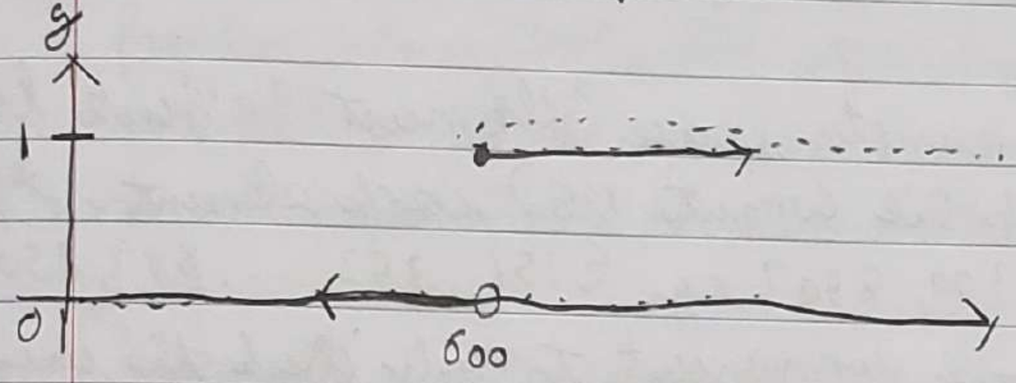
2/8/21

$$\mathcal{H} = \{ \mathbb{1}_{x \geq \theta} : \underset{\substack{\uparrow \\ \text{model parameter}}}{\theta} \in \underset{\substack{\uparrow \\ \text{parameter space}}}{\boxed{\Theta}} \}$$



prediction

$\hat{y} = g(\vec{x})$

$y = g(\vec{x}) + e = \hat{y} + e = \hat{y} + \overset{e}{\overset{\parallel}{(y - \hat{y})}}$

The algorithm A produces g. Since g is fully specified by θ, the algorithm selects / estimates / optimizes / fits a θ. Lets create an algorithm, a bad algorithm will have high estimation error.

$$\begin{array}{cc|c|c|c} & & \hat{y} & & \\ & & 0 & 1 & \\ \hline y & 0 & 0 & -1 & e \\ \hline & 1 & 1 & 0 & \\ \hline \end{array}$$

Lets define an overall error function / objective function called "misclassification error" (ME)

$$ME = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{g(\vec{x}_i) \neq y_i} = \frac{1}{n} \sum_{i=1}^{n} |e_i|$$

or accuracy (Acc) as

$$Acc = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{g(\vec{x}_i)\, y_i} = 1 - ME$$

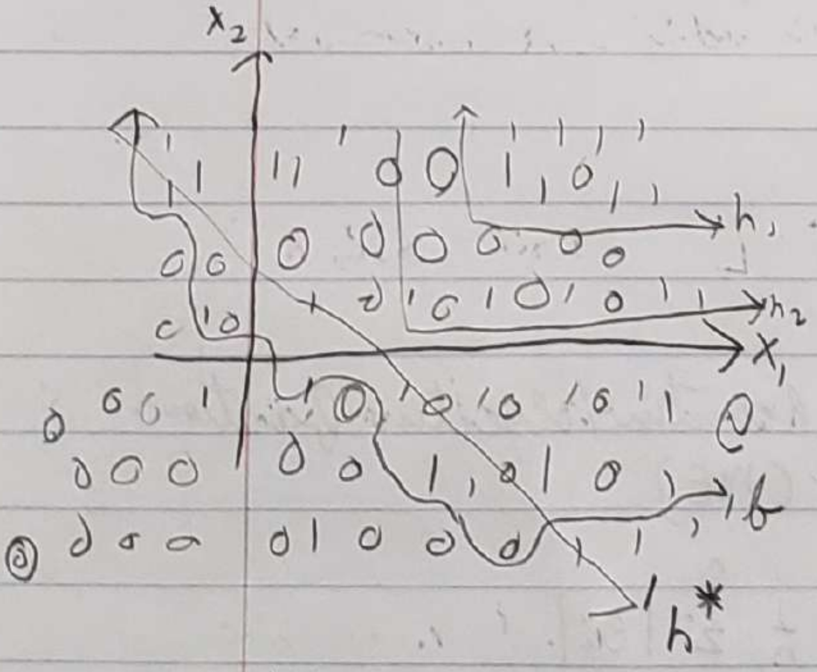
goal of the algorithm is to minimize ME (or max ACC). To do so, we check every possible $\theta \in \Theta$ and keep track of the $ME(\Theta)$ and then return the model with the lowest ME.

How to define parameter space? It must be finite because we need to check (ie compute ME) each element. Gabriel says grid up [300, 850] eg {351, 352, ..., 849, 850}. That's fine, but its more convenient to only check the unique values of x.

A producer $g(x) = \mathbb{1}_{x \geq \underset{\theta \in \text{unique}(\vec{x})}{\text{argmin}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\mathbb{1}_{x_i > \theta} \neq y_i} \right\}}$

Let's make a loan model with two continuous x's ie x_1, x_2 $(p=2)$

$\dim[\Theta] = 2 = p$



A two dimensional threshold model extending what we have before has candidate set:

$\mathcal{H} = \left\{ \mathbb{1}_{x_1 \geq \theta_1}, \mathbb{1}_{x_2 > \theta_2} : \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \in \Theta \right\}$

This candidate set of "angle bracket"-looking things is very restrictive! Which means we will probably have high misspecification error. Let's use another hypothesis set: all lines.

$$\mathcal{H} = \{ \mathbb{1}_{x_2 \geq \underset{\underset{\text{intercept}}{\nearrow}}{a} + \underset{\underset{\text{slope}}{\uparrow}}{b} x_1} : a \in \mathbb{R}, b \in \mathbb{R} \}$$

The slope and intercept provide you with enough "degree of freedom" to specify any separating line. We need an algorithm to find g ie to specify a and b. This is a hard problem so we will study it with different conditions

We will first reparameterize the hypothesis space to be:

$$\text{"} \; \mathbb{1}_{\vec{w} \cdot \vec{x} \geq 0}$$

$$\mathcal{H} = \{ \mathbb{1}_{\underset{\underset{\text{intercept term or "bias"}}{\uparrow}}{w_0} + \underset{\underset{\text{weight of the first feature}}{\uparrow}}{w_1 x_1} + \underset{\underset{\text{weight of the second feature}}{\uparrow}}{w_2} x_2 \geq 0} : w_0 \in \mathbb{R}, w_1 \in \mathbb{R}, w_2 \in \mathbb{R} \}$$

In order to fit this model, we "add" a dummy value of 1 to each data record:

$$\vec{x} = [750, \$58000] \rightarrow \vec{x} = [1 \;\; 750 \;\; \$58{,}000]$$

So we append the $\vec{1}$, the n-dim one column vector to X, the matrix of features in $\mathbb{D}$.

We only need 2 parameters (a, b) but here we have three (w_0, w_1, w_2) and hence we are "over-parameterized" meaning we have infinite solution seen here;

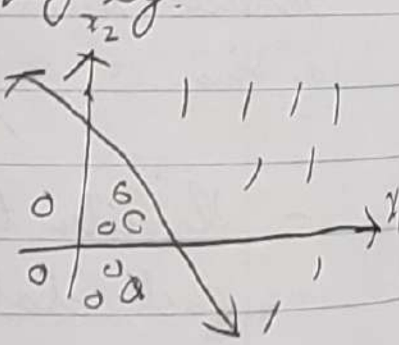
$$\mathbb{1}_{\vec{w} \cdot \vec{x} \geq 0} = \mathbb{1}_{c\vec{w} \cdot \vec{x} \geq 0} \;\; \forall c > 0 \qquad \underset{\underset{\text{pin}}{\uparrow}}{x_1 + x_2 = 7}$$

A: find w_0, w_1, w_2 to minimize ME ie

$$\vec{w}_{*} := \underset{\vec{w} \in \mathbb{R}^3}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \mathbb{1}_{\mathbb{1}_{\vec{w} \cdot \vec{x}_i > 0} \neq y_i} \right\} = \operatorname{argmin} \{ME\}$$

we have a problem here. There is no analytic solution since the indicator function is non-differentiable. We need a way to search over all possible lines. So (1) we need to reduce the number of lines like before, (2) use an iterative algorithm to find a local solution (not the best but hopefully pretty good), or (3) change our objective function

In the setting of perfect linear separability eg. where ME of that linear discrimination model is zero (ie no errors). Consider the 1957 perceptron iterative algorithm for p features:



Step 1: Initialize $\vec{w}^{\,t=0} = \vec{0}_{p+1}$ or to a random vector value.

Step 2: Compute $\hat{y}_i = \mathbb{1}_{\vec{w}^{\,t=0} \cdot \vec{x}_i \geq 0}$

Step 3: For $j = 0, 1, \ldots, p$ set

$$w_0^{t=1} = w_0^{t=0} + (y_i - \hat{y}_i)(1)$$
$$w_1^{t=1} = w_1^{t=1} + (y_i - \hat{y}_i)(x_{i,1})$$
$$\vdots$$
$$w_p^{t=1} = w_p^{t=0} + (y_i - \hat{y}_i)(x_{i,p})$$

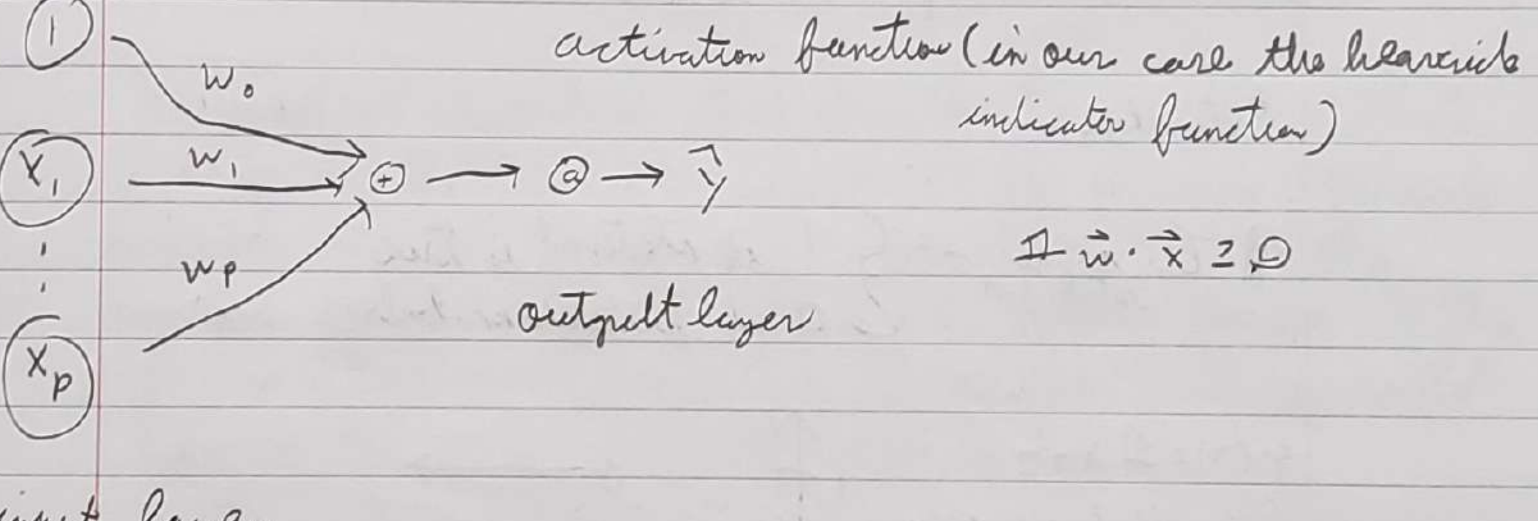
note: t is the iteration number, it starts at

Step 4: Repeate Step 2 and 3 for $i = 1, \ldots, n$ (all the observations)

Step 5: Repeat steps 2, 3 and 4 until $ME = 0$ ie all e_i's $= 0$ or until a prespecified (large) number of iterations.

The perception is proved to converged for linearly separable data sets but for non-linearly separable datasets, anything can happen so it may fail.

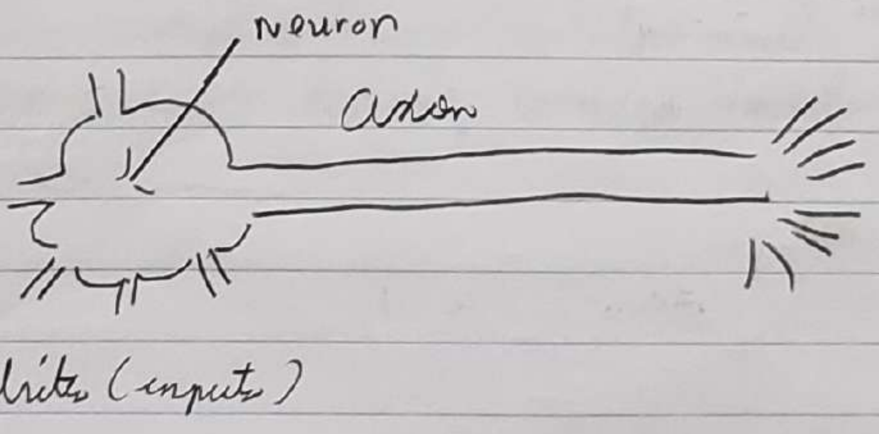
diagram of perception:



1
x_1
x_p
w_0
w_1
w_p
+
$\hat{y}$
activation function (in our case the heaviside indicator function)
$\mathbb{1}\, \vec{w} \cdot \vec{x} \geq 0$
output layer

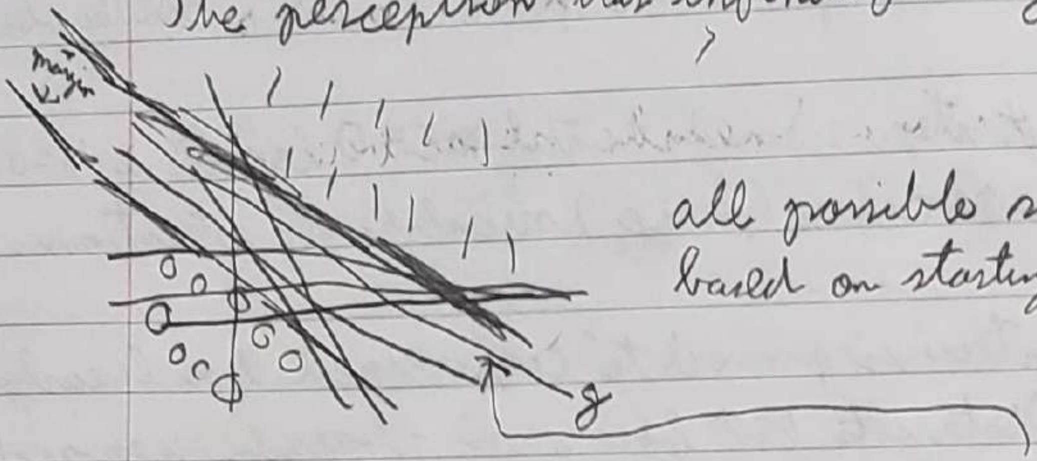
input layer

The perceptron is a type of "neural network" model. So are deep learning models. They're called neurons since they kind of act like neurons:



dendrites (inputs)

The perceptron has infinitely many solution



all possible solution which vary based on starting values

But you kinda see there is a "best" model. This best model divides the margin (aka wedge) evenly. This "best" model is called the "maximum margin hyperplane" and it was proven in 1998 to be optimal linear classifier.

Extra laws

$$\mathbb{1}_{\text{statement}} := \begin{cases} 1 & \text{if statement is true} \\ 0 & \text{if statement is false} \end{cases}$$

$g(x) = \mathbb{1}_{x>2}$