MATH 390.4 Lecture 3

Pizon Shetu

February 4 2020

*Recall

$$* = y = t(z) = f(x) + \delta = h * (x) + \epsilon$$

where epsilon is "Error"

1 Supervised Learning

<u>Supervised Learning</u>: has 3 ingredients

1. Training Data:

$$D =< x, y >$$

$$X = [x1, x2, ..., xn], \mathbf{Y} = [y1, y2, ..., yn] xi \in \mathbf{X}, yi \in \mathbf{Y}$$

2. **H**: a candidate set of functions

3. **A**, an algorithm which takes in data **D** and set **H** and produces a model **g** where

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

Question: is

$$f \in \mathbf{H}$$

generally? **NO**

*However there is a

$$\mathbf{h^*} \in \mathbf{H}$$

which is the closet possible model (function) to f

$$* = h * (x) + (f(x) - h * (x)) + (t(z) - f(x))$$

where

$$(f(x) - h * (x)) = \delta$$

and

$$(t(z) - f(x)) = \epsilon$$

Just because h* element of, H does not mean A will locate it. A will not be perfect and the value of Epsilon will confuse A. This g != h*, g is the best A can do.

$$Y = g(x) + (h * (x) - g(x)) + (f(x) - h * (x)) + (t(z) - f(x))$$

where g(x) is your model

(h*(x)-g(x)) is your estimation error

(f(x)-h*(x)) + (t(z)-f(x)) is your epsilon error where

(t(z)-f(x)) is your delta error and

(f(x)-h*(x)) is your mis-specification error

y* = g(x) where y* is the prediction of y in setting x

e = y-y* residual if x element of D (training data) otherwise they are unknown

<u>How to reduce errors</u>

1. Delta, ignorance error can be reduced by measuring more Xj's (features) of the units that contain information about Z.

2. Misspecification error can be reduced by expanded H to include more complicated functions

3. Estimation error can be reduced by increasing sample size

Example:

This is like 1. Of supervised learning Mortgages loan Y = 0,1 where 0 did not pay back and 1 paid back the loan P = 1 x1 is credit score X = [300,850]

D = ¡x,y¿ = [810,390,750. . . ...] [1,0,1. . . ....] this is data set of the relation of credit score and if they paid back the loan or not

This is like 2 in supervised learning trying to make H a set of candidate functions

$$\mathbb{1}$$

is the indicator function where

$$\mathbb{1}(w) = 1 \text{ if w} \in A$$

$$0 \text{ if w} \notin A$$

$$H = \mathbb{1}x >= \theta : \theta \in \Theta$$

$\theta =$ is a parameter sometimes denoted as β or w. . . and others.

$\Theta =$ is the set of all parameters

e.g.

$$g(x) = \mathbb{1}x >= 515.3$$

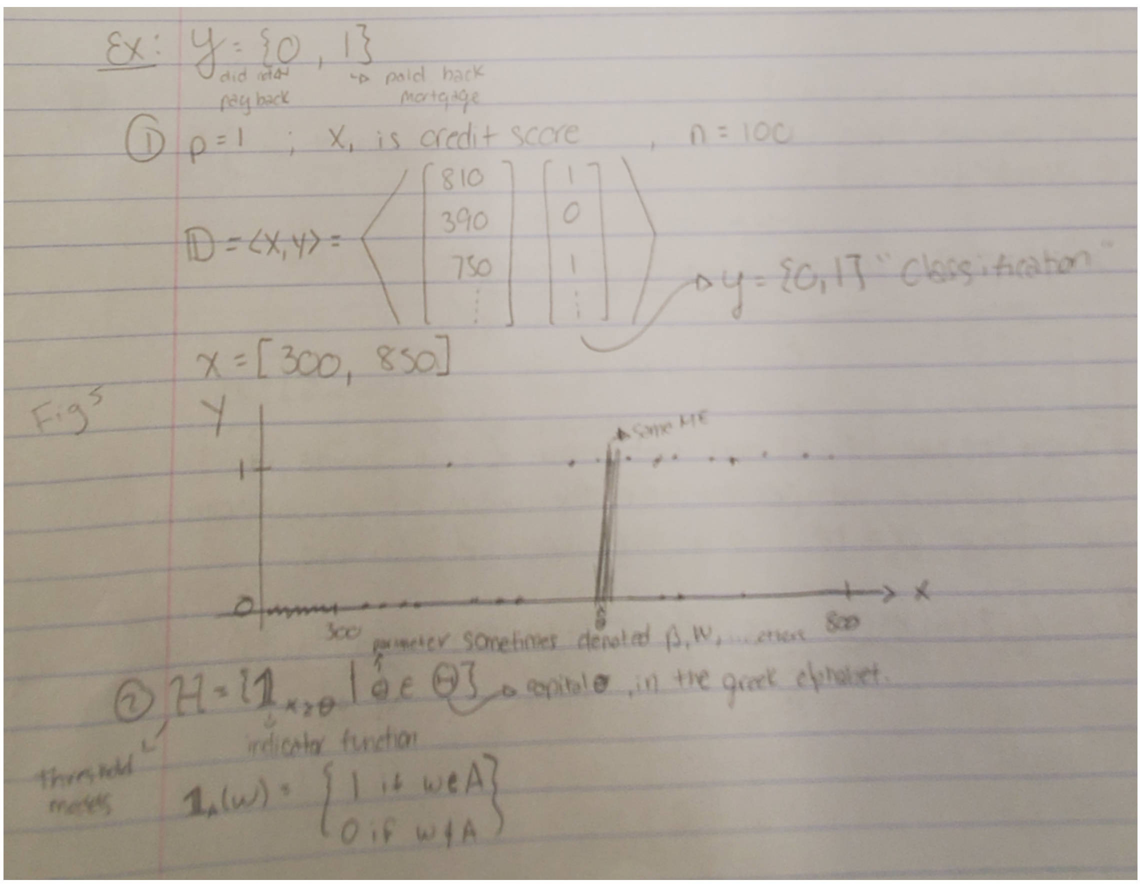
or

$$g(x) = \mathbb{1}x >= 407.9$$

3. This is 3 of supervised learning

Algorithm A is a estimation of theta $\sum_{i=1}^{n}$

First lets define "Misclassification error" ME:= 1/n $\sum_{i=1}^{n} \mathbb{1}g(xi)! = yi$



$$\text{Accuracy } ACC := 1 - ME$$

A: will minimize ME over $\theta \in \Theta$ Where $\theta \in$ Unique $x's$

1/n $\sum_{i=1}^{n} |yi - yi'| = 1/n \sum ei^2$

$1/n \sum ei^2$ is the mean squarred error(MSE) where

$\sum ei^2$ is the sum squred error (SSE)

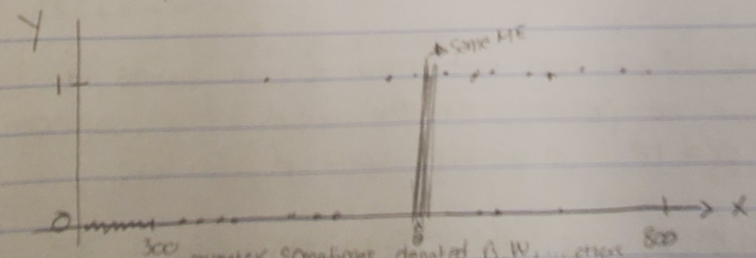
Ex: $\underbrace{\mathcal{Y} = \{0}_{\text{did not pay back}}, \underbrace{1\}}_{\text{paid back mortgage}}$

① $p = 1$; X_1 is credit score , $n = 100$

$$\mathbb{D} = \langle X, Y \rangle = \left\langle \begin{bmatrix} 810 \\ 390 \\ 750 \\ \vdots \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix} \right\rangle \rightarrow \mathcal{Y} = \{0, 1\} \text{ "Classification"}$$

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

Fig 5



parameter sometimes denoted β, w, ... etc

② $\mathcal{H} = \{\underbrace{\mathbb{1}_{x \geq \theta}}_{\text{indicator function}} \mid \theta \in \Theta\}$ → capital θ, in the greek alphabet.

Threshold models

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

$\rightarrow$ PH = 3(# of columns ; x)

$$X = \begin{bmatrix} \vdots & \\ \vec{1} & x \\ \vdots & \end{bmatrix}$$

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

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

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

This is an "over-parameterized" model, where each line has infinite $\vec{w}$'s that specify it

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

Assume that 0's and 1's are linearly seperable there exists w's such that $g(\vec{x})$ has no error

Perceptron Learning Algorithm (1957)

① Initialize $\vec{W}^{t=0} = \vec{0}$ or random . compute $\hat{y}$

② for $j = 0, 1, \ldots, p$ let, . . .

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

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

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

$$\vdots$$

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

recognize $\hat{y}$

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

③ Repeat step 2 for $i = 1 \ldots n$

④ Repeat steps 2 & 3 until no errors

Perceptron is proven to converge if the linear separability assumption is true