we will make few mistakes but we will remain with a large version space V_t.
4. This Theorem bounds the expected number of mistakes. Using concentration techniques, we can obtain a bound which holds with extremely high probability.
5. A simpler way is to explicitly derandomize the algorithm.
6. A simple derandomization is to make a deterministic prediction according to majority vote of $h \in V_t$.
7. The resulting algorithm is called **Halving**.

It is easy to construct a hypothesis class and a sequence of examples on which Consistent will indeed make $|H| - 1$ mistakes.

Halving algorithm

Let $V_1 = H$
for $t \leftarrow 1, 2, \ldots$ **do**
    Receive $\mathbf{x}_t$.
    Predict $\hat{y}_t = \underset{r \in \{0,1\}}{\operatorname{argmax}} |\{h \in V_t \mid h(\mathbf{x}_t) = r\}|$. In case of a tie predict $\hat{y}_t = 1$.
    Receive true label $y_t = h^*(\mathbf{x}_t)$.
    Update $V_{t+1} = \{h \in V_t \mid h(\mathbf{x}_t) = y_t\}$.
end for

Theorem (Mistakebound of Halving algorithm)

Let H be a finite hypothesis class. Halving algorithm has mistake bound $M_{Halving}(H) \leq \log_2|H|$.

Proof.

1. When Halving makes a mistake, we have $|V_{t+1}| \leq \frac{|V_t|}{2}$.
2. Let M be total number of mistakes, then we have

$$1 \leq |V_{T+1}| \leq |H|2^{-M}$$

3. Rearranging this inequality we conclude our proof.

□

Online learnability

1. What is the optimal online learning algorithm for a given hypothesis class H?
2. We present a dimension of hypothesis classes that characterizes the best achievable mistake bound.
3. This measure was proposed by Nick Littlestone and we therefore refer to it as $Ldim(H)$.
4. To define Ldim, we consider online learning process as a game between two players: learner versus environment.
5. On round t of the game,
   - the environment picks an instance $\mathbf{x}_t$,
   - the learner predicts a label $\hat{y}_t \in \{0, 1\}$, and
   - the environment outputs the true label, $y_t \in \{0, 1\}$.

1. Suppose that environment wants to make learner have mistake on the first T rounds of the game.
2. Then, it must output $y_t = 1 - \hat{y}_t$, and the question is how it should choose the instances $\mathbf{x}_t$ in such a way that ensures that for some $h^* \in H$, we have $y_t = h^*(\mathbf{x}_t)$ for all $t \in \{1, 2, \ldots, T\}$.
3. A strategy for an adversarial environment can be formally described as the following a binary tree.
   - Each node of the tree is associated with an instance from $\mathcal{X}$.
   - Initially, environment presents to learner the instance associated with root of tree.
   - If learner predicts $\hat{y}_t = 0$, then environment will set $y_t = 1$ and will traverse to the right child.
   - If learner predicts $\hat{y}_t = 1$, then environment will set $y_t = 0$ and will traverse to the left child.
   - This process will continue.

1. Formally, consider a full binary tree of depth T.
2. We define the depth of the tree as the number of nodes in a path from the root to a leaf.
3. This tree has $(2^T - 1)$ nodes each of which is assigned an instance. Let $\mathbf{v}_1, \ldots, \mathbf{v}_{2^T-1}$ be these instances.
4. We start from the root of the tree, and set $\mathbf{x}_1 = \mathbf{v}_1$.
5. At round t, we set $\mathbf{x}_t = \mathbf{v}_{i_t}$ where i_t is the current node.
6. If $y_t = 0$, we go to the left child of i_t, otherwise we go to the right child of i_t.
7. This results in $i_{t+1} = 2i_t + y_t$ and hence

$$i_t = 2^{t-1} + \sum_{j=1}^{t-1} y_j 2^{t-1-j}.$$

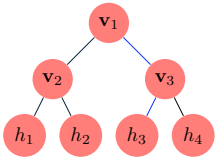
8. This **strategy succeeds** if for every $(y_1, \ldots, y_T)$ there exists a $h^* \in H$ such that for all $t \in \{1, \ldots, T\}$, we have $y_t = h^*(\mathbf{x}_t)$.

Definition (H Shattered tree)

A shattered tree of depth d is a sequence of instances $\mathbf{v}_1, \ldots, \mathbf{v}_{2^d-1}$ in $\mathcal{X}$ such that for every labeling $(y_1, \ldots, y_d) \in \{0,1\}^d$ there exists $h \in H$ such that for all $t \in \{1, \ldots, d\}$ we have $y_t = h(\mathbf{x}_{i_t})$, where

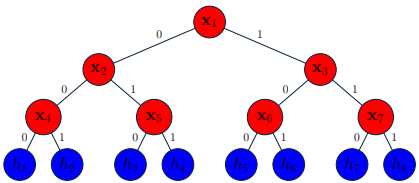
$$i_t = 2^{t-1} + \sum_{j=1}^{t-1} y_j 2^{t-1-j}.$$

	h_1	h_2	h_3	h_4
$\mathbf{v}_1$	0	0	1	1
$\mathbf{v}_2$	0	1	?	?
$\mathbf{v}_3$	?	?	0	1



Definition (Littlestone's Dimension (Ldim))

$Ldim(H)$ is the maximal integer T such that there exists a shattered tree of depth T, which is shattered by H.



$Ldim(H)$ is maximum depth of a full binary tree shattered by H.

Theorem (Lower bound of mistake)

No algorithm can have a mistake bound strictly smaller than $Ldim(H)$; namely, for every algorithm, A, we have $M_A(H) \geq Ldim(H)$.

Proof.

1. Let $T = Ldim(H)$ and let $\mathbf{v}_1, \ldots, \mathbf{v}_{2^T-1}$ be a sequence that satisfies the requirements in the definition of $Ldim$.
2. If the environment sets $\mathbf{x}_t = \mathbf{v}_{i_t}$ and $y_t = 1 - \hat{y}_t$ for all $t \in \{1, \ldots, T\}$, then the learner makes T mistakes.
3. In other hand, the definition of $Ldim$ implies that there exists a hypothesis $h \in H$ such that $y_t = h(\mathbf{x}_{i_t})$ for all $t \in \{1, \ldots, T\}$.

□

Theorem (Online learnability)

If $Ldim(H)$ is finite, then **the hypothesis class H is online learnable**.

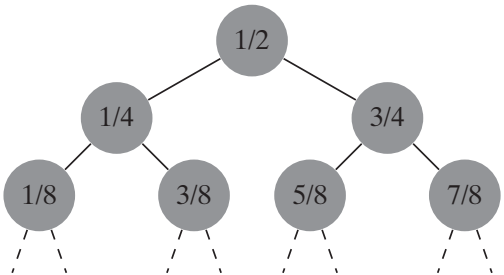
Example (Finite hypothesis class)

Let H be a finite hypothesis class. Clearly, any tree that is shattered by H has **depth of at most** $\log_2(|H|)$. Therefore, $Ldim(H) \leq \log_2(|H|)$.

Example (Threshold function)

Let $\mathcal{X} = [0, 1]$ and $H = \{\mathbf{x} \mapsto \text{sgn}(\mathbf{x} - a) \mid \mathbf{x} \in [0, 1]\}$. Then, $Ldim(H) = \infty$.

The following tree is shattered by H and, because of the density of the reals, **this tree can be made arbitrarily deep**.



Example (Finite domain/hypothesis set)

Let $\mathcal{X} = \{1, 2, \ldots, d\}$ and $H = \{h_1, \ldots, h_d\}$ where $h_j(\mathbf{x}) = 1$ iff $\mathbf{x} = j$. Then, it is easy to show that $Ldim(H) = 1$ while $|H| = d$ can be arbitrarily large.

Therefore, this example shows that $Ldim(H)$ **can be significantly smaller than** $\log_2(|H|)$.

Homework: Show that $Ldim(H) = 1$**.**

Example (Perceptron)

Let $\mathcal{X} = \left\{\mathbf{x} \in \{0,1\}^* \;\middle|\; \|\mathbf{x}\|_0 \leq r\right\}$ and $H = \left\{\mathbf{x} \mapsto \operatorname{sgn}\left(\langle \mathbf{w}, \mathbf{x}\rangle\right) \;\middle|\; \|\mathbf{w}\|_0 \leq k\right\}$. Then $|H| = \infty$ but $Ldim(H) \leq r \times k$.

Homework: Show that $Ldim(H) \leq r \times k$**.**

Is there any algorithm A such that for hypothesis class H we have $M_A(H) = Ldim(H)$?

Standard Optimal Algorithm (SOA)

Let $V_1 = H$
for $t \leftarrow 1, 2, \ldots$ **do**
    Receive $\mathbf{x}_t$.
    For $r \in \{0, 1\}$, let $V_t^{(r)} = \{h \in V_t \mid h(\mathbf{x}_t) = r\}$
    Predict $\hat{y}_t = \underset{r \in \{0,1\}}{\operatorname{argmax}} \, Ldim(V_t^{(r)})$.
    Receive true label $y_t = h^*(\mathbf{x}_t)$.
    Update $V_{t+1} = V_t^{(y_t)}$.
end for

The SOA **uses the same idea as Halving**, but instead of predicting according to **the larger class**, it predicts according to the **class with larger** *Ldim*.

Theorem (Optimality of SOA)

SOA enjoys the mistake bound $M_{SOA}(H) \leq Ldim(H)$.

Proof (Optimality of SOA).

1. It suffices to prove that when algorithm makes a mistake, we have $Ldim(V_{t+1}) \leq Ldim(V_t) - 1$.
2. We prove this claim by assuming the contrary, that is $Ldim(V_{t+1}) = Ldim(V_t)$.
3. If this is true, then definition of $\hat{y}_t$ implies $Ldim(V_{t+1}) = Ldim(V_t)$ for both $r = 1$ and $r = 0$.
4. But in this case, then we can construct a shaterred tree of depth $Ldim(V_t) + 1$ for the class V_t, which leads to the desired contradiction.

□

Corollary

Let H be any hypothesis class. Then, the standard optimal algorithm enjoys the mistake bound $M_{SOA}(H) = Ldim(H)$ and no other algorithm A can have $M_A(H) < Ldim(H)$.

Proof.

By combining Theorem (Lower bound of mistake) and Theorem (Optimality of SOA), this corollary can be proved. □

In PAC learning, learnability is characterized by the $VC(H)$.

Recall that $VC(H)$ is the maximal number d such that there are instances $\mathbf{x}_1, \ldots, \mathbf{x}_d$ that are shattered by H.

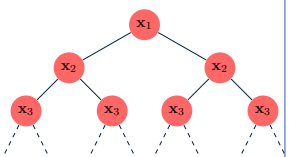
That is for any sequence of labels $(y_1, \ldots, y_d) \in \{0,1\}^d$ there exists a hypothesis $h \in H$ that gives exactly this sequence of labels.

Theorem

For any class H, we have $VC(H) \leq Ldim(H)$.

Proof.

1. Let $VC(H) = d$ and $\mathbf{x}_1, \ldots, \mathbf{x}_d$ be a shattered set.
2. Construct a full binary tree of instances $\mathbf{x}_1, \ldots, \mathbf{x}_{2^d-1}$, where all nodes at depth i are $\mathbf{x}_i$.
3. Definition of shattered sample implies we got a valid shattered tree of depth d and we conclude $VC(H) \leq Ldim(H)$.



□

Corollary

For any finite hypothesis class H, we have $V(H) \leq Ldim(H) \leq \log(|H|)$.

Example (Threshold function)

Let $\mathcal{X} = \mathbb{R}$ and $H = \{\mathbf{x} \mapsto \text{sgn}(\mathbf{x} - a) \mid \mathbf{x} \in \mathbb{R}\}$. We have shown that $VC(H) = 1$ while $Ldim(H) = \infty$.

Example

Let $\mathcal{X} = \{1, 2, \ldots, d\}$ and $H = \{h_1, h_2, \ldots, h_d\}$, where $h_k(\mathbf{x}) = 1$ iff $\mathbf{x} = k$.

1. **Show that** $Ldim(H) = 1$ while $|H| = d$ can be arbitrarily large.
2. **What is** $VC(H)$**?**

Online classification in the unrealizable case

1. Similarly to agnostic PAC model, we no longer assume that all labels are generated by some $h^* \in H$, but we **require the learner to be competitive with the best fixed predictor from** H.
2. This is captured by **the regret of the algorithm**, which measures **how sorry the learner is**.

Definition (Regret)

The **regret of an algorithm** A **relative to** h when running on a sequence of T examples is defined as

$$Regret_A(h, T) = \sup_{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)} \left[\sum_{t=1}^{T} |\hat{y}_t - y_t| - \sum_{t=1}^{T} |h(\mathbf{x}_t) - y_t| \right],$$

and the **regret of the algorithm relative to a hypothesis class** H is

$$Regret_A(H, T) = \sup_{h \in H} Regret_A(h, T).$$

The **goal of learner** is to have the **lowest possible regret relative to** H.

1. Halving simply discards experts after a single mistake.
2. Weighted majority (WM) weights importance of experts as a function of their mistake rate.
3. WM reduces weight of incorrect experts by factor of $\beta \in [0, 1)$. It reduces to Halving when $\beta = 0$.

Weighted majority algorithm (WM)

$w_{1,i} \leftarrow 1$ for all $i \in \{1, 2, \ldots, N\}$
for $t \leftarrow 1, 2, \ldots$ **do**
  Receive $\mathbf{x}_t$.
  $\hat{y}_t \leftarrow \mathbb{I}\left[\sum_{i:y_{t,i}=1} w_{t,i} \geq \sum_{i:y_{t,i}=0} w_{t,i}\right]$
  Receive true label y_t.
  if $(\hat{y}_t \neq y_t)$ **then**
    $w_{t+1,i} \leftarrow \mathbb{I}[y_{t,i} \neq y_t]\, \beta w_{t,i} + \mathbb{I}[y_{t,i} = y_t]\, w_{t,i}$ for all $i \in \{1, 2, \ldots, N\}$
  end if
end for
return $\mathbf{w}_{T+1}$

Theorem (Mistakes of WM)

Fix $\beta \in (0,1)$. Let $M_{WM}(N)$ be the number of mistakes made by **WM** *after $T \geq 1$ rounds, and m_T^* be the number of mistakes made by* **the best of the N experts**. *Then*

$$M_{WM}(N) \leq \frac{\log N + m_T^* \log(1/\beta)}{\log(\frac{2}{1+\beta})}.$$

Proof (Mistake bound of WM)

1. We use a potential function, derive its upper and lower bounds and combine them for proof.
2. For $t \geq 1$, define potential function as $W_t = \sum_{i=1}^{N} w_{t,i}$.
3. Predictions generated using weighted majority vote, if algorithm has mistake at round t, then

$$W_{t+1} \leq \left[\frac{1}{2} + \frac{1}{2}\beta\right] W_t = \left[\frac{1+\beta}{2}\right] W_t$$

Proof (Mistake bound of WM).

4. Since $W_1 = N$ and $M_{WM}(N)$ mistakes are made after T rounds, we obtain

$$W_T \leq \left[\frac{1+\beta}{2}\right]^{M_{WM}(N)} N.$$

5. Since $w_{t,i} > 0$, for all experts i, $W_T \geq w_{T,i} = \beta^{m_{T,i}}$, where $m_{T,i}$ is the number of mistakes made by the ith expert after T rounds.

6. Applying this lower bound to the best expert

$$\beta^{m_T^*} \leq W_T \leq \left[\frac{1+\beta}{2}\right]^{M_{WM}(N)} N$$

$$m_T^* \log \beta \leq \log N + M_{WM}(N) \log \left[\frac{1+\beta}{2}\right]$$

$$M_{WM}(N) \log \left[\frac{2}{1+\beta}\right] \leq m_T^* \log(\frac{1}{\beta}) + \log N$$

□

1. This theorem guarantees a bound of the following form for WM (for constant C)

$$M_{WM}(N) \leq O(\log N) + C \times m_T^*.$$

2. It guarantees that the number of mistakes is roughly a constant times m_T^*.
3. This is a remarkable result, because **it requires no assumption about the sequence of points and labels generated**.
4. In realizable case ($m_T^* = 0$), the bound reduces to $M_{WM}(N) \leq O(\log N)$ as **Halving algorithm**.
5. For deterministic algorithm A, let $H = \{h_0, h_1\}$, where $h_0(\mathbf{x}) = 0$ and $h_1(\mathbf{x}) = 1$ for all $\mathbf{x}$.
6. An adversary can make $M_A(H) = T$, by simply waiting for $\hat{y}_t$ and then give $y_t = 1 - \hat{y}_t$.

1. The error of the best expert over that sequence is at most $m_T^* \leq \frac{T}{2}$. Thus, for that sequence, we have
$$Regret_A(H, T) = M_A(H) - m_T^* \geq \frac{T}{2}$$
2. This shows that the regret of any deterministic online algorithm such as **weighted majority is not sublinear in T.**
3. **Question:** can we design an algorithm **with low regret**, meaning that $Regret_A(H, T)$ grows sublinearly with T?
4. This implies that the difference between the error rate of the **learner** and **the best hypothesis in H** tends to zero as T goes to infinity. This means that $Regret_A(H, T) = o(T)$.
5. This is **impossible** because (**no deterministic algorithm can obtain a sublinear regret bound even if $|H| = 2$**) as shown.
6. This impossibility result is attributed to Cover.

1. To sidestep Cover's impossibility result, we must further restrict the power of the adversarial environment **by allowing the learner to randomize his predictions**.
2. To make **the randomization meaningful**, we force the adversarial environment to decide on y_t without knowing the random coins flipped by the learner on round y_t.
3. The adversary can still know the learner's forecasting strategy and even the random coin flips of previous rounds, but it does not know the actual value of the random coin flips used by the learner on round t.
4. With this change, we analyze the expected number of mistakes of the algorithm, where the expectation is with respect to the learner's own randomization.

1. In this randomized scenario, we assume that a set $\mathcal{A} = \{1, \ldots, N\}$ of N actions is available.
2. At round t, algorithm A selects a distribution $\mathbf{p}_t$ over the set of actions, receives a loss vector $\mathbf{l}_t$, whose ith component $l_{t,i} \in [0, 1]$ is the loss of action i. Thus, we have
   - **the expected loss:** $L_t = \sum_{i=1}^{N} p_{t,i} l_{t,i} = \langle \mathbf{p}_t, \mathbf{l}_t \rangle$,
   - **the total loss:** $\mathcal{L}_T = \sum_{t=1}^{T} L_t$,
   - **the total loss of action i:** $\mathcal{L}_{T,i} = \sum_{t=1}^{T} l_{t,i}$,
   - **the best action loss:** $\mathcal{L}_T^{min} = \min_{i \in \mathcal{A}} \mathcal{L}_{T,i}$,
   - **the regret of algorithm:** $Regret_A(H, T) = \mathcal{L}_T - \mathcal{L}_T^{min}$.

For this algorithm, we consider zero-one losses $l_{t,i} \in \{0, 1\}$ for all t.

Randomized weighted majority algorithm (RWM)

$w_{1,i} \leftarrow 1, p_{1,i} \leftarrow \frac{1}{N}$ for all $i \in \{1, 2, \ldots, N\}$
for $t \leftarrow 1, 2, \ldots$ **do**
    Receive $\mathbf{x}_t$.
    Choose expert k with probability $p_{t,k}$ and outputs its prediction.
    Receive true label y_t.
    for $(i \leftarrow 1$ to $N)$ **do**
        if $(l_{t,i} = 1)$ **then**
            $w_{t+1,i} \leftarrow \beta w_{t,i}$
        else
            $w_{t+1,i} \leftarrow w_{t,i}$
        end if
    end for
    $W_{t+1} \leftarrow \sum_{i=1}^{N} w_{t,i}$
    $p_{t+1,i} \leftarrow \frac{w_{t+1,i}}{W_{t+1}}$ for all $i \in \{1, 2, \ldots, N\}$
end for
return $\mathbf{w}_{T+1}$

This is equivalent to $q_j = \frac{\sum_{i=1}^{N} w_{t,i} \mathbb{I}[h_i(\mathbf{x}_t) = j]}{W_t}$ for $j \in \{0, 1\}$.

Then choose $\hat{y}_t = j$ with probability q_j.

Theorem (Bounds on $Regret_{RWM}(H, T)$)

Fix $\beta \in [\frac{1}{2}, 1)$. Then for any $T \geq 1$, the loss of RWM on any sequence can be bounded as

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

In particular, for $\beta = \max\{\frac{1}{2}, 1 - \sqrt{\frac{\log\ N}{T}}\}$, the loss can be bounded as

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

Proof (Bounds on $Regret_{RWM}(H, T)$).

1. Let $W_t = \sum_{i=1}^{N} w_{t,i}$. Then, we have

$$\begin{aligned} W_{t+1} = \sum_{i:\ l_{t,i}=0} w_{t,i} + \beta \sum_{i:\ l_{t,i}=1} w_{t,i} &= W_t + (\beta - 1) \sum_{i:\ l_{t,i}=1} w_{t,i} \\ &= W_t + (\beta - 1) W_t \sum_{i:\ l_{t,i}=1} p_{t,i} \\ &= W_t + (\beta - 1) W_t L_t \\ &= W_t \left(1 - (1 - \beta) L_t\right) \end{aligned}$$

2. Since $W_1 = N$, then $W_{T+1} = N \prod_{t=1}^{T} \left(1 - (1 - \beta) L_t\right)$, $W_{T+1} \geq \max_{i \in \{1,\ldots,N\}} w_{T+1,i} = \beta^{\mathcal{L}_T^{min}}$.

Proof (Bounds on $Regret_{RWM}(H, T)$) (cont.).

3. Then using $\log(1-x) \leq -x$ valid for all $x < 1$ and $-\log(1-x) < x + x^2$ valid for all $x \in [0, \frac{1}{2}]$, we obtain $\beta^{\mathcal{L}_T^{min}} \leq N \prod_{t=1}^{N} (1 - (1-\beta)L_t)$. Then

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

$$\Longrightarrow \mathcal{L}_T^{min} \log \beta \leq \log N - (1-\beta) \sum_{t=1}^{T} L_t$$

$$\Longrightarrow \mathcal{L}_T^{min} \log \beta \leq \log N - (1-\beta)\mathcal{L}_T$$

$$\Longrightarrow \mathcal{L}_T \leq \frac{\log N}{1-\beta} - \frac{\log \beta}{1-\beta} \mathcal{L}_T^{min}$$

$$\Longrightarrow \mathcal{L}_T \leq \frac{\log N}{1-\beta} - \frac{\log(1-(1-\beta))}{1-\beta} \mathcal{L}_T^{min}$$

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

4. This shows the first statement.

Proof (Bounds on $Regret_{RWM}(H, T)$) (cont.).

5. Since $\mathcal{L}_T^{min} \leq T$, this also implies
$$\mathcal{L}_T \leq \frac{\log N}{1-\beta} + (1-\beta)T + \mathcal{L}_T^{min}$$
6. Differentiating the upper bound with respect to β and setting it to zero gives $\frac{\log\ N}{(1-\beta)^2} - T = 0$, that is $\beta = 1 - \sqrt{(log\ N)/T}$.
7. Thus, if $1 - \sqrt{(log\ N)/T} \geq \frac{1}{2}$, then $\beta_0 = 1 - \sqrt{(log\ N)/T}$ is the minimizing value of β,otherwise the boundary value $\beta_0 = \frac{1}{2}$ is the optimal value.
8. The second statement follows by replacing β with β_0.

□

This bound assumes that **the algorithm additionally receives as a parameter the number of rounds T**.

There exists a general doubling trick that can be used to relax this requirement at the price of a small constant factor increase.

For this algorithm, we consider loss $l_{t,i} \in [0,1]$ for all t.

Randomized exponential weighted majority algorithm (REWM)

Set $\eta \leftarrow \sqrt{2 \log N/T}$
$w_{1,i} \leftarrow 1, p_{1,i} \leftarrow \frac{1}{N}$ for all $i \in \{1, 2, \ldots, N\}$
for $t \leftarrow 1, 2, \ldots$ **do**
    Receive $\mathbf{x}_t$.
    Choose expert k with probability $p_{t,k}$ and outputs its prediction.
    Receive true label y_t and hence $\mathbf{l}_t = (l_{t,1}, \ldots, l_{t,N})$.
    for $(i \leftarrow 1$ to $N)$ **do**
        $w_{t+1,i} \leftarrow w_{t,i} \exp(-\eta l_{t,i})$
    end for
    $W_{t+1} \leftarrow \sum_{i=1}^{N} w_{t,i}$
    $p_{t+1,i} \leftarrow \frac{w_{t+1,i}}{W_{t+1}}$ for all $i \in \{1, 2, \ldots, N\}$
end for
return $\mathbf{w}_{T+1}$

Theorem (Bounds on $Regret_{REWM}(H, T)$)

Assuming that $T > 2\log(N)$, then REWM enjoys the bound

$$\sum_{t=1}^{T} \langle \mathbf{p}_t, \mathbf{l}_t \rangle - \min_{i \in \{1,\dots,T\}} \sum_{t=1}^{T} l_{t,i} \leq \sqrt{2T\log(N)}.$$

Proof (Bounds on $Regret_{REWM}(H, T)$).

1. We have

$$\log \frac{W_{t+1}}{W_t} = \log \sum_{i=1}^{N} \frac{w_{t,i}}{W_t} \exp\left(-\eta l_{t,i}\right) = \log \sum_{i=1}^{N} p_{t,i} \exp\left(-\eta l_{t,i}\right)$$

2. By using inequality $e^{-a} \leq 1 - a + a^2/2$, which holds for all $a \in (0, 1)$, we obtain

$$\begin{aligned} \log \frac{W_{t+1}}{W_t} &\leq \log \sum_{i=1}^{N} p_{t,i} \left(1 - \eta l_{t,i} + \eta^2 l_{t,i}^2/2\right) \\ &\leq \log \left[1 - \sum_{i=1}^{N} p_{t,i} \left(\eta l_{t,i} - \eta^2 l_{t,i}^2/2\right)\right] = \log(1 - b). \end{aligned}$$

Proof (Bounds on $Regret_{REWM}(H, T)$) (cont.).

3. Let $b = \sum_{i=1}^{N} p_{t,i} \left(\eta l_{t,i} - \eta^2 l_{t,i}^2/2\right)$.
4. Note that $b \in (0, 1)$. By using the inequality $\log(1 - b) \leq -b$, which holds for all $b \leq 1$, we obtain

$$\begin{aligned}
\log \frac{W_{t+1}}{W_t} &\leq -\sum_{i=1}^{N} p_{t,i} \left(\eta l_{t,i} - \eta^2 l_{t,i}^2/2\right) \\
&= -\eta \langle \mathbf{p}_t, \mathbf{l}_t \rangle + \eta^2 \sum_{i=1}^{N} p_{t,i} l_{t,i}^2/2 \\
&\leq -\eta \langle \mathbf{p}_t, \mathbf{l}_t \rangle + \eta^2/2.
\end{aligned}$$

5. Summing this inequality over t, we get

$$\log \frac{W_{T+1}}{W_1} = \log W_{T+1} - \log W_1 = \sum_{t=1}^{T} \log \frac{W_{t+1}}{W_t} \leq -\eta \sum_{t=1}^{T} \langle \mathbf{p}_t, \mathbf{l}_t \rangle + T\eta^2/2.$$

Proof (Bounds on $Regret_{REWM}(H, T)$) (cont.).

6. For each i, we can rewrite

$$w_{T+1,i} = \exp\left(-\eta \sum_{t=1}^{T} l_{t,i}\right)$$

7. Next, we lower bound W_{T+1}.

$$\log W_{T+1} = \log\left[\sum_{i=1}^{N} \exp\left(-\eta \sum_{t=1}^{T} l_{t,i}\right)\right] \geq \log\left[\max_i \exp\left(-\eta \sum_{t=1}^{T} l_{t,i}\right)\right]$$
$$= -\eta \min_i \sum_{t=1}^{T} l_{t,i}.$$

8. We have

$$\log W_{T+1} - \log W_1 \leq -\eta \sum_{t=1}^{T} \langle \mathbf{p}_t, \mathbf{l}_t \rangle + T\eta^2/2$$
$$\log W_{T+1} \geq -\eta \min_i \sum_{t=1}^{T} l_{t,i}.$$

Proof (Bounds on $Regret_{REWM}(H, T)$) (cont.).

9. Combining the above inequalities and the fact $\log W_1 = \log N$, we get that

$$-\eta \min_i \sum_{t=1}^{T} l_{t,i} - \log N \leq -\eta \sum_{t=1}^{T} \langle \mathbf{p}_t, \mathbf{l}_t \rangle + T\eta^2/2.$$

10. This can be rearranged as

$$\sum_{t=1}^{T} \langle \mathbf{p}_t, \mathbf{l}_t \rangle - \min_i \sum_{t=1}^{T} l_{t,i} \leq \frac{\log N}{\eta} + \frac{T\eta}{2}.$$

11. By plugging value of η into the above inequality, the proof will be completed.

□

1. Let $H = \{h_1, \ldots, h_N\}$ be a finite hypothesis class.
2. Each hypothesis h_i is an expert with advice $h_i(\mathbf{x}_t)$ and cost $l_{t,i} = |h_i(\mathbf{x}_t) - y_t|$.
3. Prediction of algorithm will be $\hat{y}_t = \sum_{i=1} p_{t,i} h_i(\mathbf{x}_t) \in [0, 1]$.
4. The loss is

$$|\hat{y}_t - y_t| = \left| \sum_{i=1}^{N} p_{t,i} h_i(\mathbf{x}_t) - y_t \right| = \left| \sum_{i=1}^{N} p_{t,i} \left(h_i(\mathbf{x}_t) - y_t \right) \right|$$

5. The last equality is concluded from
   - 5.1 if $y_t = 1$, then $h_i(\mathbf{x}_t) - y_t \leq 0$, and
   - 5.2 if $y_t = 0$, then $h_i(\mathbf{x}_t) - y_t \geq 0$.
6. This results in

$$|\hat{y}_t - y_t| = \sum_{i=1}^{N} p_{t,i} |h_i(\mathbf{x}_t) - y_t| = \langle \mathbf{p}_t, \mathbf{l}_t \rangle .$$

As conclusion, we have

Corollary

Let H be a finite hypothesis class. There exists an algorithm for online classification, whose predictions come from $[0,1]$, that enjoys the regret bound

$$\sum_{t=1}^{T} |\hat{y}_t - y_t| - \min_{h \in H} \sum_{t=1}^{T} |h(\mathbf{x}_t) - y_t| \leq \sqrt{2T \log |H|}.$$

Theorem

For every hypothesis class H, there exists an algorithm for online classification, whose predictions come from $[0,1]$, that enjoys the regret bound

$$\forall h \in H \quad \sum_{t=1}^{T} |\hat{y}_t - y_t| - \min_{h \in H} \sum_{t=1}^{T} |h(\mathbf{x}_t) - y_t| \leq \sqrt{2T \min\{\log|H|, Ldim(H)\log(eT)\}}.$$

Furthermore, **no algorithm can achieve an expected regret bound smaller** *than* $\Omega\left(\sqrt{Ldim(H)\,T}\right)$.

Proof.

1. The preceding corollary is the proof for the finite hypothesis class.
2. **Homework:** Prove the general hypothesis case.
3. **Homework:** Prove the lower bound.

□

Perceptron

1. In prediction with expert advice ,
   - we have a set of N experts,
   - each expert gives its prediction for a given input $\mathbf{z}_t$, and
   - we guess a prediction according to all experts' predictions.
2. The way of scoring is according to the best expert.
3. We could easily imagine a scenario where **no one expert is good**.
4. But, if we form a **committee of experts**, they might be much better.
5. We assume that there is a **perfect committee**, i.e. there exists a $\mathbf{w}^* \in \mathbb{R}^N$, such that for all t, we have $\operatorname{sgn}\left(\langle \mathbf{w}_t^*, \mathbf{x}_t \rangle\right) = y_t$.

Committee of experts

1: We have N experts $H = \{h_1, \ldots, h_N\}$.
2: **for** $t \leftarrow 1, 2, \ldots$ **do**
3: Receive $\mathbf{z}_t$.
4: Let $\mathbf{x}_t = (x_{t,1}, \ldots, x_{t,N}) = (h_1(\mathbf{z}_t), \ldots, h_N(\mathbf{z}_t)) \in \{-1, +1\}^N$
5: Predict $\hat{y}_t = \operatorname{sgn}(\langle \mathbf{w}_t, \mathbf{x}_t \rangle)$.
6: Receive true label y_t.
7: Update $\mathbf{w}_t$.
8: **end for**
9: **return** $\mathbf{w}_{T+1}$

Perceptron algorithm

Let $\mathbf{w}_1 \leftarrow 0$.
for $t \leftarrow 1, 2, \ldots, T$ **do**
  Receive $\mathbf{x}_t$.
  Predict $\hat{y}_t = \operatorname{sgn}\left(\langle \mathbf{w}_t, \mathbf{x}_t \rangle\right)$.
  Receive true label y_t.
  if $(\hat{y}_t \neq y_t)$ **then**
    $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_t \mathbf{x}_t$ ▷ more generally $\eta y_t \mathbf{x}_t$ and $\eta > 0$.
  else
    $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t$
  end if
end for
return $\mathbf{w}_{T+1}$

1. Before an update, $\mathbf{x}_t$ is misclassified and thus $y_t \langle \mathbf{w}_t, \mathbf{x}_t \rangle$ is negative.
2. After an update, $y_t \langle \mathbf{w}_{t+1}, \mathbf{x}_t \rangle = y_t \langle \mathbf{w}_t, \mathbf{x}_t \rangle + \eta \|\mathbf{x}_t\|^2$.
3. Thus, the update corrects the weight vector in the direction of making the inner product $y_t \langle \mathbf{x}_t, \mathbf{w}_t \rangle$ positive by augmenting it with the quantity $\eta \|\mathbf{x}_t\|^2 > 0$.

Lemma

Let $\mathbf{w}^*$ be the weight learned by Perceptron.If $\forall t$, we have $y_t \langle \mathbf{x}_t, \mathbf{w}^* \rangle \geq \rho$, then the inner product $\langle \mathbf{w}^*, \mathbf{w}_k \rangle$ increases at least linearly with each update.

Proof.

Weight vector $\mathbf{w}$ updated when the training instance is not classified correctly.
We consider the inner product $\langle \mathbf{w}^*, \mathbf{w}_k \rangle$ before and after each update.

$$
\begin{aligned}
\langle \mathbf{w}^*, \mathbf{w}_k \rangle &= \langle \mathbf{w}^*, (\mathbf{w}_{k-1} + y_k \mathbf{x}_k) \rangle \\
&= \langle \mathbf{w}^*, \mathbf{w}_{k-1} \rangle + y_k \langle w^*, \mathbf{x}_k \rangle \\
&\geq \langle \mathbf{w}^*, \mathbf{w}_{k-1} \rangle + \rho \\
&\geq \langle \mathbf{w}^*, \mathbf{w}_{k-2} \rangle + 2\rho \\
&\geq \langle \mathbf{w}^*, \mathbf{w}_{k-3} \rangle + 3\rho \\
&\vdots \\
&\geq \langle \mathbf{w}^*, \mathbf{w}_1 \rangle + k\rho = k\rho
\end{aligned}
$$

□

Lemma

If $\forall t$, we have $\|\mathbf{x}_t\| \leq r$, then the squared norm $\|\mathbf{w}_k\|^2$ increases at most linearly in the number of updates k.

Proof.

Weight vector $\mathbf{w}$ updated when the training instance is not classified correctly.
We consider the inner product $\|\mathbf{w}_k\|^2$ before and after each update.

$$\begin{aligned}
\|\mathbf{w}_k\|^2 &= \|\mathbf{w}_{k-1} + y_k \mathbf{x}_k\|^2 \\
&= \|\mathbf{w}_{k-1}\|^2 + 2y_k \langle \mathbf{w}_{k-1}, \mathbf{x}_k \rangle + \|\mathbf{x}_k\|^2 \\
&\leq \|\mathbf{w}_{k-1}\|^2 + \|\mathbf{x}_k\|^2 \\
&\leq \|\mathbf{w}_{k-1}\|^2 + r^2 \\
&\leq \|\mathbf{w}_{k-2}\|^2 + 2r^2 \\
&\vdots \\
&\leq \|\mathbf{w}_1\|^2 + kr^2 = kr^2.
\end{aligned}$$

□

Theorem (Mistake bound of Perceptron)

Suppose there exists a $\mathbf{w}^*$ *of unit length and values* $\rho > 0$ *and* $r > 0$ *such that* $\forall t$*, we have* $y_t \langle \mathbf{x}_t, \mathbf{w}^* \rangle \geq \rho$ *and* $\|\mathbf{x}_t\| \leq r$ *. Then, the number of mistakes made by the Perceptron algorithm is no more than* $\left(\frac{r}{\rho}\right)^2$ *.*

Proof.

The $\cos(\mathbf{x}, \mathbf{z})$ measures the similarity of $\mathbf{x}$ and $\mathbf{z}$.

$$\begin{aligned}\cos(\mathbf{w}^*, \mathbf{w}_k) &= \frac{\langle \mathbf{w}^*, \mathbf{w}_k \rangle}{\|\mathbf{w}^*\| . \|\mathbf{w}_k\|} \\ &\overset{1}{\geq} \frac{k\rho}{\|\mathbf{w}^*\| . \|\mathbf{w}_k\|} \overset{2}{\geq} \frac{k\rho}{\sqrt{kr^2}\|\mathbf{w}^*\|} \leq 1.\end{aligned}$$

The last inequality is because the cos is bounded by one. Hence, we have

$$\begin{aligned}k &\leq \sqrt{kr^2}\|\mathbf{w}^*\| / \rho \\ &\leq (r/\rho)^2 \|\mathbf{w}^*\|^2 = (r/\rho)^2 .\end{aligned}$$

□

Homework

1. *Consider a scenario that* $\mathbf{w}^*$ *consist of* 0 *and* 1 *and the number of* 1 *is* k.
2. *Let also* $k \ll N$.
3. *This means that* k *experts are the* **perfect committee**.
4. *By normalizing* $\|\mathbf{x}_t\|_2 \leq 1$ *and* $\|\mathbf{w}^*\|_2 = 1$, *find margin of Perceptron.*
5. *Find the number of mistakes of Perceptron algorithm.*

Winnow algorithm

1. Percepton algorithm has the following characteristics
   - bound independent of dimension and tight.
   - convergence can be slow for small margin, it can be in $\Omega(2^N)$
2. Winnow algorithm is particularly well suited to cases where
   - a relatively small number of dimensions or experts can be used to define an accurate weight vector and
   - many of the other dimensions may then be irrelevant.
3. In Winnow, weights are updated in multiplicative manner.

Winnow algorithm

Let $\mathbf{w}_1 \leftarrow \left(\frac{1}{N}, \ldots, \frac{1}{N}\right)$.
for $t \leftarrow 1, 2, \ldots, T$ **do**
    Receive $\mathbf{x}_t$.
    Predict $\hat{y}_t = \operatorname{sgn}\left(\langle \mathbf{w}_t, \mathbf{x}_t \rangle\right)$.
    Receive true label y_t.
    if $(\hat{y}_t \neq y_t)$ **then**
        $Z_t \leftarrow \sum_{i=1}^{N} w_{t,i} \exp(\eta y_t x_{t,i})$
        for $i \leftarrow 1, 2, \ldots, N$ **do**
            $$w_{t+1,i} \leftarrow \frac{w_{t,i} \exp(\eta y_t x_{t,i})}{Z_t}$$
        end for
    else
        $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_{t+1}$
    end if
end for
return $\mathbf{w}_{T+1}$

1. Winnow algorithm =WM algorithm:
   - when $y_{t,i} = \mathbf{x}_{t,i} \in \{-1, +1\}$, then $\mathrm{sgn}\left(\langle \mathbf{x}_t, \mathbf{w}_t \rangle\right)$ coincides with the majority vote.
   - multiplying by e^{η} or $e^{-\eta}$ the weight of correct or incorrect experts, is equivalent to multiplying by $\beta = e^{-2\eta}$ the weight of incorrect ones.
2. Relationships with other algorithms such as boosting and Perceptron (Winnow and Perceptron can be viewed as special instances of a general family).
3. Find such relationships as homework.

Theorem (Mistake bound for Winnow algorithm)

Assume that $\|\mathbf{x}_t\|_\infty \leq r_\infty$ *for all* $t \in \{1, 2, \ldots, T\}$ *and that for some* $r_\infty > 0$. *Assume that there exist* $\mathbf{v} \in \mathbb{R}^N$, $\mathbf{v} > 0$ *and* $\rho_\infty > 0$ *such that for all* $t \in \{1, 2, \ldots, T\}$

$$\rho_\infty \leq \frac{y_t \langle \mathbf{v}, \mathbf{x}_t \rangle}{\|\mathbf{v}\|_1}.$$

Then, for $\eta = \dfrac{\rho_\infty}{r_\infty^2}$, *the number of mistakes made by the Winnow algorithm is bounded by* $2 \left(\dfrac{r_\infty}{\rho_\infty} \right)^2 \log N$.

Proof of (Mistake bound for Winnow algorithm).

1. Define relative entropy as potential function $\Phi_t = \sum_{i=1}^{N} \frac{v_i}{\|\mathbf{v}\|_1} \log \frac{v_i / \|\mathbf{v}\|_1}{w_{t,i}}$.
2. The upper bound for each t, we have

$$
\begin{aligned}
\Phi_{t+1} - \Phi_t &= \sum_{i=1}^{N} \frac{v_i}{\|\mathbf{v}\|_1} \log \frac{w_{t,i}}{w_{t+1,i}} = \sum_{i=1}^{N} \frac{v_i}{\|\mathbf{v}\|_1} \log \frac{Z_t}{\exp(\eta y_t x_{t,i})} \\
&= \log Z_t - \eta \sum_{i=1}^{N} \frac{v_i}{\|\mathbf{v}\|_1} y_t x_{t,i} \\
&\leq \log \left[\sum_{i=1}^{N} w_{t,i} \exp(\eta y_t x_{t,i}) \right] - \eta \rho_\infty \\
&= \log \mathop{\mathbb{E}}_{i \sim \mathbf{w}_t} [\exp(\eta y_t x_{t,i})] - \eta \rho_\infty \\
&= \log \mathop{\mathbb{E}}_{i \sim \mathbf{w}_t} [\exp(\eta y_t x_{t,i} - \eta y_t \langle \mathbf{w}_t, \mathbf{x}_t \rangle + \eta y_t \langle \mathbf{w}_t, \mathbf{x}_t \rangle)] - \eta \rho_\infty \\
&\leq \log \left[\exp(\eta^2 (2 r_\infty)^2 / 8) \right] + \underbrace{\eta y_t \langle \mathbf{w}_t, \mathbf{x}_t \rangle}_{\leq 0} - \eta \rho_\infty \leq \eta^2 r_\infty^2 / 2 - \eta \rho_\infty
\end{aligned}
$$

Proof of (Mistake bound for Winnow algorithm) (cont.).

3. Summing up these inequalities over all t yields and let M be the total number of updates

$$\Phi_{T+1} - \Phi_1 \leq M \left[\eta^2 r_\infty^2/2 - \eta\rho_\infty\right].$$

4. We derive a lower bound by noting that

$$\Phi_1 = \sum_{i=1}^{N} \frac{v_i}{\|\mathbf{v}\|_1} \log \frac{v_i/\|\mathbf{v}\|_1}{1/N} = logN + \sum_{i=1}^{N} \frac{v_i}{\|\mathbf{v}\|_1} \log \frac{v_i}{\|\mathbf{v}\|_1} \leq \log N$$

5. The relative entropy is always non-negative, hence $\Phi_{T+1} \geq 0$, yielding

$$\Phi_{T+1} - \Phi_1 \geq 0 - \log N = -\log N.$$

6. By combining the upper and lower bounds we obtain

$$-\log N \leq M \left[\eta^2 r_\infty^2/2 - \eta\rho_\infty\right].$$

7. Setting $\eta = \frac{\rho_\infty}{r_\infty^2}$ yields the statement of the theorem.

□

On-line to batch conversion

1. Can online learning algorithms be used to derive hypotheses with small generalization error in the standard stochastic setting?
2. How can the intermediate hypotheses they generate be combined to form an accurate predictor?

On-line to batch setting

1. Let $H = \{h : \mathcal{X} \mapsto \mathcal{Y}'\}$ and let $L : \mathcal{Y}' \times \mathcal{Y} \mapsto \mathbb{R}_+$ be a bounded loss function ($L \leq M$ for some $M \geq 0$).
2. Let a standard supervised learning setting where a sample $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)\} \in (\mathcal{X} \times \mathcal{Y})^T$ is drawn i.i.d. according to some fixed but unknown distribution $\mathcal{D}$.
3. The sample is sequentially processed by an on-line learning algorithm A.
4. The algorithm starts with an initial hypothesis $h_1 \in H$ and generates a new hypothesis $h_{t+1} \in H$, after processing pair $(\mathbf{x}_t, y_t)$, for $t \in \{1, \ldots, T\}$.
5. The regret of the algorithm is defined as before by

$$Regret_A(H, T) = \sum_{t=1}^{T} L(h_t(\mathbf{x}_t), y_t) - \min_{h \in H} \sum_{t=1}^{T} L(h(\mathbf{x}_t), y_t)$$

6. The generalization error of a hypothesis $h \in H$ is its expected loss

$$\mathbf{R}(h) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}[L(h(\mathbf{x}), y)].$$

The following lemma gives a bound on average generalization errors of the hypotheses generated by A.

Lemma (Bound on average generalization errors of hypotheses $h_1, \ldots, h_T$)

Let $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)\} \in (\mathcal{X} \times \mathcal{Y})^T$ be a sample drawn iid according $\mathcal{D}$, $L \leq M$ a bounded loss and hypotheses $h_1, \ldots, h_T$ the sequence of hypotheses generated by an on-line algorithm A sequentially processing S. Then, for any $\delta > 0$, with probability at least $(1 - \delta)$, the following holds:

$$\frac{1}{T}\sum_{t=1}^{T} \mathbf{R}(h_t) \leq \frac{1}{T}\sum_{t=1}^{T} L(h_t(\mathbf{x}_t), y_t) + M\sqrt{\frac{2\log\frac{1}{\delta}}{T}}.$$

Proof.

1. For any $t \in \{1, \ldots, T\}$, define random variable $V_t = \mathbf{R}(h_t) - L(h_t(\mathbf{x}_t), y_t)$.
2. Observe that for any $t \in \{1, \ldots, T\}$, we have

$$\mathbb{E}\left[V_t \mid x_1, \ldots, x_{t-1}\right] = \mathbf{R}(h_t) - \mathbb{E}\left[L(h_t(\mathbf{x}_t), y_t) \mid h_t\right] = \mathbf{R}(h_t) - \mathbf{R}(h_t) = 0$$

3. Since $L \leq M$, then $V_t \in [-M, +M]$ for all $t \in \{1, \ldots, T\}$.
4. Using Azuma's inequality, we obtain $\mathbb{P}\left[\frac{1}{T}\sum_{t=1}^{T} V_t \geq \epsilon\right] \leq \exp\left(-2T\epsilon^2/(2M)^2\right)$.
5. Setting the right-hand side to be equal to $\delta > 0$ yields the statement of the lemma.

□

Summary

Summary

- We defined on line-learning problem.
- The on line learnability is characterized by *Ldim* measure.
- We analyzed several on line algorithms.
- On line algorithms can be used in batch learning.

1. Chapter 21 of (Shalev-Shwartz and Ben-David 2014).
2. Chapter 8 of (Mohri, Rostamizadeh, and Talwalkar 2018).
3. Paper (Ben-David, Pál, and Shalev-Shwartz 2009).
4. The interested reader is referred to (Cesa-Bianchi and Lugosi 2006) .

References

- Ben-David, Shai, Dávid Pál, and Shai Shalev-Shwartz (2009). "Agnostic Online Learning". In: *Proceedings of the 22nd Conference on Learning Theory, Montreal, Quebec, Canada, June 18-21*.
- Cesa-Bianchi, Nicolò and Gábor Lugosi (2006). *Prediction, learning, and games*. Cambridge University Press.
- Mohri, Mehryar, Afshin Rostamizadeh, and Ameet Talwalkar (2018). *Foundations of Machine Learning*. Second Edition. MIT Press.
- Shalev-Shwartz, Shai and Shai Ben-David (2014). *Understanding machine learning: From theory to algorithms*. Cambridge University Press.

Questions?