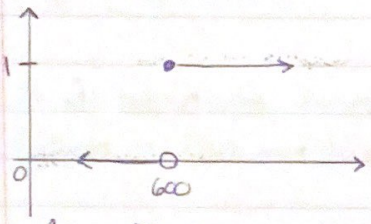
$$\mathcal{H} = \{ \mathbb{1}_{x \geq 0} : \theta \in \Theta \}$$

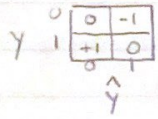
model parameter; parameter space



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

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

The algorithm $\mathcal{A}$ produces g. Since g is fully specified by θ, the algorithm selects/estimates/optimizes/fits a θ. Let's create an algorithm. A bad algorithm will have high estimation error.



Let's 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

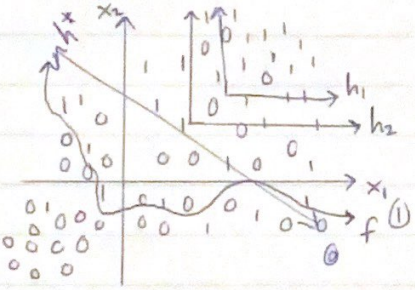
$$ACC = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{g(\vec{x}_i) = y_i} = 1 - ME$$

goal of the algorithm is to minimize ME (or maximize 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 (i.e. compute ME) each element. Gabriel says grid up $[300, 850]$ e.g. $\{351, 352, \ldots, 850\}$. That's fine, but it's more convenient to only check the unique values of x.

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

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



$\dim\left[\text{H}\right] = 2 = p$

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

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

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.

$\mathcal{H} = \left\{ \mathbb{1}_{x_2 \geq a + b x_1} : a \in \mathbb{R}, b \in \mathbb{R} \right\}$

intercept, slope

The slope and intercept provide you with enough "degree of freedom" to specify any separating line. We need an algorithm to find g i.e. 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:

$\mathcal{H} = \left\{ \mathbb{1}_{w_0 + w_1 x_1 + w_2 x_2 \geq 0} : w_0 \in \mathbb{R}, w_1 \in \mathbb{R}, w_2 \in \mathbb{R} \right\}$, $\mathbb{1}_{\vec{w} \cdot \vec{x} \geq 0}$

intercept term or "bias"; weight of first feature; weight of second feature

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

$\vec{x} = [750 \ \$58000] \rightsquigarrow \vec{x} = [1 \ 750 \ \$58000]$

So we append the $\vec{1}$, the n-dim 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 solutions seen here

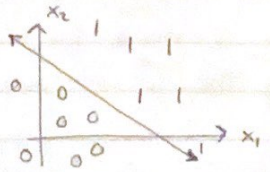
$$\mathbb{1}_{\vec{w}\cdot\vec{x}\geq 0} = \mathbb{1}_{c\vec{w}\cdot\vec{x}\geq 0} \quad \forall c \neq 0$$

$\mathcal{A}$: find w_0, w_1, w_2 to minimize ME i.e.

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

$\mathbb{1}$ is not differentiable

We have a problem here. There is no analytic solution. 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 e.g. where ME of that linear discrimination model is zero (i.e. 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 $\hat{y}_i = \mathbb{1}_{\vec{w}^{t}\cdot\vec{x}_i\geq 0}$

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

$w_0^{t+1} = w_0^{t} + (y_i - \hat{y}_i)(1)$

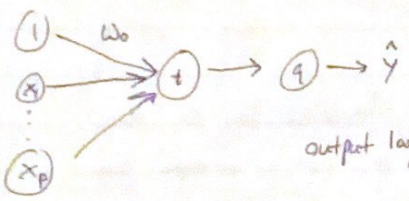
$w_1^{t+1} = w_1^{t} + (y_i - \hat{y}_i)(x_{i,1})$

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

Step 4: Repeat steps 2 and 3 for $i = 1, \ldots, n$ (all the observations)
Step 5: Repeat steps 2, 3, and 4 until $ME = 0$ i.e. all e_i's $= 0$ or until a prespecified (large) number of iterations.

The perceptron is proved to converge for linearly separable datasets but for non-~~linearly~~ linearly separable datasets, anything can happen so it may fail.

Diagram of perceptron



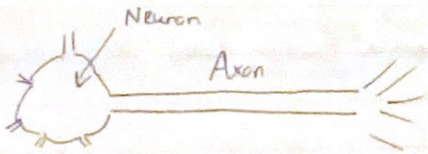
activation function (in our case the Heaviside indicator function)

$1_{\vec{w} \cdot \vec{x} \geq 0}$

output layer

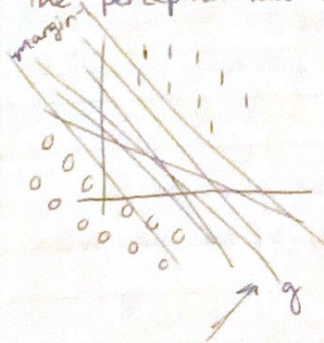
input layer

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 infintely many solutions



all possible solutions which vary based on starting values.

"best" model (g). This best model divides the margin/wedge evenly. This "best" model is called the "maximum margin hyperplane". Optimal linear classifier (proved in 1998)