Machine Learning Theory (CSC 431/531) - Lecture 2

Nishant Mehta

1 Realizable case and the Mistake Bound Model

We begin our course with classification under the strong assumption of *realizability*: there exists some boolean function $c \in \mathcal{C}$ which determines the labels, so that labeled examples always take the form $(x, c(x))$. Here, $\mathcal{C}$ is the *concept class*, a set of boolean functions. Learner knows $\mathcal{C}$ but of course does not know c. The reason for using the notation $\mathcal{C}$ rather than $\mathcal{F}$ is to allow for the possibility that Learner predicts according to a different set of hypotheses than those in $\mathcal{C}$.

We first study this setting under the online learning protocol. Consider an arbitrary sequence of examples, where the length of the sequence also can be arbitrarily large. Since we are in the realizable case, there exists a perfect hypothesis $c \in \mathcal{C}$. Suppose that Learner takes $\mathcal{F} = \mathcal{C}$, eventually plays hypothesis c, and predicts according to it thereafter. Then the total number of mistakes made by Learner is simply the number of mistakes made by Learner prior to playing hypothesis c. Thus, it is natural to try to bound the total number of mistakes made by Learner.

Definition 1. An algorithm $\mathcal{A}$ learns a class $\mathcal{C}$ in the *mistake bound model* if there there is a constant $M < \infty$ such that, for any $c \in \mathcal{C}$ and any sequence of examples $(x_1, c(x_1)), \ldots, (x_T, c(x_T))$ of any length T, the total number of mistakes made by $\mathcal{A}$ on the sequence is at most M.

Two remarks are in order. First, for many concept classes there is a natural way to define the concept class for each choice of the dimension d of the input. We then would typically want a mistake bound M that grows only polynomially in d. Second, we typically also care about *efficient learnability*. That is, we would like the runtime of the algorithm to be at most polynomial in d for each round t.

1.1 Learning monotone conjunctions

Let the input space be $\{0,1\}^d$ and the label space $\mathcal{Y}$ be $\{0,1\}$. Consider the concept class $\mathcal{C}$ consisting of the set of all monotone conjunctions. That is, each element of $\mathcal{C}$ is of the form $x_{i_1} x_{i_2} \cdots x_{i_k}$ for some $k \in \{0, 1, \ldots, d\}$; if $k = 0$, then the predicted label is always positive.

There is a simple algorithm for learning monotone conjunctions in the mistake bound model:

1. Initialize hypothesis f to the conjunction $f(x) = x_1 x_2 \cdots x_d$.
2. While there are still examples in the sequence:

   Predict the label of the next example using f. If the true label is 1 but the predicted label was 0, update f by removing from the conjunction all components x_j that are zero in the example.

The idea behind the above algorithm is to begin with the most restrictive hypothesis which labels everything but the all ones vector as negative. Thus, mistakes can occur only on positive examples. In the event that a mistake occurs on a positive example, we are guaranteed that any component set to zero in the example cannot be part of the correct conjunction c, and we may thus remove such components. The algorithm thus only removes terms from the conjunction in f which are inconsistent with the data observed thus far. Moreover, each mistake leads to the removal of at least one term from the conjunction, and so there can be at most d mistakes. Therefore, the above algorithm learns the class of monotone conjunctions in the mistake bound model and makes at most d mistakes.

2 Halving algorithm

When the concept class is finite, there is a surprisingly simple algorithm that obtains a mistake bound of $\log_2 |\mathcal{C}|$. This algorithm is called the Halving algorithm, and it uses two key ideas.

The first idea is that of a *version space*. The version space is the set of hypotheses that are consistent with the data observed thus far. Thus, at the start of round t, the version space $\mathcal{V}_t$ is the subset of hypotheses from $\mathcal{C}$ which are consistent with $(x_1, y_1), \ldots, (x_{t-1}, y_{t-1})$.

The second idea is to predict according to a majority vote. For a set of hypotheses $\mathcal{F}$, define the majority vote based on $\mathcal{F}$ as

$$\textsc{mv}_{\mathcal{F}}(x) = \begin{cases} 1 & \text{if } |\{f \in \mathcal{F} : f(x) = 1\}| \geq |\mathcal{F}|/2; \\ 0 & \text{otherwise.} \end{cases}$$

The Halving algorithm simply predicts according to the majority vote with respect to the version space in every round.

Algorithm 1: HALVING ALGORITHM

```
$\mathcal{V}_1 \leftarrow \mathcal{C}$
for $t = 1 \to T$ do
    Observe $x_t$
    $f_t \leftarrow \textsc{mv}_{\mathcal{V}_t}$ (and predict $\hat{y}_t = \textsc{mv}_{\mathcal{V}_t}(x_t)$)
    Observe true label $y_t = c(x_t)$
    Set $V_{t+1} \leftarrow \{f \in V_t : f(x_t) = y_t\}$
end
```

How many mistakes does this algorithm make? Because it predicts according to the majority vote, wherever the algorithm makes a mistake it is guaranteed that at least half the hypotheses in the version space were wrong; thus, the version space is halved on each mistake. Formally, if the algorithm makes a mistake in round t, it holds that $|\mathcal{V}_{t+1}| \leq |\mathcal{V}_t|/2$. We initially have $\mathcal{V}_1 = \mathcal{C}$, and so if we have made M_t mistakes at the beginning of round t, it follows that $|V_t| \leq |\mathcal{C}|/2^{M_t}$. Since there exists a perfect hypothesis $c \in \mathcal{C}$, the algorithm can make at most $\log_2 |\mathcal{C}|$ mistakes.

We have just shown that any finite concept class is learnable in the mistake bound model using the Halving algorithm.

Theorem 1. *The Halving algorithm learns any finite concept class $\mathcal{C}$ in the mistake bound model and makes at most $\log_2 |\mathcal{C}|$ mistakes.*

Unfortunately, the runtime of the Halving algorithm is linear in $|\mathcal{C}|$, which can be exorbitant. Why is this bad? In many situations, the size of the concept class $|\mathcal{C}|$ can be exponential in the dimension of the data, in which case the runtime of the Halving algorithm is *exponential* in d (!). For instance, the class of monotone conjunctions has cardinality 2^d. In other cases, such as the case of linear separators, the concept class can even be infinite.

3 Learning linear separators (with margin) in the mistake bound model: Perceptron

We next consider the problem of linear classification in the realizable case. Specifically, we will look at the subclass of linear classifiers known as *homogenous linear separators.*

Let $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{-1, +1\}$. The concept class of homogenous linear separators is defined as $\mathcal{C} = \{f_w : w \in \mathbb{R}^d\}$, for hypotheses $f_w(x) = \operatorname{sgn}(\langle w, x\rangle)$. Here, sgn is the sign function, defined[1] as the map

$$\operatorname{sgn}(z) = \begin{cases} +1 & \text{if } z \geq 0; \\ -1 & \text{if } z < 0. \end{cases}$$

The reason these linear separators are called homogenous is because the concept class lacks a bias term; consequently, the linear separator corresponding to a vector w is the hyperplane normal to w that passes through the origin, i.e. $\{x \in \mathbb{R}^d : \langle w, x\rangle = 0\}$. If we also allowed for a bias term $b \in \mathbb{R}$, the class would be upgraded to the set of non-homogenous linear separators (which do not necessarily pass through the origin); this class also is commonly referred to as *halfspaces*, and each hypothesis is of the form $x \mapsto \operatorname{sgn}(\langle w, x\rangle + b)$.

As before, we make the realizability assumption with respect to concept class $\mathcal{C}$. The sequence of examples is thus linearly separable, meaning that there exists a vector $w^* \in \mathbb{R}^d$ for which

$$y_t = \operatorname{sgn}(\langle w^*, x_t\rangle \qquad \text{for all } t \in [T].$$

Since the output of any classifier f_w is invariant to scaling of w, without loss of generality we assume that w^* has unit ℓ_2 norm, i.e., $\|w^*\|_2 = \left(\sum_{j=1}^d (w_j^*)^2\right)^{1/2} = 1$.

At this stage, we lack the tools to provide a mistake bound for learning the class of homogenous linear separators. Were the concept class finite, we could use our mistake bound for the Halving algorithm, but alas, the concept class is not even countable. However, with one additional assumption, we will be able to use an elegant algorithm called the Perceptron algorithm to obtain a finite mistake bound.

Assuming separability with margin. We further assume that the positive and negative examples are linearly separable by some positive margin. To make this precise, for any $w \in \mathbb{R}^d$, define the *margin with respect* $w \in \mathbb{R}^d$ (and the sequence of examples) to be

$$\gamma_w = \min_{t \in [T]} \frac{y_t \langle w, x_t\rangle}{\|w\|_2}.$$

If f_w correctly classifies (x_t, y_t), it is not hard to see (try drawing a diagram) that γ_w is equal to the Euclidean distance from x_t to the hyperplane $\{x \in \mathbb{R}^d : \langle w, x\rangle = 0\}$.

The *margin* γ is then

$$\gamma = \gamma_{w^*} = \min_{t \in [T]} y_t \langle w^*, x_t\rangle.$$

We now formalize our assumption of separability with margin:

[1] The sign function usually maps zero to zero, but we map zero to one so that our classifiers take values in $\{-1, +1\}$.

Assumption 1. *There exists a unit vector $w^* \in \mathbb{R}^d$ for which*

$$\gamma = \min_{t \in [T]} y_t \langle w^*, x_t \rangle > 0.$$

We now consider an efficient algorithm for learning linear separators with a finite number of mistakes, under the assumption that the data is separable with margin γ. This algorithm is the *Perceptron* algorithm.

Algorithm 2: PERCEPTRON

```
$w_0 \leftarrow \mathbf{0}$
$m = 0$
for $t = 1 \to T$ do
    Observe $x_t$
    Predict $\hat{y}_t = \text{sgn}(\langle w_m, x_t \rangle)$
    Observe true label $y_t = c(x_t)$
    if $\hat{y}_t \neq y_t$ then
        $w_{m+1} \leftarrow w_m + y_t x_t$
        $m \leftarrow m + 1$
    end
end
```

Define $R := \max_{t \in [T]} \|x_t\|_2$; this is the radius of the smallest Euclidean ball (centered at the origin) that contains the data.

Theorem 2. *Let $(x_1, y_1), \ldots, (x_T, y_T)$ be a sequence of examples that is linearly separable with margin $\gamma > 0$. Then on this sequence the Perceptron algorithm makes at most $\frac{R^2}{\gamma^2}$ mistakes.*

Remarkably, this mistake bound is *independent* of the dimension d. Before showing the proof, let's build some intuition for Perceptron's update rule in the case of a mistake. Suppose that Perceptron makes its m^{th} mistake on a positively labeled example $(x, 1)$, resulting in the update $w_{m+1} = w_m + x$. Then $\langle w_{m+1}, x \rangle = \langle w_m, x \rangle + \|x\|^2 > \langle w_m, x \rangle$, and so the new hypothesis is closer to classifying x as positive. This same intuition works for mistakes on negative examples as well.

Proof. The proof is based on two claims.

The first claim is that the norm of w_m is never too big:

$$\|w_m\| \le R\sqrt{m}. \tag{1}$$

Let's prove this claim. For $j \ge 1$, let $(\tilde{x}_j, \tilde{y}_j)$ denote the j^{th} example on which Perceptron made a mistake. It therefore holds that

$$w_m = w_{m-1} + \tilde{y}_m\, \tilde{x}_m.$$

Then

$$\begin{aligned} \|w_m\|^2 &= \|w_{m-1} + \tilde{y}_m\, \tilde{x}_m\|^2 \\ &= \|w_{m-1}\|^2 + \|\tilde{x}_m\|^2 + 2\tilde{y}_m \langle w_{m-1}, \tilde{x}_m \rangle \\ &\le \|w_{m-1}\|^2 + R^2, \end{aligned}$$

where the inequality follows because $\tilde{y}_m \langle w_{m-1}, \tilde{x}_m \rangle \le 0$ since w_{m-1} made a mistake on $(\tilde{x}_m, \tilde{y}_m)$. Repeating this argument all the way back to $w_0 = \mathbf{0}$ yields the claim.

The second claim is that the inner product $\langle w_m, w^* \rangle$ grows quickly with m:

$$\langle w_m, w^* \rangle \ge \gamma \cdot m. \tag{2}$$

To see this, observe that

$$\begin{aligned} \langle w^*, w_m \rangle &= \langle w^*, w_{m-1} + \tilde{y}_m\, \tilde{x}_m \rangle \\ &= \langle w^*, w_{m-1} \rangle + \tilde{y}_m \langle w^*, \tilde{x}_m \rangle \\ &\ge \langle w^*, w_{m-1} \rangle + \gamma. \end{aligned}$$

Applying this argument recursively yields $\langle w^*, w_m \rangle \ge \gamma \cdot m$, proving the claim.

Now, the inner product $\langle w_m, w^* \rangle$ grows *at most* linearly in $\|w_m\|$ (from the Cauchy-Schwarz inequality), which itself grows no faster than the square root of m (from the first claim). Consequently, $\langle w_m, w^* \rangle = O(\sqrt{m})$. On the other hand, this inner product also grows *at least* linearly in m (from the second claim), and so it must be the case that m is bounded as otherwise we would arrive at a contradiction. Indeed, applying Cauchy-Schwarz to (2) and using (1) yields

$$\gamma \cdot m \overset{(2)}{\le} \langle w^*, w_m \rangle \le \|w^*\| \cdot \|w_m\| = \|w_m\| \overset{(1)}{\le} R\sqrt{m},$$

and so $m \le \frac{R^2}{\gamma^2}$. □