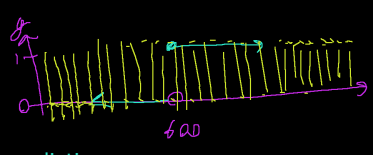
$$\mathcal{H} = \{ \mathbb{1}_{x \geq \theta} : \theta \in \Theta \}$$

model parameter

parameter space

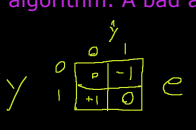


prediction

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

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

The algorithm A produces g. Since g is fully specified by theta, the algorithm selects / estimates / optimizes / fits a theta. 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

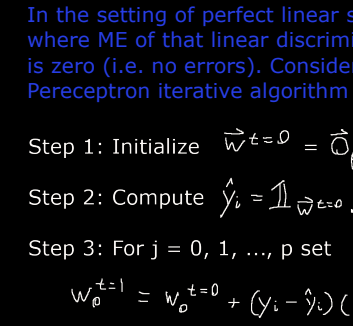
$$ACC = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{g(\vec{x}_i) = y_i} = 1 - ME$$

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, ..., 849, 850}. That's fine, but it's more convenient to only check the unique values of x.

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

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



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

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

$$\mathcal{H} = \left\{ \mathbb{1}_{x_1 \geq \theta_1, \mathbb{1}_{x_2 \geq \theta_2}} : \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \in \mathbb{R}^2 \right\}$$

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.

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

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:

$$\mathcal{H} = \{ \mathbb{1}_{w_0 + w_1 x_1 + w_2 x_2 \geq 0} : w_0 \in \mathbb{R}, w_1 \in \mathbb{R}, w_2 \in \mathbb{R} \}$$

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

weight of the first feature, weight of second feature

intercept term or "bias"

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

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

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:

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

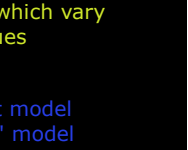
$x_1 + x_2 = 7$ ↑ pick

A: find w_0, w_1, w_2 to minimize ME i.e.

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

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 e.g. where ME of that linear discrimination model is zero (i.e. no errors). Consider the 1957 Pereceptron 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, ..., p set

$$w_0^{t=1} = w_0^{t=0} + (y_i - \hat{y}_i)(1)$$

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

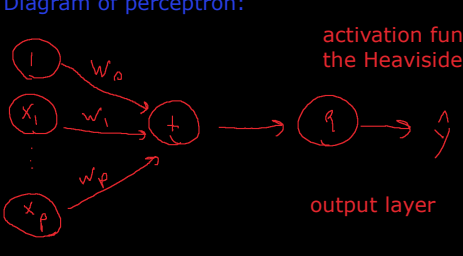
$$\vdots$$

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

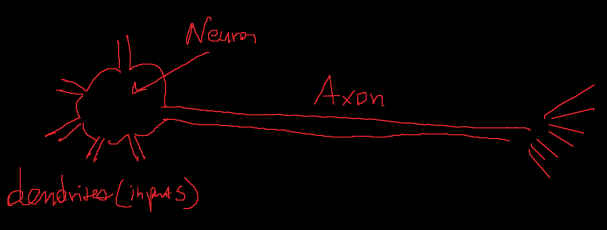
Step 4: Repeat steps 2 and 3 for i = 1, ..., n (all the observations).
Step 5: Repeat steps 2,3 and 4 until ME = 0 i.e. all e_i's = 0 or until a prespecified (large) number of iterations.
Note: t is the iteration number. It starts at 0. t = 1 is first iteration..

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

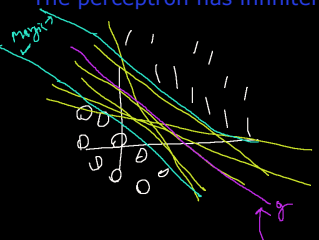
Diagram of perceptron:



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:



The perceptron has infinitely many solutions



all possible solutions which vary based on starting values

But you kinda see there's 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 the optimal linear classifier.