CS 7545: Machine Learning Theory **Fall 2021**

Lecture 8: Supervised Learning

Instructor: Santosh Vempala *Lecture date: 10/04-06-13/2021*

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

In supervised learning, the input is a set of labelled examples $(x^1, \ell(x^1)), (x^2, \ell(x^2)), \ldots, (x^n, \ell(x^n))$ where $x \in \Omega$, and $\ell : \Omega \to L$ is an unknown labelling function. The goal is to learn ℓ.

A supervised learning algorithm outputs a function $h : \Omega \to L$.

- How close is h to ℓ?
- What is the Sample complexity?
- What is the Time complexity of the algorithm?

Example 8.1 (Conjunctions)**.** *For $\Omega = \{0,1\}^n$ and $L = \{0,1\}$, the number of possible labellings in 2^{2^n}. The number of conjunctions on Ω is 3^n.*

8.1 PAC Learning

In the PAC Learning model, we assume that

- the data is i.i.d. from an unknown distribution $\mathcal{D}$, and
- there is an unknown labelling function $h \in \mathcal{H}$. (Correct hypothesis class is known)

The goal is to output a hypothesis $g \in \mathcal{H}$ which is close to h.

Definition 8.2. *An algorithm is an (ϵ, δ)-PAC learning algorithm if with probability at least $1-\delta$, the output hypothesis correctly classifies at least $1-\epsilon$ of $\mathcal{D}$, i.e.,*

$$\Pr_{x \sim \mathcal{D}}(g(x) \neq h(x)) \leq \epsilon.$$

An algorithm is efficient if sample complexity/time complexity is bounded by $\text{poly}\big(\log |\mathcal{H}|, \frac{1}{\epsilon}, \log \frac{1}{\delta}\big)$. In PAC model, we can bound the number of samples needed so that our algorithms can "generalize" well for future examples. Assume $|\mathcal{H}|$ is finite. Given m samples, suppose a function $g \in \mathcal{H}$ is not ϵ-accurate with respect to $\mathcal{D}$, i.e., $\Pr_{\mathcal{D}}(g(x) \neq y) > \epsilon$, then

$$\Pr(g \text{ is correct on } m \text{ samples}) \leq (1-\epsilon)^m.$$

So, the probability that there exists a hypothesis g such that g is not ϵ-accurate but correctly classifies m examples is

$$\Pr(\exists g \in \mathcal{H} \text{ s.t. } g \text{ is not } \epsilon\text{-accurate and correct on } m \text{ examples}) \leq |\mathcal{H}|(1-\epsilon)^m.$$

If $m > \frac{1}{\epsilon} \log \frac{|\mathcal{H}|}{\delta}$, this probability is at most δ.

Example 8.3 (Conjunctions)**.** *For the case of conjunctions, $m = O(\frac{n}{\epsilon} \log \frac{1}{\delta})$ samples suffice.*

An algorithm to find a conjunction which is consistent with the examples is as follows. Start with a set of all possible literals. On seeing a positive example, for each variable x_i, remove x_i from the set if $x_i = 0$ and remove $\bar{x}_i$ otherwise. For every positive example, at least one literal corresponding to every variable is removed.

Example 8.4 (Decision Lists)**.** *Let $\Omega = \{0,1\}^n$ and $L = \{0,1\}$. A decision list is defined as*

```
if $x_1 = b_1$ then
    output $o_1$
else if $x_2 = b_2$ then
    output $o_2$
    ⋮
else if $x_n = b_n$ then
    output $o_n$
else 0/1,
```

where $b_i, o_i \in \{0,1\}$. The number of decision lists on n variables is $|\mathcal{H}| \le 4^n n!$. So, $m = O(\frac{n}{\epsilon}\log\frac{1}{\delta})$ examples suffice.

An algorithm to find a consistent decision list is as follows. In each iteration, find a rule consistent with all the examples. Eliminate those examples that fire the rule. Continue this process until there is no examples left. Note that in each iteration at least one example will be eliminated, so the algorithm terminates in at most m steps.

8.2 Mistake Bound Model

In this model, a sequence of examples (x^t, y^t) is revealed as follows. For each t,

- x^t is revealed
- Algorithm predicts label z
- True label y^t is revealed
- If $z \neq y^t$, mistake++

The goal is to bound the maximum number of mistakes over all possible sequence of data.

A simple (although inefficient) algorithm is to predict the majority of the predictions of all consistent hypotheses. Whenever this majority algorithm makes a mistake, it can eliminate at least half of the remaining hypotheses, so it will make at most $\log(|\mathcal{H}|)$ mistakes.

8.3 PAC/Mistake Bound Learning of Halfspaces

A halfspace or a Linear Threshold Function with parameters $w \in \mathbb{R}^d$, $b \in \mathbb{R}$ is defined as:

$$H(w,b) = \{x \in \Omega : w^\top x \ge b\}.$$

Some hypothesis classes that can be encoded as halfspaces:

- Conjunctions: $x \in \{0,1\}^d$, $\ell(x) = x_{i_1} \wedge \bar{x}_{i_2} \ldots \wedge x_{i_k}$

$$\ell(x) = 1 \Leftrightarrow x_{i_1} + (1 - x_{i_2}) + \ldots + x_{i_k} \ge k.$$

- Disjunctions: $x \in \{0,1\}^d$, $\ell(x) = x_{i_1} \vee \bar{x}_{i_2} \ldots \vee x_{i_k}$

$$\ell(x) = 1 \Leftrightarrow x_{i_1} + (1 - x_{i_2}) + \ldots + x_{i_k} \ge 1.$$

- Decision List: $x \in \{0,1\}^d$,

```
if $x_1 = 1$ then
   output 1
else if $x_2 = 0$ then
   output 0
else 1
```

$$\ell(x) = 1 \Leftrightarrow 4x_1 - 2(1-x_2) + 1 \geq 1.$$

To use the sample complexity bound from PAC-learning, we need to bound the number of distinct halfspaces in $\mathbb{R}^d$ when $\Omega = \{0,1\}^d$. Any halfspace in $\mathbb{R}^d$ is defined by d linearly independent vectors. When $\Omega = \{0,1\}^d$, $|\mathcal{H}| \leq \binom{2^d}{d} \leq 2^{d^2}$. Without loss of generality, we can assume that $\|w\| = 1$, and $\|x\| \leq 1$, and $b = 0$, as

$$w^\top x - b \geq 0 \Leftrightarrow \begin{bmatrix} w & b \end{bmatrix}^\top \begin{bmatrix} x \\ -1 \end{bmatrix} \geq 0.$$

Algorithm 1 Perceptron

```
Initialize $w^0 \leftarrow 0$
for $t = 1, \ldots, T$
    if $\ell(x^t) \neq \text{sign}(w^t \cdot x^t)$
        $w^{t+1} \leftarrow w^t + \ell(x^t)x^t$
    else
        $w^{t+1} \leftarrow w^t$
```

Lemma 8.5. *Given data classifiable by halfspace w^*, with $\|w^*\| = 1$ and margin $\gamma = \min_x |(w^* \cdot x)|$, the number of mistakes made by Perceptron is at most $1/\gamma^2$.*

Proof. Let $\phi_t = \frac{w^* \cdot w^t}{\|w^t\|} = cos(\theta(w^*, w^t))$ and let m_t denote the number of mistakes made by the algorithm before step t. Then, $m_1 = 0$, $\phi_0 = 0$, and $\phi_t \leq 1$ for all t. Note that if $w^t \cdot x^t \neq \ell(x^t)$, then $w^{t+1} = w^t + \ell(x^t)x^t$. Since $\ell(x^t)$ and $\langle w^*, x^t \rangle$ have the same sign,

$$w^* \cdot w^{t+1} = w^* \cdot w^t + \ell(x^t)(w^* \cdot x^t) \geq m_{t+1}\gamma. \tag{8.1}$$

If the algorithm makes a mistake at step t, $\ell(x^{t-1})w^{t-1} \cdot x^{t-1} < 0$, therefore,

$$\left\|w^{t+1}\right\|^2 = \left\|w^t\right\|^2 + \left\|x^t\right\|^2 + 2\ell(x^t)(w^t \cdot x^t) \leq \left\|w^t\right\|^2 + 1 \leq m_{t+1}. \tag{8.2}$$

From (**??**) and (**??**), for all $t \geq 1$,

$$1 \geq \phi_t = \frac{w^* \cdot w^t}{\|w^t\|} \geq \frac{\gamma m_t}{\sqrt{m_t}} \Rightarrow m_t \leq \frac{1}{\gamma^2}.$$

□

More generally, the number of mistakes is bounded by $\frac{\|w^*\|_2^2 \|x\|_2^2}{\gamma^2}$.

8.3.1 Kernels

Definition 8.6 (Kernel)**.** *$K : \Omega \times \Omega \to \mathbb{R}$ is a legal kernel if $\exists \phi : \mathbb{R}^d \to \mathbb{R}^m$ such that $K(x, y) = \langle \phi(x), \phi(y) \rangle$.*

A Gram matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ with respect to $\{x_1, \ldots, x_n\}$, where $\mathbf{K}_{ij} = K(x_i, x_j)$, must be positive semi-definite. Some examples of legal kernels are:

- $K(x, y) = \langle x, y \rangle^d$
- $K(x, y) = (1 + \langle x, y \rangle)^d$
- $K(x, y) = e^{-\|x-y\|^2}$

Suppose no halfspace matches $\ell(x)$ but there exists a map $\phi : \mathbb{R}^d \to \mathbb{R}^m$ such that $\ell(x)$ can be classified by a halfspace on $\phi(x)$. If we know ϕ, then apply perceptron to $\phi(x)$. However, if we have access to only $K(x, y)$ or ϕ is in a very high (or infinite) dimension, we can still use the Perceptron algorithm.

Let M_t be the set examples on which Perceptron makes mistakes before step t. The algorithm maintains $w^t = \sum_{i \in M_t} \ell(x^i) x^i$ and for its prediction is the sign of $w^t \cdot x = \sum_{i \in M_t} \ell(x^i)(x \cdot x^i)$.

Rather than explicitly maintaining w^t, Kernel Perceptron maintains the set of all examples on which a mistake was made. Let $w^t = \sum_{i \in M_t} \ell(x^i)\phi(x^i)$. Then, for any $x \in \mathbb{R}^d$,

$$w^t \cdot \phi(x) = \sum_{i \in M_t} \ell(x^i)(\phi(x^i) \cdot \phi(x)) = \sum_{i \in M_t} \ell(x^i) K(x^i, x).$$

So, the prediction on x at step t of Kernel Perceptron will be

$$\text{sign}\left(\sum_{i \in M_t} \ell(x^i) K(x^i, x) \right).$$

8.3.2 Modified Perceptron

Algorithm 2 Modified Perceptron

```
Initialize $w^0 \leftarrow$ random unit vector
for $t = 0, \ldots, T$
    if $\ell(x^t) \neq \text{sign}(w^t \cdot x^t)$ and $|w^t \cdot x^t| > \sigma \|w^t\|$
        $w^{t+1} \leftarrow w^t + \ell(x^t)(w^t \cdot x^t) x^t$
    else
        $w^{t+1} \leftarrow w^t$
```

Theorem 8.7. *Modified Perceptron algoritm updates at most $O(\log n / \sigma^2)$ w.h.p.*

Proof. W.l.o.g, assume all labels are negative (because we can negate the feature part). With probability at least 1/8, $w^* \cdot w^0 \geq 1/\sqrt{n}$. If $\ell(x^t) \neq \text{sign}(\langle w^t, x^t \rangle)$ and $|\langle w^t, x^t \rangle| > \sigma \|w^t\|$, then $(w^t \cdot x^t) \geq 0$ and

$$w^* \cdot w^{t+1} = w^* \cdot w^t + \ell(x^t)(w^t \cdot x^t)(w^* \cdot x^t) \geq w^* \cdot w^t \geq w^* \cdot w^0 \geq \frac{1}{\sqrt{n}}. \tag{8.3}$$

Let m_t be number of updates before step $t + 1$, then

$$\left\|w^{t+1}\right\|^2 = \left\|w^t\right\|^2 + (w^t \cdot x^t)^2 \left\|x^t\right\|^2 + 2\ell(x^t)(w^t \cdot x^t)^2 \leq \left\|w^t\right\|^2 (1 - \sigma^2) \leq (1 - \sigma^2)^{m_t}. \tag{8.4}$$

After t steps, $1 \geq \frac{w^t \cdot w^*}{\|w^t\|} \geq \frac{1/\sqrt{n}}{(1-\sigma^2)^{m_t}}$, thus $m_t \leq \frac{\log n}{\sigma^2}$ is the maximum possible number of updates. □

8.4 Weighted Majority and Winnow

Predicting from Expert advice. In this setting, there are n experts, and each expert makes $\{0,1\}$ predictions. The algorithm makes a prediction based on expert advice and observes the outcome.

The goal is to predict nearly as well as the best expert in hindsight. Let

$$M := \#\text{ mistakes by the algorithm}$$
$$m := \#\text{ mistakes by the best expert in hindsight}$$

If there is a perfect expert, i.e., it makes no mistakes, then by predicting according to the majority of experts and removing the erring experts in each round ensures $M \le \log_2 n$.

If the best expert makes m mistakes, we can restart the above process whenever there are no experts left. In each round, the number of mistakes made by the algorithm is at most $\log_2 n$, and best (every) expert makes at least 1 mistake. Thus $M \le m \log_2 n$.

8.4.1 Weighted Majority Algorithm

Algorithm 3 Weighted Majority (Multiplicative Weights)

```
Initialize $w_i \leftarrow 1$ for all $i \in [n]$
for $t = 1, \dots, T$
    Predict according to the weighted majority
    On a mistake, if expert $i$ predicts incorrectly, then $w_i \leftarrow w_i/2$
```

Let $W = \sum_{i=1}^n w_i$. Initially, $W = n$.

- On a mistake, total weight goes down by at least 3/4
- After M mistakes, the total weight is at most $n(3/4)^M$
- The weight of the best expert is always at least $(1/2)^m$

Therefore,

$$(1/2)^m \le W \le n(3/4)^M,$$

which implies

$$M \le \frac{\log n}{\log 4/3} + \frac{m}{\log_2 4/3},$$

Randomized Weighted Majority Algorithm

Algorithm 4 Randomized Weighted Majority

```
Initialize $w_i \leftarrow 1$ for all $i \in [n]$
for $t = 1, \dots, T$
    Predict according to expert $i$ w.p. $\frac{w_i}{\sum_j w_j}$
    If expert $i$ predicts incorrectly, then $w_i \leftarrow w_i(1-\epsilon)$
```

Let f_t be the weighted fraction of experts who made a mistake at time t and let M be the total function of mistakes from until time T. Then

$$\mathbb{E}[M] = \sum_{t=1}^{T} f_t.$$

Let $W = \sum_{i=1}^n w_i$. Initially, $W = n$.

- At every time step, $W^{(t+1)} \leq W^{(t)}(1 - \epsilon f_t)$
- After T step, the total weight is at most

$$W^{(T)} \leq n \prod_{t=1}^{T} (1 - \epsilon f_t).$$

- The weight of the best expert is always at least $(1 - \epsilon)^m$.

Therefore,

$$(1 - \epsilon)^m \leq W \leq n \prod_{t=1}^{T} (1 - \epsilon f_t).$$

Taking logs and using the fact that $\log(1 - \epsilon) \geq -\epsilon(1 + \epsilon)$,

$$\mathbb{E}[M] = \sum_{t=1}^{T} f_t \leq \frac{\log n}{\epsilon} + m(1 + \epsilon).$$

Optimizing for ϵ, with $\epsilon = \sqrt{\frac{\log n}{m}}$,

$$\frac{\mathbb{E}[M]}{T} \leq \frac{m}{T} + 2\frac{\sqrt{m \log n}}{T} \leq \frac{m}{T} + 2\sqrt{\frac{\log n}{T}}.$$

8.4.2 Winnow

Algorithm 5 Winnow

```
Initialize $w_i \leftarrow 1$ for all $i \in [n]$
for $t = 1, \ldots, T$
    Predict + if $w \cdot x \geq n$, − otherwise
    For mistake on a positive example, for all $i$ with $x_i = 1$, $w_i \leftarrow 2w_i$
    For mistake on a negative example, for all $i$ with $x_i = 1$, $w_i \leftarrow w_i/2$
```

Learn disjunction of r out of n variables.

Let the disjunction be $x_1 \vee x_2 \ldots \vee x_r$. We will call $\{x_1, x_2, \ldots, x_r\}$ relevant variables.

- Weights of the relevant variables cannot go above $2n$ and weights of the relevant variables never decrease
- For every mistake on a positive example, the weight of at least one relevant variable doubles
- The number of positive mistakes M_+ is at most

$$M_+ \leq r(1 + \log_2 n)$$

- On a positive mistake, the total weight increases by at most n and on a negative mistake, the total weight decreases by at least $n/2$, since the total weight is always non-negative,

$$0 \leq W \leq n + nM_+ - nM_-/2 \Rightarrow M_- \leq 2 + 2M_+.$$

- The total number of mistakes is bounded by $M \leq 2 + 3r(1 + \log_2 n)$.

Learn k-out-of-r Majority function

Let the relevant variables be $x_1, x_2, \ldots, x_r$. An example is positive iff $\sum_{i=1}^{r} x_i \geq k$. In this setting, change the update rules to

- For mistake on a positive example, for all i with $x_i = 1$, $w_i \leftarrow w_i(1+\epsilon)$
- For mistake on a negative example, for all i with $x_i = 1$, $w_i \leftarrow w_i/(1+\epsilon)$

Weights of the relevant variables cannot go above $(1+\epsilon)n$, however they can decrease.

- If variable i is involved in a_i positive mistakes and b_i negative mistakes, then it weight after T steps is $w_i = (1+\epsilon)^{a_i - b_i} \leq (1+\epsilon)n \Rightarrow a_i - b_i \leq 1 + \log_{(1+\epsilon)} n$.
- A mistake on a positive example causes the weights of at least k relevant variables to increase by $(1+\epsilon)$, and a mistake on a negative example causes the weights of at most $k-1$ relevant variables to decrease by $(1+\epsilon)$. Therefore, $\sum_i a_i \geq kM_+$ and $\sum_i b_i \leq (k-1)M_-$.
- Summing the mistakes over all relevant variables,
$$kM_+ - (k-1)M_- \leq r(1 + \log_{(1+\epsilon)} n). \tag{8.5}$$
- On a positive mistake, the total weight of relevant variables increases by at most ϵn, and on a negative mistake, the total weight decreases by at least $n\epsilon/(1+\epsilon)$, since the total weight is always non-negative,
$$\frac{\epsilon n M_-}{(1+\epsilon)} \leq n + \epsilon n M_+. \tag{8.6}$$
- From (**??**) and (**??**),
$$(k - (k-1)(1+\epsilon))M_+ \leq r(1 + \log_{(1+\epsilon)} n) + (k-1)\frac{1+\epsilon}{\epsilon}.$$
- Setting $\epsilon = \frac{1}{2(k-1)}$, $M_+ = O(kr \log n)$ and $M_- = O(kr \log n)$.

Halfspaces

Let $x \in \{0,1\}^n$ and the halfspace to be learned be

$$w_1^* x_1 + w_2^* x_2 + \ldots + w_n^* x_n \geq w_0^*.$$

Without loss of generality, we can assume that

- $w_i^* \geq 0$ for all $i \in [n]$, otherwise use $y_i = 1 - x_i$
- w_i^* is integer for all $i \in [n]$ by appropriate scaling.

Duplicate each variable $W = \sum_{i=1}^{n} w_i^*$ times and this problem becomes w_0^*-out-of-W majority problem.

- The number of mistakes by winnow is bounded by $O(w_0^* W \log_2(nW)) = O(W^2 \log_2(nW))$
- Given margin $\gamma = \min_x |w^* \cdot x|$, the number of mistakes is at most
$$M \leq \frac{\|w^*\|_1^2 \, \|x\|_\infty^2 \log(n \, \|w^*\|_1)}{\gamma^2}.$$

Winnow vs Perceptron

- Let $x \in \{0,1\}^n$, $\|x\|_\infty = 1$, and $\|x\|_2^2 = n$. If $w^* = (1, \dots, 1, 0, \dots, 0)$ (k ones), then $\|w^*\|_1 = k$, $\|w^*\|_2 = \sqrt{k}$. The number of mistakes of the 2 models:
  - Winnow: $O(\frac{k^2 \log(n)}{\gamma^2})$
  - Perceptron: $O(\frac{kn}{\gamma^2})$
- Let $\|x\|_\infty = 1$, and $\|x\|_2^2 \leq 1$. If $w^* = (\pm\frac{1}{\sqrt{n}}, \pm\frac{1}{\sqrt{n}}, \dots, \pm\frac{1}{\sqrt{n}})$, then $\|w^*\|_1^2 = n$, $\|w^*\|_2^2 = 1$. The number of mistakes of the 2 models:
  - Winnow: $O(\frac{n \log(n)}{\gamma^2})$
  - Perceptron: $O(\frac{1}{\gamma^2})$