Linear Regression

Mahdi Roozbahani
Georgia Tech

These slides are inspired based on slides from Le Song, Chao Zhang, Yaser Abu-Mostafa, Andrew Zisserman.

The Moment

Incoming ML HW3

You; after HW2

Outline

- Supervised Learning
- Linear Regression
- Extension

Supervised Learning: Overview

$X_{n\times d}$ $Y_{n\times 1}$

Functions $\mathcal{F}$

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

Training data

$\{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}$

LEARNING

find $\hat{f} \in \mathcal{F}$
s.t. $y_i \approx \hat{f}(x_i)$

Learning machine

PREDICTION

$y = \hat{f}(x)$

New data

x

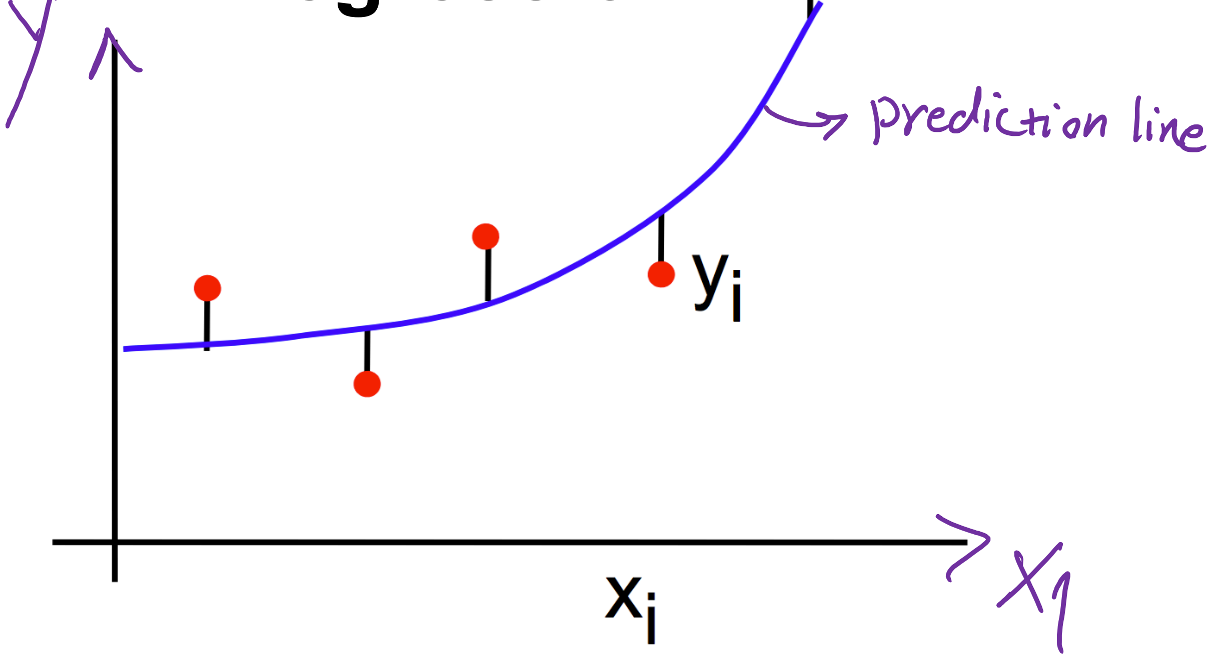
Supervised Learning: Two Types of Tasks

Given: training data $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$

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

When y is continuous:

1. Regression

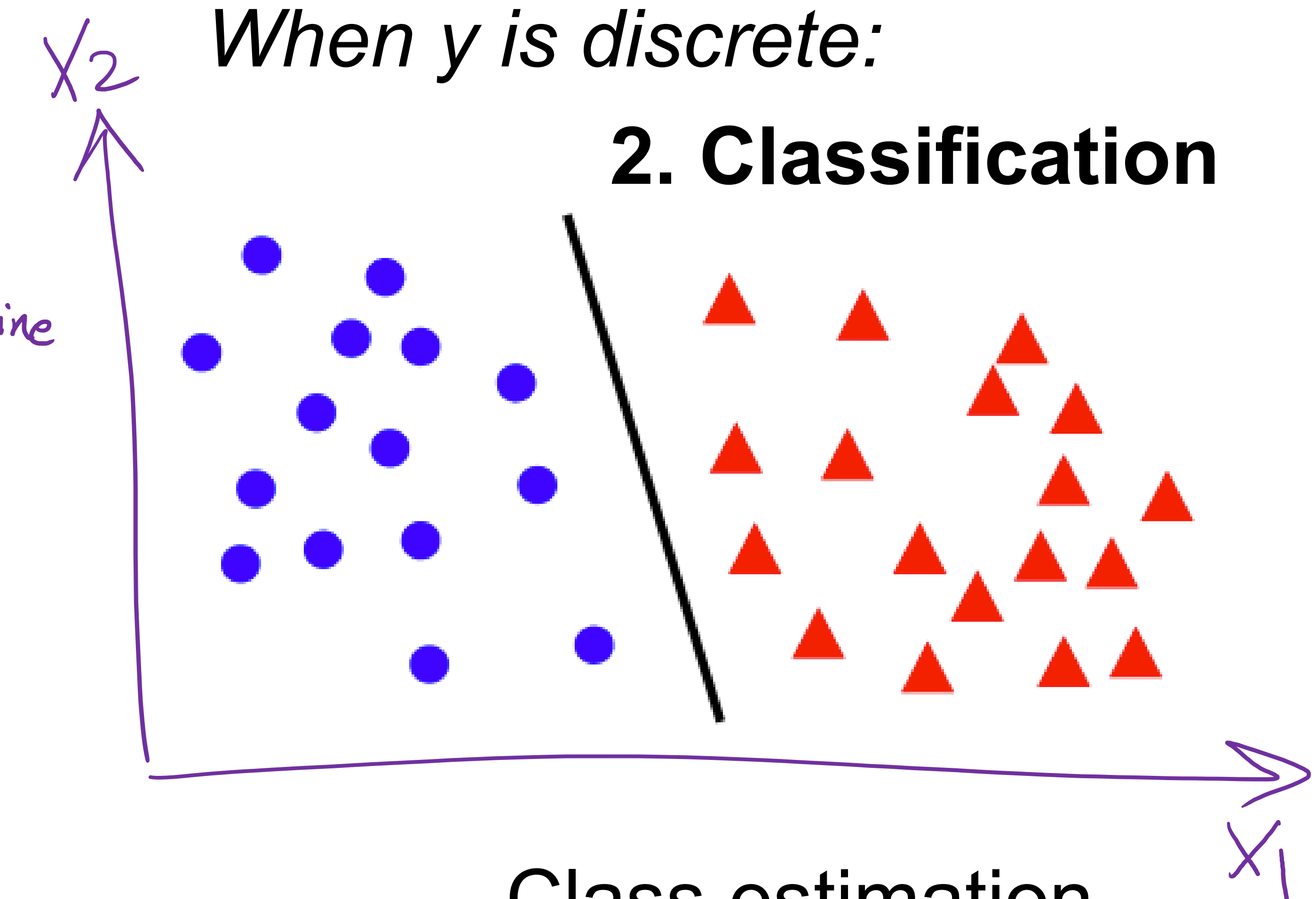


Curve fitting

difference between actual & predicted

When y is discrete:

2. Classification

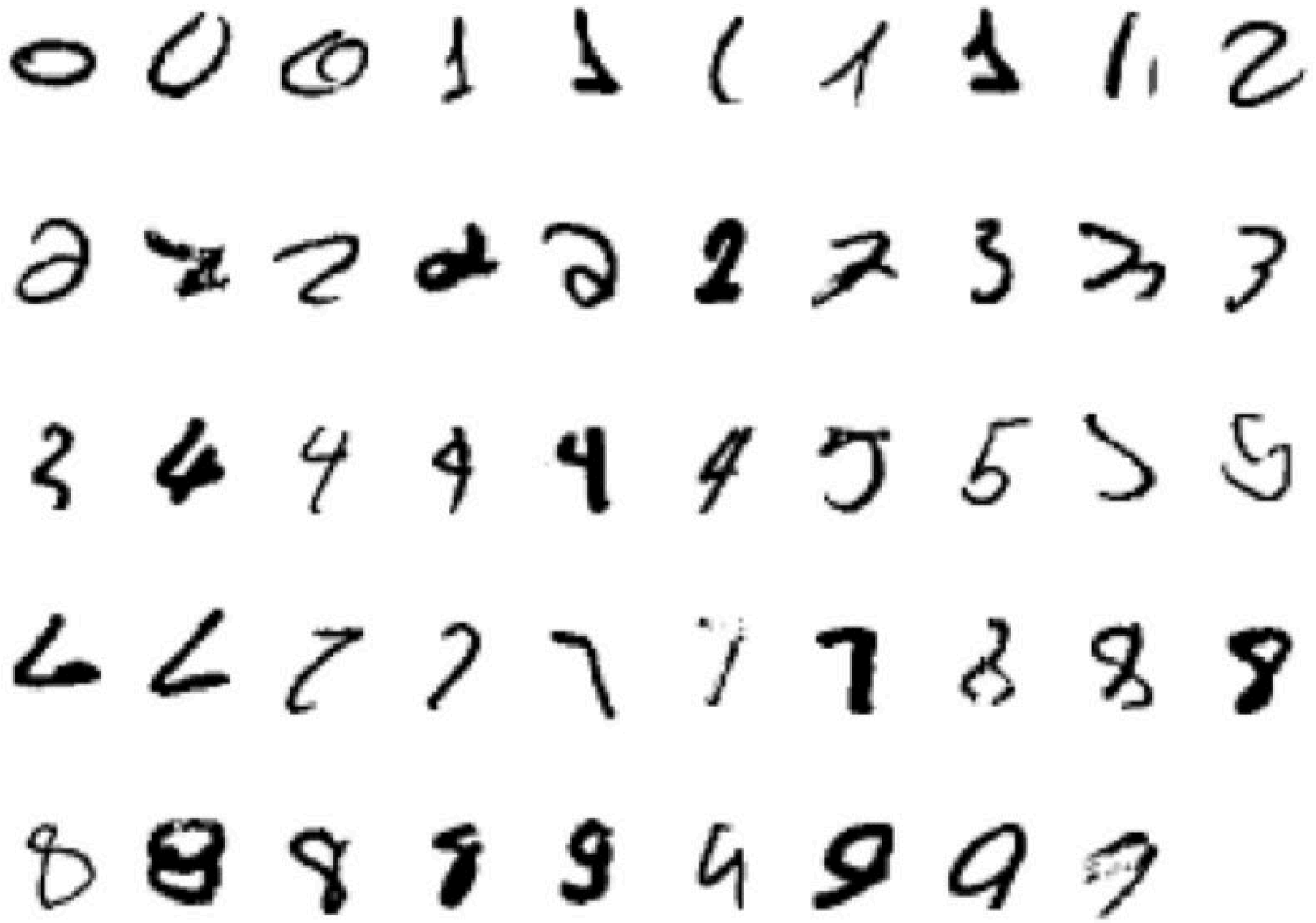


Class estimation

Classification Example 1: Handwritten digit recognition

As a supervised classification problem

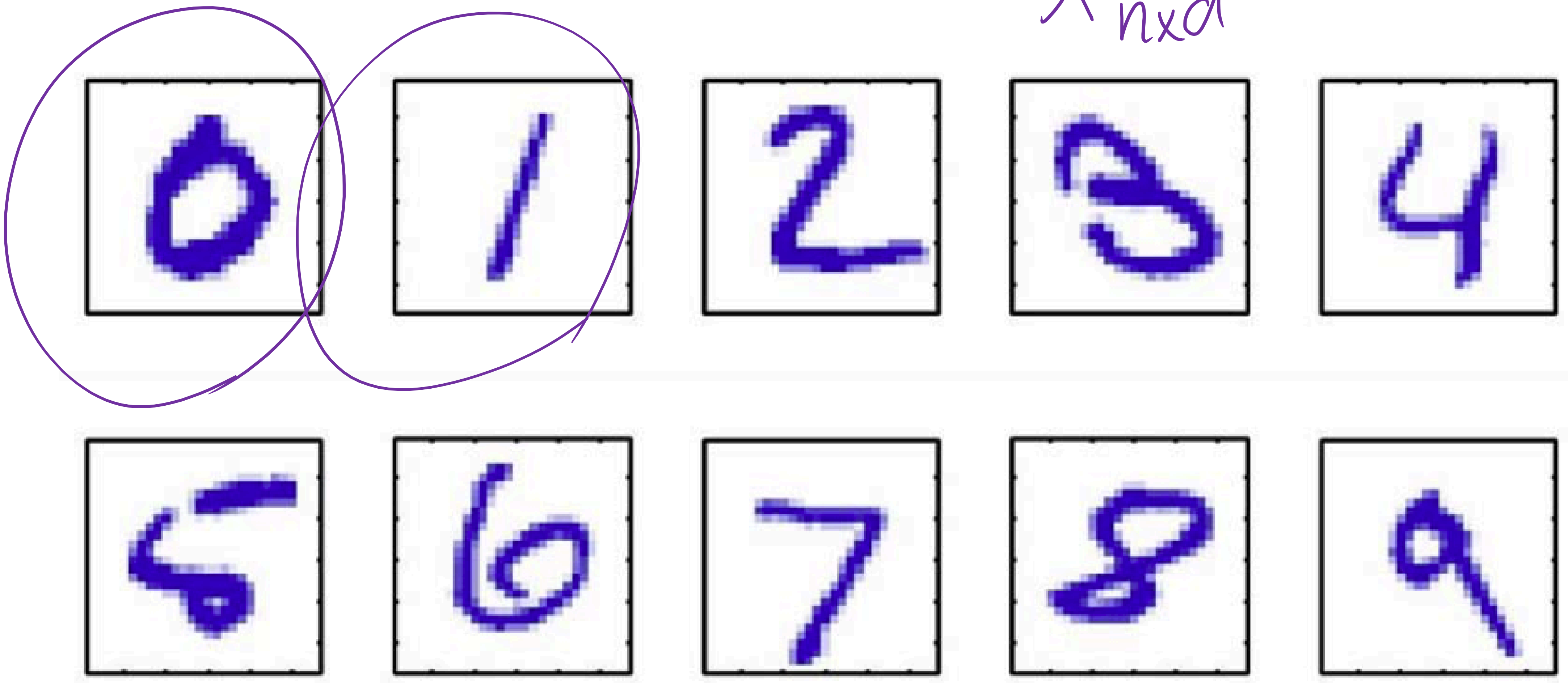
Start with training data, e.g. 6000 examples of each digit



- 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

$X_{n \times d}$



Images are 28 x 28 pixels

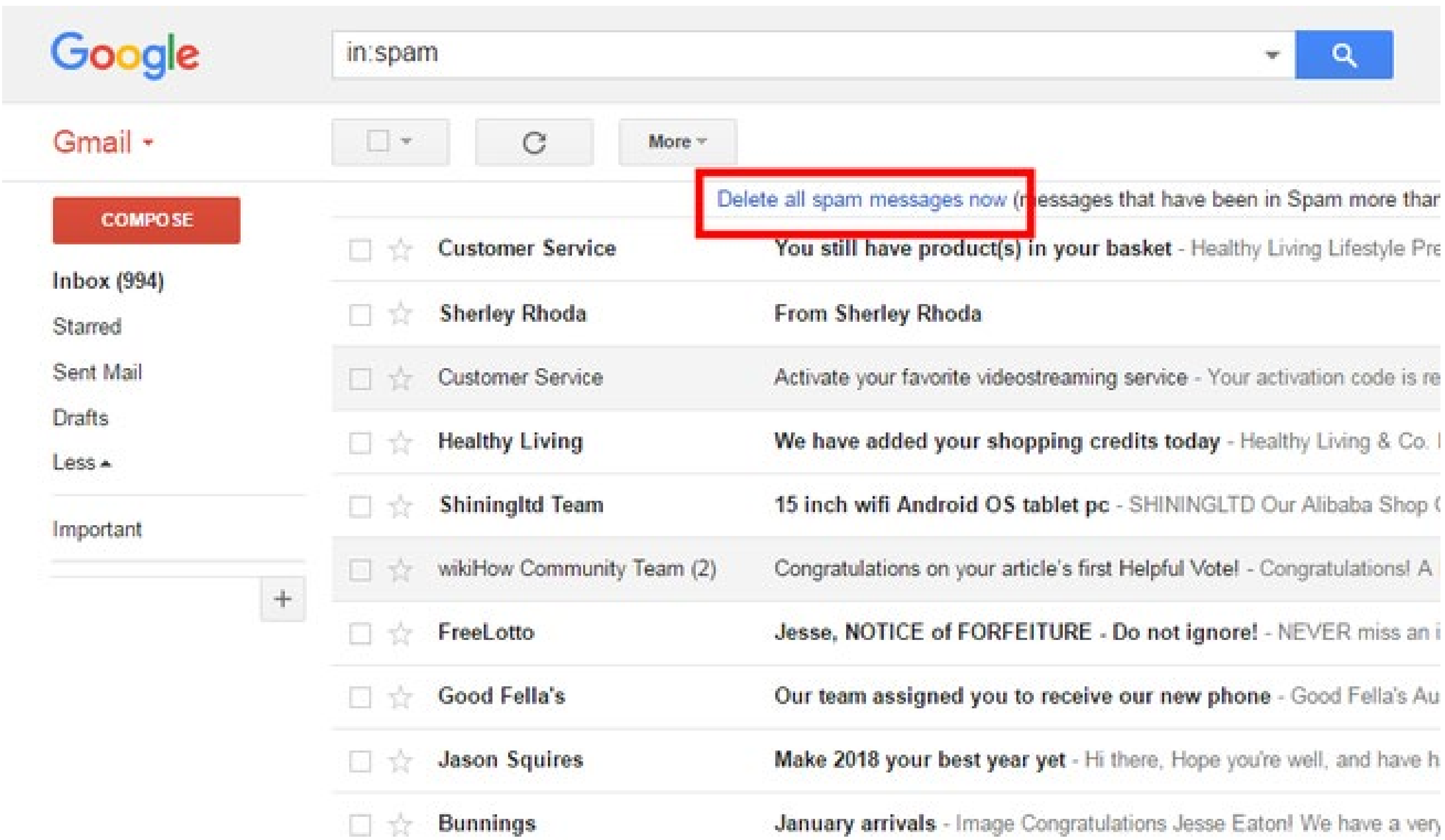
A classification problem

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

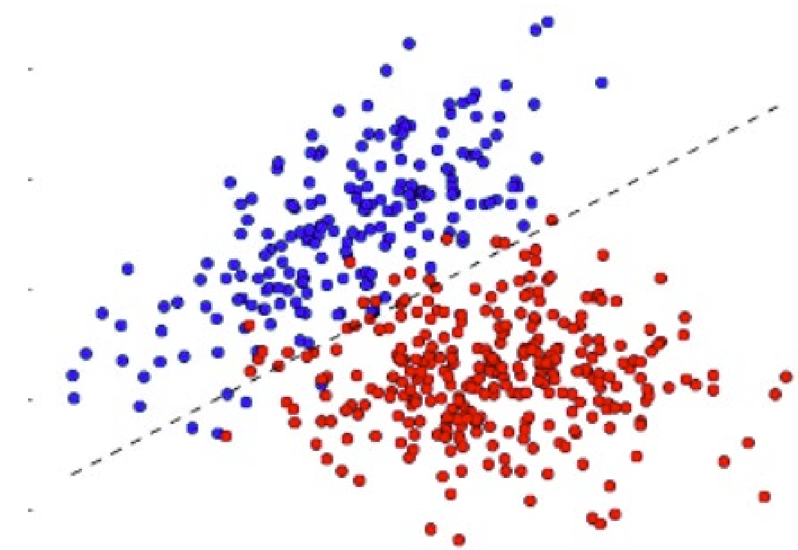
Learn a classifier $f(\mathbf{x})$ such that,

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

Classification Example 2: Spam Detection



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

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 …

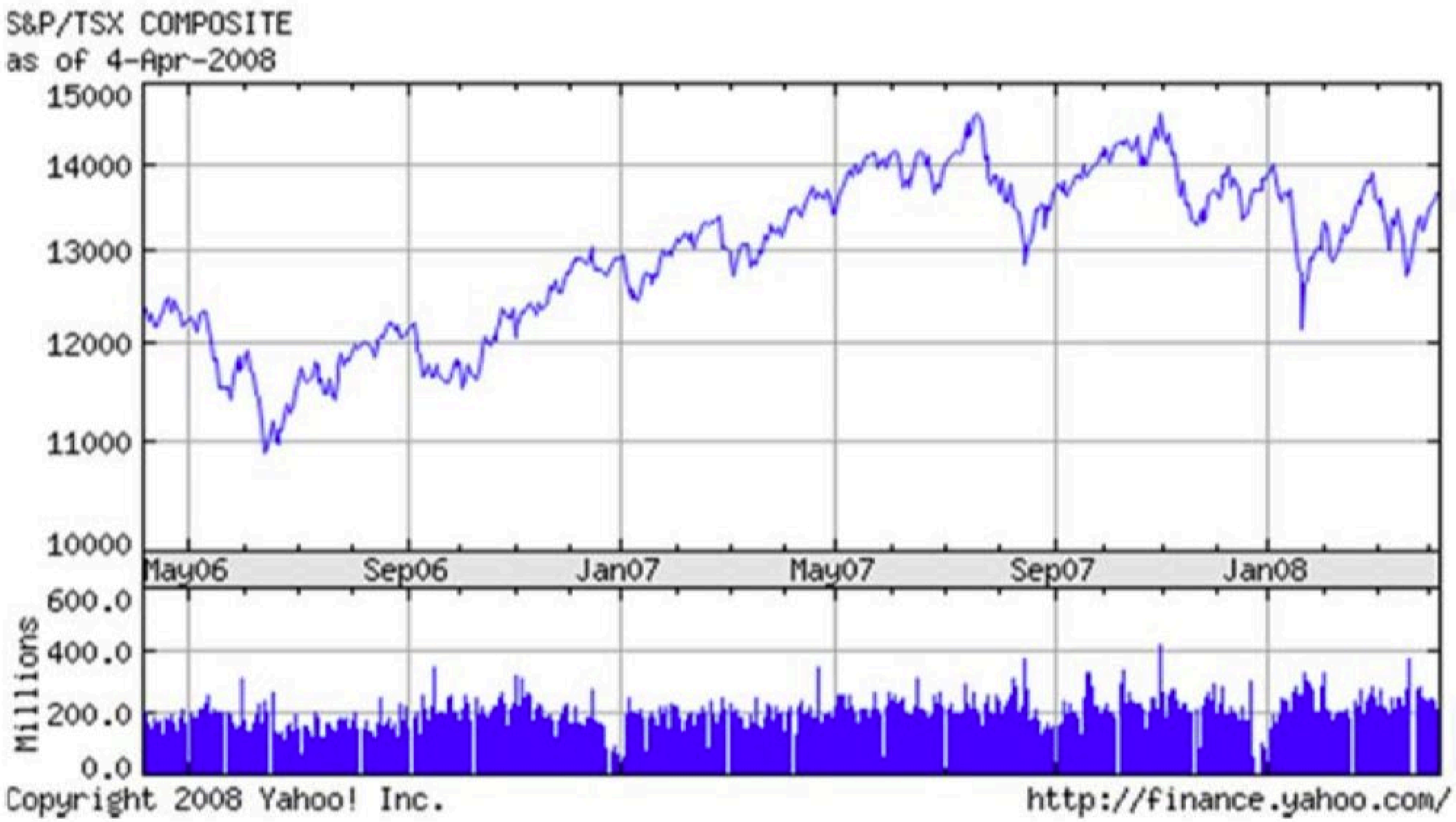
y

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

A regression problem

y = Target values

Regression Example 2: Stock Price Prediction



- Task is to predict stock price at future date

A regression problem

$X_{n\times d}$ $Y_{n\times 1}$

$Y = \underbrace{m}_{\text{slope}}^{\text{parameter}} x + b \rightarrow \boxed{\text{bias term}}$

Parameters (m, b)

- **Features:**
  - Living area, distance to campus, # bedroom …
  - Denote as $x = (x_1, x_2, \ldots, x_d)$

$\Theta = [\Theta_0, \Theta_1]^T$

- **Target:**
  - Rent
  - Denoted as y

$$\boxed{Y = \Theta_0 + \Theta_1 X_1}$$

$$\underbrace{X^{\{1\}} = [1 \quad X_1 \quad X_2]}_{1\times(d+1)} \qquad \Theta = \underbrace{\begin{bmatrix} \Theta_0 \\ \Theta_1 \\ \Theta_2 \end{bmatrix}}_{(d+1)\times 1}$$

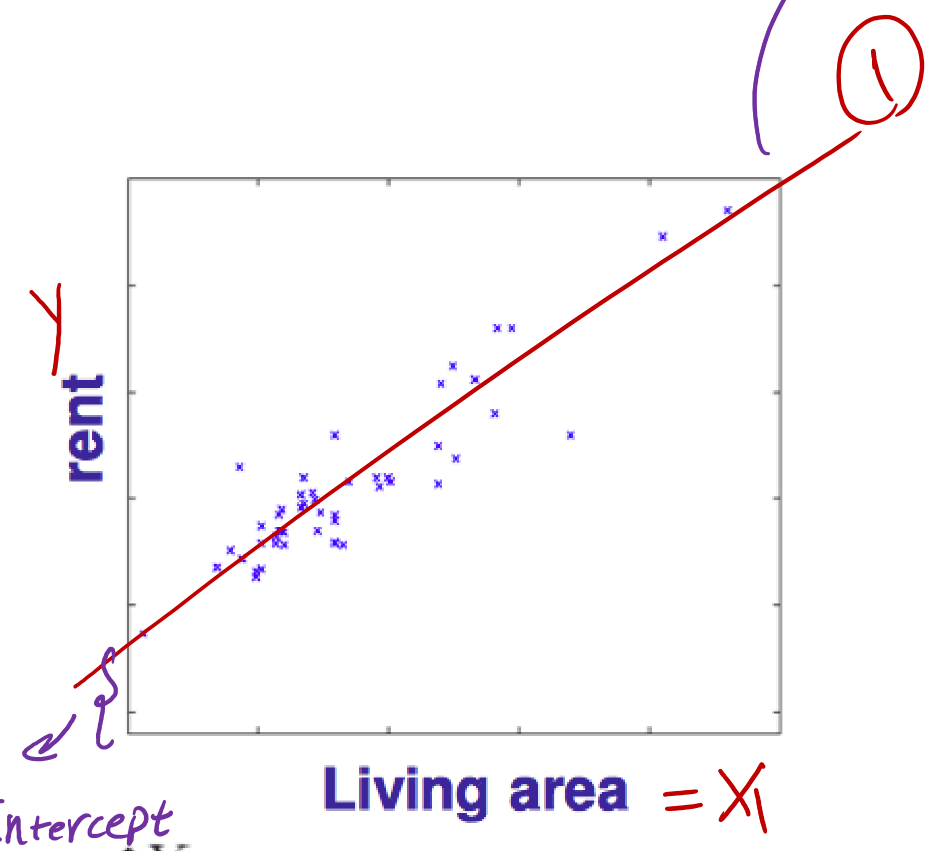
- **Training set:**
  - $x = \{x_1, x_2, \ldots, x_n\} \in R^d$
  - $y = \{y_1, y_2, \ldots, y_n\}$

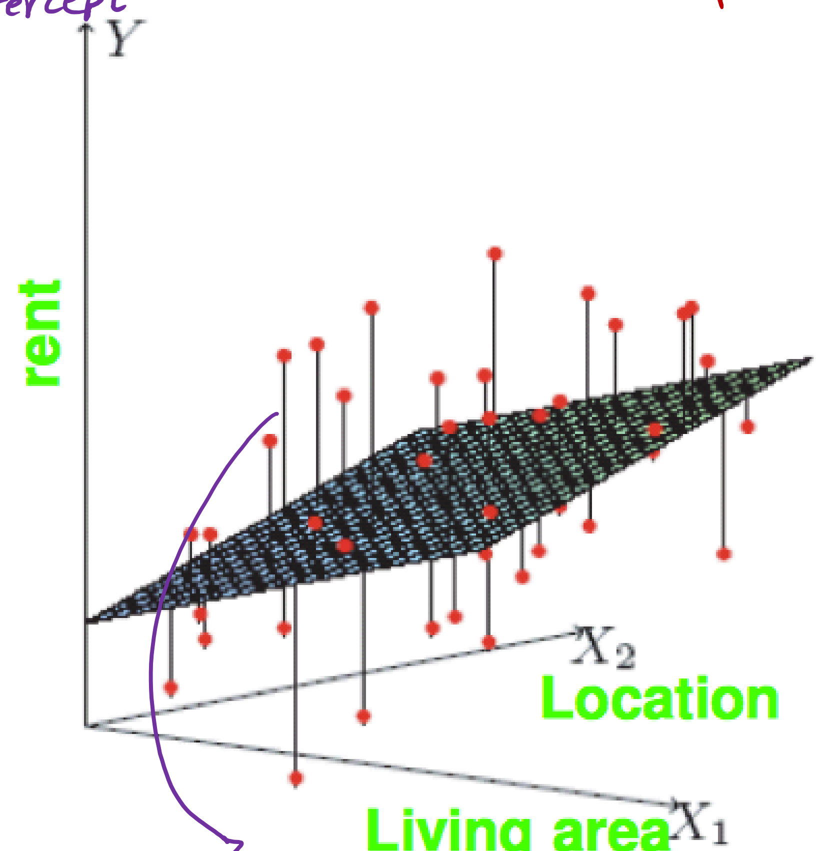
$\Theta = [\Theta_0, \Theta_1, \Theta_2]^T$

$$\underbrace{Y}_{\text{predicted}} = \Theta_0 + \Theta_1 X_1 + \Theta_2 X_2 = X\Theta$$

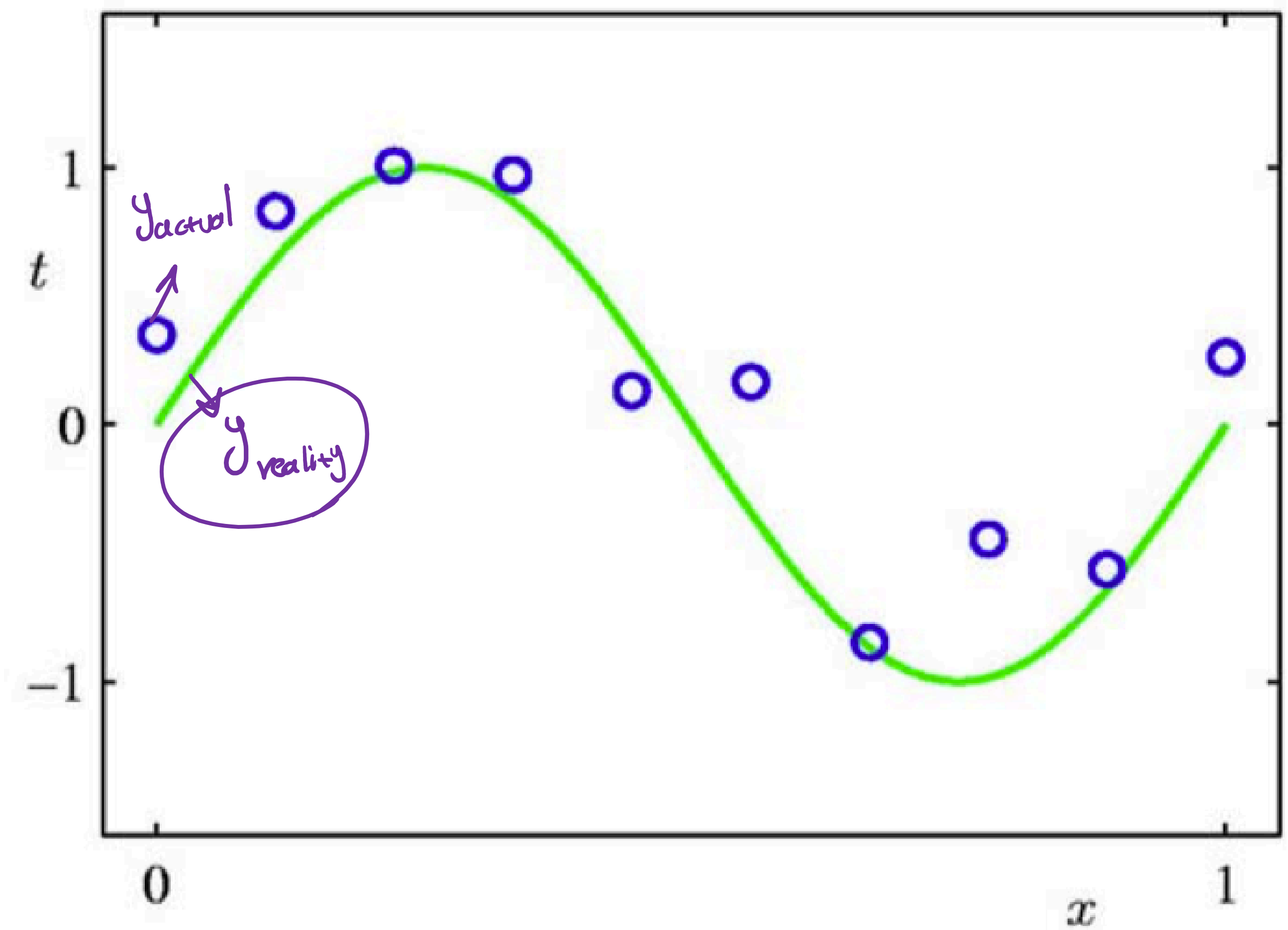
dot product

linear combination of features





Regression: Problem Setup



$x_i \in \mathbb{R}^d$

$y_i \in \mathbb{R}$

- Suppose we are given a training set of N observations

$$(x_1, \ldots, x_N) \text{ and } (y_1, \ldots, y_N), x_i, y_i \in \mathbb{R}$$

- Regression problem is to estimate y(x) from this data

Outline

- Supervised Learning
- Linear Regression
- Extension

Linear Regression

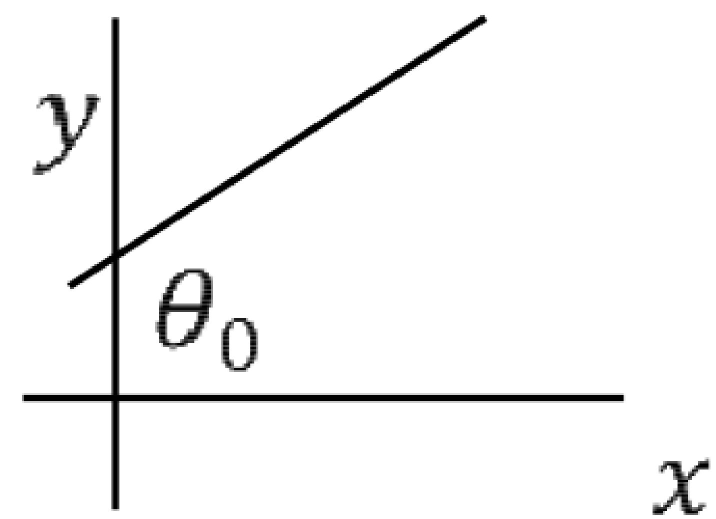
- Assume y is a linear function of x (features) plus noise ϵ

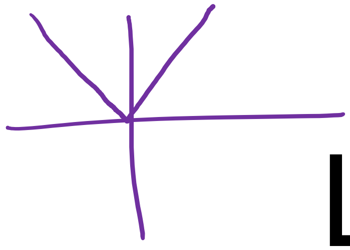
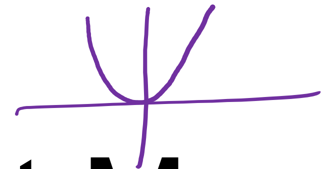
$$y = \underbrace{\theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d}_{x\theta} + \epsilon$$

- where ϵ is an error term of unmodeled effects or [random noise]()

- Let $\theta = (\theta_0, \theta_1, \ldots, \theta_d)^T$, and augment data by one dimension

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



$y = |x|$  $y = x^2$ 

$x_{i\,1\times(d+1)}$ $\theta_{(d+1)\times 1}$

$x_i\theta \in \mathbb{R}$

Least Mean Square Method

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

$E(\theta)$ Error

$\mathrm{Loss}(\theta)$

averaging $E[(y_i - x_i\theta)^2]$

Training $$\hat{\theta} = argmin_{\,\theta}\, L(\theta) = \boxed{\frac{1}{n}\sum_{i=1}^{n}} (\underbrace{y_i}_{y_{actual} \in \mathbb{R}} - \underbrace{x_i\theta}_{y_{predicted}})^2$$

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - [\theta_0 + \theta_1 x_i])^2$$

$Y = X\theta \Rightarrow \theta = \frac{Y}{X} \approx X^{-1}Y$ pseudo inverse

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

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

$$\frac{\partial L(\theta)}{\partial \theta} = -\frac{2}{n}\cancel{\sum_{i=1}^{n}} x_i^T y_i + \frac{2}{n}\cancel{\sum_{i=1}^{n}} x_i^T x_i\theta = 0$$

Matrix form

$$x = \begin{bmatrix} 1 & x_1^{\{1\}} & \cdots & x_1^{\{d\}} \\ 1 & x_2^{\{1\}} & \ddots & x_2^{\{d\}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n^{\{1\}} & \cdots & x_n^{\{d\}} \end{bmatrix}_{n \times (d+1)} \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}_{n \times 1} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}_{(d+1) \times 1}$$

$$MSE(\theta) = argmin_{\,\theta}\, L(\theta) = \frac{1}{n} \underbrace{\boxed{(y - x\theta)^{\mathrm{T}}}}_{1 \times n} \underbrace{\boxed{(\mathrm{y} - x\theta)}}_{n \times 1}$$

$$x\theta = \begin{bmatrix} \theta_0 + \theta_1 x_1^{\{1\}} + \theta_2 x_1^{\{2\}} + \cdots + \theta_d x_1^{\{d\}} \\ \theta_0 + \theta_1 x_2^{\{1\}} + \theta_2 x_2^{\{2\}} + \cdots + \theta_d x_2^{\{d\}} \\ \vdots \\ \theta_0 + \theta_1 x_n^{\{1\}} + \theta_2 x_n^{\{2\}} + \cdots + \theta_d x_n^{\{d\}} \end{bmatrix}_{n \times 1} = \hat{Y}$$

$$Y_{\text{actual}} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}$$

Matrix Version and Optimization

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

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

$X_{n \times d}$

$\boxed{X^T X}$ $d \times d$

$\overset{inv}{\approx} \frac{1}{X^T X} X^T \approx (X^T X)^{-1} X^T$

Let's rewrite it as:

$$\frac{\partial L(\theta)}{\partial \theta} = -\frac{2}{n}(x_1, \ldots, x_n)^T (y_1, \ldots, y_n) + \frac{2}{n}(x_1, \ldots, x_n)^T (x_1, \ldots, x_n)\theta = 0$$

Define $\mathrm{X} = (x_1, \ldots, x_n)$ and $\mathrm{y} = (y_1, \ldots, y_n)$

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

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

X^+ is the **pseudo-inverse** of X

$$X^T X X^+ = X^T$$

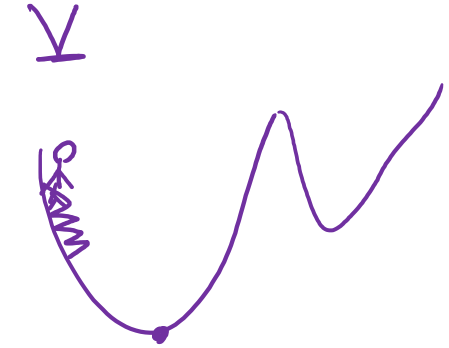
$$\theta = (X^T X)^{-1} X^T y = X^{+}\, y$$

$X_{n\times d}$ $\quad$ $n =$ instances $\quad$ $d =$ dimension

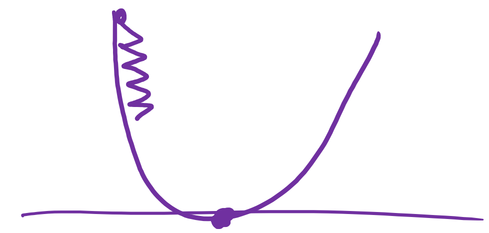
$$X^T X = \begin{bmatrix} d \times n \end{bmatrix} \begin{bmatrix} n \times d \end{bmatrix} = \begin{bmatrix} d \times d \end{bmatrix}$$

Not a big matrix because $n \gg d$This matrix is invertible most of the times. If we are VERY unlucky and columns of $\mathbf{X^T}\boldsymbol{X}$ 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



$f(x) = x^2$

$x^{\{t+1\}} \leftarrow x^{\{t\}} - \alpha \frac{\partial f(x)}{\partial x}$

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

- $$\frac{\partial L(\theta)}{\partial \theta} = -\frac{2}{n} \sum_{i=1}^{n} x_i^T (y_i - x_i \theta)$$

- Gradient descent

GD

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

Batch Gradient descent

BGD

- 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

$$\hat{\theta}^{t+1} \leftarrow \hat{\theta}^{t} + \beta_t \ \times x_i^T (y_i - x_i \theta)$$

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

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

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

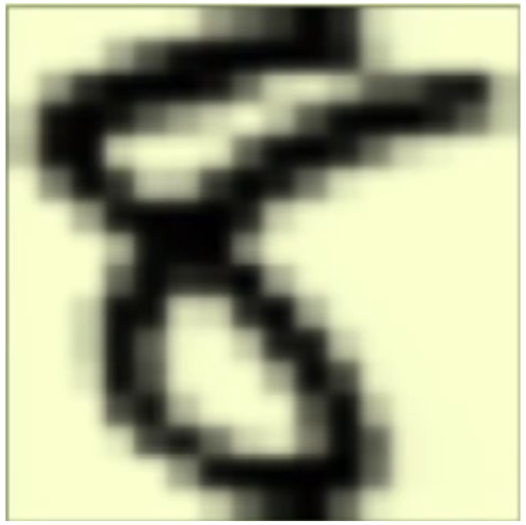
$$\theta = (X^T X)^{-1} X^T y$$

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



Extract useful information

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

$Sum\ up\ all\ the\ pixels = intensity$

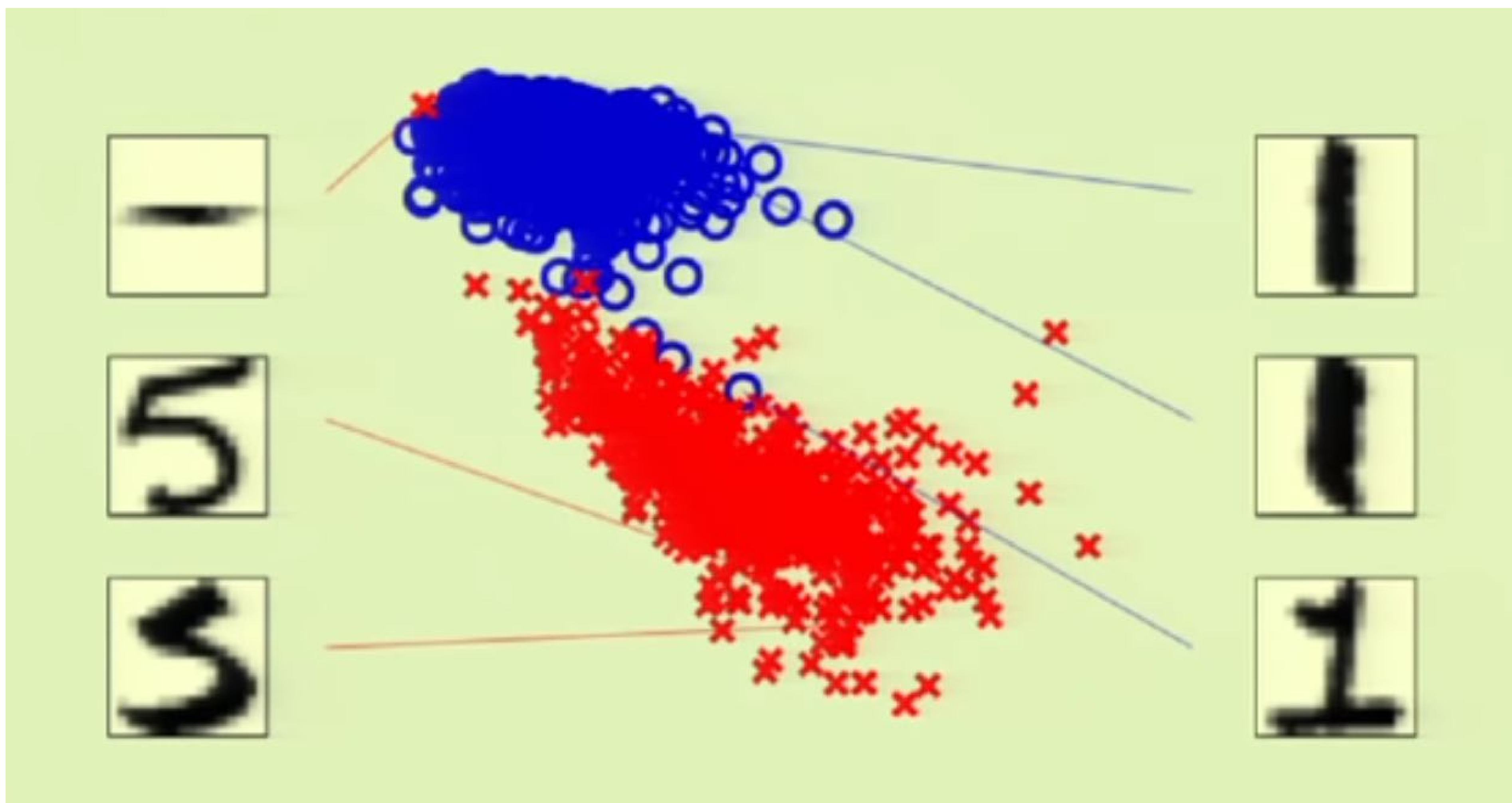
$Symmetry =$ -(*difference between flip version*)

$x = (1, x_1, x_2)$

$x_1 = intensity \; x_2 = symmetry$

It is almost linearly separable

symmetry



intensity

Linear regression for classification

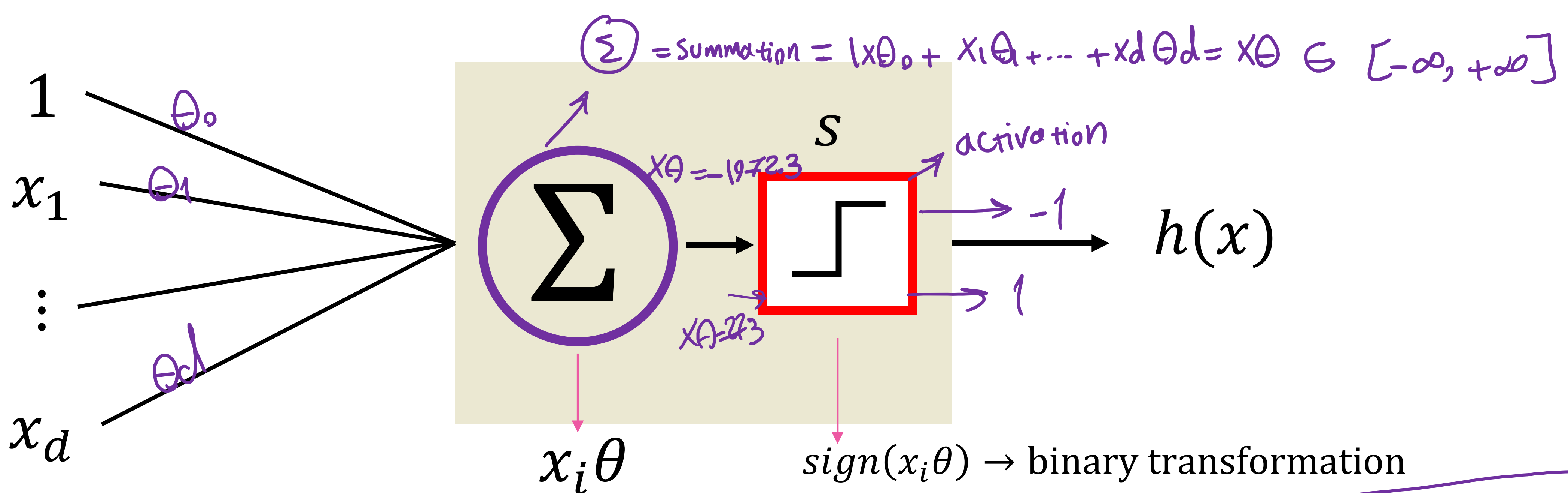
Binary-valued functions are also real-valued $\pm 1 \in R$

Use linear regression $x_i\theta \approx y_n = \pm 1$ $\quad i =$ index of a data-point

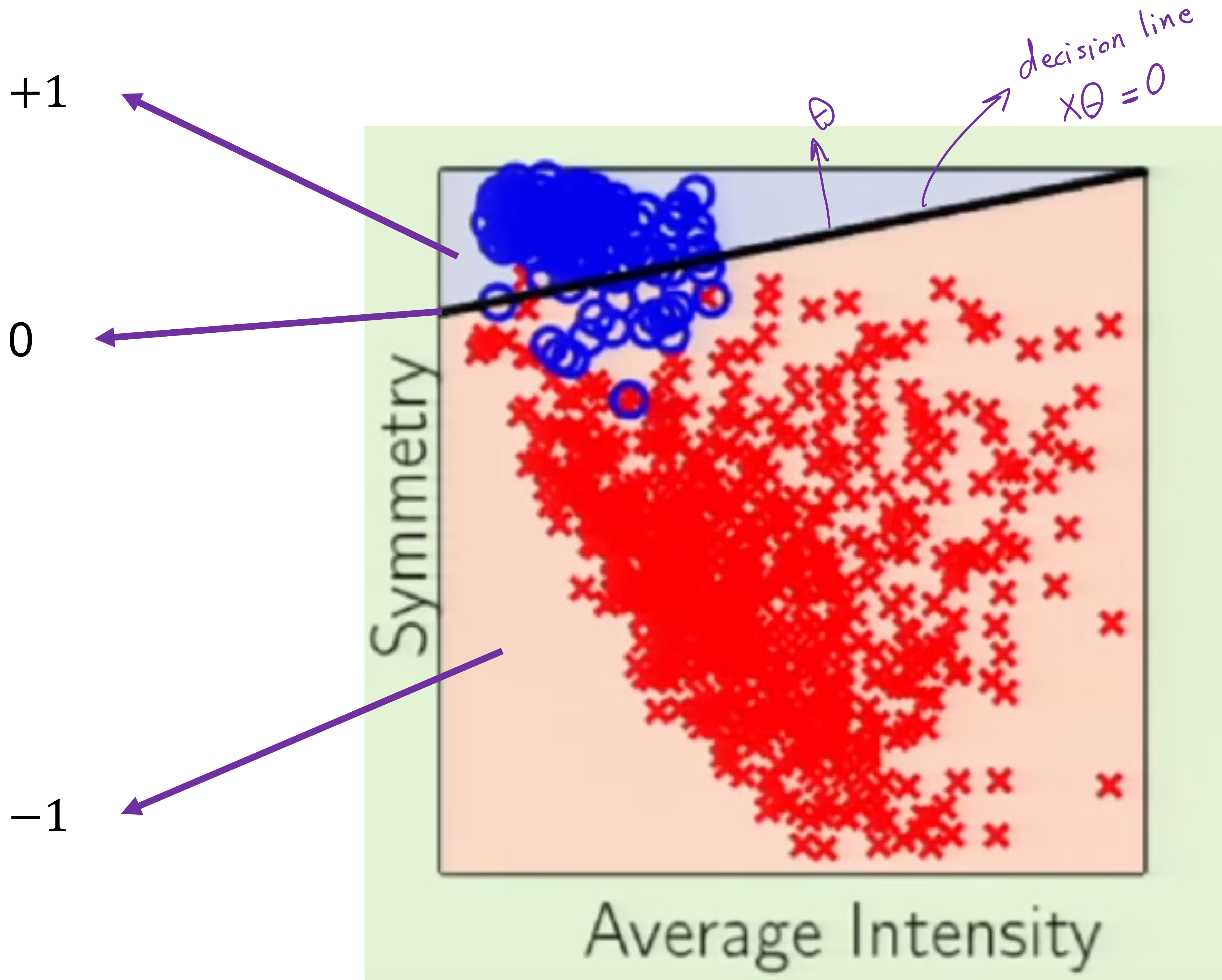
Let's calculate, $sign(x_i\theta) = \begin{cases} -1 & x_i\theta < 0 \\ 0 & x_i\theta = 0 \\ 1 & x_i\theta > 0 \end{cases}$

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

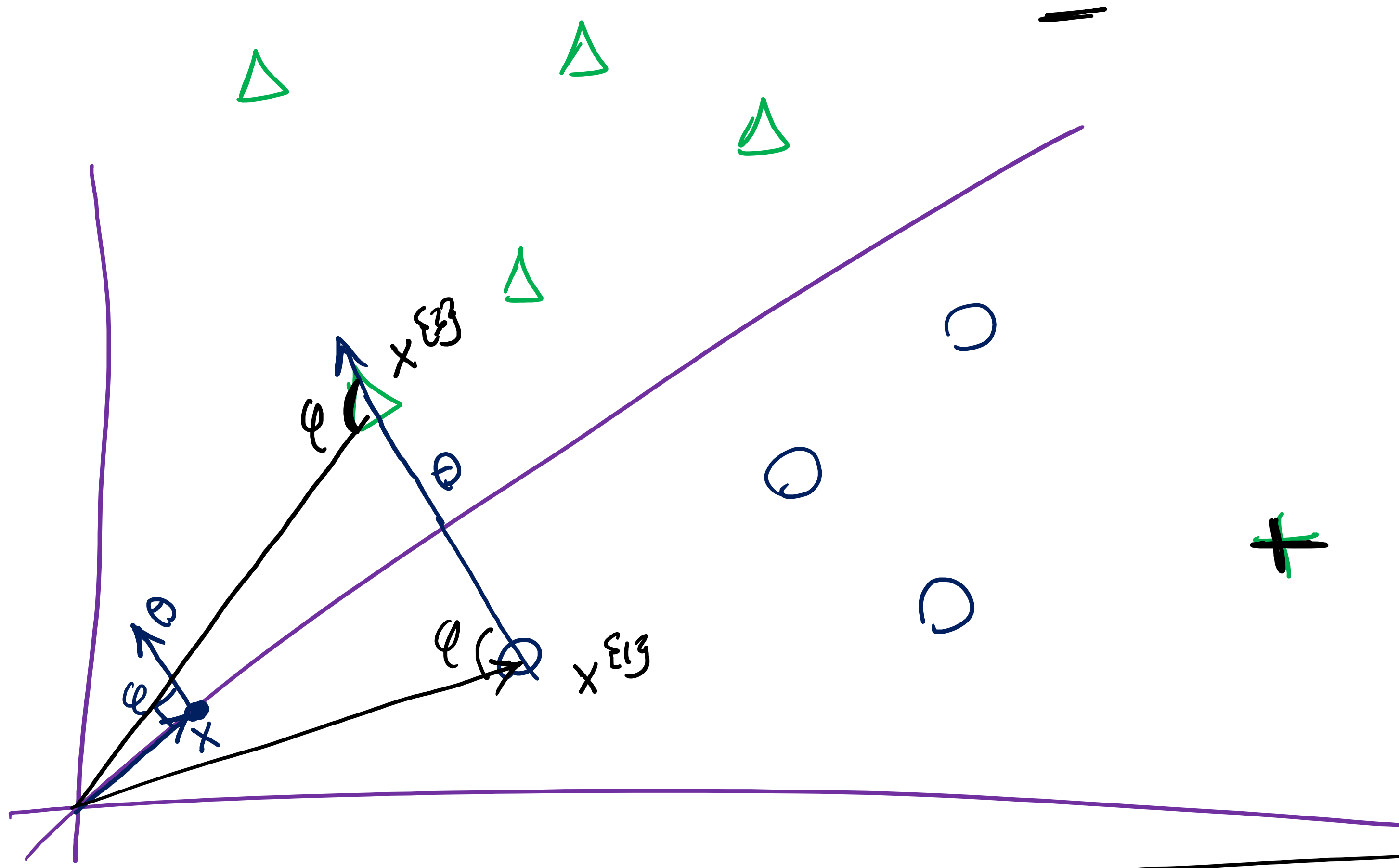
x_i



a learning block



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



$$X\Theta = |x|\,|\theta|\,\cos\varphi = 0$$

$\varphi < 90$

$$\boxed{x^{\{1\}}\theta = |x^{\{1\}}|\,|\theta|\,\cos\varphi} \Rightarrow > 0 \overset{\text{Sign}}{\Rightarrow} +1$$

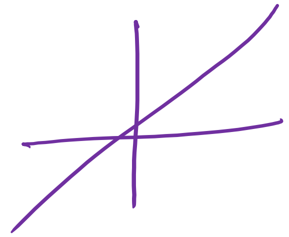
$$x^{\{2\}}\theta \Rightarrow \varphi > 90 \Rightarrow < 0 \overset{\text{sign}}{\Rightarrow} -1$$

Outline

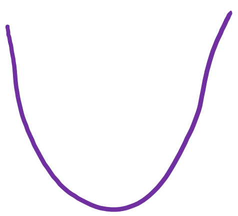
- Supervised Learning
- Linear Regression
- Extension

Extension to Higher-Order Regression

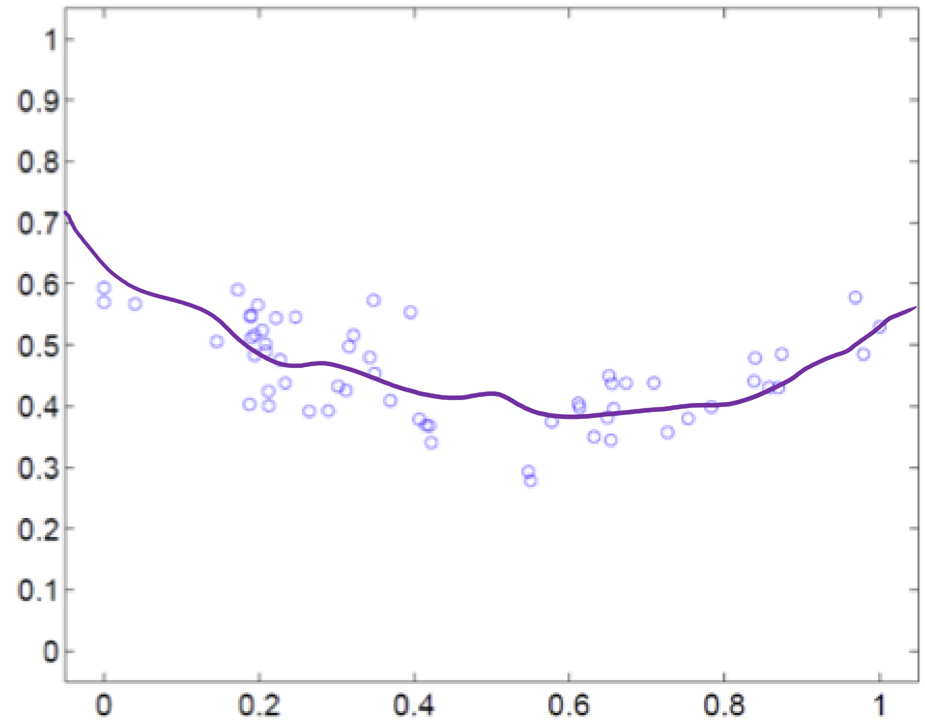
$X = \begin{bmatrix} \text{height} = h \\ \\ \end{bmatrix}$



$X = \begin{bmatrix} h^2 \\ \\ \end{bmatrix}$



h^3



$Z = \begin{bmatrix} h & h^2 & h^3 & \cdots & h^d \\ & & & & \\ \end{bmatrix}$

- Want to fit a polynomial regression model

$$\boxed{y = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_d x^d + \epsilon}$$

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

$$y = z\theta$$

Least Mean Square Still Works the Same

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

$$\hat{\theta} = argmin_{\ \theta}\ L(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - z_i\theta)^2$$

- 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

$z = \{1, x, x^2, \dots, x^d\} \in R^d \qquad\qquad y = \{y_1, y_2, \dots, y_n\}$

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

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

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

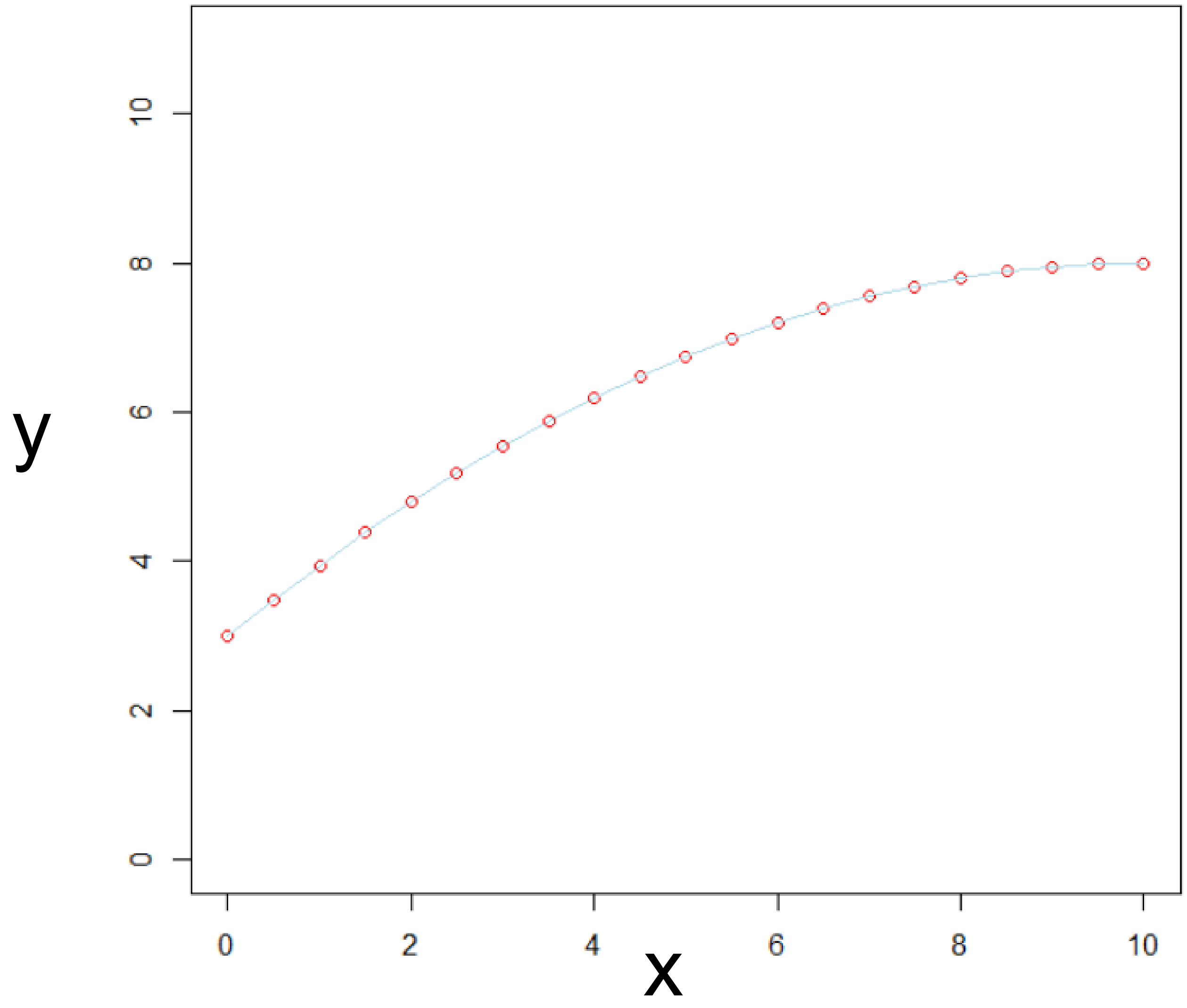
What is happening in polynomial regression?

$$x = [0,0.5,1,\dots,9.5,10]$$

$$y = [3,3.4875,3.95,\dots,7.98,8]$$

$$f = \theta_0 + \theta_1 x + \theta_2 x^2$$

$$\theta_0 = 3; \theta_1 = 1; \theta_2 = -0.5$$

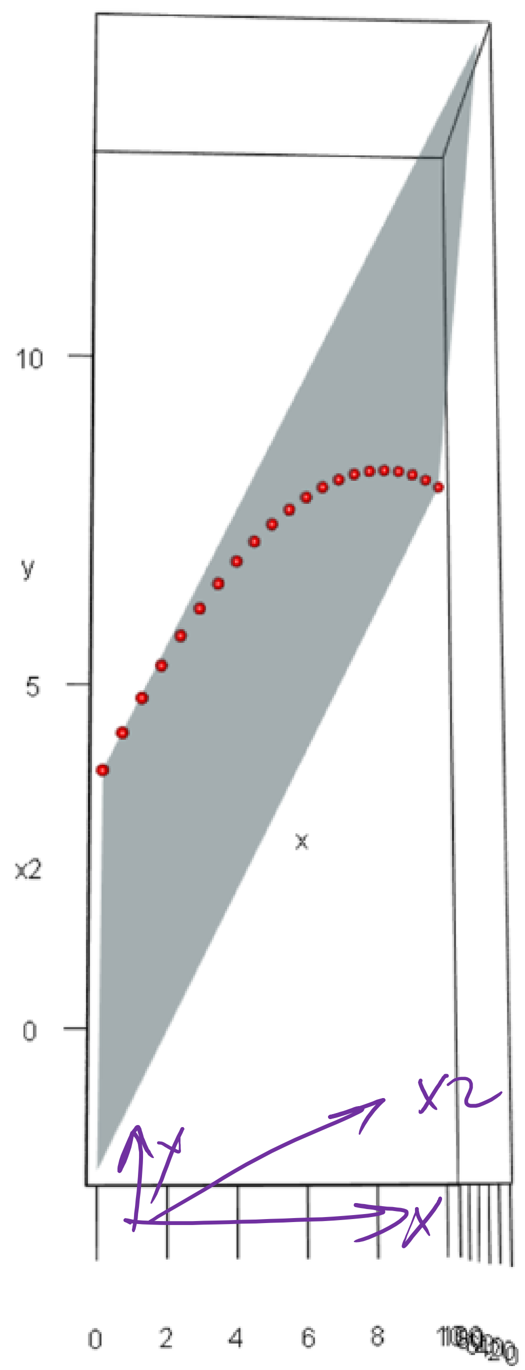


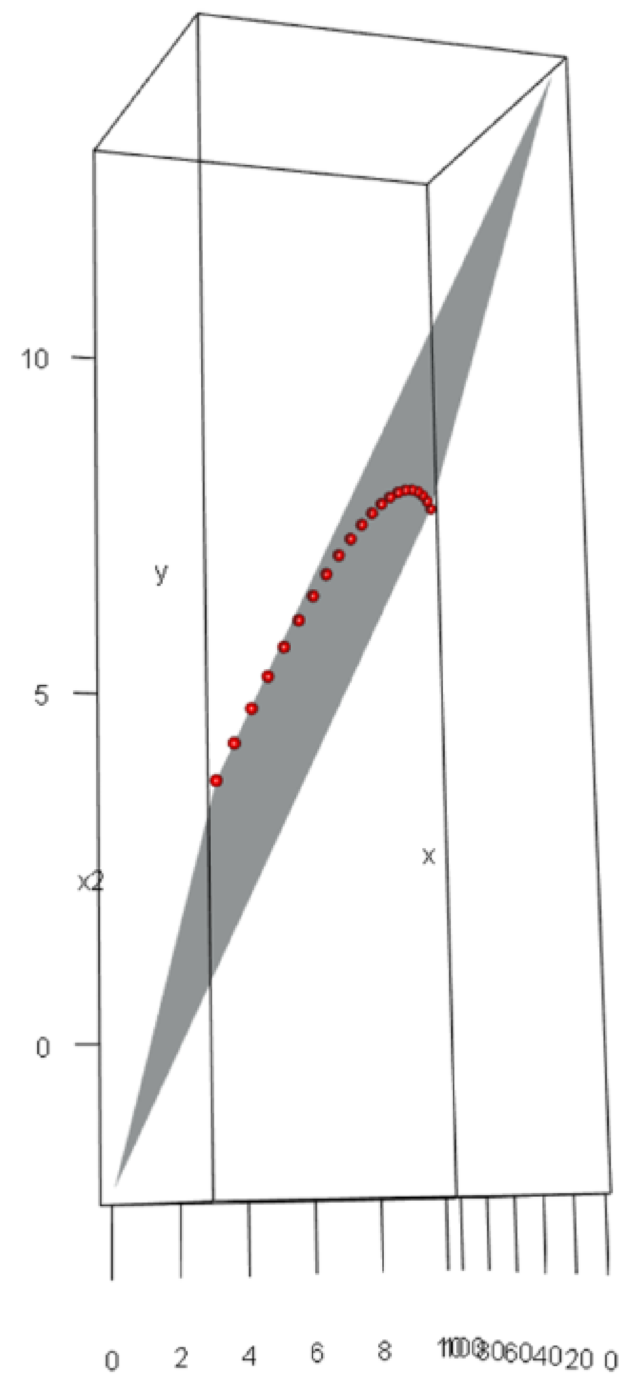
RMSE=0

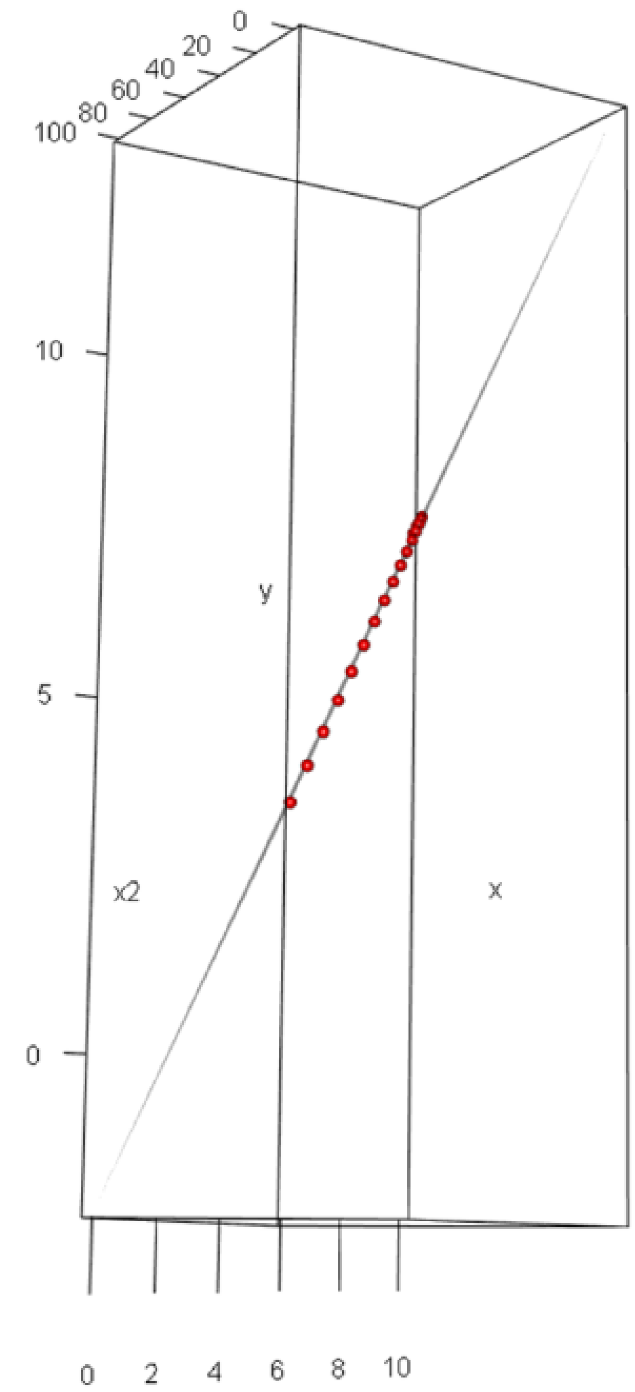
Let's add to the feature space

$x_1 = [0, 0.5, 1, \ldots, 9.5, 10]$ $\qquad$ $x_2^2 = [0, 0.25, 1, \ldots, 90.25, 100]$

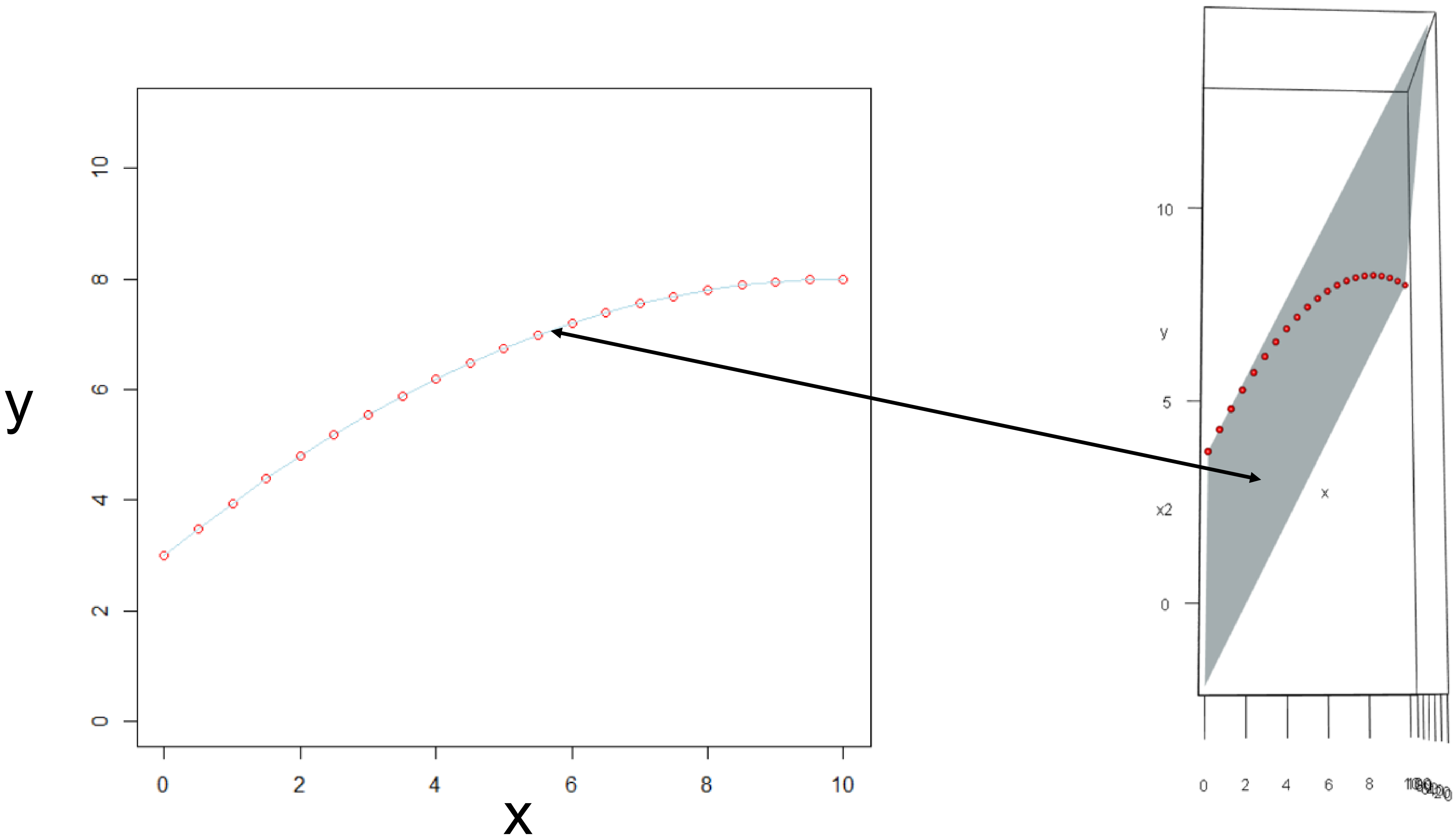
$$y = [3, 3.4875, 3.95, \ldots, 7.98, 8]$$







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.

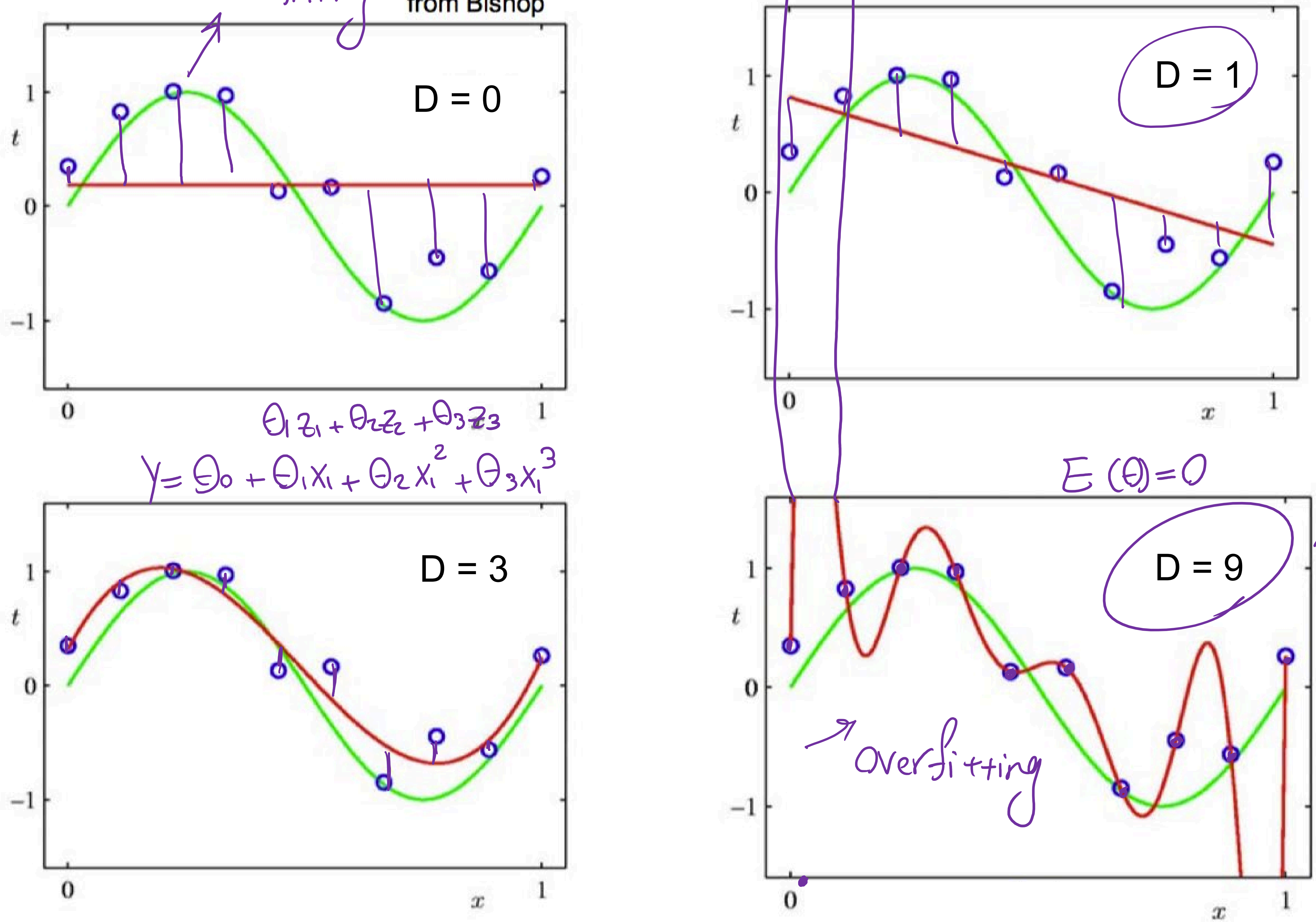


$y = \Theta_0$

Increasing the Maximal Degree

$y = \Theta_0 + \Theta_1 x_1$

from Bishop



Bias-Variance Trade off

Animation

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

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

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

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

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

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

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

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

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

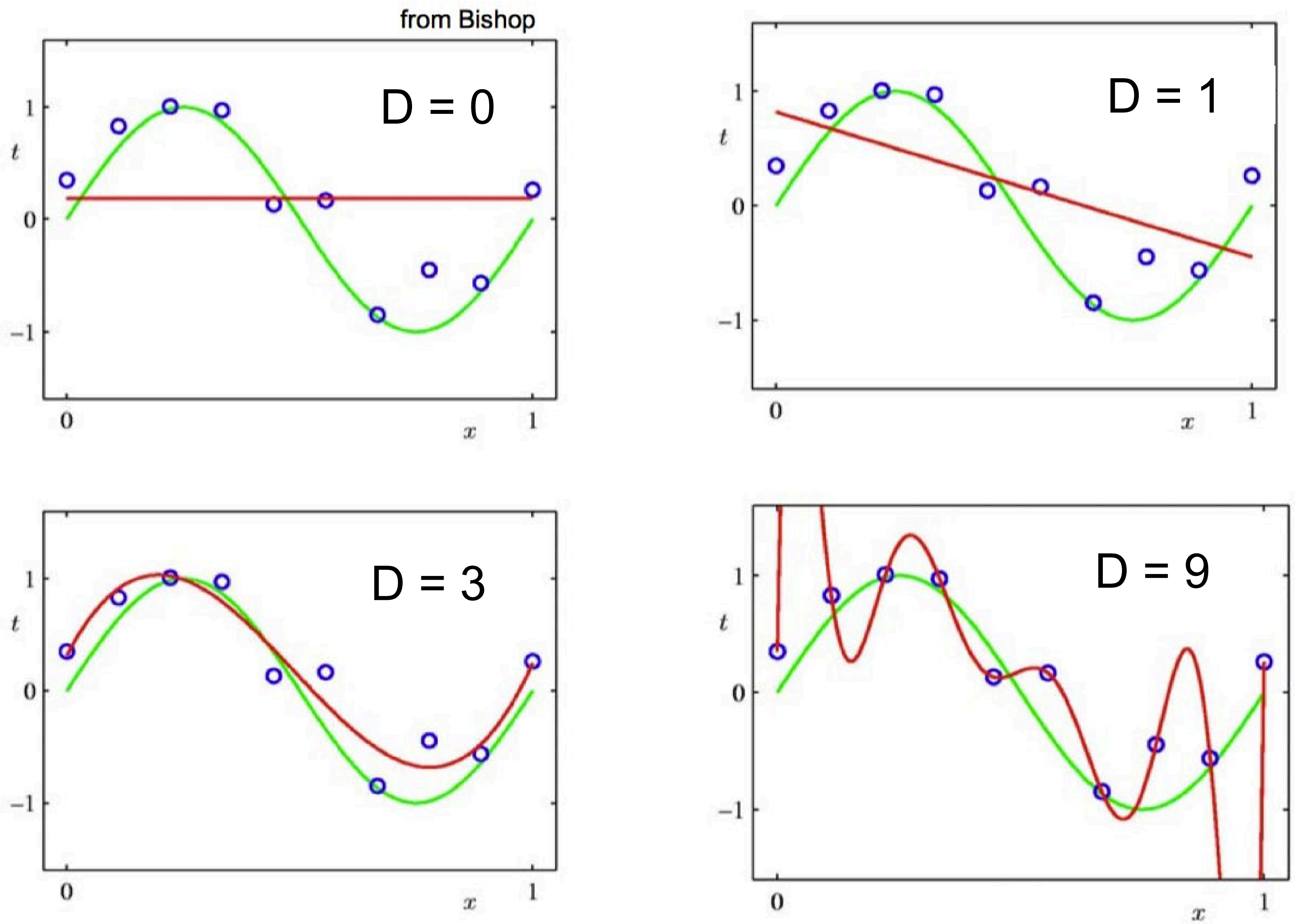
$$E[2(y_a - E[y_p])(E[y_p] - y_p)]$$

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

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

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

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