

Topic 3: Online concept learning

3.1 Concept classes and the basic scenario

3.1.1 Concept classes

Another departure from the expert advice scenario puts structure on the experts' predictions. Consider a scenario in which experts correspond to $\{0,1\}$ -valued functions from some class of functions $\mathcal{C} \subseteq \{0,1\}^{\mathcal{X}}$ defined on a domain $\mathcal{X}$. We refer to the function class $\mathcal{C}$ as a *concept class*, and the individual functions as *concepts*. Each domain element $x \in \mathcal{X}$ —a.k.a. *input* or *input point*—represents the available information about a present circumstance, and a concept $c \in \mathcal{C}$ represents a way of classifying the circumstance as positive ($c(x) = 1$) or negative ($c(x) = 0$) as a function of the available information. The output $c(x)$ is also called the *label* of x provided by c .

3.1.2 Online concept learning scenario

We will be interested in learning algorithms that take advantage of the structure of a concept class. Just as in the expert advice scenario, a learner makes predictions in a sequence of rounds. But unlike the expert advice scenario, and at the start of each round, the learner is provided an input $x \in \mathcal{X}$, which determines the predictions of all concepts $c \in \mathcal{C}$. We call this the *online concept learning scenario*. The *realizability assumption* from the expert advice scenario can be adapted to online concept learning. The *$\mathcal{C}$ -realizability assumption* is the assumption that, at any time, all actual outcomes that have been revealed (i.e., the “true” labels) are consistent with the labels provided by some “true” concept $c^* \in \mathcal{C}$. We call the pair (x, y) a *labeled example*.¹ The goal of the learner is to never make more than a finite number of mistakes, i.e., ensure a finite mistake bound.²

The online concept learning scenario can capture the expert advice scenario: consider $\mathcal{X} = \{0,1\}^N$ and $\mathcal{C} = \text{EXPERTS}_N := \{c_i \mid i \in [N]\}$ where $c_i(x_1, \dots, x_N) = x_i$ for all $i \in [N]$.³ Conversely, if $|\mathcal{C}| < \infty$, then every algorithm we have considered for the expert advice scenario can be used in the online concept learning scenario, too, simply by treating each concept as an expert and computing the experts' predictions in each round. However, we will be particularly interested in efficient algorithms when $|\mathcal{C}|$ is very large compared to the size of the input x .

3.2 Learning concepts over the Boolean domain

3.2.1 Concepts over the Boolean n -cube

In this section, we let $\mathcal{X} = \{0,1\}^n$ for some positive integer n , i.e., the *Boolean n -cube*. So each input $x = (x_1, \dots, x_n) \in \{0,1\}^n$ is described by n Boolean attributes (with 1 corresponding to true and 0 corresponding to false). A concept $c: \{0,1\}^n \rightarrow \{0,1\}$ maps each of the 2^n possible settings of the n attribute values to either true (0) or false (1). A concept can be described by its “truth table”

¹A labeled example (x, y) is called a positive example if $y = 1$, and a negative example if $y = 0$.

²The online concept learning scenario with the $\mathcal{C}$ -realizability assumption is also called the *online mistake bound learning model* with concept class $\mathcal{C}$.

³In social choice theory, the class EXPERTS_N is also sometimes called a class of *dictators*.

which gives this mapping explicitly; there are exactly 2^{2^n} possible truth tables. Thus every concept class $\mathcal{C}$ on the domain $\{0, 1\}^n$ has cardinality $|\mathcal{C}| \leq 2^{2^n}$, so the Halving algorithm ensures a mistake bound of $\log_2(|\mathcal{C}|) \leq 2^n$ with the realizability assumption. This mistake bound is also achieved by the naïve algorithm that simply memorizes every input/label-pair that has been encountered and hence can avoid mistakes on repeated inputs.

3.2.2 Monotone conjunctions

The first class we consider is that of monotone conjunctions. For each subset $S \subseteq [n] := \{1, \dots, n\}$, define the concept $c_S: \{0, 1\}^n \rightarrow \{0, 1\}$ by

$$c_S(x_1, \dots, x_n) := \bigwedge_{i \in S} x_i.$$

The value of $c_S(x_1, \dots, x_n)$ is true precisely when all x_i for $i \in S$ are true. The concept class $\text{MCONJ}_n := \{c_S \mid S \subseteq [n]\}$ contains all 2^n such concepts c_S , and each concept is succinctly described by an attribute index set $S \subseteq [n]$.

We may adapt the Halving algorithm from Section 1.2 to guarantee a mistake bound of $\log_2(|\text{MCONJ}_n|) = \log_2(2^n) = n$ with the realizability assumption. However, a naïve adaptation of Halving would explicitly track which of the 2^n concepts are “alive” at any point in time, which is computationally expensive. An alternative is the following algorithm, called Attribute Elimination, essentially due to Valiant (1984) and Angluin (1988).

Algorithm 3.1: Attribute Elimination for online learning monotone conjunctions over $\{0, 1\}^n$.

- Initially, set $S_1 := [n]$.
- In round t :
 - Observe $x_t = (x_{t,1}, \dots, x_{t,n}) \in \{0, 1\}^n$.
 - Predict $a_t := c_{S_t}(x_t)$.
 - Observe label $y_t \in \{0, 1\}$.
 - Update hypothesis:

$$S_{t+1} := \begin{cases} S_t & \text{if } a_t = y_t, \\ S_t \setminus \{i \in [n] \mid x_{t,i} = 0\} & \text{if } a_t \neq y_t. \end{cases}$$

Attribute Elimination maintains a “hypothesis” set of attribute indices that is guaranteed to always contain the index set S^* for a “true” concept c_{S^*} (under the realizability assumption). After a mistake, the Attribute Elimination updates the hypothesis set by removing indices of attributes that can be deduced to not belong to S^* . Both the “size” of the hypothesis and the number of operations needed for each update is linear in the number of attributes n .

Theorem 3.1. *In the online concept learning scenario with the MCONJ_n -realizability assumption, Attribute Elimination makes at most n mistakes.*

Proof. Let $S^* \subseteq [n]$ be attribute set for a “true” concept c_{S^*} that never makes a mistake. Also, define M_t to be the number of mistakes of Attribute Elimination through round t , with $M_0 := 0$.

We show by induction that, for all rounds $t \in \mathbb{N}$, we have

$$S^* \subseteq S_t \quad \text{and} \quad M_{t-1} \leq |S_1| - |S_t|. \quad (3.1)$$

The claim is true for $t = 1$ since $S_1 = [n]$ and $M_0 = 0$ by definition. So consider any round $t \geq 1$, and assume that (3.1) holds as the inductive hypothesis. We now show that $S^* \subseteq S_{t+1}$ and $M_t \leq |S_1| - |S_{t+1}|$.

If $a_t = y_t$, then $S^* \subseteq S_{t+1} = S_t$ and $M_t = M_{t-1} \leq |S_1| - |S_t| = |S_1| - |S_{t+1}|$. So assume instead that $a_t \neq y_t$. Since $S^* \subseteq S_t$ by (3.1), we have

$$a_t = \underbrace{\left(\bigwedge_{i \in S^*} x_{t,i} \right)}_{y_t} \wedge \left(\bigwedge_{i \in S_t \setminus S^*} x_{t,i} \right).$$

If $y_t = 0$, then $a_t = 0$ as well, which contradicts the assumption that the learner makes a mistake. So it must be that $y_t = 1$ and $a_t = 0$, and

$$\left(\bigwedge_{i \in S^*} x_{t,i} \right) = 1 \quad \text{and} \quad \left(\bigwedge_{i \in S_t \setminus S^*} x_{t,i} \right) = 0.$$

This implies the following facts:

1. $x_{t,i} = 1$ for all $i \in S^*$;
2. there exists some $i \in S_t \setminus S^*$ such that $x_i = 0$.

Combining the first fact with (3.1) gives $S^* \subseteq S_{t+1}$. The second fact implies $|S_{t+1}| \leq |S_t| - 1$; combining this with (3.1) (and the assumption $a_t \neq y_t$) gives

$$M_t = M_{t-1} + 1 \leq |S_1| - (|S_t| - 1) \leq |S_1| - |S_{t+1}|.$$

This completes the inductive step. The theorem follows since $|S_1| = n$. $\square$

3.2.3 Monotone disjunctions

The second class we consider is that of monotone disjunctions. For each subset $S \subseteq [n]$, define the concept $d_S: \{0, 1\}^n \rightarrow \{0, 1\}$ by

$$d_S(x_1, \dots, x_n) := \bigvee_{i \in S} x_i.$$

The value of $d_S(x_1, \dots, x_n)$ is true precisely when any x_i for $i \in S$ is true. The concept class $\text{MDISJ}_n := \{d_S \mid S \subseteq [n]\}$ contains all 2^n such concepts d_S .

By De Morgan's law,

$$\neg \bigvee_{i \in S} x_i = \bigwedge_{i \in S} \neg x_i$$

(where $\neg b = 1 - b$ for $b \in \{0, 1\}$). In other words, the negation of a monotone disjunction is the same as a monotone disjunction over the negated attributes. Therefore, learning monotone disjunctions reduces to learning monotone conjunctions (and vice versa) by a simple transformation of the inputs $(x_{t,1}, \dots, x_{t,n}) \mapsto (\neg x_{t,1}, \dots, \neg x_{t,n})$ and labels $y_t \mapsto \neg y_t$.

Theorem 3.2. *In the online concept learning scenario with the MDISJ_n -realizability assumption, Attribute Elimination with the appropriate input and label transformations makes at most n mistakes.*

Exercise 3.1. Design a learning algorithm for the online concept learning scenario with the $\text{MCONJ}_n \cup \text{MDISJ}_n$ -realizability assumption with mistake bound $O(n)$ and time complexity $O(n)$ per round. $\diamond$

3.2.4 Conjunctions

The third class we consider is that of conjunctions. For each pair of disjoint subsets $S, T \subseteq [n]$, define the concept $c_{S,T}: \{0,1\}^n \rightarrow \{0,1\}$ by

$$c_{S,T}(x_1, \dots, x_n) := \left(\bigwedge_{i \in S} x_i \right) \wedge \left(\bigwedge_{i \in T} \neg x_i \right).$$

The value of $c_{S,T}(x_1, \dots, x_n)$ is true precisely when all x_i for $i \in S$ are true, and all x_i for $i \in T$ are false. The concept class $\text{CONJ}_n := \{c_{S,T} \mid S, T \subseteq [n], S \cap T = \emptyset\}$ contains all 3^n such concepts $c_{S,T}$.

Suppose we extend the n Boolean attributes $(x_1, \dots, x_n)$ to $2n$ Boolean attributes $(x_1, \dots, x_{2n})$, where $x_{n+i} := \neg x_i$ for all $i \in [n]$. Then

$$c_{S,T}(x_1, \dots, x_n) = \bigwedge_{i \in S \cup (n+T)} x_i.$$

In other words, $c_{S,T}$ has a representation as a monotone conjunction over the extended attributes $(x_1, \dots, x_{2n})$. Therefore, learning conjunctions reduces to learning monotone conjunctions (over a larger domain with $2n$ attributes) by a simple transformation of the inputs $(x_{t,1}, \dots, x_{t,n}) \mapsto (x_{t,1}, \dots, x_{t,n}, \neg x_{t,1}, \dots, \neg x_{t,n})$.

Theorem 3.3. *In the online concept learning scenario with the CONJ_n -realizability assumption, Attribute Elimination with the appropriate input transformation makes at most $2n$ mistakes.*

Exercise 3.2. Let DISJ_n be the concept class of disjunctions, defined as you would expect. What are the appropriate input and label transformations needed for Attribute Elimination for the online concept learning scenario with the DISJ_n -realizability assumption? $\diamond$

Exercise 3.3. Improve the bound from Theorem 3.3 to $n + 1$ mistakes. $\diamond$

3.2.5 Linear threshold functions

The class of linear threshold functions (LTFs) is a very well-studied concept class in part for its connections to neural networks (Minsky and Papert, 1969). A function $f: \{0,1\}^n \rightarrow \{0,1\}$ is a linear threshold function (LTF) on $\{0,1\}^n$ if there exists $w \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$ such that

$$f(x) = \mathbf{1}\{\langle w, x \rangle > \theta\} = \begin{cases} 1 & \text{if } \sum_{i=1}^n w_i x_i > \theta, \\ 0 & \text{otherwise} \end{cases}$$

That is, $f(x)$ is true precisely when the standard inner product $\langle w, x \rangle$ of w and x exceeds θ . In this case, we say that the LTF f is represented by (w, θ) , where w is the weight vector w , and θ is

the *threshold*. Let LTF_n denote the concept class of all LTFs on $\{0, 1\}^n$. Although every LTF has infinitely-many representations, the number of LTFs is finite; in fact, Muroga (1971) showed that the number of LTFs satisfies

$$\log_2(|\text{LTF}_n|) = \Theta(n^2).$$

(An even sharper result of Zuev (1989) implies that $\lim_{n \rightarrow \infty} \log_2(|\text{LTF}_n|) = n^2$.) The upper bound $\log_2(|\text{LTF}_n|) = O(n^2)$ is a consequence of Sauer's lemma, which will be discussed later. The lower bound $\log_2(|\text{LTF}_n|) = \Omega(n^2)$ is the following proposition.

Proposition 3.1. $|\text{LTF}_n| \geq 2^{2 + \frac{n(n-1)}{2}}.$

Proof. We first claim that every $f \in \text{LTF}_n$ has a representation $(w, \theta) \in \mathbb{R}^n \times \mathbb{R}$ such that

$$|\{\langle w, x \rangle \mid x \in \{0, 1\}^n\}| = 2^n. \quad (3.2)$$

There are many ways to see this; here is one. Pick any representation $(a, t) \in \mathbb{R}^n \times \mathbb{R}$ for f . Let

$$\gamma := \min\{\langle a, x \rangle - t \mid x \in \{0, 1\}^n, \langle a, x \rangle > t\} > 0.$$

Let $v := (2^0, 2^1, \dots, 2^{n-1})$; set $w := a + \epsilon v$ and $\theta := t + \gamma/2$. We show that there is a choice of $\epsilon > 0$ such that (w, θ) is a representation for f and (3.2) holds.

- If $\epsilon \leq \gamma/2^{n+1}$, then for all $x \in \{0, 1\}^n$, $0 \leq \epsilon \langle v, x \rangle \leq \epsilon 2^{n-1} \leq \gamma/2$. Therefore:

$$\begin{aligned} \langle a, x \rangle \leq t &\implies \langle w, x \rangle = \langle a, x \rangle + \epsilon \langle v, x \rangle \leq t + \gamma/2 \leq \theta; \\ \langle a, x \rangle > t &\implies \langle w, x \rangle = \langle a, x \rangle + \epsilon \langle v, x \rangle \geq t + \gamma > \theta. \end{aligned}$$

So (w, θ) is a representation for f .

- Consider any distinct $x, x' \in \{0, 1\}^n$. Then $\langle v, x \rangle \neq \langle v, x' \rangle$. So if $\langle w, x \rangle = \langle w, x' \rangle$ for a particular choice of ϵ , then it must be that

$$\epsilon = \frac{\langle a, x \rangle - \langle a, x' \rangle}{\langle v, x \rangle - \langle v, x' \rangle}.$$

Consequently, there are finitely-many choices of $\epsilon > 0$ that lead to a weight vector w with $\langle w, x \rangle = \langle w, x' \rangle$ for some distinct $x, x' \in \{0, 1\}^n$. If ϵ is strictly smaller than all such choices, then (3.2) holds.

We now claim that, for all $n \in \mathbb{N}$,

$$|\text{LTF}_{n+1}| \geq 2^n |\text{LTF}_n|.$$

Fix any $f \in \text{LTF}_n$, and take a representation (w, θ) for f such that (3.2) holds. From this f , we construct 2^n distinct LTFs over $\{0, 1\}^{n+1}$ that behave the same as f on the sub-cube $\{(x_1, \dots, x_n, x_{n+1}) \in \{0, 1\}^{n+1} \mid x_{n+1} = 0\}$, but behave differently from each other on the other sub-cube $\{(x_1, \dots, x_n, x_{n+1}) \in \{0, 1\}^{n+1} \mid x_{n+1} = 1\}$. Set $S := \{\langle w, x \rangle \mid x \in \{0, 1\}^n\}$. For each $s \in S$, define the LTF $F_s: \{0, 1\}^{n+1} \rightarrow \{0, 1\}$ by

$$F_s(x, x_{n+1}) := \mathbb{1}\{\langle w, x \rangle + (\theta - s)x_{n+1} > \theta\}.$$

Observe that $F_s(x, 0) = f(x)$, and $F_s(x, 1) = \mathbb{1}\{\langle w, x \rangle > s\}$. Since the values in $S = \{\langle w, x \rangle \mid x \in \{0, 1\}^n\}$ are distinct, the 2^n choices of $s \in S$ lead to 2^n distinct LTFs over $\{0, 1\}^{n+1}$. So we have the desired construction of 2^n distinct LTFs over $\{0, 1\}^{n+1}$ corresponding to f . This gives the claimed bound $|\text{LTF}_{n+1}| \geq 2^n |\text{LTF}_n|$.

For $n = 1$, there are $|\text{LTF}_1| = 4$ distinct LTFs. Therefore

$$|\text{LTF}_n| \geq 4 \prod_{k=1}^{n-1} 2^k = 2^{2 + \frac{n(n-1)}{2}}$$

as claimed. $\square$

So there are many more LTFs than there are, say, conjunctions or disjunctions. In fact, conjunctions and disjunctions can be represented as LTFs. So algorithms for learning LTFs can be used to learn conjunctions or disjunctions (as well as many other well-known concept classes).

We may adapt the Halving algorithm from Section 1.2 to guarantee a mistake bound of $\log_2(|\text{LTF}_n|) = \log_2(2^{O(n^2)}) = O(n^2)$ for the online concept learning scenario with the LTF_n -realizability assumption; but as usual, a naïve implementation is computationally expensive. There is a polynomial-time algorithm that achieves mistake bound $O(n^2 \log(n))$ (Maass and Turán, 1994), but it is an open question whether there is a polynomial-time algorithm that achieves an asymptotically better mistake bound.

3.3 Perceptron and margins

3.3.1 Linear threshold functions over Euclidean space

Perhaps the most well-known algorithm for learning LTFs is (the online variant of) the Perceptron algorithm (Rosenblatt, 1958). Perceptron is applicable to learning LTFs over Euclidean n -space, $\mathbb{R}^n$ (or really, any inner product space):

$$\text{LTF}_{\mathbb{R}^n} := \left\{ f: \mathbb{R}^n \rightarrow \{0, 1\} \mid \exists (w, \theta) \in \mathbb{R}^n \times \mathbb{R} \text{ s.t. } f(x) = \mathbb{1}\{\langle w, x \rangle > \theta\} \right\}.$$

To use Perceptron to learn LTFs over the Boolean n -cube, we can “embed” $\{0, 1\}^n$ in $\mathbb{R}^n$.

The number of LTFs over $\mathbb{R}^n$ is not finite, so the Halving algorithm is not directly applicable. In fact, no algorithm can guarantee a finite mistake bound, even with the realizability assumption.

Theorem 3.4. *In the online concept learning scenario, for each learning algorithm, there is an infinite sequence of input/label-pairs consistent with some $c^* \in \text{LTF}_{\mathbb{R}^1}$ on which the learner makes a mistake in every round.*

Proof. To construct the infinite sequence, simulate the learning algorithm in the online concept learning scenario and set the true labels $(y_t)_{t \in \mathbb{N}}$ to be the opposite of the learner’s predictions $(a_t)_{t \in \mathbb{N}}$. We need to determine inputs $(x_t)_{t \in \mathbb{N}}$ so that for every $t \in \mathbb{N}$, the choice of x_t depends only on $y_1, \dots, y_{t-1}$, and there is some $c^* \in \text{LTF}_{\mathbb{R}^1}$ consistent with $(x_1, y_1), \dots, (x_t, y_t)$. This must be done for every possible sequence $(y_t)_{t \in \mathbb{N}}$.

For $t \in \mathbb{N}$, define

$$x_t := \sum_{i=1}^{t-1} (1 - y_i) 2^{-i} + 2^{-t}.$$

So, initially $x_1 = 1/2$; on round $t > 1$,

$$x_t = x_{t-1} + \left(\frac{1}{2} - y_{t-1}\right)2^{-(t-1)} = \begin{cases} x_{t-1} + 2^{-t} & \text{if } y_{t-1} = 0, \\ x_{t-1} - 2^{-t} & \text{if } y_{t-1} = 1. \end{cases}$$

We claim that for every $t \in \mathbb{N}$, there exists some $\theta_t \in \mathbb{R}$ such that

$$y_\tau = \mathbb{1}\{x_\tau > \theta_t\} \quad \text{for all } \tau \in \{1, \dots, t\},$$

i.e., the concept $c^* \in \text{LTF}_{\mathbb{R}^1}$ given by $c^*(x) = \mathbb{1}\{x > \theta_t\}$ is consistent with $(x_1, y_1), \dots, (x_t, y_t)$. A choice for θ_t that works is

$$\theta_t := \sum_{i=1}^t (1 - y_i)2^{-i}.$$

Indeed, for any $\tau \in \{1, \dots, t\}$,

$$x_\tau - \theta_t = \sum_{i=1}^{\tau-1} (1 - y_i)2^{-i} + 2^{-\tau} - \sum_{i=1}^t (1 - y_i)2^{-i} = y_\tau 2^{-\tau} - \sum_{i=\tau+1}^t (1 - y_i)2^{-i}$$

which is positive if and only if $y_\tau = 1$. □

The proof of Theorem 3.4 applies to the subset of $\text{LTF}_{\mathbb{R}^1}$ in which the weight vector is required to be 1 and the threshold is required to lie in $[0, 1)$. The constructed sequence of input/label-pairs ensures that

$$\begin{aligned} x_t + 2^{-t} &= \min\{x_\tau \mid \tau \in \{1, \dots, t\}, y_\tau = 1\} \cup \{1\}, \\ x_t - 2^{-t} &= \max\{x_\tau \mid \tau \in \{1, \dots, t\}, y_\tau = 0\} \cup \{0\}. \end{aligned}$$

In other words, $x_t + 2^{-t}$ is the leftmost positive example seen so far (or 1), and $x_t - 2^{-t}$ is the rightmost negative example seen so far (or 0). Let $V_t := \{\theta \in (0, 1) \mid \mathbb{1}\{x_\tau > \theta\} = y_\tau \forall \tau \in \{1, \dots, t\}\}$ be the possible values of the threshold that are consistent with the input/label-pairs prior to round t . The interval $(x_t - 2^{-t}, x_t + 2^{-t})$ is the “uncertainty region” for the threshold: for every $x \in (x_t - 2^{-t}, x_t + 2^{-t})$, there are thresholds $\theta, \theta' \in V_t$ for which $\mathbb{1}\{x > \theta\} = 0$ and $\mathbb{1}\{x > \theta'\} = 1$. The input in round t is placed in the middle of this interval, and no matter what the learner predicts, there is a threshold $\theta \in V_t$ for which $\mathbb{1}\{x_t > \theta\}$ differs from the learner’s prediction. The width of the interval also shrinks exponentially with the number of rounds t .

It is pathologies like the above that defeat every learning algorithm for LTFs over $\mathbb{R}^n$. Fortunately, this is the only kind of pathology that can defeat Perceptron.

3.3.2 Margins

To rule out the pathology from Section 3.3.1, we formalize a strengthened version of the realizability assumption for LTFs. We assume that the sequence $((x_t, y_t))_{t \in \mathbb{N}}$ satisfies the following for some parameter $\gamma > 0$ called the margin:

- $\|x_t\| \leq 1$ for all $t \in \mathbb{N}$ (where $\|\cdot\|$ denotes the Euclidean norm); and

- there exists $(w^*, \theta^*) \in \mathbb{R}^n \times \mathbb{R}$ with $\|w^*\| \leq 1$ and $|\theta^*| \leq 1$ such that, for all $t \in \mathbb{N}$,

$$\begin{aligned} y_t = 1 &\implies \langle w^*, x_t \rangle \geq \theta^* + \gamma, \\ y_t = 0 &\implies \langle w^*, x_t \rangle \leq \theta^* - \gamma. \end{aligned} \tag{3.3}$$

This assumption (both parts together) is called the *γ -margin assumption*. The requirement that $\|x_t\| \leq 1$ and $\|w^*\| \leq 1$ fixes a scale to ensure, via Cauchy-Schwarz, that $\langle w^*, x_t \rangle$ lies in $[-1, 1]$. (The restriction $\theta^* \in [-1, 1]$ is then without loss of generality.) The margin γ is on the same scale as $\langle w^*, x_t \rangle$ and θ^* . The larger the value of γ , the stronger the assumption.

The γ -margin assumption rules out the pathology from Theorem 3.4: as soon as the width of the “uncertainty region” drops to 2γ , the “true” threshold θ^* is determined.

3.3.3 Perceptron

The Perceptron algorithm of Rosenblatt (1958) is as follows.

Algorithm 3.2: Perceptron for online learning linear threshold functions over $\mathbb{R}^n$.

- Initially, set $w_1 := 0 \in \mathbb{R}^n$ and $\theta_1 := 0 \in \mathbb{R}$.
- In round t :
 - Observe $x_t \in \mathbb{R}^n$.
 - Predict $a_t := \mathbb{1}\{\langle w_t, x_t \rangle > \theta_t\}$.
 - Observe label $y_t \in \{0, 1\}$.
 - Update hypothesis: If $a_t \neq y_t$, then

$$\begin{aligned} w_{t+1} &:= \begin{cases} w_t + x_t & \text{if } y_t = 1, \\ w_t - x_t & \text{if } y_t = 0; \end{cases} \\ \theta_{t+1} &:= \begin{cases} \theta_t - 1 & \text{if } y_t = 1, \\ \theta_t + 1 & \text{if } y_t = 0. \end{cases} \end{aligned}$$

If $a_t = y_t$, then $w_{t+1} := w_t$ and $\theta_{t+1} := \theta_t$.

Intuitively, the update made by Perceptron after a mistake changes the value of the inner product between the weight vector and the input and the threshold to “move in the right direction”. For example, if $y_t = 1$ but $a_t = 0$, then $\langle w_t, x_t \rangle$ was too small or θ_t was too large; so the update causes $\langle w_{t+1}, x_t \rangle$ to be larger than $\langle w_t, x_t \rangle$ and θ_{t+1} to be smaller than θ_t .

It is somewhat simpler to view Perceptron as a learning algorithm for *homogeneous linear threshold functions* over $\mathbb{R}^{n+1}$, denoted by $\text{HLTF}_{\mathbb{R}^{n+1}}$, which are the LTFs over $\mathbb{R}^{n+1}$ with threshold 0. By mapping each input $x \in \mathbb{R}^n$ to $\tilde{x} := (x, -1) \in \mathbb{R}^{n+1}$ and each $(w, \theta) \in \mathbb{R}^n \times \mathbb{R}$ to a weight vector $\tilde{w} := (w, \theta) \in \mathbb{R}^{n+1}$, we see that

$$\langle w, x \rangle - \theta = \langle \tilde{w}, \tilde{x} \rangle.$$

So every LTF over $\mathbb{R}^n$ can be viewed as a homogeneous LTF over $\mathbb{R}^{n+1}$.

It is also somewhat simpler to use $\{-1, 1\}$ -valued labels instead of $\{0, 1\}$ -valued labels, so that the Perceptron update step (when $a_t \neq y_t$) can be written as

$$\tilde{w}_{t+1} := \tilde{w}_t + \tilde{y}_t \tilde{x}_t$$

where $\tilde{y}_t := 1$ if $y_t = 1$ and $\tilde{y}_t := -1$ if $y_t = 0$.

Under the γ -margin assumption (for $\text{LTF}_{\mathbb{R}^n}$), and using $\{-1, 1\}$ -valued labels, we have

- $\|\tilde{x}_t\| \leq \sqrt{2}$ for all $t \in \mathbb{N}$; and
- there exists $\tilde{w}^* \in \mathbb{R}^{n+1}$ with $\|\tilde{w}^*\| \leq \sqrt{2}$ such that, for all $t \in \mathbb{N}$,

$$\tilde{y}_t \langle \tilde{w}^*, \tilde{x}_t \rangle \geq \gamma.$$

We call the above two bullets together the homogeneous γ -margin assumption (in $\mathbb{R}^{n+1}$, although the dimension of the space does not explicitly play a role in the following theorem).

Theorem 3.5 (Novikoff, 1962). *In the online concept learning scenario with the homogeneous γ -margin assumption, Perceptron makes at most $4/\gamma^2$ mistakes.*

Note that if the norm bounds in the homogeneous γ -margin assumption are changed to $\|\tilde{x}_t\| \leq X$ and $\|\tilde{w}^*\| \leq W$, then the mistake bound in Theorem 3.5 changes to $X^2 W^2 / \gamma^2$. Also, Theorem 3.5 does not only apply in cases where the $\tilde{x}_t$ and $\tilde{w}^*$ are derived from the transformations described above; rather, the $\tilde{x}_t$ and $\tilde{w}^*$ can be any vectors that satisfy the conditions in the two bullets.

Proof of Theorem 3.5. At the start of round $t = 1$, we have $\langle \tilde{w}^*, \tilde{w}_1 \rangle = 0$ and $\|\tilde{w}_1\|^2 = 0$. Suppose Perceptron makes a mistake in round t . Using the sign-valued labels, a mistake in round t implies

$$\tilde{y}_t \langle \tilde{w}_t, \tilde{x}_t \rangle \leq 0. \tag{3.4}$$

We now consider the effect of the update on $\langle \tilde{w}^*, \tilde{w}_{t+1} \rangle$ and $\|\tilde{w}_{t+1}\|^2$ relative to $\langle \tilde{w}^*, \tilde{w}_t \rangle$ and $\|\tilde{w}_t\|^2$. First, we have

$$\begin{aligned} \langle \tilde{w}^*, \tilde{w}_{t+1} \rangle &= \langle \tilde{w}^*, \tilde{w}_t + \tilde{y}_t \tilde{x}_t \rangle \\ &= \langle \tilde{w}^*, \tilde{w}_t \rangle + \tilde{y}_t \langle \tilde{w}^*, \tilde{x}_t \rangle \\ &\geq \langle \tilde{w}^*, \tilde{w}_t \rangle + \gamma \end{aligned}$$

where the inequality uses the γ -margin assumption. Next, by “expanding the square”, we have

$$\begin{aligned} \|\tilde{w}_{t+1}\|^2 &= \|\tilde{w}_t + \tilde{y}_t \tilde{x}_t\|^2 \\ &= \|\tilde{w}_t\|^2 + 2\tilde{y}_t \langle \tilde{w}_t, \tilde{x}_t \rangle + \|\tilde{y}_t \tilde{x}_t\|^2 \\ &\leq \|\tilde{w}_t\|^2 + \|\tilde{y}_t \tilde{x}_t\|^2 \\ &\leq \|\tilde{w}_t\|^2 + 2 \end{aligned}$$

where the first inequality uses (3.4), and the second inequality uses the assumption $\|\tilde{x}_t\| \leq 1$ (which implies $\|\tilde{x}_t\|^2 \leq 2$).

Therefore, if there have been M_t mistakes after t rounds, then

$$\langle \tilde{w}^*, \tilde{w}_{t+1} \rangle \geq M_t \gamma \quad \text{and} \quad \|\tilde{w}_{t+1}\|^2 \leq 2M_t.$$

By Cauchy-Schwarz and the assumption that $\|\tilde{w}^*\| \leq \sqrt{2}$,

$$\langle \tilde{w}^*, \tilde{w}_{t+1} \rangle \leq \|\tilde{w}^*\| \|\tilde{w}_{t+1}\| \leq \sqrt{2} \|\tilde{w}_{t+1}\|.$$

Combining these inequalities gives $M_t \leq 4/\gamma^2$. $\square$

Exercise 3.4. A new university president has assembled a committee of N experts to help her make yes/no decisions every day; for every decision she must make, she receives yes/no recommendations from all experts. The committee contains a special subcommittee of s experts (for some odd number $s \in [N]$) that is “good”, meaning that the majority vote among this subcommittee always gives the correct yes/no decision. The president unfortunately does not know the identity or membership of this special subcommittee. Explain how the president can use Perceptron to make decisions—in roughly $O(N)$ time per decision—and what is the resulting mistake bound? Could she do better if she had more time per decision? $\diamond$

Exercise 3.5. Consider the online concept learning scenario with the MCONJ_n -realizability assumption. Suppose Perceptron is used with the following embedding of $\{0, 1\}^n$ into $\mathbb{R}^n$:

$$\{0, 1\}^n \ni (x_1, \dots, x_n) \mapsto \frac{1}{\sqrt{n}}(2x_1 - 1, \dots, 2x_n - 1) \in \mathbb{R}^n.$$

Note that the embedding maps every element of $\{0, 1\}^n$ into $\{x \in \mathbb{R}^n \mid \|x\| = 1\}$, the unit sphere of $\mathbb{R}^n$. What mistake bound (solely as a function of n) is implied by Theorem 3.5? $\diamond$

A different algorithm called Winnow, due to Littlestone (1988), learns LTFs over $\mathbb{R}^n$ under a different kind of margin assumption (essentially replacing the Euclidean norms on inputs and weight vectors by ℓ^∞ - and ℓ^1 -norms).

3.4 Optimal mistake bounds

3.4.1 Mistake trees

The strategies we used to prove the mistake lower bounds in Theorem 1.3⁴ and Theorem 3.4 share a common structure. We construct a rooted perfect binary tree of some depth d with each of the 2^d leaf nodes corresponding to some concept $c \in \mathcal{C}$, and each non-leaf node annotated with some input $x \in \mathcal{X}$. (The 2^d concepts will need to be distinct, but the inputs used to annotate the non-leaf nodes need not be distinct.) Each edge is annotated with either 0 or 1 so that each binary sequence in $\{0, 1\}^d$ appears along the edges of some root-to-leaf path. If the root-to-leaf path ending at $c \in \mathcal{C}$ is annotated by the binary sequence $(y_1, \dots, y_d) \in \{0, 1\}^d$ (on the edges) and the input sequence $(x_1, \dots, x_d)$ (on the non-leaf nodes), then it must be that $c(x_i) = y_i$ for all $i \in [d]$. A tree with this property is called a *(complete) mistake tree* for $\mathcal{C}$. See Figure 3.1 for an example.

The following theorem is immediate from the definition of a mistake tree.

Theorem 3.6 (Littlestone, 1988). *Suppose $\mathcal{C}$ has a mistake tree of depth d . Then in the online concept learning scenario with concept class $\mathcal{C}$, for each learning algorithm, there is a sequence of d input/label-pairs consistent with some $c^* \in \mathcal{C}$ on which the learner makes d mistakes.*

⁴Recall from Section 3.1.2 that the expert advice scenario can be simulated in the online concept learning scenario.

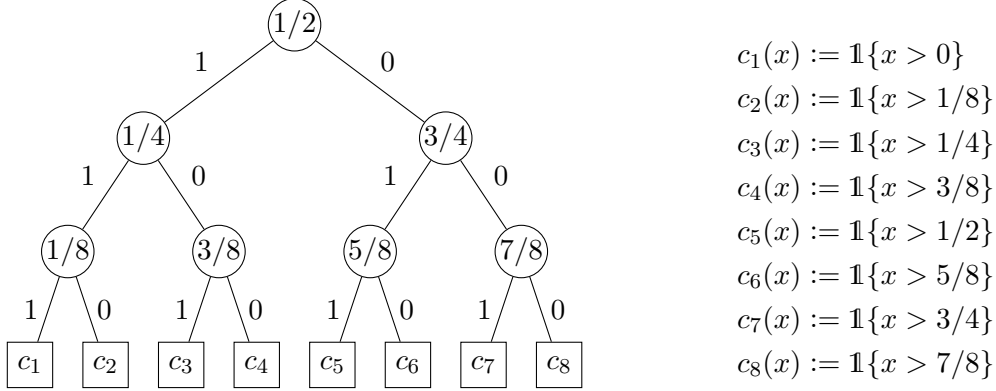


Figure 3.1: A depth $d = 3$ mistake tree for $\text{LTF}_{\mathbb{R}^1}$.

Exercise 3.6. Recall the EXPERTS_N concept class. Let $N = 8$, and construct a mistake tree for EXPERTS_N of depth 3. Generalize the construction to any $N \in \mathbb{N}$ that is a power-of-two. $\diamond$

Exercise 3.7. Let $n = 3$, and construct a mistake tree for MCONJ_n of depth 3. Generalize the construction to any $n \in \mathbb{N}$. $\diamond$

3.4.2 Littlestone dimension

Theorem 3.6 shows that the existence of deep mistake trees puts lower bounds on the mistake bounds achievable by any learning algorithm. It turns out that there is an algorithm that can use the non-existence of a deep mistake tree to guarantee a mistake bound.

Define the *Littlestone dimension* of a non-empty concept class $\mathcal{C}$, denoted $\text{Ldim}(\mathcal{C})$, to be the largest integer d such that there exists a mistake tree for $\mathcal{C}$ of depth d . If mistake trees for $\mathcal{C}$ of arbitrarily large d exist, then we say $\text{Ldim}(\mathcal{C}) = \infty$. For convenience, it is also useful to define $\text{Ldim}(\emptyset) = -1$.

The following facts about Ldim are easy to show.

- Ldim is monotone:

$$\mathcal{C}' \subseteq \mathcal{C} \implies \text{Ldim}(\mathcal{C}') \leq \text{Ldim}(\mathcal{C}).$$

- Only singleton sets have zero Ldim :

$$\text{Ldim}(\mathcal{C}) = 0 \iff |\mathcal{C}| = 1.$$

- The construction in Figure 3.1 shows that $\text{Ldim}(\text{LTF}_{\mathbb{R}^1}) \geq 3$. In fact, the proof of Theorem 3.4 shows that $\text{Ldim}(\text{LTF}_{\mathbb{R}^1}) = \infty$.
- If there is a learning algorithm for the online concept learning scenario with concept class $\mathcal{C}$ and the realizability assumption with mistake bound M , then $\text{Ldim}(\mathcal{C}) \leq M$. For example, we have the following:
 - $\text{Ldim}(\mathcal{C}) \leq \log_2(|\mathcal{C}|)$ whenever $\mathcal{C}$ is finite (via Halving).
 - $\text{Ldim}(\text{CONJ}_n) \leq n + 1$ (via Attribute Elimination; see Exercise 3.3).

Exercise 3.8. Prove the facts about Ldim in the bullets above. $\diamond$

Littlestone (1988) showed that the following algorithm, called the *Standard Optimal Algorithm* (SOA), guarantees a mistake bound of $\text{Ldim}(\mathcal{C})$, which is optimal by Theorem 3.6.

Algorithm 3.3: Standard Optimal Algorithm for the online concept learning scenario with concept class $\mathcal{C}$ and the realizability assumption.

- Initially, set $V_1 := \mathcal{C}$.
- In round t :
 - Observe $x_t \in \mathcal{X}$.
 - Let $V_t^y := \{c \in V_t \mid c(x_t) = y\}$ for each $y \in \{0, 1\}$.
 - Predict $a_t := \mathbb{1}\{\text{Ldim}(V_t^1) > \text{Ldim}(V_t^0)\}$.
 - Observe label $y_t \in \{0, 1\}$.
 - Update version space: $V_{t+1} := V_t^{y_t}$.

SOA maintains a *version space* based on input/label-pairs seen so far, i.e., $V_t \subseteq \mathcal{C}$ is the subset of concepts that are consistent with the first $t - 1$ input/label-pairs. (To borrow a term from FTL and Halving, these are the concepts that are still “alive” at the start of round t .) In round t , the version space V_t is partitioned into two sets, V_t^0 and V_t^1 , depending on how the concepts classify x_t . Unlike Halving, which just considers the cardinalities of the two sets to make its own prediction, SOA considers the Ldim of these two sets and follows the set with larger Ldim .

Theorem 3.7 (Littlestone, 1988). *In the online concept learning scenario with the $\mathcal{C}$ -realizability assumption, SOA makes at most $\text{Ldim}(\mathcal{C})$ mistakes.*

Proof. We may assume without loss of generality that $\text{Ldim}(\mathcal{C}) < \infty$. As mentioned above, for all $t \in \mathbb{N}$, we have that $V_t = \{c \in \mathcal{C} \mid c(x_i) = y_i \ \forall i \in \{1, \dots, t-1\}\}$ is the version space at the start of round t . With the realizability assumption, V_t can never become empty, and hence $\text{Ldim}(V_t) \geq 0$ for all $t \in \mathbb{N}$. If $\text{Ldim}(V_t) = 0$, then V_t is a singleton, and the definition of Ldim and the realizability assumption ensure that SOA does not make a mistake.

Suppose SOA makes a mistake in round t . We claim that the updated version space, V_{t+1} , after the mistake satisfies

$$\text{Ldim}(V_{t+1}) \leq \text{Ldim}(V_t) - 1.$$

To show this, assume for sake of contradiction that $\text{Ldim}(V_{t+1}) = \text{Ldim}(V_t) =: d_t$. Since $V_{t+1} = V_t^{y_t}$, and by the monotonicity of Ldim and the choice of a_t ,

$$\text{Ldim}(V_t) \geq \text{Ldim}(V_t^{a_t}) \geq \text{Ldim}(V_t^{y_t}),$$

it must be that the above inequalities actually hold with equality; in particular,

$$\text{Ldim}(V_t^0) = \text{Ldim}(V_t^1).$$

We construct a mistake tree for V_t of depth $d_t + 1$ as follows: take depth d_t mistake trees for V_t^0 and V_t^1 , and let their roots be the children of a new root node labeled by x_t . This is a mistake tree for

V_t and it has depth $d_t + 1$, which is a contradiction of the fact that $\text{Ldim}(V_t) = d_t$. So we conclude that $\text{Ldim}(V_{t+1}) \leq \text{Ldim}(V_t) - 1$, i.e., every mistake leads to a reduction in Ldim by at least 1.

Since $\text{Ldim}(V_1) = \text{Ldim}(\mathcal{C})$ and $\text{Ldim}(V_t) \geq 0$ for all $t \in \mathbb{N}$, it must be that the number of mistakes is at most $\text{Ldim}(\mathcal{C})$. $\square$

Exercise 3.9. Let $K_n = (V, E)$ denote the complete graph on $n \geq 4$ vertices.

- (a) Let $\mathcal{C}_{K_n} := \{c_v \mid v \in V\}$ be the concept class on the domain E , where for each $v \in V$,

$$c_v(e) := \mathbf{1}\{v \in e\} \quad \forall e \in E.$$

Determine the value of $\text{Ldim}(\mathcal{C}_{K_n})$ by exhibiting a mistake tree of the requisite depth and also giving a learning algorithm with the requisite mistake bound.

- (b) If $\mathcal{C}$ is a concept class on the domain $\mathcal{X}$, then its dual class $\mathcal{C}^* := \{\mathbf{e}_x \mid x \in \mathcal{X}\}$ is a concept class on the domain $\mathcal{C}$, where for each $\mathbf{e}_x \in \mathcal{C}^*$ (called the point evaluation function for x),

$$\mathbf{e}_x(c) := c(x) \quad \forall c \in \mathcal{C}.$$

Determine the value of $\text{Ldim}(\mathcal{C}_{K_n}^*)$ (as above).

- (c) Recall the EXPERTS_N concept class on the domain $\{0, 1\}^N$. Describe the dual class EXPERTS_N^* , and determine the value of $\text{Ldim}(\text{EXPERTS}_N^*)$ (as above). $\diamond$