CIS5200: Machine Learning Spring 2023

Lecture 1: Binary Classification with Linear Predictors

Date: January 17, 2023 *Author: Surbhi Goel*

Acknowledgements. These notes are heavily inspired by Chapter 9 of Understanding Machine Learning: From Theory to Algorithms (UML) and Cornell University's CS 4/5780 — Spring 2022.

Disclaimer. These notes have not been subjected to the usual scrutiny reserved for formal publications. If you notice any typos or errors, please reach out to the author.

1 Linear Predictors

In this lecture (and the next), we will focus on the hypothesis class of linear predictors. The class of linear functions is perhaps one the most useful and widely used largely due to the fact that it is intuitive to understand/interpret, and also computationally efficient to train and evaluate (in most cases).

2 Binary Classification with Linear Predictors

Let us formalize the setup we will use.

- Input space $\mathcal{X} = \mathbb{R}^d$
- Label space $\mathcal{Y} = \{-1, 1\}$[1]
- Hypothesis class $\mathcal{F} := \{x \mapsto \text{sign}(w^\top x + b) \mid w \in \mathbb{R}^d, b \in \mathbb{R}\}$ where $\text{sign}(a) = \begin{cases} 1 & \text{if } a \geq 0, \\ -1 & \text{otherwise.} \end{cases}$

  Here w is often referred to as the weight vector and b as the bias. We will use w and b to index the function class.
- Loss function $\ell(\hat{y}, y) = \begin{cases} 0 & \text{if } \hat{y} = y, \\ 1 & \text{otherwise.} \end{cases}$

  Note that this is the 0/1 loss which gives a loss of 1 for each incorrect prediction, and 0 otherwise.

Remark. Without loss of generality, we can assume that the bias term $b = 0$. This is because, a halfspace with bias, can be transformed into a halfspace without bias using the following mapping

[1] In the last lecture we used $\{0, 1\}$. For technical reasons, it will be easier to use $\{-1, 1\}$.

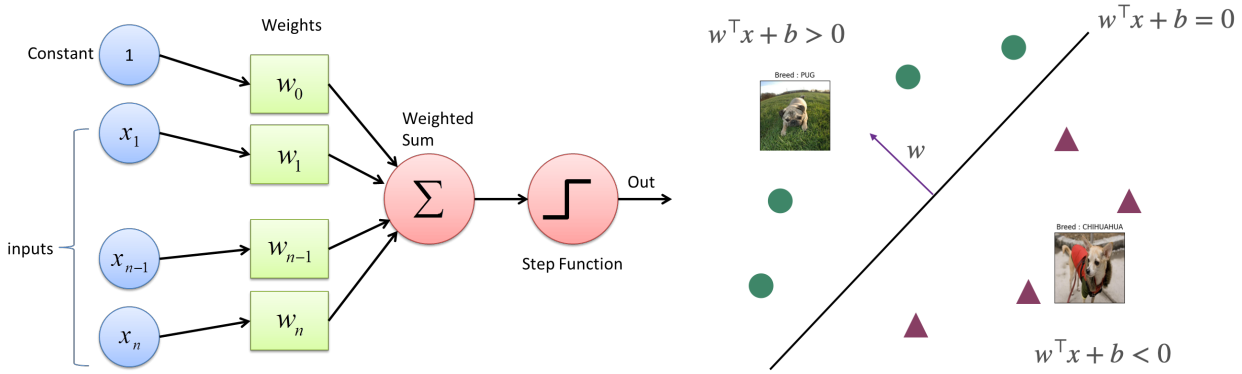

Figure 1: Linear separator or halfspace. (*left*) Perceptron view [source: `https://deepai.org/machine-learning-glossary-and-terms/perceptron`], (*right*) geometric visualization of a 2D halfspace. Note that w is perpendicular to the hyperplane (you can try to prove this). The points above the line are labelled positive and those below are labelled negative.

$x \mapsto \begin{bmatrix} x \\ 1 \end{bmatrix}$ and $w \mapsto \begin{bmatrix} w \\ b \end{bmatrix}$. This increases the dimension of the weight vector by 1, and this extra dimension absorbs the bias. Going forward, we will assume $b = 0$.

Now given a training dataset $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, we want to find the empirical risk minimzer by minimizing the following:

$$\hat{w} = \arg\min_w \frac{1}{m} \sum_{i=1}^{m} 1[\text{sign}(w^\top x_i) \neq y_i]$$

ERM for linear classifiers is computationally hard to solve in the worst case. To make this tractable, we will make a *separability* assumption. In particular, we will assume that there is a halfspace that correctly classifies the data. In particular, there exists w_* such that for all $(x_i, y_i) \in S$), $y_i = \text{sign}(w_*^\top x_i)$. We say the data is **linearly separable** if it satisfies this condition.

3 Perceptron

This model (Perceptron) was an early attempt to model a biological neuron dating to McCulloch & Pitts in 1943. Figure 1 (left) shows a pictorial representation of the "neuron" which is what comprises neural networks today, albeit with different "activation functions" than the sign/step function. The term MLP, common jargon in deep learning, stands for Multi-layer Perceptron.

In 1957, Frank Rosenblatt came up with an algorithm called "Perceptron" to learn this model from data. The invention of the algorithm gained wide popularity and created a huge wave of excitement. Check out this article from the New York Times titled *Electronic 'Brain' Teaches Itself* in 1958 (`https://www.nytimes.com/1958/07/13/archives/electronic-brain-teaches-itself.html`). Perhaps, this was the first instance of what we call deep learning now.

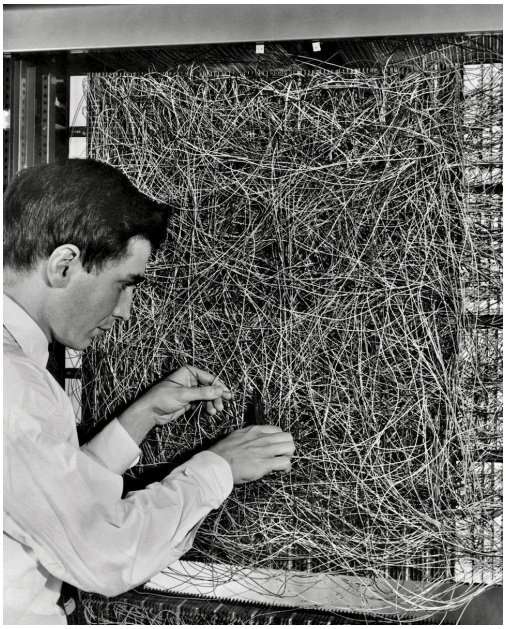

Figure 2: Frank Rosenblatt with a Mark I Perceptron computer in 1960.

3.1 Algorithm

We will now see how the Perceptron algorithm (Algorithm 1) solves the ERM problem in the linearly separable case. The Perceptron algorithm can also be run with data in an online fashion, but we will discuss the batch version here.

Algorithm 1: Perceptron

Initialize $w_1 = 0 \in \mathbb{R}^d$
for $t = 1, 2, \ldots$ **do**
  if $\exists i \in \{1, \ldots, m\}$ *s.t.* $y_i \neq \text{sign}\left(w_t^\top x_i\right)$ **then** update $w_{t+1} = w_t + y_i x_i$
  else output w_t
end

The update may seem strange, but let us see the intuition behind this. Suppose x_i is the example that is misclassified and assume $\|x_i\|_2 = 1$.

- If the true label was positive, then $w_{t+1}^\top x_i = w_t^\top x_i + \|x_i\|^2 = w_t^\top x_i + 1$.
- If the true label was negative, then $w_{t+1}^\top x_i = w_t^\top x_i - \|x_i\|^2 = w_t^\top x_i - 1$.

In both cases, the update moves the weight in the right direction. Also see Figure 3 for a visual interpretation of this. This does not imply that the update would be good for the other examples.

Given the intuition, let us try and formalize it. For simplicity, we will assume the data lies in the unit ball, so that $\|x_i\|_2 \leq 1$. We will also assume that $\|w_*\|_2 = 1$. Lastly, we will define a very useful quantity in machine learning called the margin. The margin γ is the minimum distance of any point from the hyperplane. Recall, in Homework 0, we calculated the distance of a point from

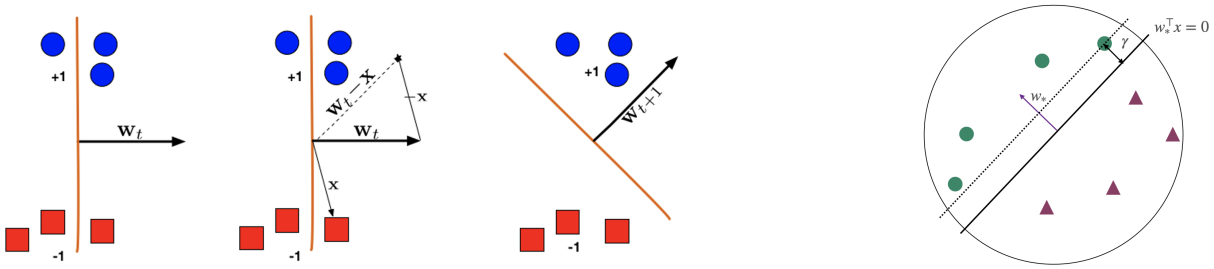

Figure 3: (*left*) The update of the Perceptron Algorithm using a mislabelled example. Note that after this update, the Perceptron algorithm will terminate. (*right*) Margin defined as the distance from the hyperplane of the closest example. *Source: Lecture notes from Cornell CS 4/5780* `https://www.cs.cornell.edu/courses/cs4780/2022sp/notes/Notes06.pdf`.

the separating hyperplane $w_*^\top x = 0$. Using that, we get

$$\gamma = \min_{i \in \{1,\dots,m\}} |w_*^\top x_i|.$$

Lemma 1. *For all $i \in \{1, \dots, m\}$, $y_i(w_*^\top x_i) \geq \gamma$.*

Proof. Recall that $y_i = \text{sign}(w_*^\top x_i)$ based on our separation assumption. Thus we have,

$$\begin{aligned} y_i(w_*^\top x_i) &= \text{sign}(w_*^\top x_i)(w_*^\top x_i) \\ &= |w_*^\top x_i|. \end{aligned}$$

Here the second equality follows from observing that $\text{sign}(a)a = |a|$ for all a. Now, using our margin assumption, we know that $|w_*^\top x_i| \geq \min_{j \in \{1,\dots,m\}} |w_*^\top x_j| \geq \gamma$. □

Lemma 2. *For all $i \in \{1, \dots, m\}$ such that $y_i \neq \text{sign}(w_t^\top x_i)$, we have $y_i(w_t^\top x_i) \leq 0$.*

Proof. Since $y_i \neq \text{sign}(w_t^\top x_i) \implies y_i = -\text{sign}(w_t^\top x_i)$. Thus we have,

$$\begin{aligned} y_i(w_t^\top x_i) &= -\text{sign}(w_t^\top x_i)(w_t^\top x_i) \\ &= -|w_t^\top x_i| \leq 0. \end{aligned}$$

Here the second equality follows from observing that $\text{sign}(a)a = |a|$ for all a. And the last inequality follows from observing that $|a| \geq 0$ for all a. □

Now with these lemmas in hand, let us state the main convergence theorem.

Theorem 3. *The Perceptron algorithm stops after at most $1/\gamma^2$ rounds, and returns a hyperplane w such that $\forall i \in [m]$, $\text{sign}(w^\top x_i) = y_i$, that is, all points are correctly classified.*

Proof. By the terminating condition of the algorithm, if the algorithm terminates, it must have all points correctly classified. Let us know show that the algorithm does terminate and in fact in time independent of the input dimension.

We want to show that we are making progress, and w_t is getting closer to w_*. One way to measure this is by looking at the cosine of the angle between w_t and w_*, that is, $\frac{w_*^\top w_t}{\|w_t\|}$.

Suppose at time t, we update with (x_i, y_i) for some i, then

$$\begin{aligned} w_*^\top w_{t+1} &= w_*^\top w_t + y_i w_*^\top x_i \\ &\geq w_*^\top w_t + \gamma. \end{aligned}$$

Here the first equality follows from the definition of the update. The second inequality follows from Lemma 1.

Also observe that

$$\begin{aligned} \|w_{t+1}\|_2^2 &= (w_t + y_i x_i)^\top (w_t + y_i x_i) \\ &= \|w_t\|_2^2 + \underbrace{y_i^2 \|x_i\|_2^2}_{\leq 1} + 2\underbrace{y_i w_t^\top x_i}_{\leq 0} \\ &\leq \|w_t\|_2^2 + 1. \end{aligned}$$

Here the last inequality follows from (a) our assumption that $\|x_i\|_2^2 \leq 1$, and (b) from Lemma 2 since (x_i, y_i) is misclassified. □

This implies that after T rounds, we have,

$$\begin{aligned} w_*^\top w_{T+1} &\geq w_*^\top w_T + \gamma \\ &\geq (w_*^\top w_{T-1} + \gamma) + \gamma \\ &\vdots \\ &\geq (w_*^\top w_1) + T\gamma \\ &= T\gamma. \end{aligned}$$

The above follows from substituting the recurrence repeatedly and observing that $w_*^\top w_1 = 0$ since $w_1 = 0$.

A similar calculation gives us,

$$\|w_{T+1}\|_2^2 \leq T \implies \|w_{T+1}\|_2 \leq \sqrt{T}.$$

This implies that the cosine satisfies,

$$\frac{w_*^\top w_{T+1}}{\|w_{T+1}\|_2} \geq \frac{\gamma T}{\sqrt{T}} = \gamma\sqrt{T}.$$

Now we know that the cosine is bounded above by 1. Thus

$$1 \geq \frac{w_*^\top w_{T+1}}{\|w_{T+1}\|_2} \geq \gamma\sqrt{T} \implies T \leq 1/\gamma^2.$$

This completes the proof.

Note: It is helpful to go through the proof and see exactly where what assumption is used, and how necessary it is.

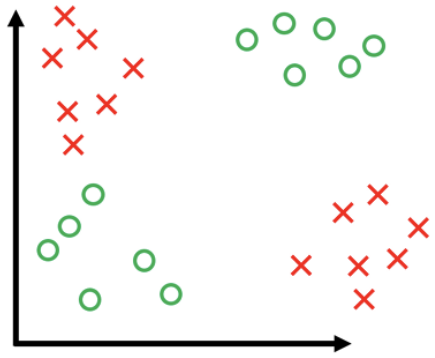

Figure 4: Data drawn from an XOR function. *Source: Lecture notes from Cornell CS 4/5780* *https://www.cs.cornell.edu/courses/cs4780/2022sp/notes/Notes06.pdf.*

Limitations of Perceptron. Minksy and Papert wrote a book titled 'Perceptrons' in 1969 which showed that the Perceptron algorithm could not learn XOR functions (Figure 4). Such data is not linearly separable.

The book became widely popular as a criticism for neural networks (or perceptrons) and is said to have led to the AI winter which lasted till the mid-90s.

Apart from not working on non-linearly separable data, it cannot handle noise in the label. This has led to developments of other techniques that we will study in the later part of the course.