Assignment 4 Solutions

COMP 599 Fall 2020 McGill University

November 18, 2020

Question 1[20 points] Find a hypothesis class and a sequence of examples so that the mistake bound of the halving algorithm is tight.

Solution. This very nice solution is due to Dragos Cristian Manta.

We will show an example of a hypothesis class $\mathcal{H} = \{h_1, h_2, \ldots,\}$ with a sequence of examples $\{x_i\}$ such that the halving algorithm makes exactly $M = \log_2(n)$ mistakes. For the purpose of this demonstration, we will assume that $n = 2^k$ for some k (to allow for $\log_2(n)$ to be an integer since M has to be an integer). We will also assume WLOG (I'll add a word on that at the end) that, in case that there is an equal number of hypotheses in $\mathcal{H}$ predicting labels 0 and 1 for a sample x_i, then the algorithm will assign the label 1 to x_i. Otherwise, it follows its usual prediction rule : $\hat{y}_i = \arg\max_r \left\{|h \in \mathcal{H} : \langle(\S_\rangle) = \nabla|\right\}$ (the majority vote among all hypotheses $h \in \mathcal{H}$ for each class $r \in \{0, 1\}$).

Let $x \in \mathbb{R}$. $\forall 1 \leq i \leq n$ define

$$h_i(x) := \begin{cases} 0 & \text{if } x < i \\ 1 & \text{if } x \geq i. \end{cases}$$

Now define the following sequence of samples:

$$x_i = \frac{1}{2} + 2^{k-i}(2^i - 1) = 2^k - 2^{k-i} + \frac{1}{2} \; \forall 1 \leq i \leq k. \tag{1}$$

(recall that $|\mathcal{H}| = \setminus = \in^{\|}$). Let the true hypothesis be $h^* = h_n = h_{2^k}$. We present to the algorithm the samples in exactly the following order: $x_1, x_2, x_3, ..., x_k$. We claim that the number of mistakes is $k = \log_2(n)$, matching the theoretical upper bound, thus making it tight.

To see this, it is helpful to look at how the sequence of x_i's is generated. The expression (1) can be re-written as

$$x_i = 2^{k-1} + 2^{k-2} + ... + 2^{k-i} + \frac{1}{2}.$$

Notice that $x_1 = 2^{k-1} + \frac{1}{2}$ is between the following 2 sets:

$$\left\{1, 2, ..., 2^{k-1}\right\} \text{ and}$$
$$\left\{2^{k-1} + 1, 2^{k-1} + 2, ..., 2^k\right\},$$

both having equal sizes. Therefore $h_1(x_1) = h_2(x_1) = ... = h_{2^{k-1}}(x_1) = 1$ and $h_{2^{k-1}+1}(x_1) = h_{2^{k-1}+2}(x_1) = ... = h_{2^k}(x_1) = 0$. Note that, since we fixed $h^* = h_n (= h_{2^k})$ at the beginning, then $h^*(x_1) = 0$. Since an equal number of hypotheses assign the label 0 and 1 to x_1, then the algorithm will assign the label 1 to x_1. It is thus wrong on x_1. Hence the algorithm discards $h_1, h_2, ..., h_{2^{k-1}}$. For notational convenience, for the remainder of this problem, instead of writing "$\{h_{2^p}, ...\}$", we will say " the hypotheses corresponding to $\{2^p, ...\}$ ".

For general x_i, assume that the algorithm was wrong on $x_1, x_2, ..., x_{i-1}$. Then it discarded, up to this point, the first $2^{k-1} + 2^{k-2} + ... + 2^{k-i+1}$ hypotheses. Hence the number of remaining hypotheses is $2^k - (2^{k-1} + 2^{k-2} + ... + 2^{k-i+1}) = 2^k - (2^k - 2^{k-i+1}) = 2^{k-i+1}$. This set of remaining hypotheses corresponds to $\left\{2^k - 2^{k-i+1} + 1, 2^k - 2^{k-i+1} + 2, ..., 2^k\right\}$. It can be partitioned in the following way:

$$\begin{aligned}
&\left\{2^k - 2^{k-i+1} + 1, 2^k - 2^{k-i+1} + 2, ..., 2^k - 2^{k-i+1} + 2^{k-i}\right\} \\
&\cup \left\{2^k - 2^{k-i} + 1, 2^k - 2^{k-i} + 2, ..., 2^k\right\} \\
&= \left\{2^k - 2^{k-i+1} + 1, 2^k - 2^{k-i+1} + 2, ..., 2^k - 2^{k-i}\right\} \cup \left\{2^k - 2^{k-i} + 1, 2^k - 2^{k-i} + 2, ..., 2^k\right\}.
\end{aligned}$$

However, by (1), $x_i = 2^k - 2^{k-i} + \frac{1}{2}$, which sits exactly between the 2 sets partitioning the remaining hypotheses (both of which have the same size). Hence the algorithm will assign the label 1 to x_i, which is again wrong by the fact that h^* assigns 0 to x_i $\forall 1 \leq i \leq k$ since x_k (the biggest one) $= 2^k - \frac{1}{2} < 2^k$. Therefore, in total, there were $k = \log_2(n)$ mistakes made (one for each sample), as claimed.

A word about the "WLOG": We can re-do the same analysis if the algorithm assigned 0 (instead of 1) in case of equality of votes by defining

$$h_i(x) := \begin{cases} 1 & \text{if } x < i \\ 0 & \text{if } x \geq i \end{cases}$$

instead.

Question 2[30 points] For $i = 1, \ldots, n$, define $\vec{x}_i \in \mathbb{R}^n$ and $y_i \in \{-1, 1\}$ by

$$\vec{x}_i = (\underbrace{(-1)^i, \ldots, (-1)^i, (-1)^{i+1}}_{i \text{ first components}}, 0, \ldots, 0) \quad \text{and} \quad y_i = (-1)^{i+1}.$$

Suppose that the perceptron algorithm is run cyclically over the sequence $S = (\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$ until it makes no more mistakes. We are using perceptron in off-line mode. We run it again and again on this data *in the given order* until it makes no more mistakes.

Show that the total number of mistakes is *at least* 2^{n-3}.

Just to make sure that you understand the $\vec{x}_i$, here are the first few:
$\vec{x}_1 = (1, 0, 0, \ldots, 0)$
$\vec{x}_2 = (1, -1, 0, 0, \ldots, 0)$

$\vec{x}_3 = (-1, -1, 1, 0, 0, \ldots, 0)$
$\vec{x}_4 = (1, 1, 1, -1, 0, 0, \ldots, 0)$

Hint. Note that this algorithm will only ever assign integer values to the weights so it suffices to consider vectors in $\mathbb{Z}$. Let $\vec{w} = (w_1, \ldots, w_n) \in \mathbb{Z}^n$ be (a normal vector of) any linear separator. Give lower bounds on the magnitude of the w_i. Then argue that the n-th component is incrased by at most 1 in every update so the size of w_n gives your lower bound.

Solution.

Following the hint, for i odd we have $\boldsymbol{w} \cdot \boldsymbol{x}_i \geq 0$, i.e., $w_i \geq w_{i-1} + \ldots w_1$. Likewise, for i even we have $w_i > w_{i-1} + \ldots w_1$. Thus we have the recurrence $w_i \geq w_{i-1} + \ldots w_1$ for all i with, in particular, $w_1 \geq 0, w_2 \geq 1$. Then for all $i \geq 2$ we have $\sum_{j=1}^{i} w_j \geq 2^{i-2}$ (by induction on i). We conclude that $w_n \geq \sum_{i=1}^{n-1} w_i \geq 2^{n-3}$.

Since each mistake by the Perceptron algorithm increases the n-th component of the hypothesis vector by at most 1, there are at least 2^{n-3} mistakes.

Question 3[50 points] In this question we will consider how to modify the perceptron algorithm to deal with non-separable data. We start by introducing an apparently new algorithm called the *dual perceptron*. Instead of a weight vector $\mathbf{w}$ we maintain a set of coefficients $\{\alpha_i\}_{i=1}^{n}$ where each α_i is $+1, 0$ or -1. The basic loop of the algorithm runs for $t = 1$ up to $t = T$:

- Receive $\mathbf{x}_t$. If $\displaystyle\sum_{i=1}^{i=t-1} \alpha_i(\mathbf{x}_i \cdot \mathbf{x}_t) \geq 0$ then predict $+1$, otherwise predict -1.
- Receive correct label y_t. If there is a mistake then $\alpha_t \leftarrow y_t$ otherwise $\alpha_t \leftarrow 0$.

Note that this algorithm makes *exactly the same* predictions as perceptron. However, instead of maintaining $\mathbf{w}$ explicitly we maintain it implicitly as the vector $\sum_{i=1}^{i=t-1} \alpha_i \mathbf{x}_i$.

Consider a sequence of labelled examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$. We do not assume that these data are necessarily separable. Fix $\gamma > 0$, you can think of this as the desired margin, if you like. Now to cope with the fact that there might not be a separating hyperplane we introduce a *loss function*. Suppose that we have a threshold function defined by a hyperplace with unit normal $\mathbf{u}$. We define the loss ℓ_t on input $(\mathbf{x}_t, y_t)$ as:

$$\ell_t = \max(0, \gamma - y_t(\mathbf{u} \cdot \mathbf{x_t})).$$

Notice that ℓ_t is non-negative and is zero if and only if the threshold function has margin at least γ at $(\mathbf{x}_t, y_t)$. The cumulative loss L of $\mathbf{u}$ over the sequence $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_T, y_T)$ is defined by $L^2 = \sum_{t=1}^{t=T} \ell_t^2$. Then $L = 0$ if and only if $\mathbf{u}$ represents a threshold function with margin at least γ on the above sequence. So the loss measures how far we are from our desired margin. We can now state our mistake bound, which specialises to the bound in the separable case by taking $L = 0$.

Prove the following statement:

Suppose that there is a linear threshold function with squared loss L^2 on $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_T, y_T)$. We assume that the norms are bounded: there exists $D > 0$, such that $\|\mathbf{x}_t\|^2 \leq D$ for all t. Then the perceptron algorithm makes at most $(\frac{D+L}{\gamma})^2$ mistakes.

Hint: Embed the data in a higher-dimensional space $\mathbf{R}^{n+T}$ by changing each vector $\mathbf{x}_t$ to a new vector

$$\tilde{\mathbf{x}}_t = (\mathbf{x}_t, 0, 0, \ldots, \Delta, 0, \ldots, 0),$$

where $\Delta > 0$ is some fixed positive number. Show that if the labels are not changed the predictions are not changed. Show how to define a modified version of $\mathbf{u}$ so that the new data are separable with a suitable margin (which won't just be γ). Now use the separable mistake bound to derive a mistake bound in terms of γ, Δ, L. Show that by choosing Δ to minimise this expression we get the claimed mistake bound.

Solution. In short, the idea of the proof is to add new features to make the data linearly separable, and then to appeal to the mistake bound in the separable case. Fix $\Delta > 0$. We map each vector $\mathbf{x}_t \in \mathbf{R}^n$ to a new vector $\tilde{\mathbf{x}}_t \in \mathbf{R}^{n+T}$, with T extra features, where

$$\tilde{\mathbf{x}}_t = (\mathbf{x}_t, 0, \ldots, 0, \Delta, 0, \ldots, 0)$$

has a Δ in the t-th extra dimension. We leave each label y_t unchanged.

Notice immediately that the predictions and mistakes of the (Dual) Perceptron algorithm are unaffected by augmenting the inputs in this way since $\mathbf{x}_t \cdot \mathbf{x}_s = \tilde{\mathbf{x}}_t \cdot \tilde{\mathbf{x}}_s$ for $1 \leq s \leq t \leq T$. Now let $u \in \mathbf{R}^n$ be a unit vector realising the loss L on the original data. Define

$$\tilde{\mathbf{u}} = (\mathbf{u}, \frac{y_1 \ell_1}{\Delta}, \ldots, \frac{y_T \ell_T}{\Delta}).$$

Then $\|\tilde{\mathbf{u}}\| = \|\mathbf{u}\| + \frac{L^2}{\Delta^2} = 1 + \frac{L^2}{\Delta^2}$. Also, we have

$$y_t(\tilde{\mathbf{u}} \cdot \tilde{\mathbf{x}}_t) = y_t(\mathbf{u} \cdot \mathbf{x}_t) + y_t^2 \ell_t = y_t(\mathbf{u} \cdot \mathbf{x}_t) + \ell_t \geq \gamma,$$

for all t. Thus, the transformed sequence $(\tilde{\mathbf{x}_1}, y_1), \ldots, (\tilde{\mathbf{x}_T}, y_t)$ is separable with margin at least $\frac{\gamma}{\sqrt{(1+\frac{L^2}{\Delta^2})}}$. We now have the bound

$$\|\tilde{\mathbf{x}}_t\|^2 = \|\mathbf{x}_t\|^2 + \Delta^2 \leq D^2 + \Delta^2.$$

Now using our result for the separable case we conclude that Perceptron makes at most

$$\frac{(D^2 + \Delta^2)(1 + \frac{L^2}{\Delta^2})}{\gamma^2}$$

on the transformed and hence the original input. Now Δ is a new variable that we introduced. We minimize it by taking $\Delta = \sqrt{DL}$, yielding a mistake bound of $(\frac{D+L}{\gamma})^2$.