

Linear Regression

Mahdi Roozbahani
Georgia Tech

EVERY GROUP PROJECT



**DOES 99%
OF THE WORK**

**HAS NO IDEA
WHAT'S GOING
ON THE
WHOLE TIME**

**SAYS HE'S
GOING TO
HELP
BUT HE'S
NOT**

**DISAPPEAR
AT THE VERY
BEGINNING AND
DOESN'T SHOW
UP AGAIN TIL
THE VERY END**

Outline

- Supervised Learning 
- Linear Regression
- Extension

$X_{1000 \times 20}$

Supervised Learning: Overview

$X \rightsquigarrow$ $\begin{bmatrix} 80\% \\ \text{training} \\ 800 \end{bmatrix}$ $\begin{bmatrix} 20\% \\ \text{testing} \\ 200 \end{bmatrix}$ OR $\begin{bmatrix} \text{training} \\ 70\% \end{bmatrix}$ $\begin{bmatrix} \text{validation} \\ 10\% \\ \text{hyper parameter} \end{bmatrix}$ $\begin{bmatrix} \text{testing} \\ 20\% \end{bmatrix}$

Functions $\mathcal{F}$ Training data

$$f: \mathcal{X} \rightarrow \mathcal{Y}$$

$$\{(x^{i\}}, y^{i\}) \in \mathcal{X} \times \mathcal{Y}\}$$

$$X = \begin{bmatrix} x_1 & x_d \\ \vdots & \vdots \end{bmatrix}_{n \times d} \quad Y = \begin{bmatrix} y_a \\ \vdots \end{bmatrix}_{n \times 1}$$

LEARNING

$$\begin{array}{l} \text{find } \hat{f} \in \mathcal{F} \\ \text{s.t. } y_a \approx \underbrace{f(x^{i\})}_{\hat{y}_p} \end{array}$$

ML model



Learning machine

PREDICTION

$$\hat{y}_p = f(x^{i\})$$

New data

$$x$$

Supervised Learning: Two Types of Tasks

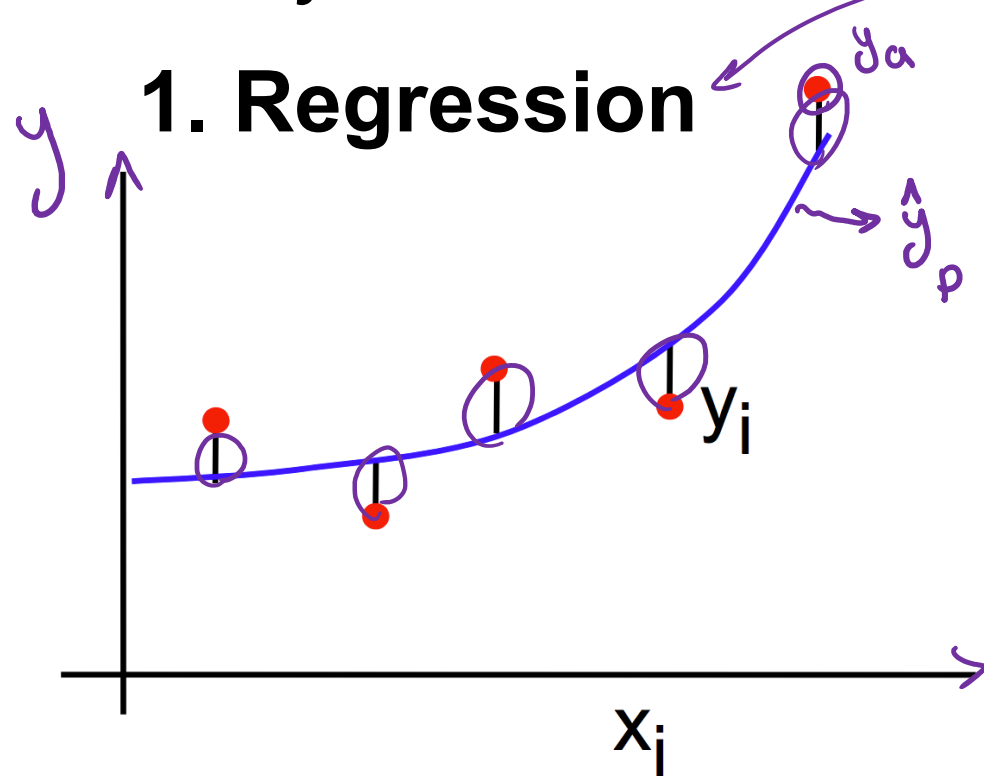
Given: training data $\{(x^{\{1\}}, y^{\{1\}}), (x^{\{2\}}, y^{\{2\}}), \dots, (x^{\{n\}}, y^{\{n\}})\}$

Learn: a function $f(\mathbf{x}) : y = f(\mathbf{x})$

$$X = \begin{bmatrix} & & & & \end{bmatrix}_{n \times d} \quad Y = \begin{bmatrix} 27.5 \\ 26 \\ \vdots \\ i \end{bmatrix}_{n \times 1} \quad Y = \begin{bmatrix} \text{cat} \\ \text{dog} \\ \vdots \end{bmatrix}_{n \times 1}$$

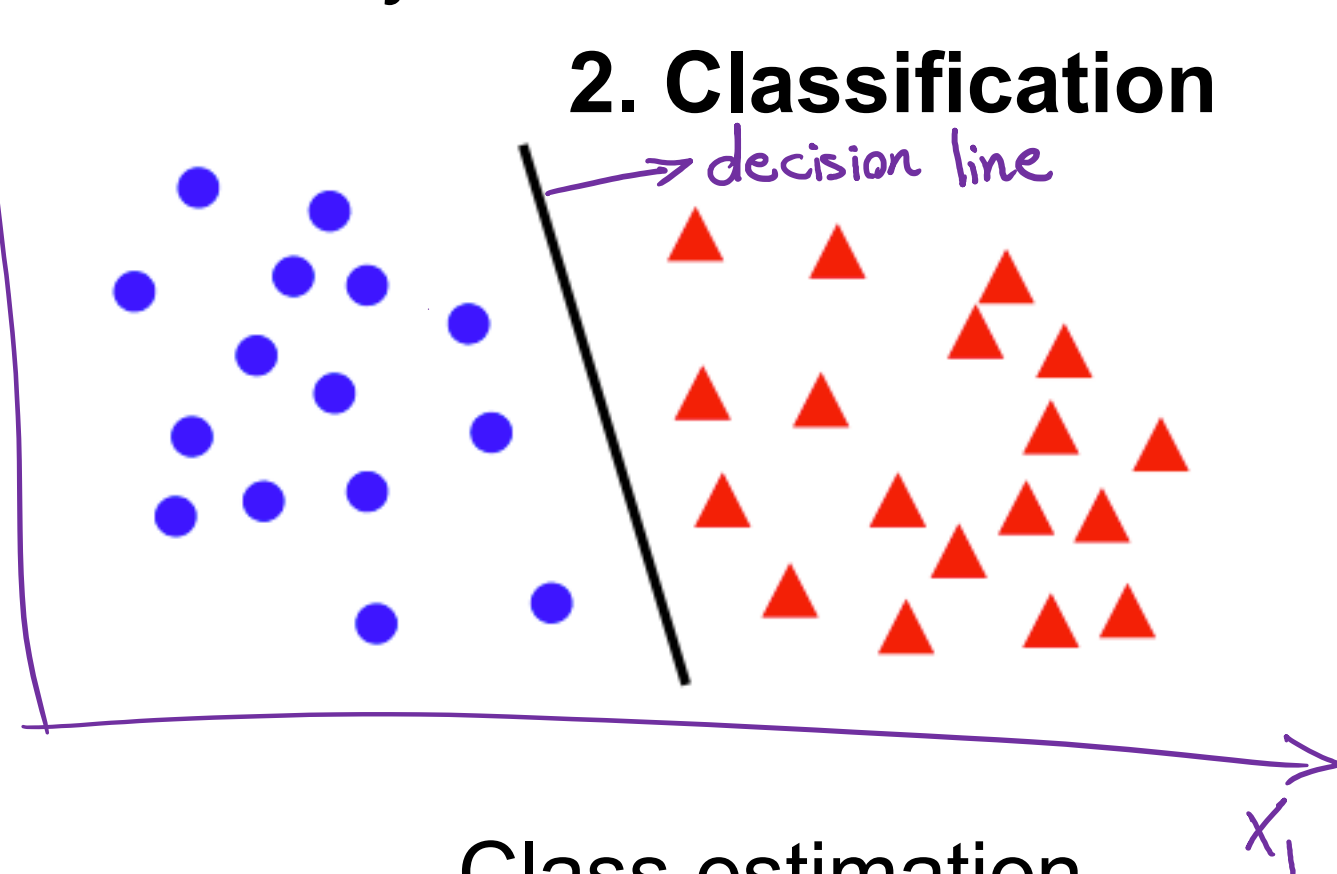
When y is continuous:

When y is discrete:



Curve fitting

Minimize the difference between y_a & $\hat{y}_p$



Class estimation

Classification Example 1: Handwritten digit recognition

As a supervised classification problem

Start with training data, e.g. 6000 examples of each digit

0 0 0 1 1 1 1 1 1 2

2 2 2 2 2 2 2 3 3 3

3 4 4 4 4 4 5 5 5 5

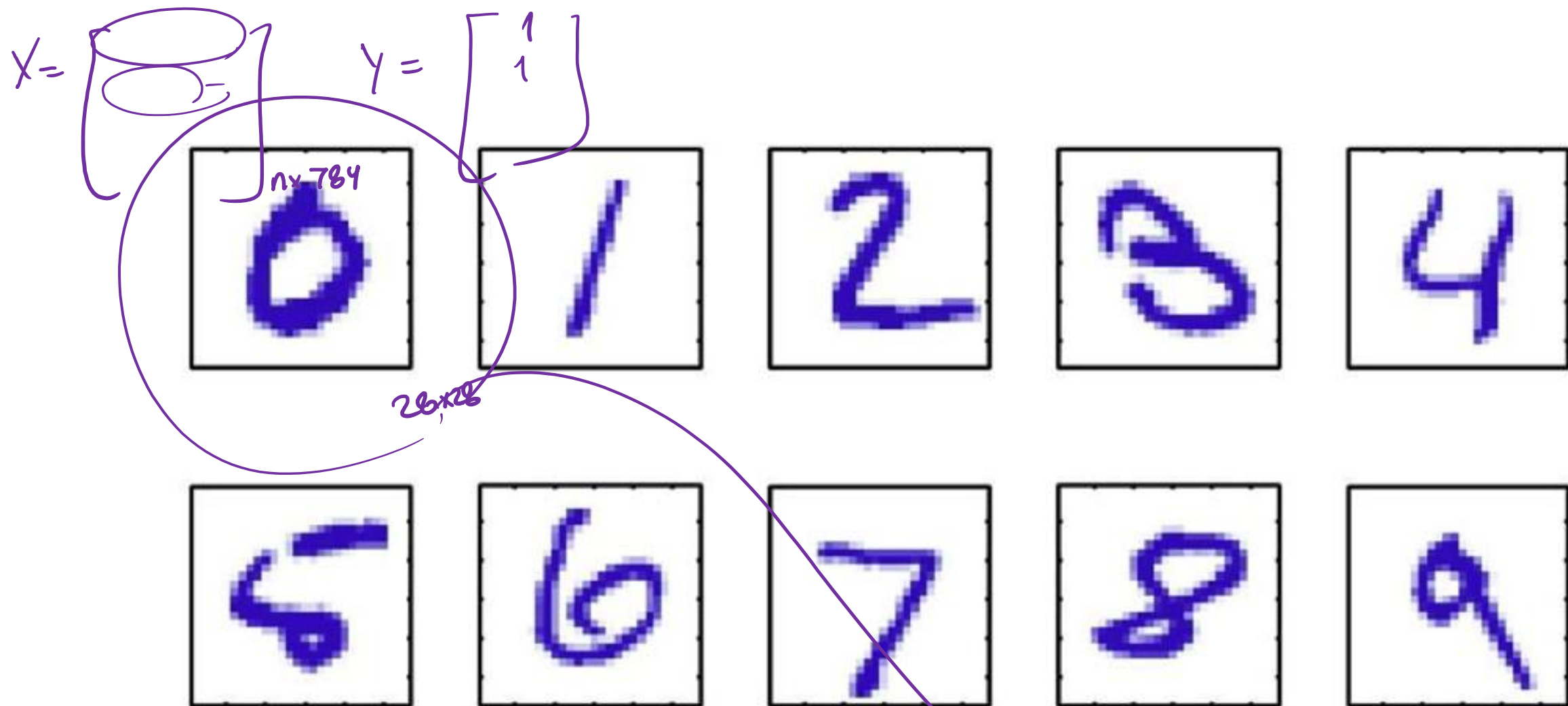
6 6 7 7 7 7 7 8 8 8

8 8 8 8 8 9 9 9 9

Multi class classification

- Can achieve testing error of 0.4%
- One of first commercial and widely used ML systems (for zip codes & checks)

Classification Example 1: Hand-Written Digit Recognition



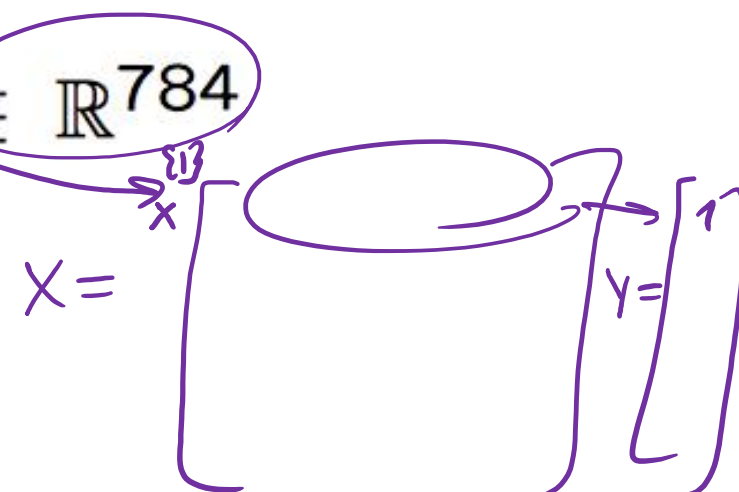
Images are 28 x 28 pixels

A classification problem

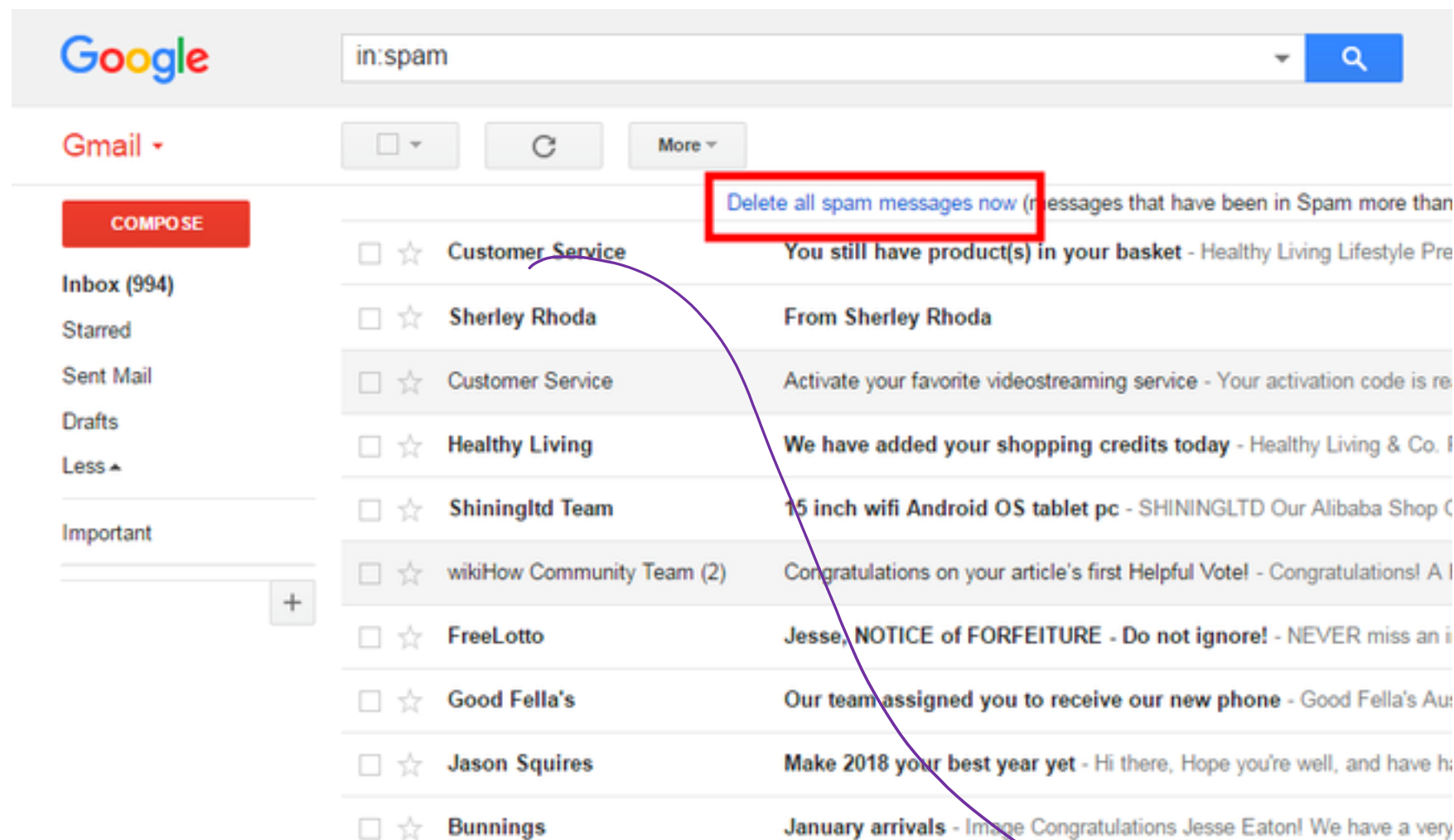
Represent input image as a vector $\mathbf{x} \in \mathbb{R}^{784}$

Learn a classifier $f(\mathbf{x})$ such that,

$$f : \mathbf{x} \rightarrow \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

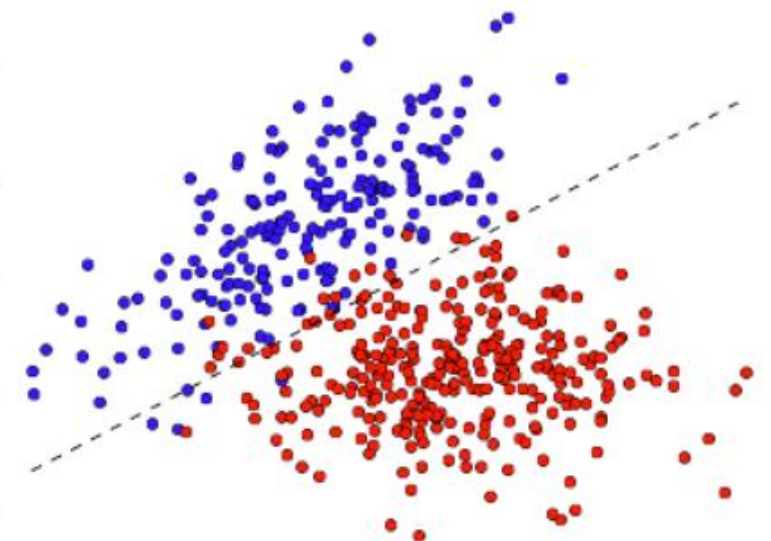


Classification Example 2: Spam Detection



Binary

NOT SPAM



SPAM

A classification problem

- This is a classification problem
- Task is to classify email into spam/non-spam
- Data x_i is word count
- Requires a learning system as “enemy” keeps innovating

BoW

one Email
 $X =$



$Y = \begin{bmatrix} S \\ NS \end{bmatrix}$

Regression Example 1: Apartment Rent Prediction

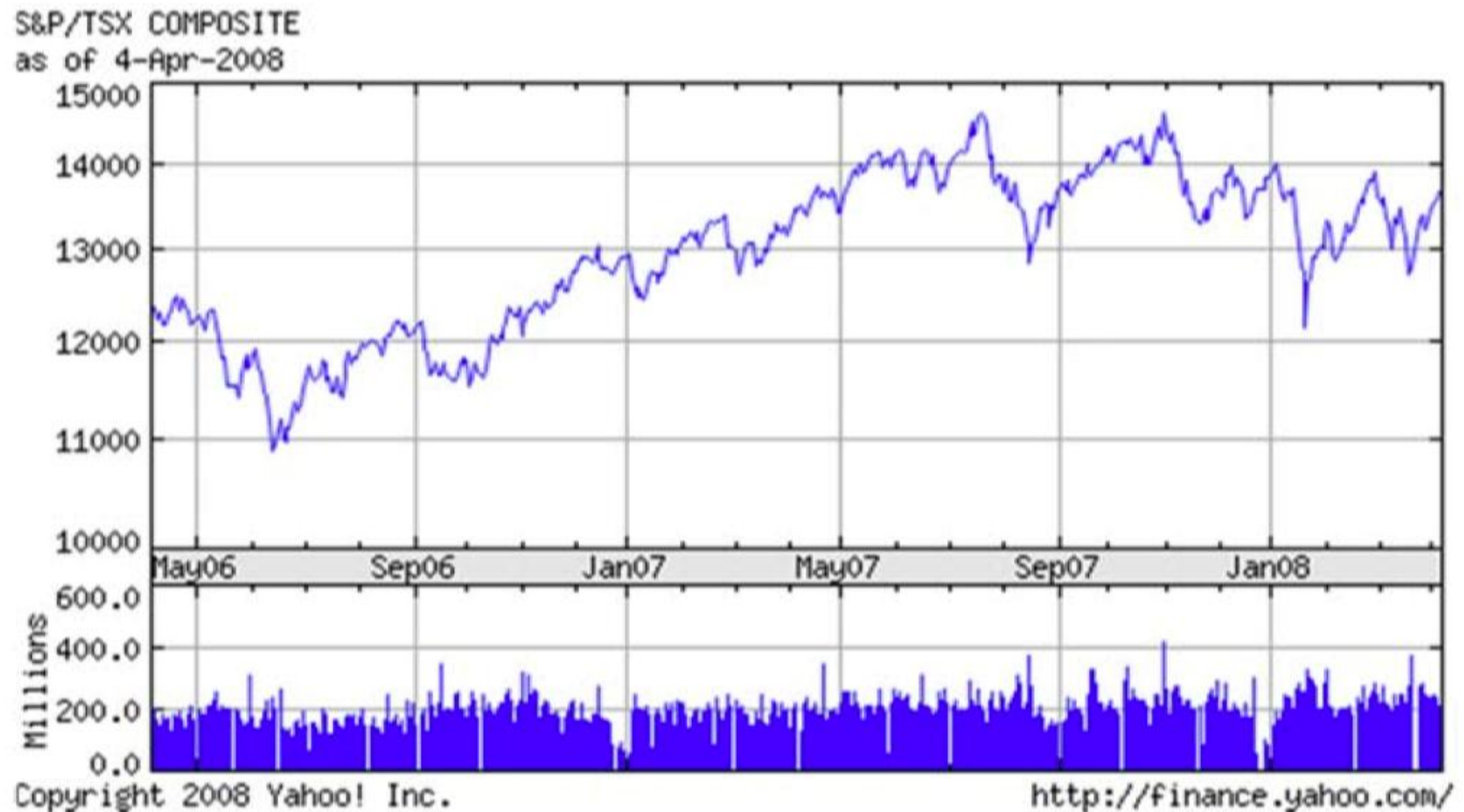
- Suppose you are to move to Atlanta
- And you want to find the **most reasonably priced** apartment satisfying your **needs**:

square-ft., # of bedroom, distance to campus ...

A regression problem

Living area (ft ²)	# bedroom	Rent (\$)
230	1	600
506	2	1000
433	2	1100
109	1	500
...		
150	1	?
270	1.5	?

Regression Example 2: Stock Price Prediction



- Task is to predict stock price at future date

A regression problem

$$y = \underbrace{m}_\text{slope} x + \underbrace{b}_\text{Intercept} \Rightarrow \hat{y}_p = f(x) = \underbrace{\theta_0}_\text{bias term} + \underbrace{\theta_1}_{\text{parameter}} (x_1) \rightarrow \text{the parameter of feature } \underline{1}$$

Features:

- Living area, distance to campus, # bedroom ...

- Denote as $x = (x_1, x_2, \dots, x_d)$

$$X^{(1)} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \end{bmatrix}$$

$$\Theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}_{1 \times d+1}$$

Target:

- Rent

$$LCF = \hat{y}_p = X^{(1)} \cdot \Theta$$

- Denoted as y

$$\hat{y}_p = \theta_0 + \theta_1 x_1^{(1)} + \dots + \theta_d x_d^{(1)} = X \Theta$$

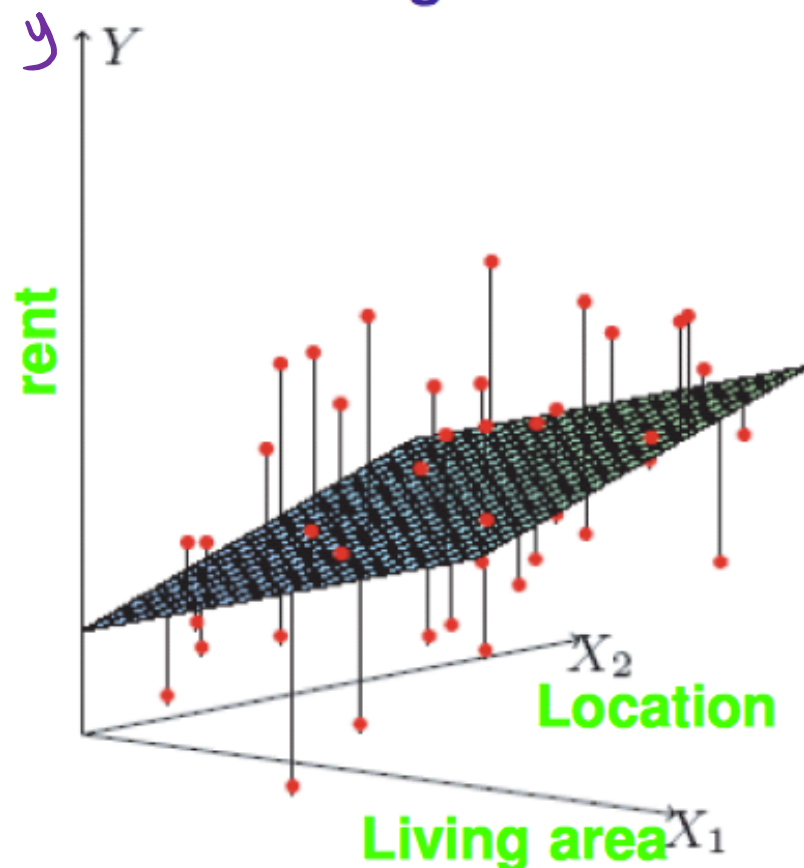
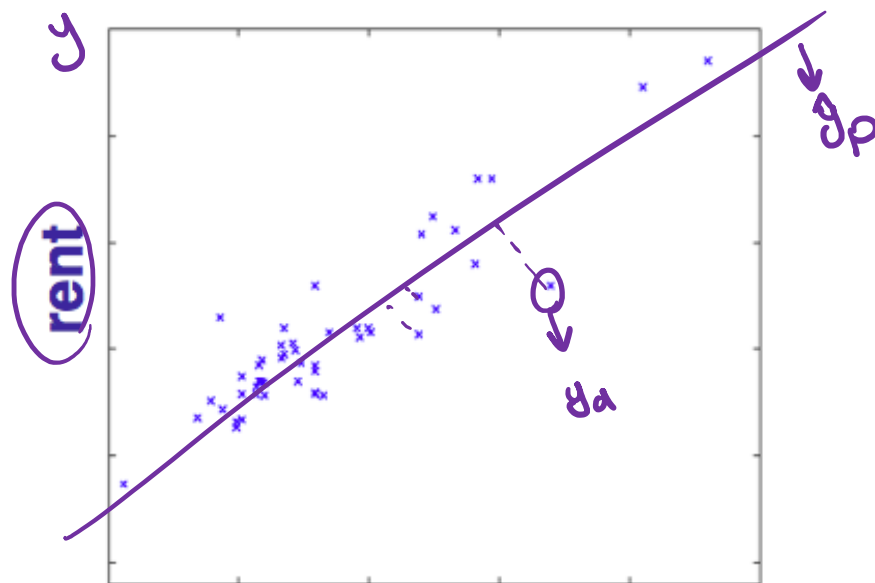
$$\boxed{\hat{y}_p = X \Theta} \rightarrow LCF$$

Training set:

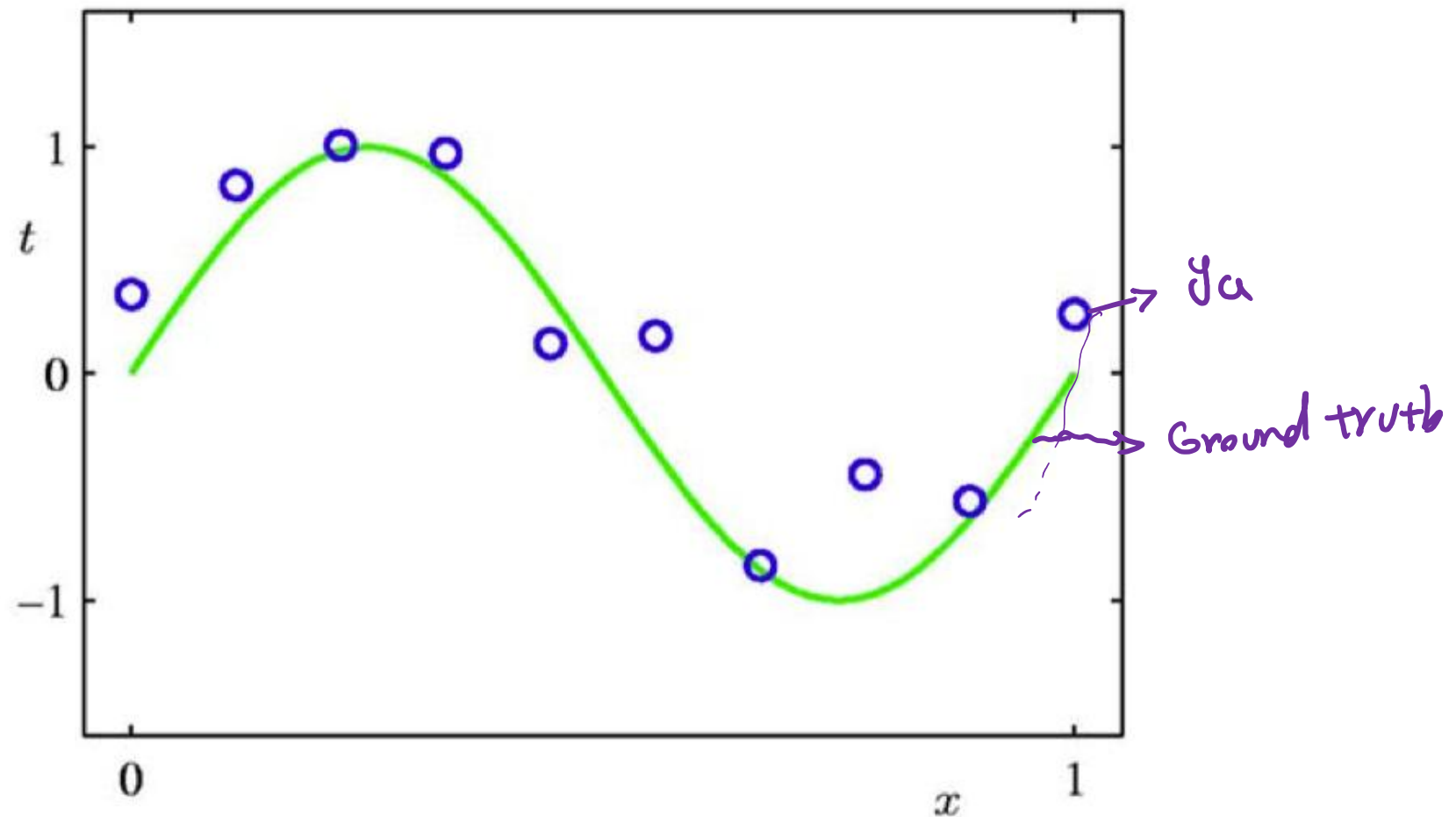
$$x = \{x^{(1)}, x^{(2)}, \dots, x^{(n)}\} \in R^d$$

$$y = \{y^{(1)}, y^{(2)}, \dots, y^{(n)}\}$$

$$\hat{y}_p = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \rightarrow LCF \text{ linear combination of features}$$



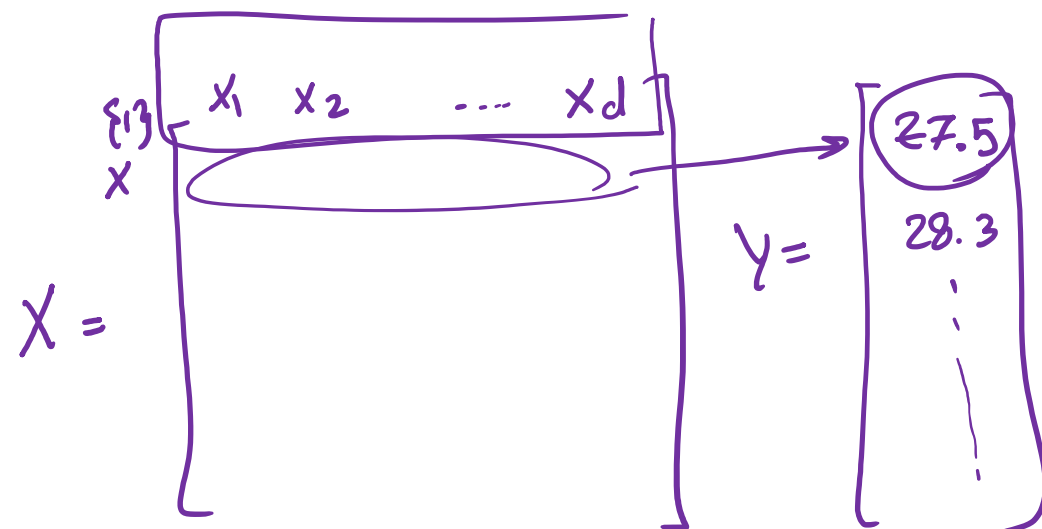
Regression: Problem Setup



- Suppose we are given a training set of N observations
- Regression problem is to estimate $y(x)$ from this data

Outline

- Supervised Learning
- Linear Regression ←
- Extension



bias term

$$\underbrace{\theta_0}_{\text{bias term}} + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_d x_d \approx 27.5$$

LCF

$$\Theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_d \end{bmatrix}_{d+1 \times 1}$$

$$x^{(1)} = \begin{bmatrix} 1 & x_1 & \dots & x_d \end{bmatrix}$$

$$\hat{y}_p = X\Theta$$

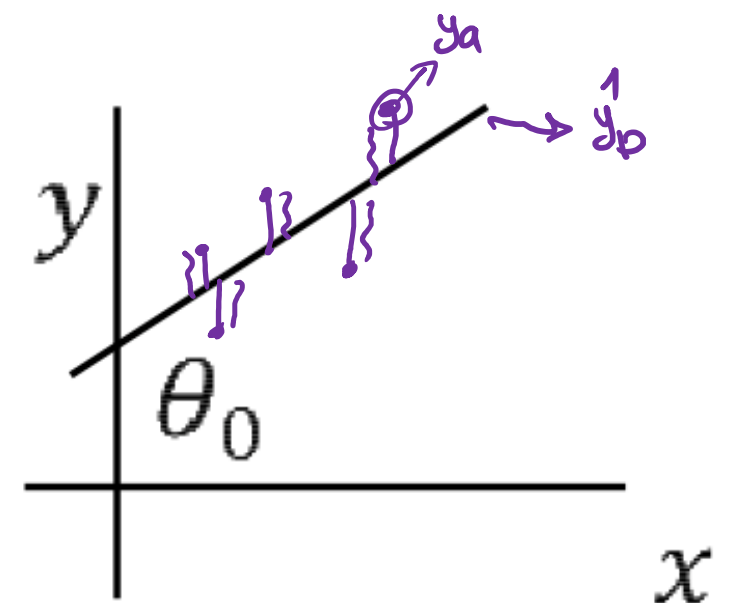
Linear Regression

- Assume y is a linear function of x (features) plus noise ϵ

$$y = \theta_0 + \theta_1 x_1 + \dots + \theta_d x_d + \epsilon$$

- where ϵ is an error term of unmodeled effects or random noise
- Let $\theta = (\theta_0, \theta_1, \dots, \theta_d)^T$, and augment data by one dimension

- Then $y = x\theta + \epsilon$



$X_{n \times d+1}$ $y_{n \times 1}$ $\theta_{d+1 \times 1}$

Convex 

Least Mean Square Method

- Given n data points, find θ that minimizes the mean square error

Training $\hat{\theta} = \underset{\theta \in \mathbb{R}^d}{\operatorname{argmin}} \underbrace{L(\theta)}_{\text{loss}} = \frac{1}{n} \sum_{i=1}^N \underbrace{(y^{(i)} - x^{(i)} \theta)^2}_{\text{Error } E(\theta)}$

$E[(y^{(i)} - x^{(i)} \theta)^2]$

$|y^{(i)} - x^{(i)} \theta| \sim \checkmark$

- Our usual trick: set gradient to 0 and find parameter $\frac{\partial L(\theta)}{\partial \theta} = 0$

$$\frac{\partial L(\theta)}{\partial \theta} = -\frac{2}{n} \sum_{i=1}^N (x^{(i)})^T (y^{(i)} - x^{(i)} \theta) = 0$$

$$\frac{\partial L(\theta)}{\partial \theta} = -\frac{2}{n} \sum_{i=1}^N (x^{(i)})^T y^{(i)} + \frac{2}{n} \sum_{i=1}^N (x^{(i)})^T x^{(i)} \theta = 0$$

$$\underline{X}_{n \times d+1} \quad Y_{n \times 1} \quad \underline{\Theta}_{d+1 \times 1}$$

$$\cancel{\frac{1}{N} \sum_{i=1}^N (y_a^{\{i\}} - x^{\{i\}} \Theta)^2} = \frac{1}{N} \underbrace{\underbrace{(Y - X\Theta)}_{n \times 1}^T}_{1 \times n} \underbrace{\underbrace{(Y - X\Theta)}_{n \times 1}}_{n \times 1}$$

$$y = \underbrace{X\Theta}_{n \times d+1} \Rightarrow \Theta = X^+ y$$

Matrix form

$$x = \begin{bmatrix} 1 & x_1^{\{1\}} & \dots & x_d^{\{1\}} \\ 1 & x_1^{\{2\}} & \ddots & x_d^{\{2\}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{\{n\}} & \dots & x_d^{\{n\}} \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}$$

$n \times (d+1)$
 $n \times 1$
 $(d+1) \times 1$

$$MSE(\theta) = \operatorname{argmin}_{\theta} L(\theta) = \frac{1}{n} (y - x\theta)^T (y - x\theta)$$

$$x\theta = \begin{bmatrix} \theta_0 + \theta_1 x_1^{\{1\}} + \theta_2 x_2^{\{1\}} + \dots + \theta_d x_d^{\{1\}} \\ \theta_0 + \theta_1 x_1^{\{2\}} + \theta_2 x_2^{\{2\}} + \dots + \theta_d x_d^{\{2\}} \\ \vdots \\ \theta_0 + \theta_1 x_1^{\{n\}} + \theta_2 x_2^{\{n\}} + \dots + \theta_d x_d^{\{n\}} \end{bmatrix}_{n \times 1}$$

LCF for y_1

Matrix Version and Optimization

$$\frac{\partial L(\theta)}{\partial \theta} = \left[-\frac{2}{n} \sum_{i=1}^N (x^{\{i\}})^T y^{\{i\}} \right] + \left[\frac{2}{n} \sum_{i=1}^N (x^{\{i\}})^T x^{\{i\}} \theta \right] = 0$$

Let's rewrite it as:

$$\frac{\partial L(\theta)}{\partial \theta} = -\frac{2}{n} (x^{\{1\}}, \dots, x^{\{n\}})^T (y^{\{1\}}, \dots, y^{\{n\}}) + \frac{2}{n} (x^{\{1\}}, \dots, x^{\{n\}})^T (x^{\{1\}}, \dots, x^{\{n\}}) \theta = 0$$

Define $X = (x^{\{1\}}, \dots, x^{\{n\}})$ and $y = (y^{\{1\}}, \dots, y^{\{n\}})$

$$\frac{\partial L(\theta)}{\partial \theta} = \left[-\frac{2}{n} X^T y \right] + \left[\frac{2}{n} X^T X \theta \right] = 0$$

$\Rightarrow \overset{d+1 \times 1}{\theta} = \underbrace{(X^T X)^{-1}}_{d+1 \times d+1} \underbrace{X^T}_{d+1 \times n} \underbrace{y}_{n \times 1} = X^+ y$

X^+ is the **pseudo-inverse** of X
 $X^T X X^+ = X^T$

$$\theta = (X^T X)^{-1} X^T y = X^+ y$$

$X_{n \times d}$

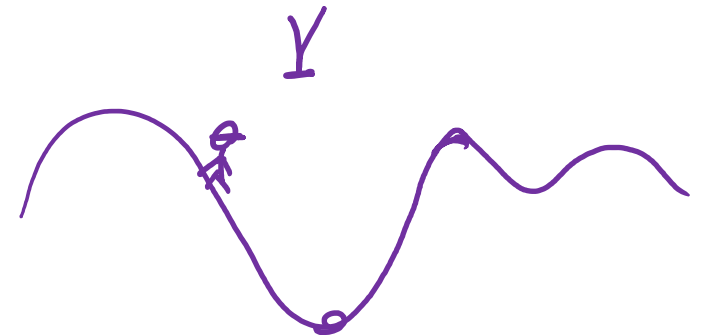
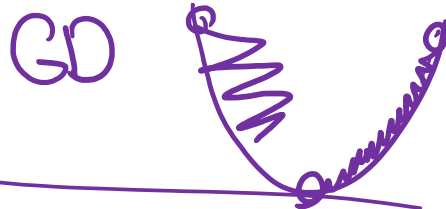
$n = \text{instances}$ $d = \text{dimension}$

$$X^T X = \left[\begin{array}{c} d \times n \end{array} \right] \left[\begin{array}{c} n \times d \end{array} \right] = \left[\begin{array}{c} d \times d \end{array} \right]$$

Not a big matrix because $n \gg d$ This matrix is invertible most of the times. If we are VERY unlucky and columns of $X^T X$ are not linearly independent (it's not a full rank matrix), then it is not invertible.

Alternative Way to Optimize

- The matrix inversion in $\hat{\theta} = (X^T X)^{-1} X^T y$ can be very expensive to compute



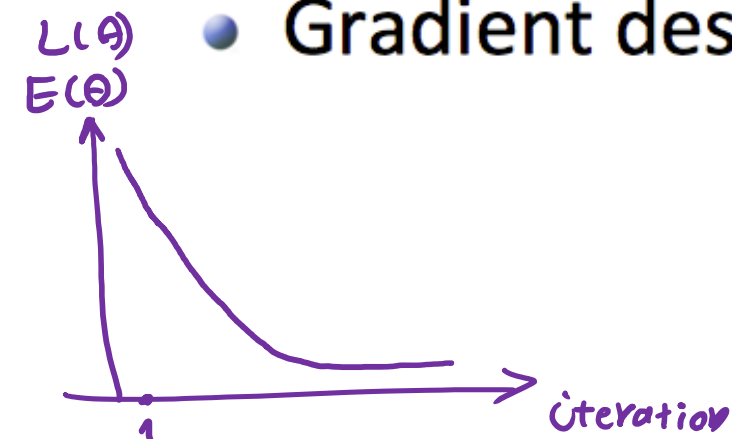
$$\frac{\partial L(\theta)}{\partial \theta} = -\frac{2}{n} \sum_{i=1}^N (x^{(i)})^T (y^{(i)} - x^{(i)} \theta)$$

$$\theta^{t+1} \leftarrow \theta^t - \alpha \left(\frac{\partial L(\theta)}{\partial \theta} \right)$$

- Gradient descent

$$\hat{\theta}^{t+1} \leftarrow \hat{\theta}^t + \frac{\alpha}{n} \sum_{i=1}^N (x^{(i)})^T (y^{(i)} - x^{(i)} \theta)$$

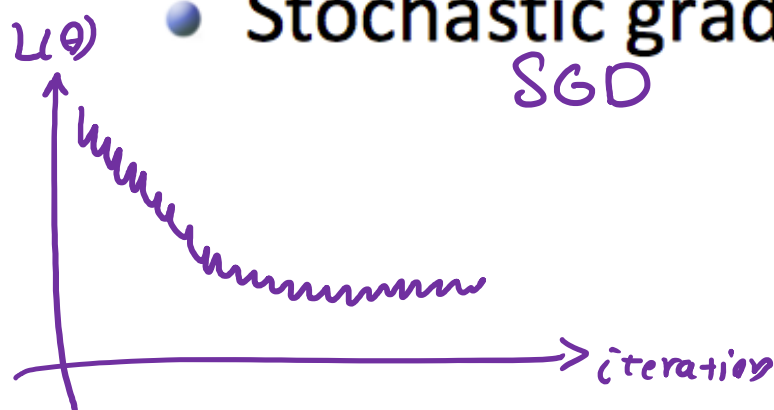
FGD



- Stochastic gradient descent (use one data point at a time)

SGD

$$\hat{\theta}^{t+1} \leftarrow \hat{\theta}^t + \beta_t \times (x^{(i)})^T (y^{(i)} - x^{(i)} \theta)$$



Recap

- Stochastic gradient update rule

$$\hat{\theta}^{t+1} \leftarrow \hat{\theta}^t + \beta_t \times (x^{\{i\}})^T (y^{\{i\}} - x^{\{i\}} \theta)$$

SGD



BGD



- Pros: on-line, low per-step cost
- Cons: coordinate, maybe slow-converging

- Gradient descent

$$\hat{\theta}^{t+1} \leftarrow \hat{\theta}^t + \frac{\alpha}{n} \sum_{i=1}^N (x^{\{i\}})^T (y^{\{i\}} - x^{\{i\}} \theta)$$

FGD

- Pros: fast-converging, easy to implement
- Cons: need to read all data

- Solve normal equations

$$\theta = (X^T X)^{-1} X^T y$$

- Pros: a single-shot algorithm! Easiest to implement.
- Cons: need to compute inverse $(X^T X)^{-1}$, expensive, numerical issues (e.g., matrix is singular ..)

Linear regression for classification

Raw Input $x = (1, x_1, \dots, x_{256})$

Linear model $(\theta_0, \theta_1, \dots, \theta_{256})$

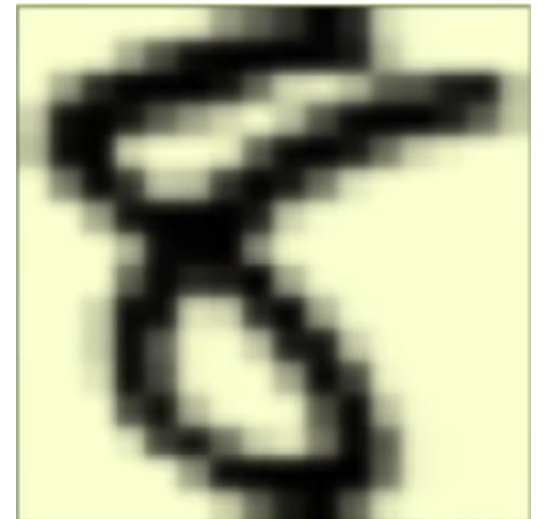
"Feature engineering"

Extract useful information

intensity and symmetry $x = (1, \underline{x_1}, \underline{x_2})$

Sum up all the pixels = intensity

Symmetry = -(difference between flip version)



16

16

5 1

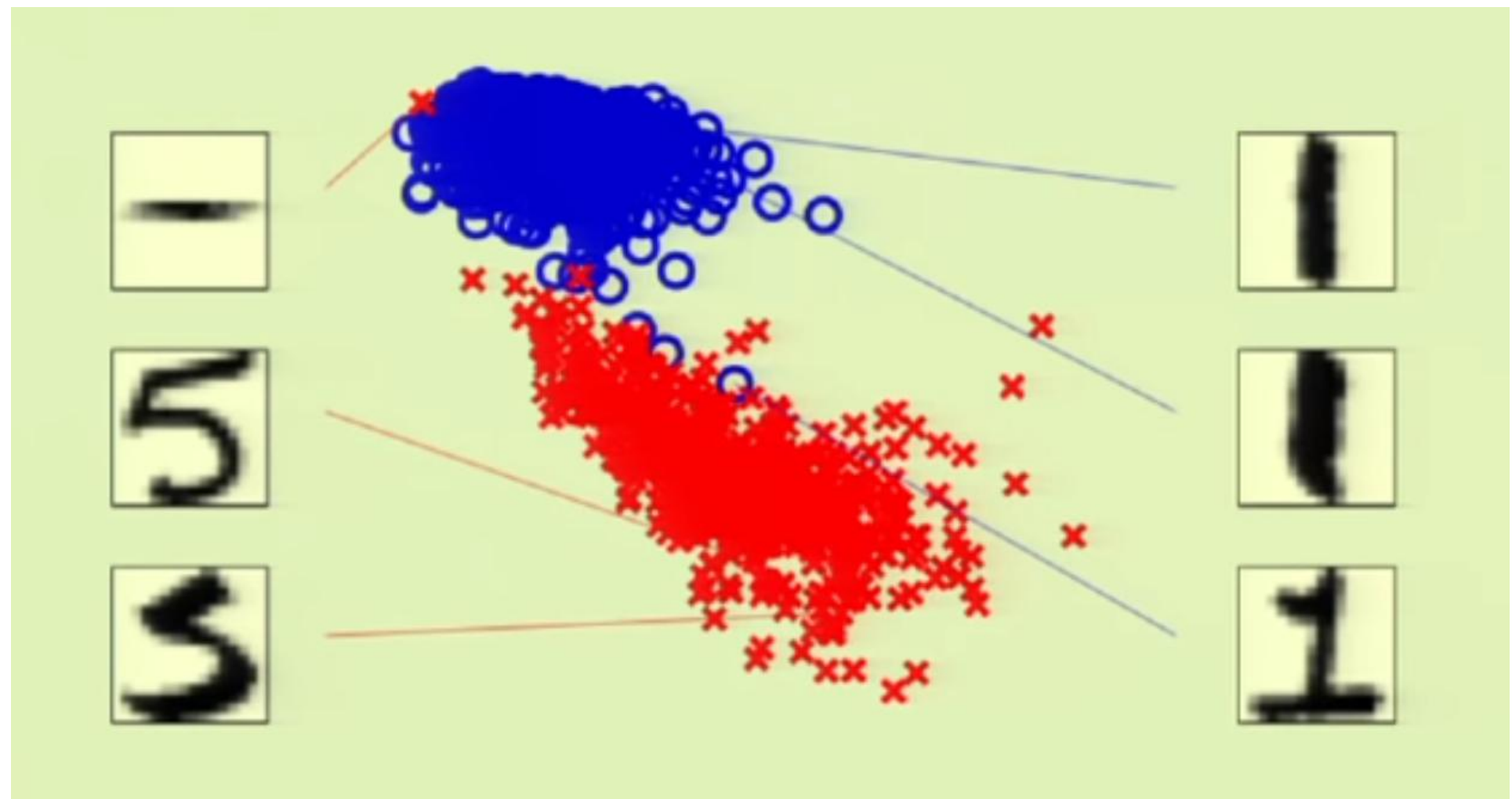
1 ↓

$$x = (1, x_1, x_2)$$

$$x_1 = \text{intensity} \quad x_2 = \text{symmetry}$$

$$y = \sum_{i=1}^d x_i \theta_i \in \mathbb{R}$$

It is almost linearly separable



Linear regression for classification

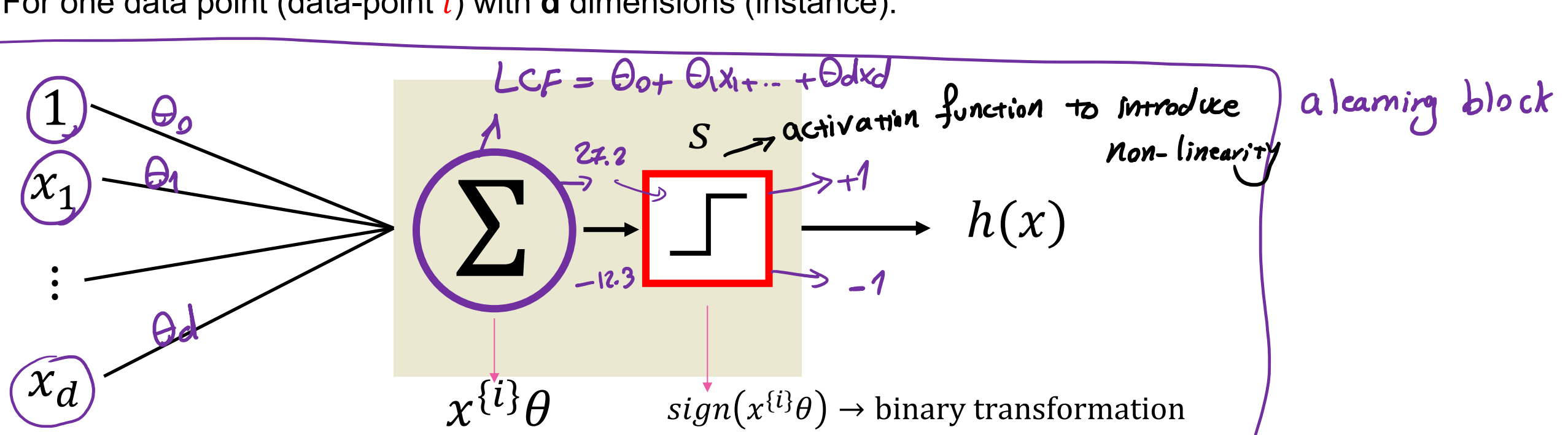
Binary-valued functions are also real-valued $\pm 1 \in \mathbb{R}$

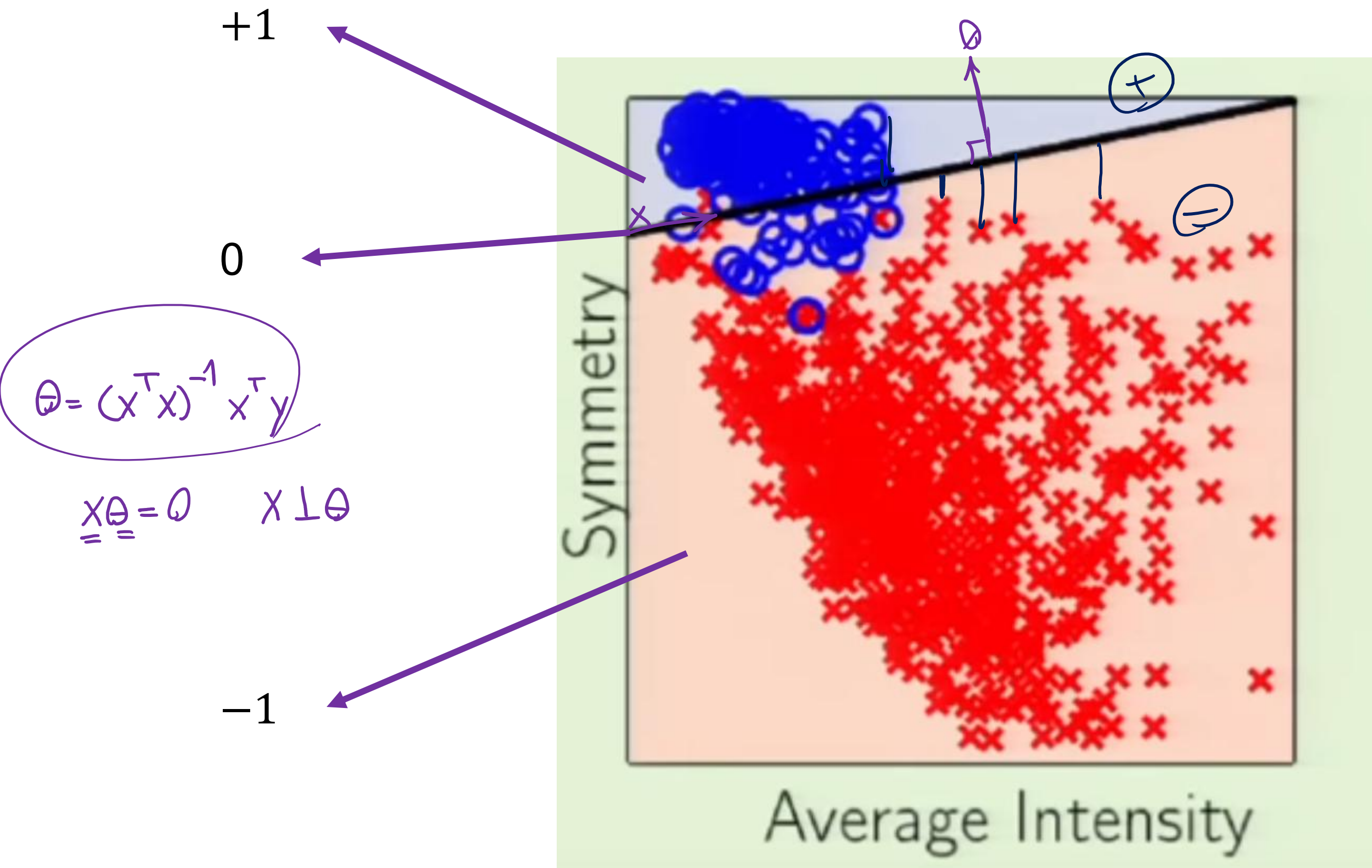
Use linear regression $x^{(i)}\theta \approx y^{(i)} = \pm 1$ $i = \text{index of a data-point}$

Let's calculate, $\text{sign}(x^{(i)}\theta) = \begin{cases} -1 & x^{(i)}\theta < 0 \\ 0 & x^{(i)}\theta = 0 \\ 1 & x^{(i)}\theta > 0 \end{cases}$

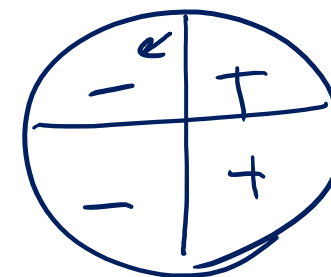
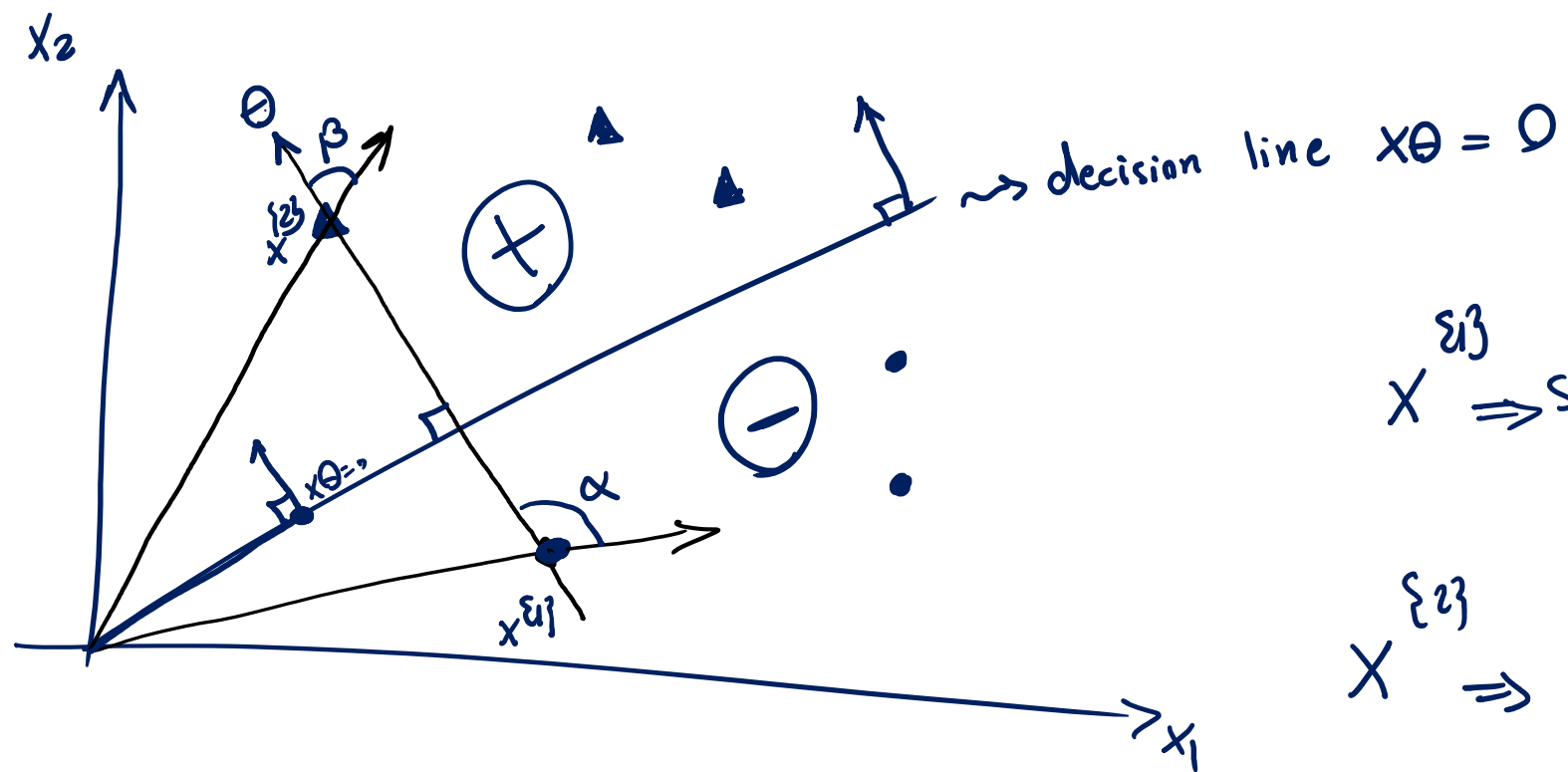
Handwritten notes: $+1 \rightarrow 5$, $-1 \rightarrow 1$

For one data point (data-point i) with d dimensions (instance):





Not really the best for classification, but t's a good start



$$x^{(1)} \Rightarrow \text{Sign} \left(\underbrace{x^{(1)} \theta}_{+} = \underbrace{\|x^{(1)}\| \| \theta \| \cos \alpha}_{-} \right) = -1$$

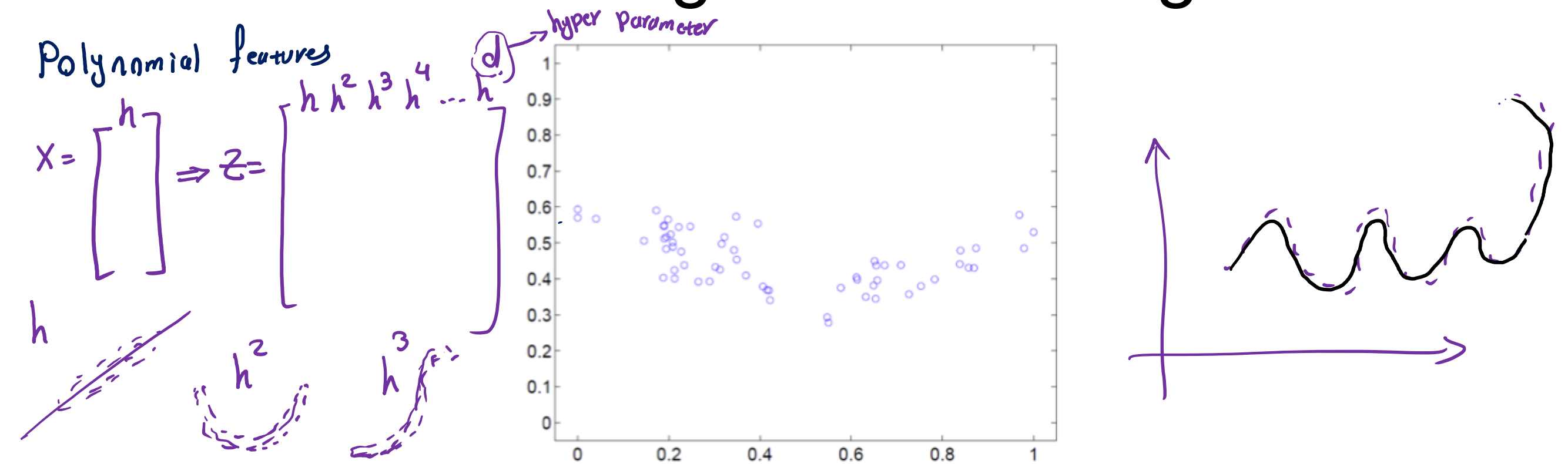
$$x^{(2)} \Rightarrow \text{Sign} \left(\underbrace{\|x^{(2)}\| \| \theta \| \cos \beta}_{+} \right) = +1$$

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \rightarrow \text{plane}$$

Outline

- Supervised Learning
- Linear Regression
- Extension 

Extension to Higher-Order Regression



- Want to fit a polynomial regression model

$$y = \theta_0 + \theta_1 \underbrace{x}_{z_1} + \theta_2 \underbrace{x^2}_{z_2} + \dots + \theta_d \underbrace{x^d}_{z_d} + \epsilon$$

- $z = \{1, x, x^2, \dots, x^d\} \in R^d$ and $\theta = (\theta_0, \theta_1, \theta_2, \dots, \theta_d)^T$

$$y = z\theta$$

Least Mean Square Still Works the Same

- Given n data points, find θ that minimizes the mean square error

$$\hat{\theta} = \operatorname{argmin}_{\theta} L(\theta) = \frac{1}{n} \sum_{i=1}^N (y^{\{i\}} - \underline{z^{\{i\}}} \theta)^2$$

- Our usual trick: set gradient to 0 and find parameter

$$\frac{\partial L(\theta)}{\partial \theta} = -\frac{2}{n} \sum_{i=1}^N (z^{\{i\}})^T (y^{\{i\}} - z^{\{i\}} \theta) = 0$$

$$\frac{\partial L(\theta)}{\partial \theta} = -\frac{2}{n} \sum_{i=1}^N (z^{\{i\}})^T y^{\{i\}} + \frac{2}{n} \sum_{i=1}^N (z^{\{i\}})^T z^{\{i\}} \theta = 0$$

Matrix Version of the Gradient

$$z = \{1, x, x^2, \dots, x^d\} \in R^d \quad y = \{y^{\{1\}}, y^{\{2\}}, \dots, y^{\{n\}}\}$$

$$\frac{\partial L(\theta)}{\partial \theta} = -\frac{2}{n} z^T y + \frac{2}{n} z^T z \theta = 0$$

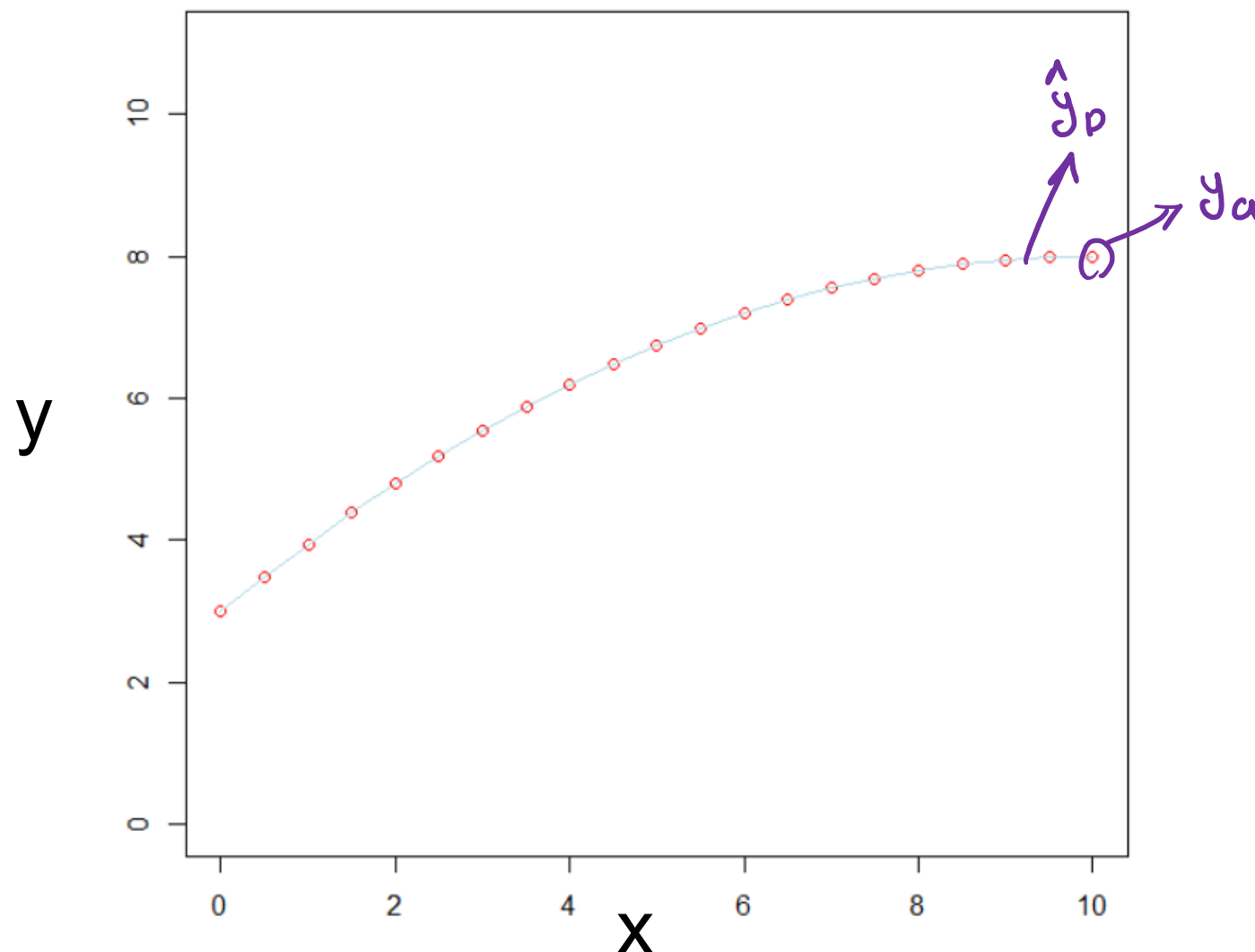
$$\Rightarrow \theta = (z^T z)^{-1} z^T y = z^+ y$$


z space

- If we choose a different maximal degree **d** for the polynomial, the solution will be different.

What is happening in polynomial regression?

$$\left. \begin{array}{l} x = [0, 0.5, 1, \dots, 9.5, 10] \\ y = [3, 3.4875, 3.95, \dots, 7.98, 8] \end{array} \right\} \begin{array}{l} f = \theta_0 + \theta_1 \underline{x} + \theta_2 \underline{x}^2 \\ \theta_0 = 3; \theta_1 = 1; \theta_2 = -0.5 \end{array}$$

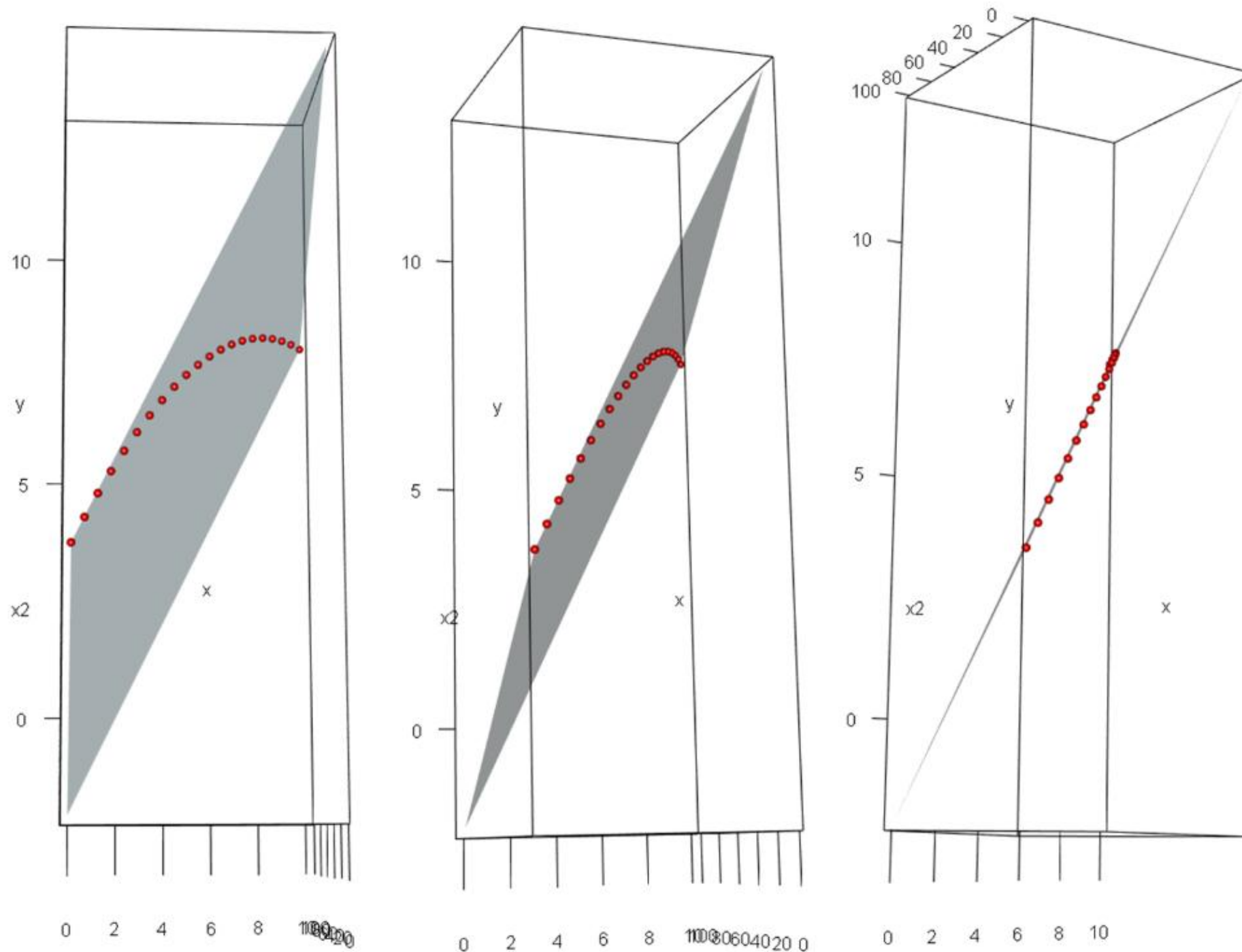


$$L(\theta) = 0$$

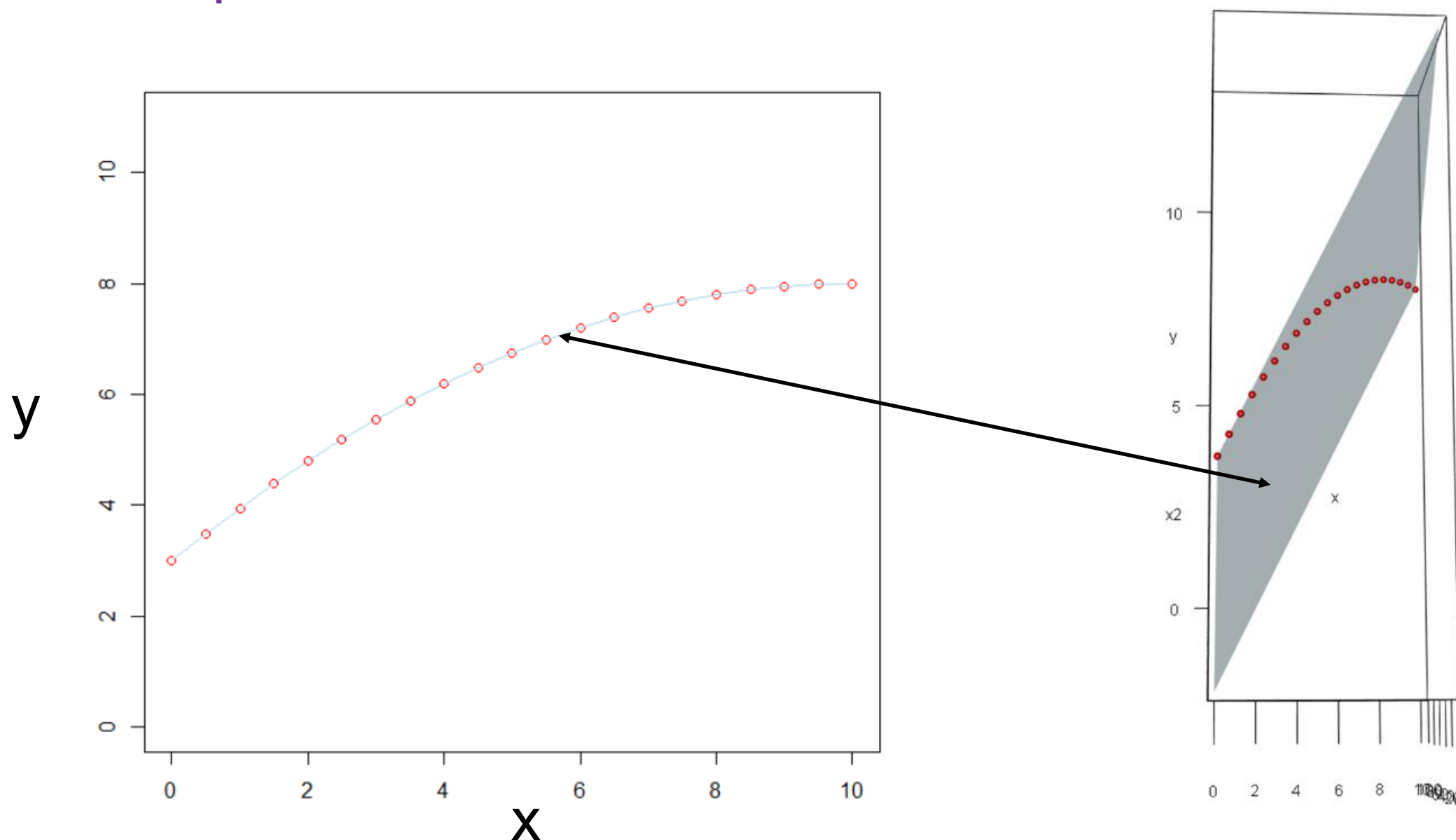
RMSE=0

Let's add to the feature space

$$x_1 = [0, 0.5, 1, \dots, 9.5, 10] \quad x_2^2 = [0, 0.25, 1, \dots, 90.25, 100]$$
$$y = [3, 3.4875, 3.95, \dots, 7.98, 8]$$



We are fitting a D -dimensional hyperplane in a $D+1$ dimensional hyperspace (in above example a 2D plane in a 3D space). That hyperplane really is 'flat' / 'linear' in 3D. It can be seen a non-linear regression (a curvy line) in our 2D example in fact it is a flat surface in 3D. So the fact that it is mentioned that the model is linear in parameters, it is shown here.

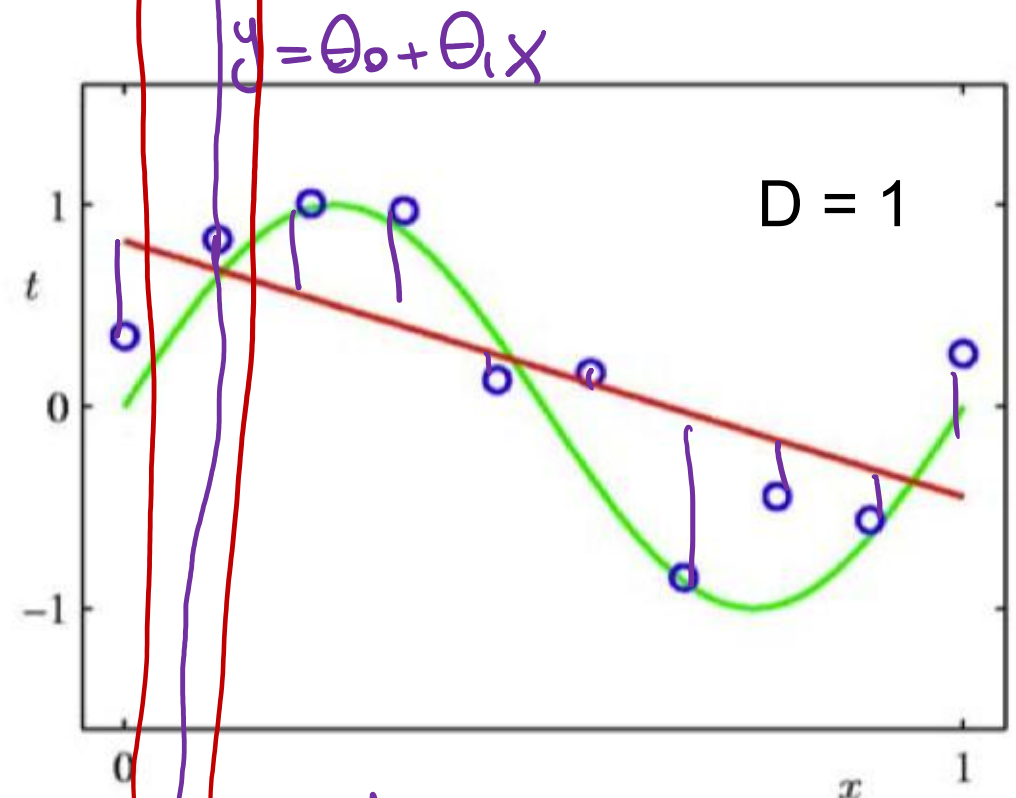
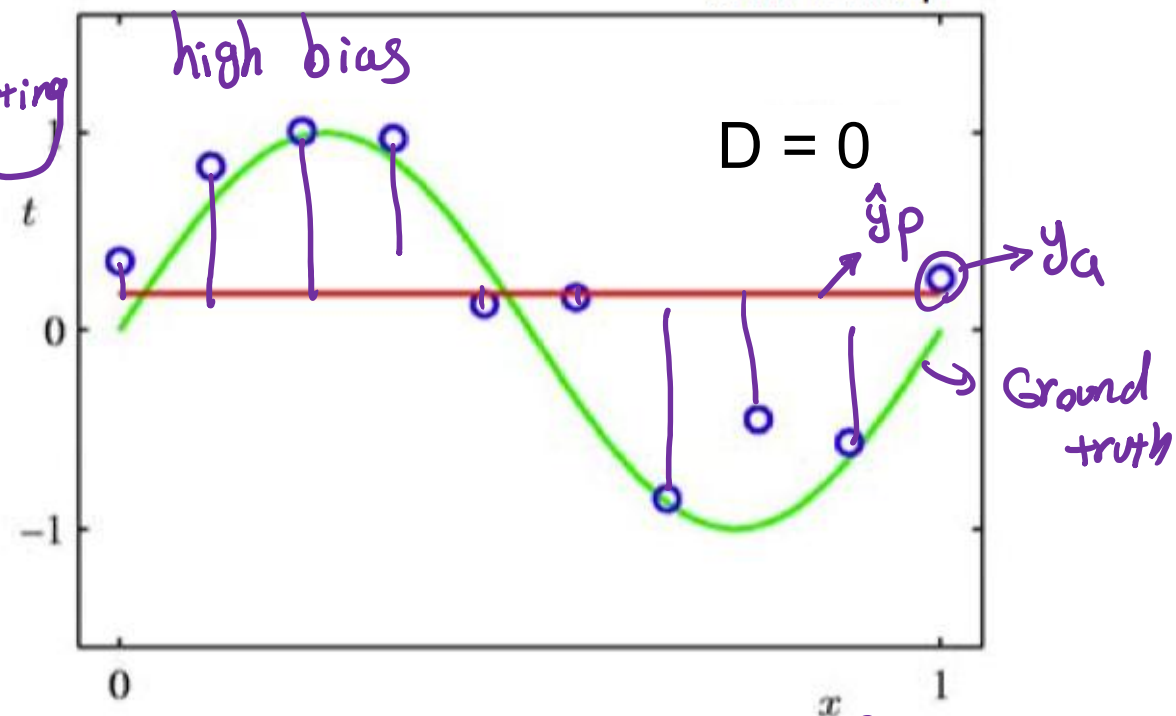


Increasing the Maximal Degree

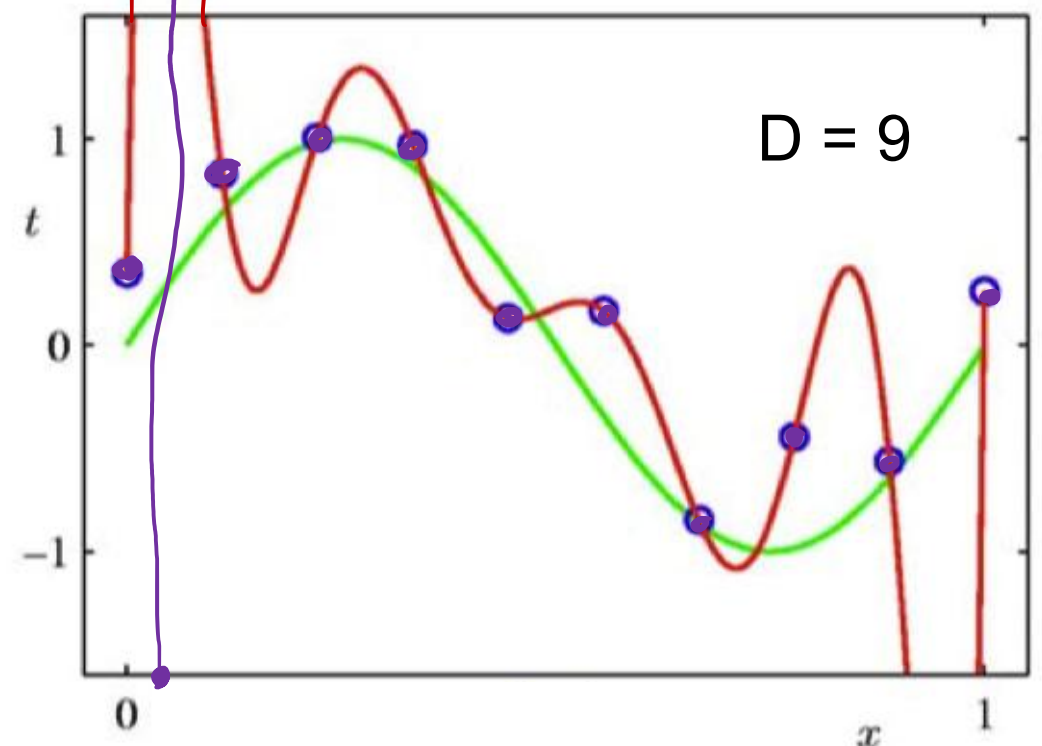
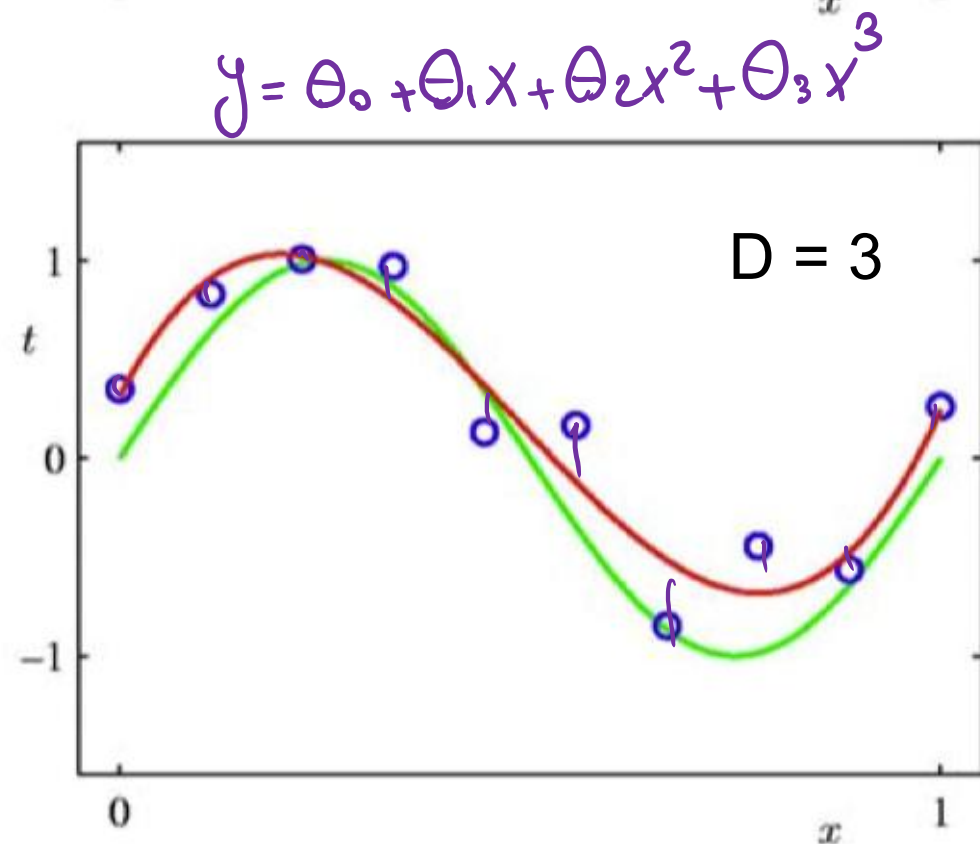
$$y = \theta_0 + \theta_1 x = \theta_0 + \theta_1$$

from Bishop

underfitting



high variance - Overfitting

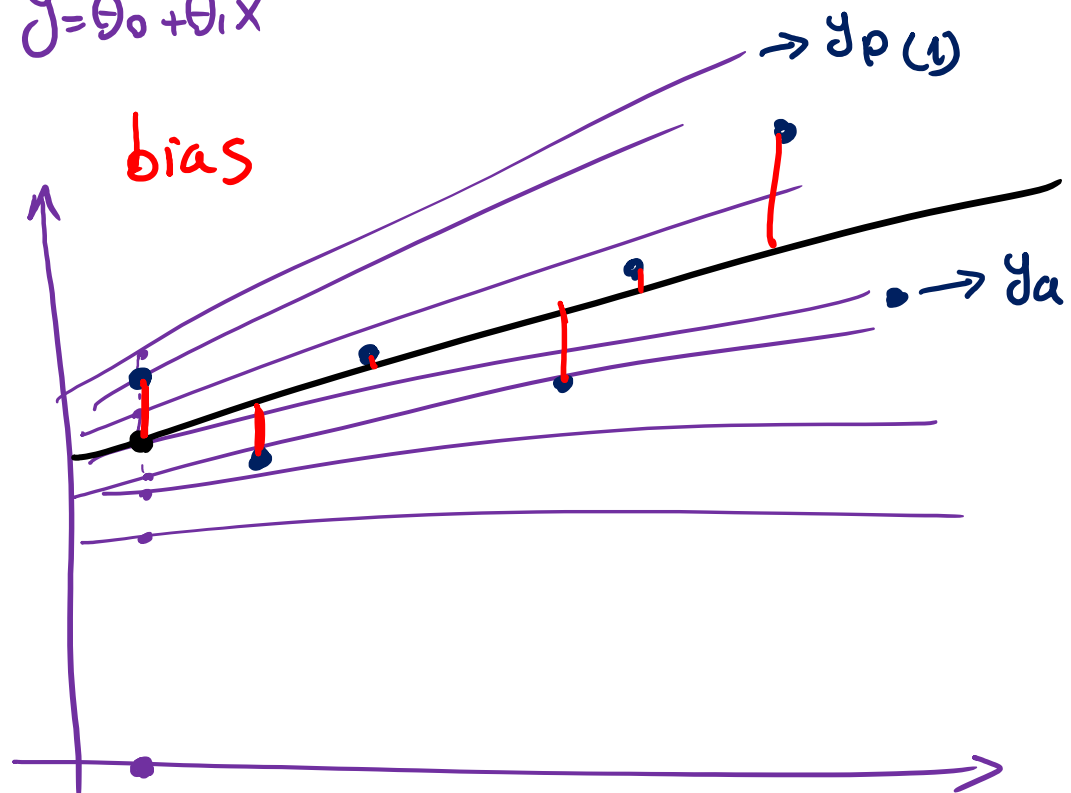


$$X = \begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix}_{1000 \times 10} \quad X_{(1)} = \begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix}_{100 \times 10} \quad \dots \quad X_{(10)} = \begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix}_{100 \times 10}$$

$\underbrace{\hspace{10em}}_{LR(1)} \qquad \qquad \qquad LR(10)$

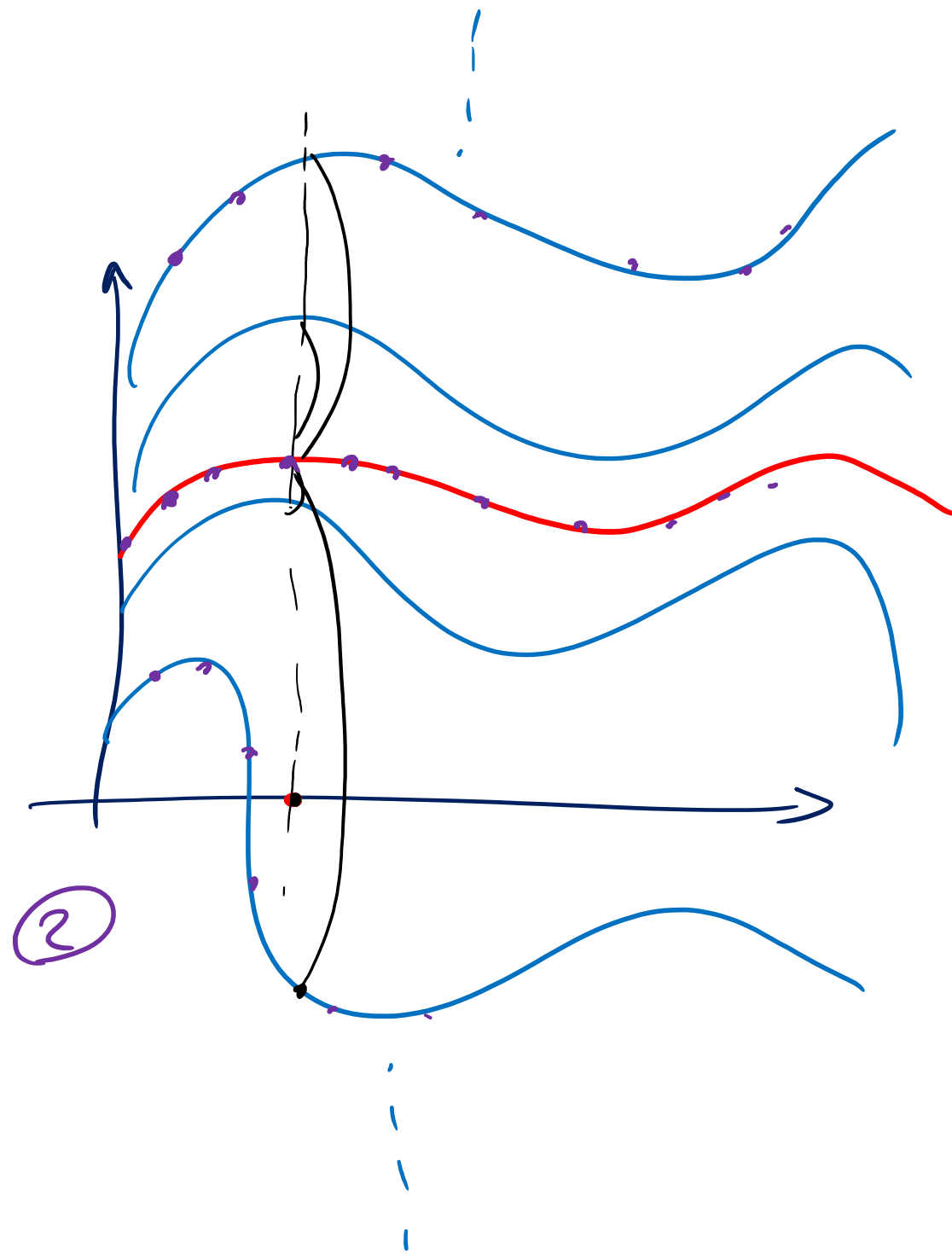
$$y = \theta_0 + \theta_1 x$$

bias



$$\frac{1}{10} \sum_{i=1}^{10} \hat{y}_{p(i)} = E[y_p] \in \mathbb{R}$$

①



Bias-Variance Trade off

[Animation](#)

We will have multiple prediction values (i.e. through Cross validation) $E[y_p]$

$$L(\theta) = \frac{1}{n} \sum_{i=1}^N (y^{\{i\}} - x^{\{i\}} \theta)^2 = \vec{E} \left[(\underbrace{y_a}_{\text{actual}} - \underbrace{y_p}_{\text{predicted}})^2 \right]$$

$$(\underbrace{y_a - y_p}_{\text{error}})^2 = (y_a - E[y_p] + E[y_p] - y_p)^2$$

$$= (y_a - E[y_p])^2 + (E[y_p] - y_p)^2 + 2(y_a - E[y_p])(E[y_p] - y_p)$$

$$E \left[\underbrace{(y_a - E[y_p])^2}_a + \underbrace{(E[y_p] - y_p)^2}_b + 2 \underbrace{(y_a - E[y_p])}_c \underbrace{(E[y_p] - y_p)}_d \right]$$

$E[a+b+c] = E[a] + E[b] + E[c]$ $E[ab] = E[a]E[b]$

$$= \cancel{E[(\underbrace{y_a}_{\text{Scalar}} - \underbrace{E[y_p]}_{\text{Scalar}})^2]} + E[(\underbrace{E[y_p]}_{\text{Scalar}} - \underbrace{y_p}_{\text{vector}})^2] + 2 \cancel{E[y_a - E[y_p]] E[E[y_p] - y_p]}$$

$$= (\underbrace{y_a}_{\text{Scalar}} - \underbrace{E[y_p]}_{\text{Scalar}})^2 + \frac{1}{N} \sum (\underbrace{E[y_p]}_{\text{Scalar}} - \underbrace{y_p}_{\text{vector}})^2 + 0$$

$$= \text{bias}^2 + \text{Variance}$$

$$E[E[y_p]] - E[y_p] = E[y_p] - E[y_p] = 0$$

$$L(\theta) = \text{bias}^2 + \text{Variance}$$

Bias-Variance Trade off

[Animation](#)

We will have multiple prediction values (i.e. through Cross validation) $E[y_p]$

$$L(\theta) = \frac{1}{n} \sum_{i=1}^N (y^{\{i\}} - x^{\{i\}} \theta)^2 = E \left[(y_a - y_p)^2 \right]$$

$$(y_a - y_p)^2 = (y_a - E[y_p] + E[y_p] - y_p)^2$$

$$= (y_a - E[y_p])^2 + (E[y_p] - y_p)^2 + 2(y_a - E[y_p])(E[y_p] - y_p)$$

$$E \left[(y_a - y_p)^2 \right] = (y_a - E[y_p])^2 + E \left[(E[y_p] - y_p)^2 \right]$$

$$= [Bias]^2 + Variance$$

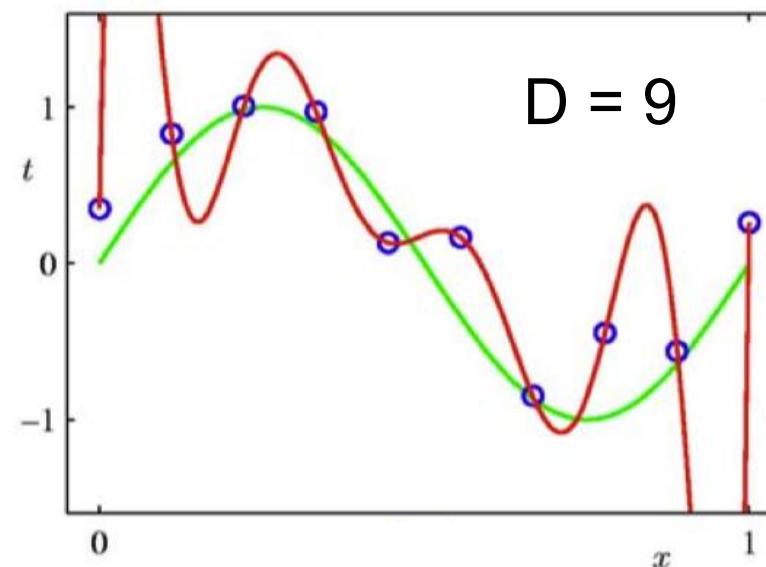
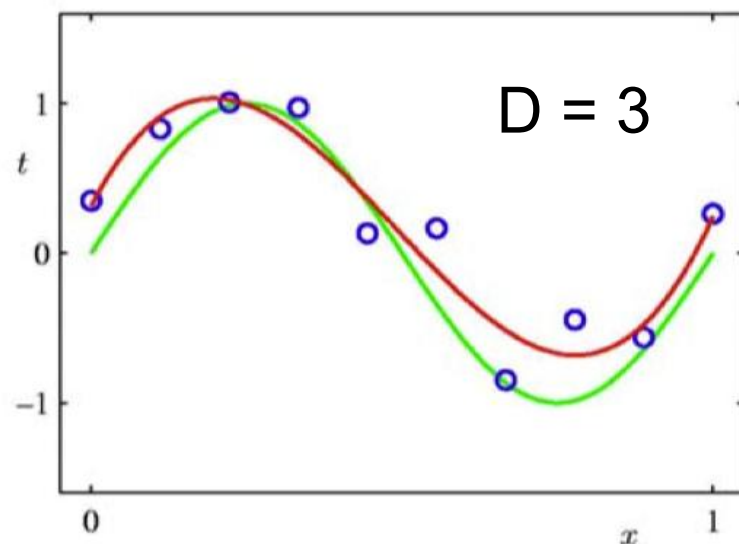
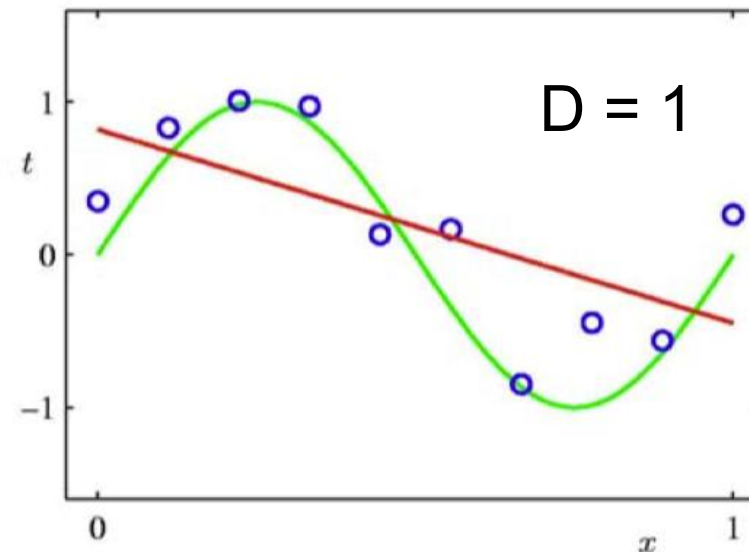
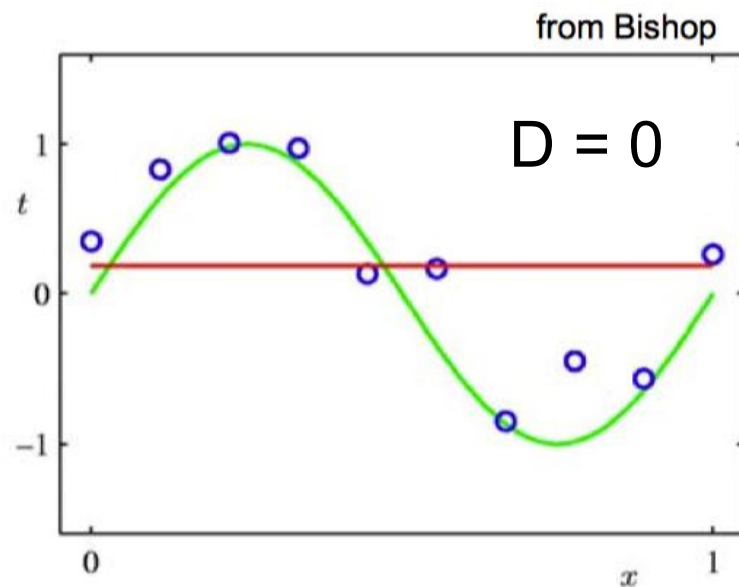
$$= [true\ value - mean(predictions)]^2 - mean[(mean(prediction) - prediction)^2]$$

Why $E[2(y_a - E[y_p])(E[y_p] - y_p)] = 0$?

$y_a - E[y_p]$ is a scalar, therefore $E[y_a - E[y_p]] = y_a - E[y_p]$

$$\begin{aligned} & E[2(y_a - E[y_p])(E[y_p] - y_p)] \\ &= 2(y_a - E[y_p])E[E[y_p] - y_p] \\ &= 2(y_a - E[y_p])\left(E[E[y_p]] - E[y_p]\right) \\ &= 2(y_a - E[y_p])(E[y_p] - E[y_p]) = 0 \end{aligned}$$

Which One is Better?



- Can we increase the maximal polynomial degree to very large, such that the curve passes through all training points?
 - We will know the answer in next lecture.

Take-Home Messages

- Supervised learning paradigm
- Linear regression and least mean square
- Extension to high-order polynomials