Recall:

$$y = t(\vec{z}) = f(\vec{x}) + \delta$$

δ: error due to ignorance

Note: the space of f is very large, so we need to constrain it.

- Supervised Learning

1. Training Data: $\mathbb{D} = \langle X, y \rangle$, where

$$X = \begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \vdots \\ \vec{x}_n \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix},$$

$\vec{x}_i \in \mathcal{X}$
$y_i \in Y$

2. $\mathcal{H}$: a candidate set of functions

3. $\mathcal{A}$: an algorithm which takes in data $\mathbb{D}$ + set $\mathcal{H}$ + produces a model, g.

$$g = \mathcal{A}(\mathbb{D}, \mathcal{H})$$

Q: is $f \in \mathcal{H}$ generally?

- A: No.

However, there is $h^* \in \mathcal{H}$, which is the closest possible model to f.
(function)

We have

$$y = t(\vec{z}) = f(\vec{x}) + \delta = h^*(\vec{x}) + \underbrace{\mathcal{E}}_{\text{epsilon error}}$$

$$= h^*(\vec{x}) + \underbrace{\underbrace{\left[f(\vec{x}) - h^*(\vec{x})\right]}_{\text{misspecification error}} + \underbrace{\left[t(\vec{z}) - f(\vec{x})\right]}_{\delta}}_{\mathcal{E}}$$

- Just because $h^* \in \mathcal{H}$ does not mean $\mathcal{A}$ will locate it. $\mathcal{A}$ will not be perfect & the value of $\mathcal{E}$ will confuse $\mathcal{A}$. Thus $g \neq h^*$, g is the best $\mathcal{A}$ can do.

$$y = \underbrace{g(\vec{x})}_{\text{Model}} + \underbrace{\underbrace{\left[h^*(\vec{x}) - g(\vec{x})\right]}_{\text{"estimation error"}} + \underbrace{\left[f(\vec{x}) - h^*(\vec{x})\right] + \underbrace{\left[t(\vec{z}) - f(\vec{x})\right]}_{\delta}}_{\mathcal{E}}}_{e \text{ "residual"}}$$

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

prediction of y is getting $\vec{x}$.

$e = y - \hat{y}$ residual if $\vec{x} \in \mathbb{D}$, otherwise they're unknown.

- How to reduce errors

1. δ, ignorance error can be reduced by measuring more x_i's (features) of the units that contain information about $\vec{z}$.
2. Misspecification error can be reduced by expanding $\mathcal{H}$ to include more complicated functions.
3. Estimation error can be reduced by increasing sample size n.

$$SE[\hat{\theta}] = \frac{\theta}{\sqrt{n}}$$

ex, $Y = \{0, 1\}$ "classification"

did not pay back mortgage — paid back mortgage

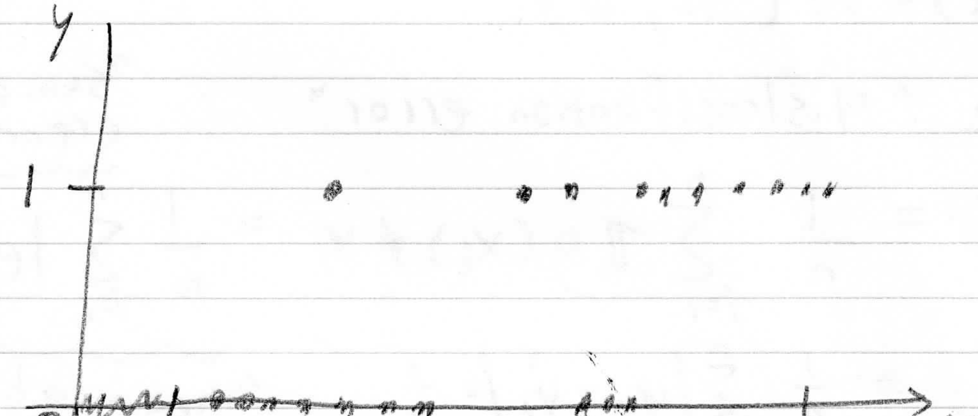
(1) $p=1$ (number of variables per person)

x is credit score

$n=100$

$$\mathbb{D} = \langle X, y \rangle = \left\langle \begin{bmatrix} 810 \\ 390 \\ 750 \\ \vdots \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix} \right\rangle$$

$\mathcal{X} = [300, 850]$



(2)

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

$\mathcal{H}$: threshold models

$$\mathbb{1}_A(w) := \begin{cases} 1 & \text{if } w \in A \\ 0 & \text{if } w \notin A \end{cases}$$

θ: parameter, sometimes denoted $\beta, w, \ldots$ others,
Capital θ is Θ in the Greek alphabet

e.g.

$$g(x) = \mathbb{1}_{x \geq 515.3}$$
$$g(x) = \mathbb{1}_{x \geq 407.9}$$

(3) Algorithm $\mathcal{A}$
Recall

$$g(x) = \mathcal{A}(\mathcal{H}, \mathbb{D})$$

Define "Mislassification error"

$$ME := \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{g(x_i) \neq y_i} = \underbrace{\frac{1}{n} \sum_{i=1}^{n} |e_i|}_{\text{"Sum absolute error" (SAE)}}$$

$$= \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| = \frac{1}{n}$$

"mean absolute error" (MAE

"Accuracy" ACC =

$$\cancel{\frac{1}{n}} \sum e_i^2$$

"Sum squared error" (SSE)

"Accuracy" $Acc = 1 - ME$

$\mathcal{A}$: minimize ME over $\Theta \in \{$unique x's$\}$

ex. $y \in Y = \{0, 1\}$

x_1 : credit score $\in [300, 850]$

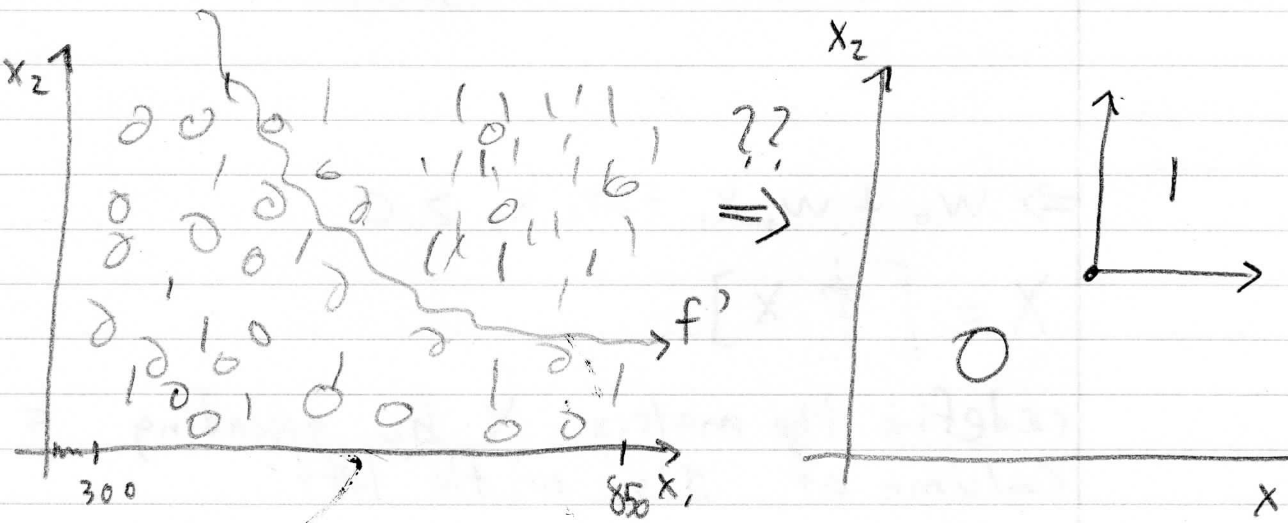
x_2 : salary (in \$1000's) $\in \mathbb{R}$

$$\mathbb{D} = \left\langle \begin{bmatrix} 810 & 63.1 \\ 390 & 58.7 \\ 750 & 132.6 \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix} \right\rangle$$

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

eg.

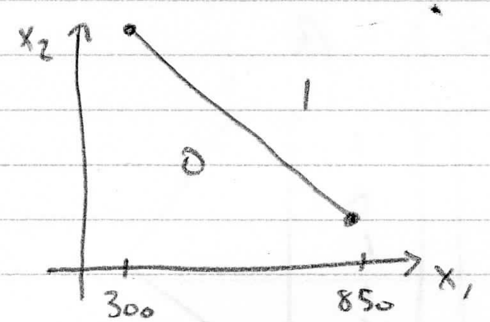
$$g(x) = \mathbb{1}_{x_1 \geq 650 + x_2 \geq 100}$$



f' could be real f, but may be hard to get.

Better guess using a linear function:

$$\mathcal{H} = \left\{ \mathbb{1}_{x_2 \geq a + bx_1} : a, b \in \mathbb{R} \right\}$$



Parameter space has dimension 2, i.e 2 degrees of freedom.

$$x_2 \geq a + bx_1 \implies \underbrace{-a}_{w_0} \underbrace{- \underbrace{b}_{w_1} x_1 + \underbrace{(1)}_{w_2} x_2}_{\text{"weight"}} \geq 0$$

w_0 : "bias"

$$\implies w_0 + w_1 x_1 + w_2 x_2 \geq 0$$

$$X = [\,\vec{1} \;\vdots\; X\,]$$

redefine the matrix X by appending a column of 1's on the left.

$$\vec{x} = [\,1 \; x_1 \; x_2\,]$$

$p+1 = 3$ (# of columns in X)

for the bias term

credit score / salary.

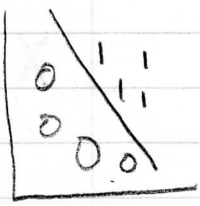
$\mathcal{H} = \{ \mathbb{1}_{\vec{w} \cdot \vec{x} \geq 0} : \vec{w} \in \mathbb{R}^3 \}$

This is an "overparametrized" model.
Each line has infinite $\vec{w}$'s that specify it.

Need algorithm $\mathcal{A}$.

$g = \mathcal{A}(\mathbb{D}, \mathcal{H})$.

Assume the 0's & 1's are linearly seperable.
Then $\exists\, \vec{w}$ s.t. $g(\vec{x})$ has no error.



Perceptron Learning Algorithm (1957)

1. Initialize $\vec{w}^{\,t=0} = \vec{0}$ or random. Compute $\hat{y}$

2. Let

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

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

$$w_2^{t=1} = w_2^{t=0} + (y_i - \hat{y}_i)\,x_{1,2}$$

$$\vdots$$

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

$$X = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \dots & x_{1,p} \\ 1 & x_{2,1} & x_{2,2} & \dots & x_{2,p} \\ \vdots & & & & \vdots \\ 1 & x_{n,1} & x_{n,2} & \dots & x_{n,p} \end{bmatrix}$$

3. Repeat step 2 for $i = 1, \dots, n$.

4. Repeat steps 2, 3 until no errors.

Fact: Perceptron is proven to converge if the linear seperability assumption is true.