

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

Supervised Learning: has 3 ingredients

1. Training Data:

$$D = \langle x, y \rangle$$

$$X = [x_1, x_2, \dots, x_n], Y = [y_1, y_2, \dots, y_n] x_i \in \mathbf{X}, y_i \in \mathbf{Y}$$

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

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

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

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

where

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

and

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

Just because $h^* \in H$ does not mean A will locate it. A will not be perfect and the value of ϵ will confuse A . This $g \neq h^*$, 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

$$(h^*(x) - g(x)) \text{ is your estimation error}$$

$$(f(x) - h^*(x)) + (t(z) - f(x)) \text{ is your epsilon error where}$$

$$(t(z) - f(x)) \text{ is your delta error and}$$

$$(f(x) - h^*(x)) \text{ is your mis-specification error}$$

$$y^* = g(x) \text{ where } y^* \text{ is the prediction of } y \text{ in setting } x$$

$$e = y - y^* \text{ residual if } x \text{ element of } D \text{ (training data) otherwise they are unknown}$$

How to reduce errors

1. Delta, ignorance error can be reduced by measuring more X_j '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 $X = [300,850]$

$$D = \langle x, y \rangle = [810, 390, 750, \dots][1, 0, 1, \dots]$$

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

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

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

$$H = \{x \mapsto \theta : \theta \in \Theta\}$$

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

Θ = is the set of all parameters

e.g.

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

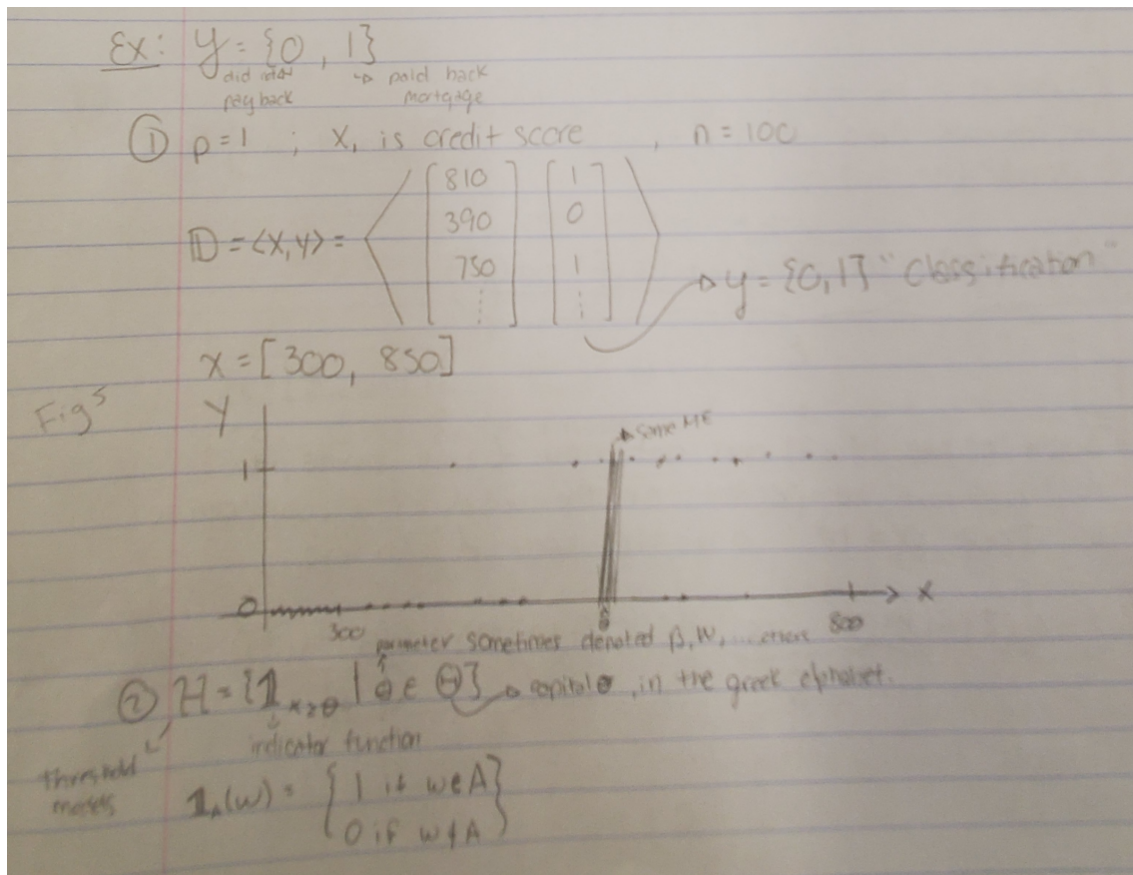
or

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

3. This is 3 of supervised learning

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

First lets define "Misclassification error" $ME := 1/n \sum_{i=1}^n \mathbb{1}_{g(x_i) \neq y_i}$



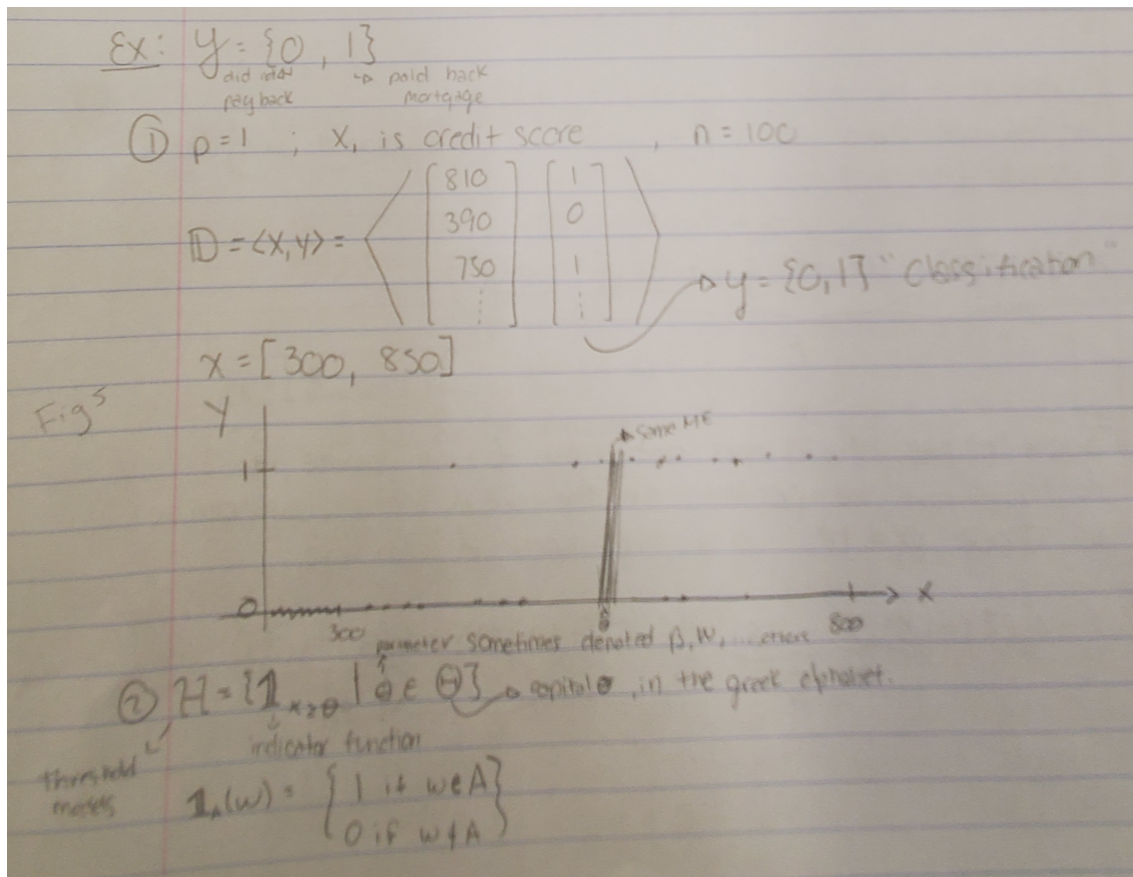
$$\text{Accuracy } ACC := 1 - ME$$

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

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

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

$\sum e_i^2$ is the sum squared error (SSE)



$$\Rightarrow PH = 3(\# \text{ of columns in } x)$$

$$X = \begin{bmatrix} \vdots \\ \vec{x} \\ \vdots \end{bmatrix} \quad \begin{array}{l} \text{redefine the matrix } X \\ \text{by appending a column of} \\ \text{1's on the left} \end{array}$$

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

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

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

Need Algorithm A $g = A(D, H)$
 Assume that 0's and 1's are linearly separable 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 $j=0, 1, \dots, p$: let, \dots

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

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

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

\vdots

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

regress \hat{y}

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

③ Repeat step 2 for $i=1 \dots n$

④ Repeat steps 2 & 3 until no errors

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