Lecture 04. Linear Regression

Xin Chen

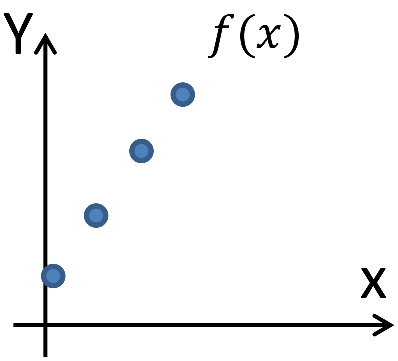
These slides are based on slides from Mahdi Roozbahani

Outline

- Supervised Learning
- Linear regression
- Extension

Recall one of our examples

Problem 1: 1 → 3 → 5 → 7 → ?



Data → Model $f(x)$

Define your dataset: (X, Y)

Learning (training the data) and predicting

Supervised Learning: overview

Learning:

Training data
$\{ (x_i, y_i) \in x \times y \}$

Find $f \in F$
s.t. $v_i \approx \hat{f}(x_i)$

Functions F
$f : x \to y$

Prediction:

New data: x

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

Result

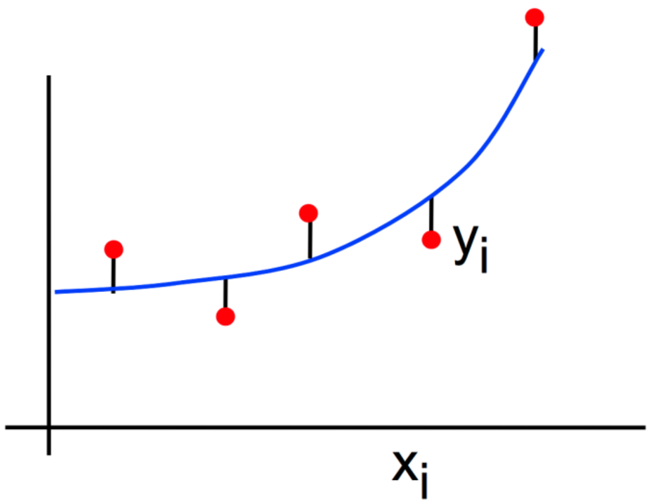
Supervised Learning: two types of tasks

Given: training data: $\{(x_1, y_1), (x_1, y_1), \ldots, (x_n, y_n)\}$

Learn: a function $f(x)$: $y = f(x)$

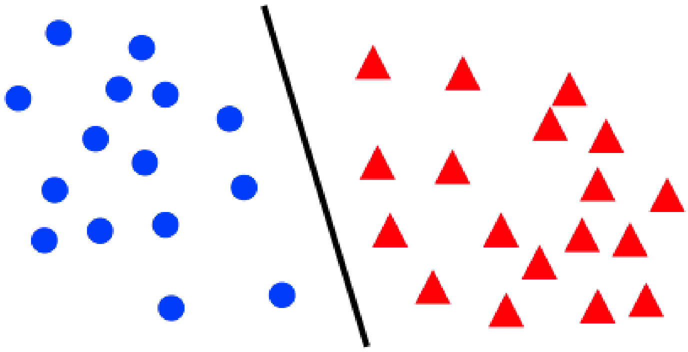
When y is continuous

1. Regression



When y is discrete

2. Classification



Example 1: Apartment Rent Prediction

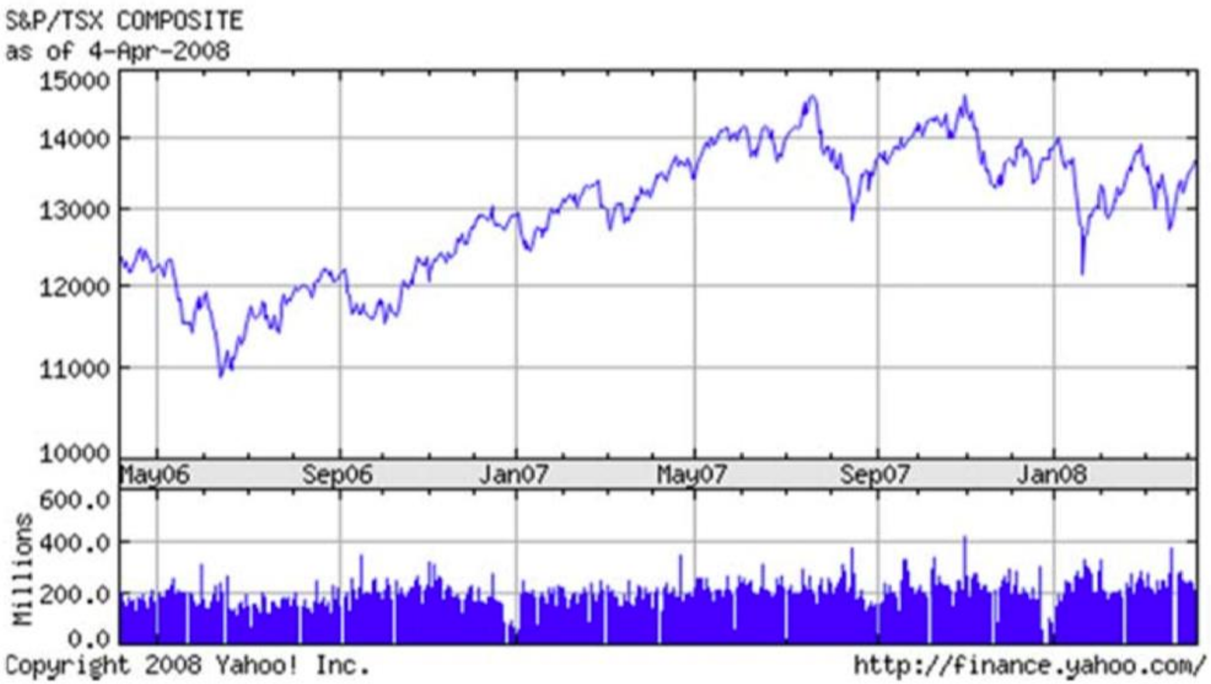
- Suppose you are to move to Altanta
- You want to find the most reasonably priced apartment satisfying your needs: (square-ft, # of bedrooms, rent price)

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

Example 2: Stock Price Prediction

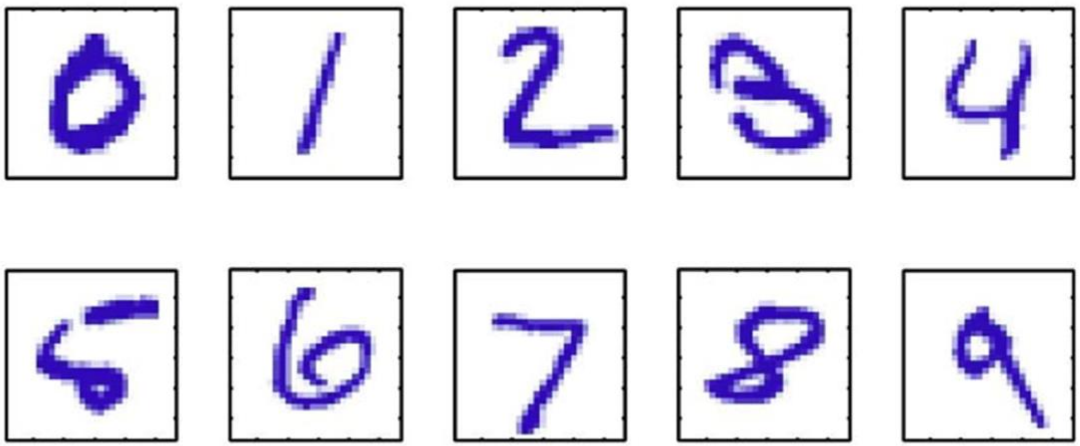
- The task is to predict stock prices at a future date.



A regression problem

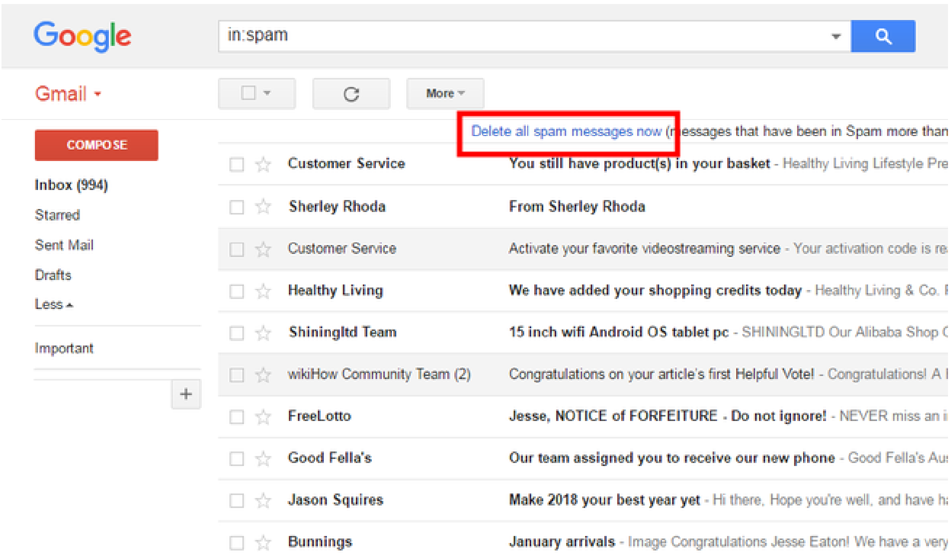
Example 3: Hand-written Digit Recognition

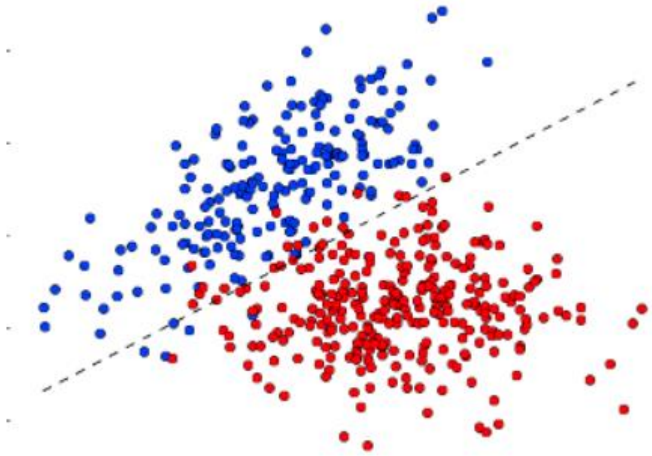
- Represent input image as a vector $x \in \mathbb{R}^{784}$
- Learn a classifier $f(x)$ such that,
  - $f(x) \rightarrow \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$



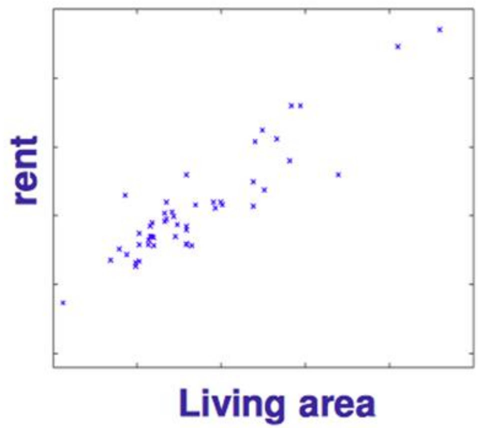
Example 4: Spam Detection

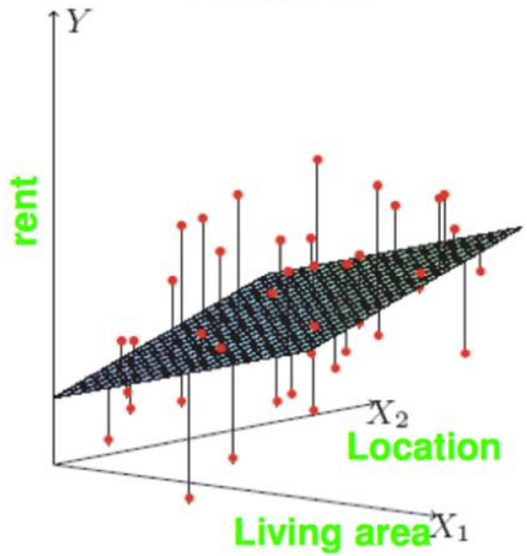
- The task is to classify emails into spam/non-spam.
  - Data x_i is word count
  - This requires a learning system as “enemy” keeps innovating.





A regression problem

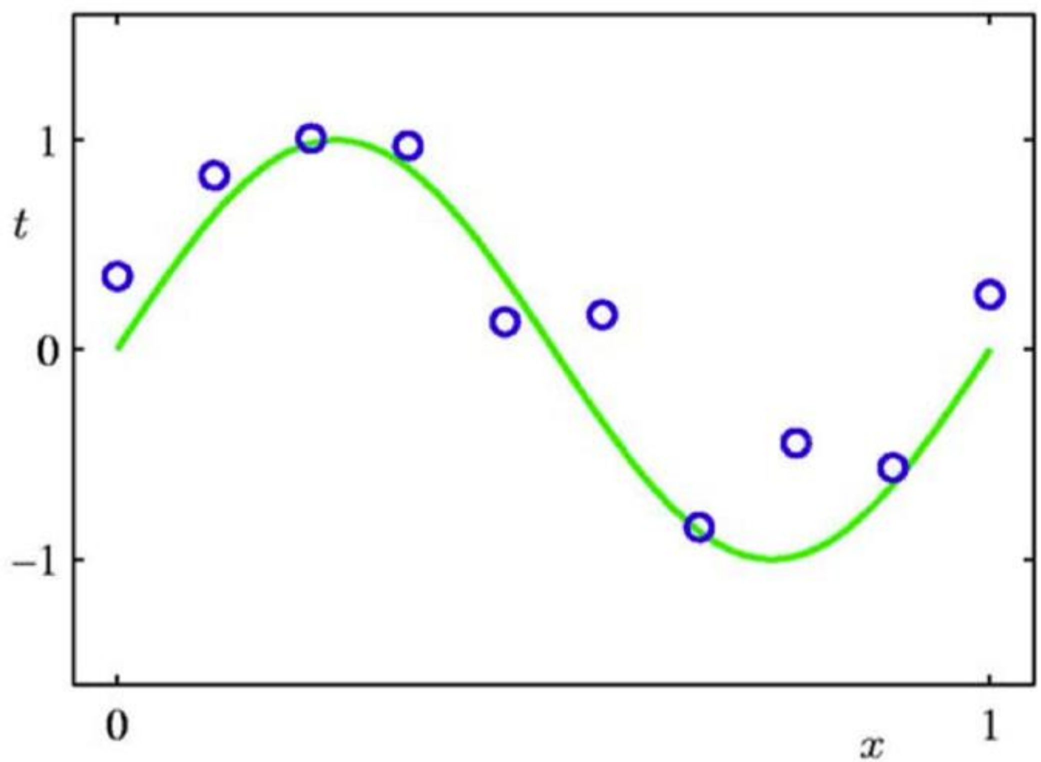




- Features
  - Living area, distance to campus, # bedroom
  - Denote as $x = (x_1, x_2, \ldots, x_d)$
- Target:
  - Rent
  - Denoted as y
- Training set:
  - $x = \{x_1, x_2, \ldots, x_n\} \in R^d$
  - $y = \{y_1, y_2, \ldots, y_n\}$

Regression: Problem setup

- Suppose we are given a training set of N observations $(x_1, x_2, \ldots, x_d)$ and $(y_1, y_2, \ldots, y_n)$, $x_i, y_i \in \mathbb{R}$
- Regression problem is to estimate $y(x)$ from this data



Outline

- Supervised Learning
- Linear regression
- Extension

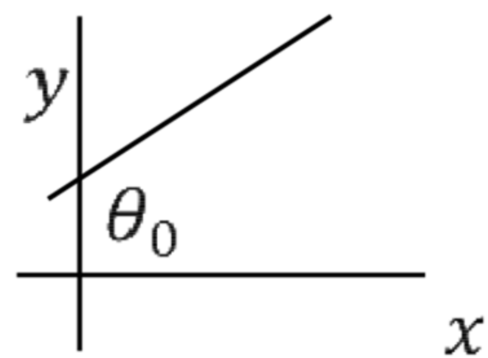
Linear Regression

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

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

Where ϵ is an error term of unmodeled effects of random noise.

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



Least Mean Square Method

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

Definition of Mean Square Error: $L(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i\theta)^2$

- Training: $\hat{\theta} = argmin_{\theta}\ L(\theta)$

The trick is to set the gradient to 0 and find the parameter θ that $\frac{\partial L(\theta)}{\partial \theta}$= 0

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

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

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

- Let's rewrite it as
- $\frac{\partial L(\theta)}{\partial \theta}$=-$\frac{2}{n}(x_1,\ x_2,\dots,\ x_n)^T(y_1,\ y_2,\dots,\ y_n)$+$\frac{2}{n}(x_1,\ x_2,\dots,\ x_n)^T(x_1,\ x_2,\dots,\ x_n)\theta$=0

Define $X = (x_1,\ x_2,\dots,\ x_n)$ and $Y = (y_1,\ y_2,\dots,\ y_n)$

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

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

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

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix}, x = \begin{bmatrix} 1 & {x_1}^{\{1\}} & \dots & {x_1}^{\{d\}} \\ 1 & {x_2}^{\{1\}} & \dots & {x_2}^{\{d\}} \\ \dots & \dots & \dots & \dots \\ 1 & {x_n}^{\{1\}} & \dots & {x_n}^{\{d\}} \end{bmatrix}, \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dots \\ \theta_n \end{bmatrix}$$

$X_{n \times d}$ n =#instances , d = dimension

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

- This is not a big matrix because of $n \gg d$
  - Most times this matrix is invertible.
  - If $X^T X$ are not linearly independent (it’s not a full rank matrix), then it is not invertible.

Alternative ways to optimize

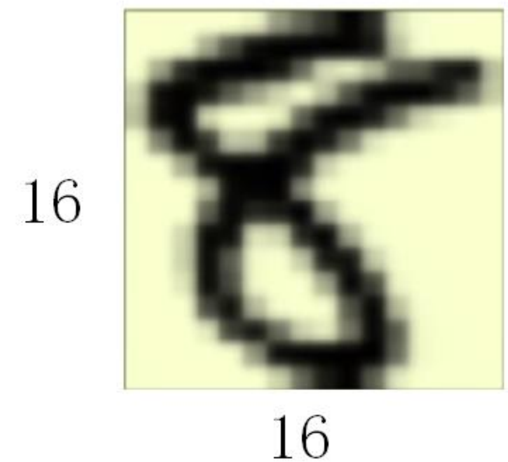
The matrix operation is still can be very expensive to compute

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

- Gradient descent:
  - $\hat{\theta}^{t+1} \leftarrow \hat{\theta}^{t} + \frac{a}{n}\sum_{i=1}^{n} {x_i}^T(y_i - x_i\theta)$
- Stochastic gradient descent (use one data point at a time):
  - $\hat{\theta}^{t+1} \leftarrow \hat{\theta}^{t} + \beta_t * {x_i}^T(y_i - x_i\theta)$

Linear regression for classification

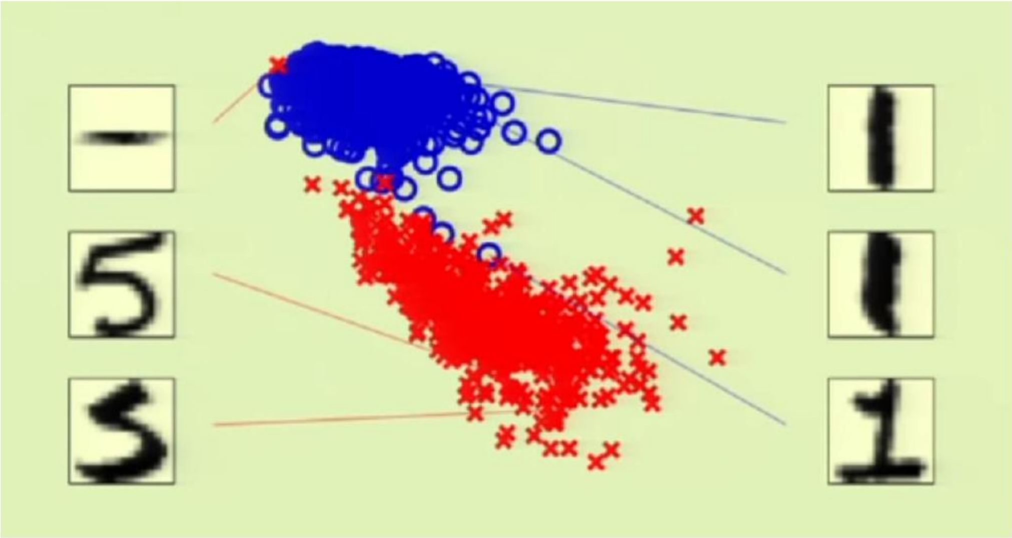
- Raw input $x = (x_1, x_2, \dots, x_{256})$
- Linear model $\theta = (\theta_0 + \theta_1 + \dots + \theta_{256})$
- Extract useful information
  - Include intensity and symmetry $x = (x_0, x_1, x_2)$
  - Intensity = sum up all the pixels
  - Symmetry = -(difference between flip versions)



$x = (x_0, x_1, x_2)$, x_1= intensity, x_2=symmetry

It is almost linearly separable

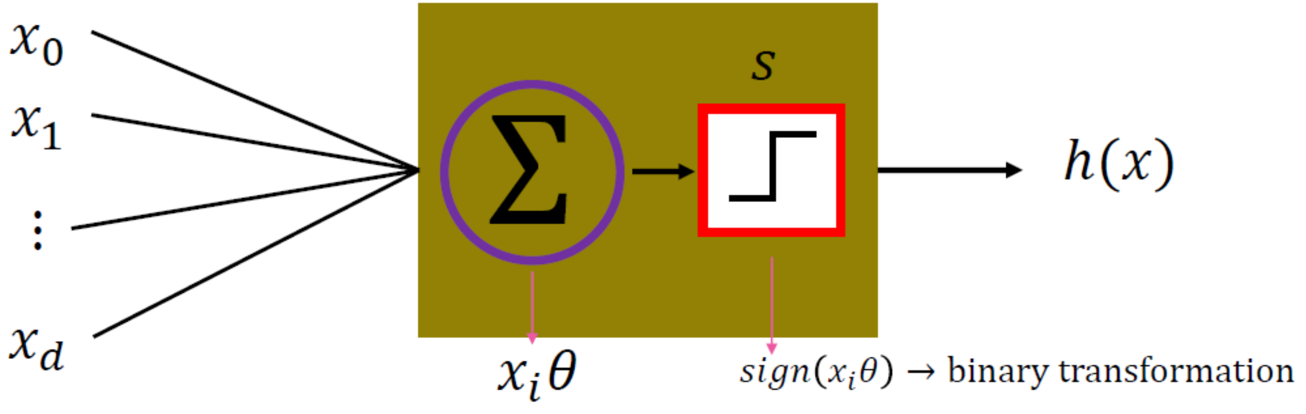
symmetry

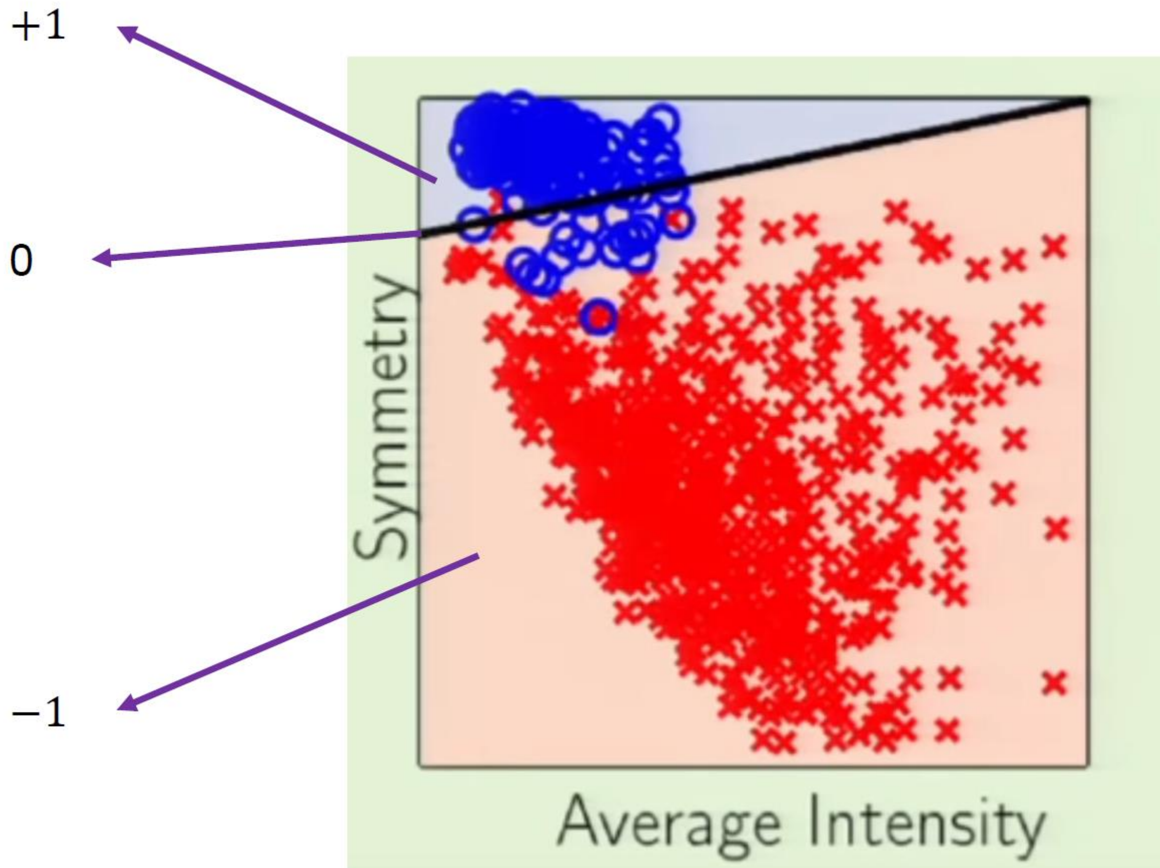


intensity

Linear regression for classification

- Binary-value functions are also real-valued $\pm \in R$
- Use linear regression $x_i\theta \approx y_n = \pm 1$, i =index of a data point.
- Let's calculate, $sign(x_i\theta) = \begin{cases} -1 & x_i\theta < 0 \\ 0 & x_i\theta = 0 \\ 1 & x_i\theta > 0 \end{cases}$



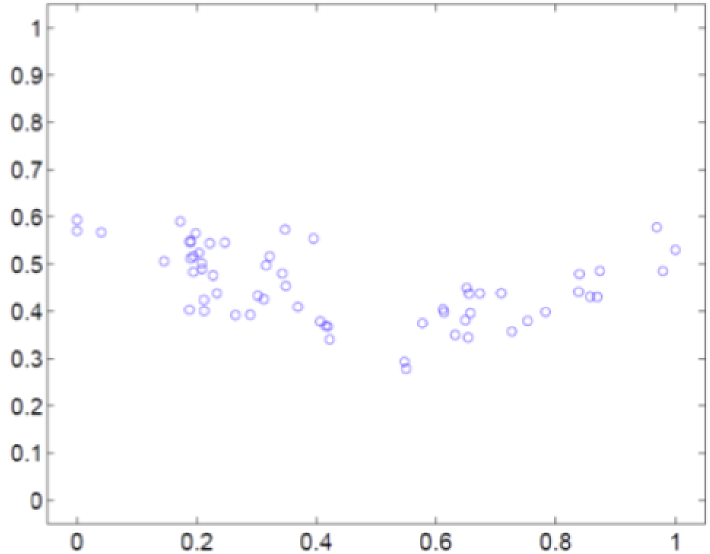


Not really the best for classification, but it's a good start

Outline

- Supervised Learning
- Linear regression
- Extension

Extension to higher-order regression



- Want to fit it into a polynomial regression model:
  $y = \theta_0 + \theta_1 x^1 + \ldots + \theta_d x^d + \epsilon$
- Let $z = \{1, x^1, x^2, \ldots x^d\} \in R^d$ and $\theta = (\theta_0 + \theta_1 + \ldots + \theta_d)^T$

$$\rightarrow y = z\theta$$

Least mean square still works here

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

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

$$\theta = (Z^T Z)^{-1} Z^T Y = Z^+ Y$$

where $Z = (1, x^1, x^2, \ldots x^d)$ and $Y = (y_1, y_2, \ldots, y_n)$

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

What is happening in polynomial regression

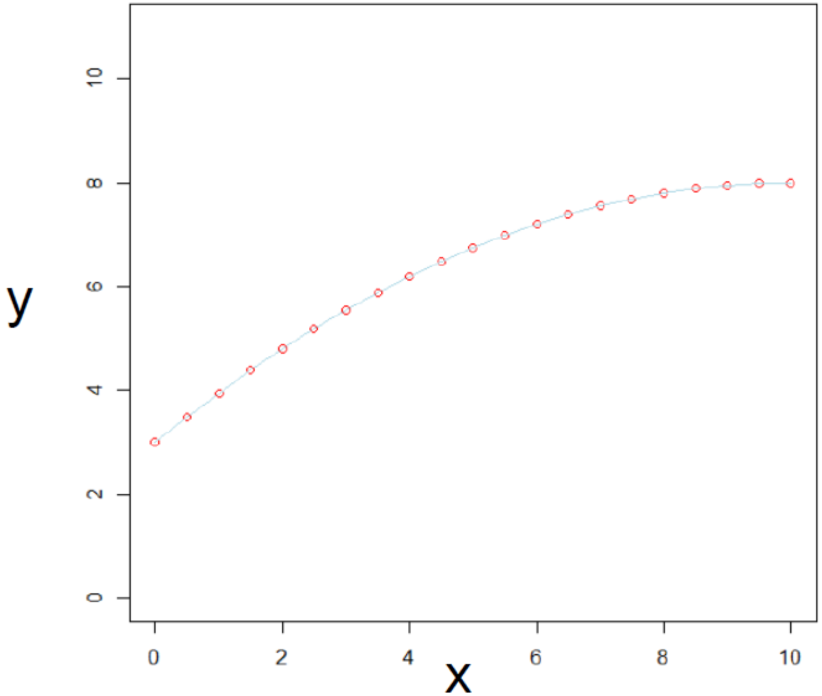
$$x = [0, 0.5, 1, \ldots, 9.5, 10]$$

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

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

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

d=2



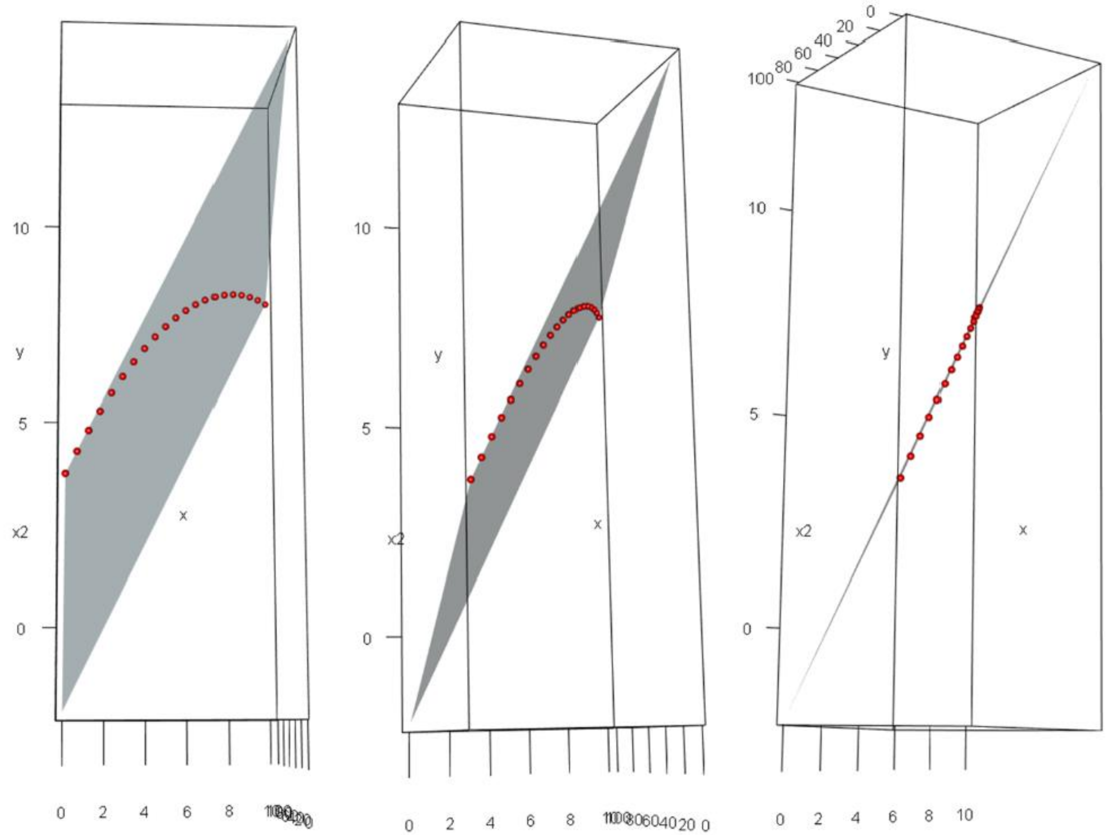
MSE=0

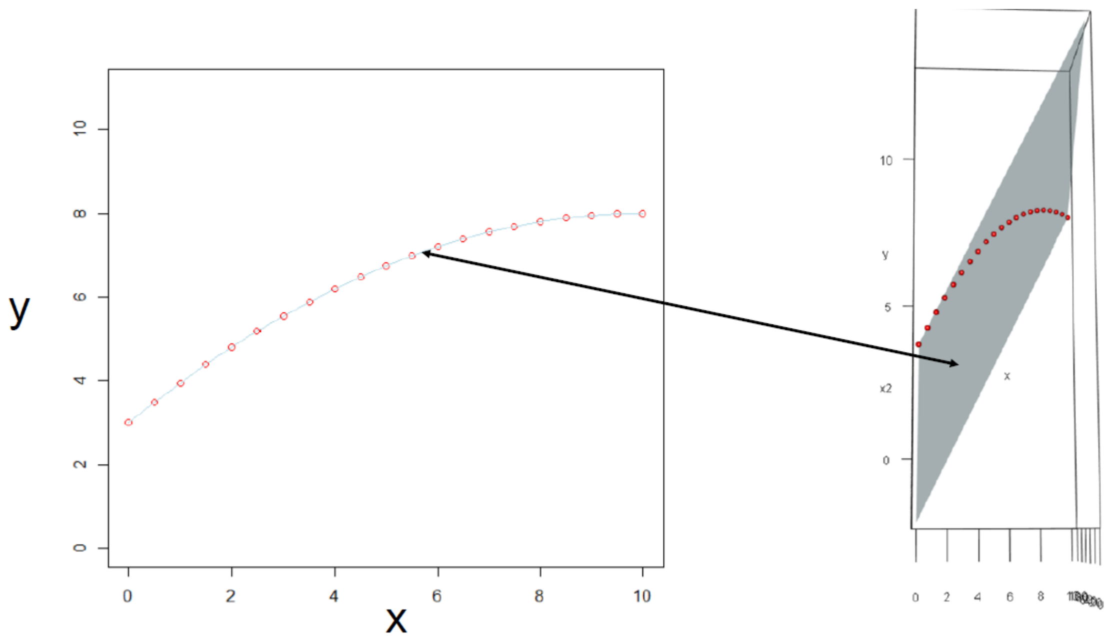
Let's add one more for the feature space

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

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

d=3

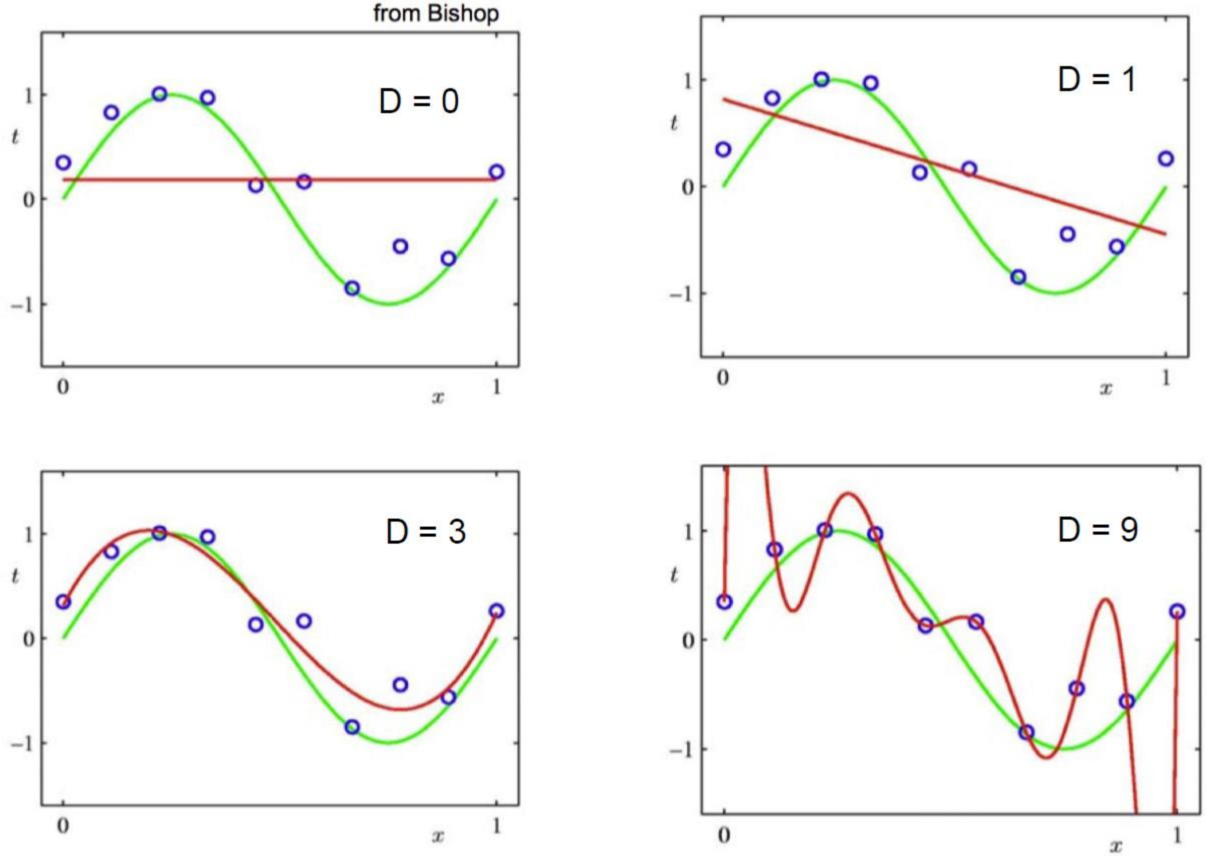




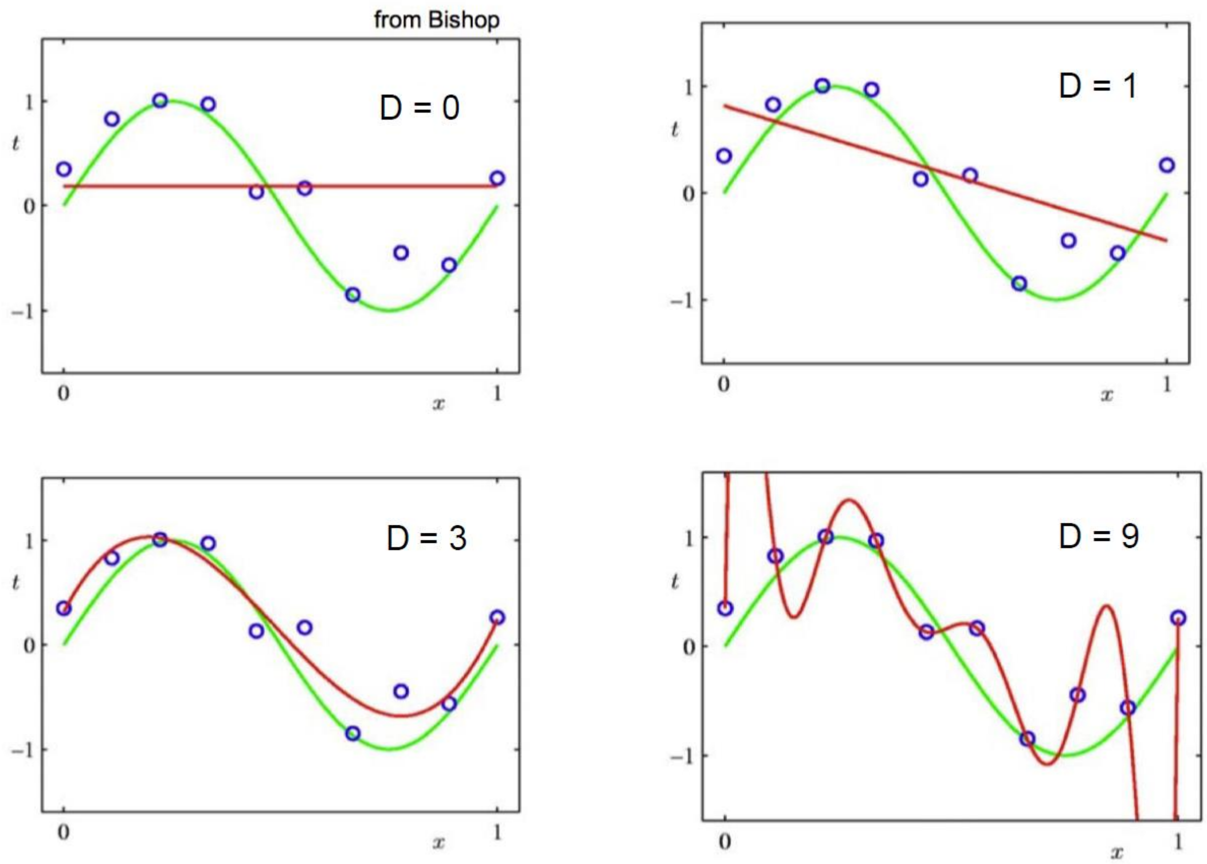
- We are fitting a d -dimensional hyperplane in a $d + 1$ dimensional hyperspace.
- The hyperplane is still ‘flat’/’linear’ in 3D, with a non-linear regression (a curvy line).

Increasing the maximal degree

from Bishop



Which one is better?



Can we increase the maximal polynomial degree to a very large dimension, as a “safe” solution?

- See it in our next lecture