$X = [\ h \quad w \quad a\ ] \rightsquigarrow Z = [\ h\ ] \rightsquigarrow$ feed it in an ML algorithm (NB, SVM, ...)

↳ acc = 80%

$Z = [\ w\ ] \rightsquigarrow$ acc = 70%

$Z = [\ a\ ] \rightsquigarrow$ acc = 65%

$Z = [\ h \quad w\ ] \rightsquigarrow$ acc = 85%

$Z = [\ h \quad a\ ] \rightsquigarrow$ acc = 58%

Forward Feature Selection

Backward Feature Selection

$$X = \begin{bmatrix} h & w & a \\ & & \\ & & \end{bmatrix} \not{\ell}$$

→ LDA

Georgia Tech

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

1000 → 20% to testing
→ 10% to validation
→ 70% training

Functions $\mathcal{F}$

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

discrete ~ classification

Continaus ~ regression

Training data

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

$x^{\{i\}}$

$X = \begin{bmatrix} \\ \\ \end{bmatrix}_{n \times d}$ $y_a = \begin{bmatrix} 0 \\ \\ \end{bmatrix}_{n \times 1}$

LEARNING

find $\hat{f} \in \mathcal{F}$

s.t. $y_a \approx f(x^{\{i\}})$

Learning machine

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

$\hat{Y}_p = \begin{bmatrix} 0 \\ \\ \end{bmatrix}$

$f(\cdot)$

ML

PREDICTION

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

New data

x

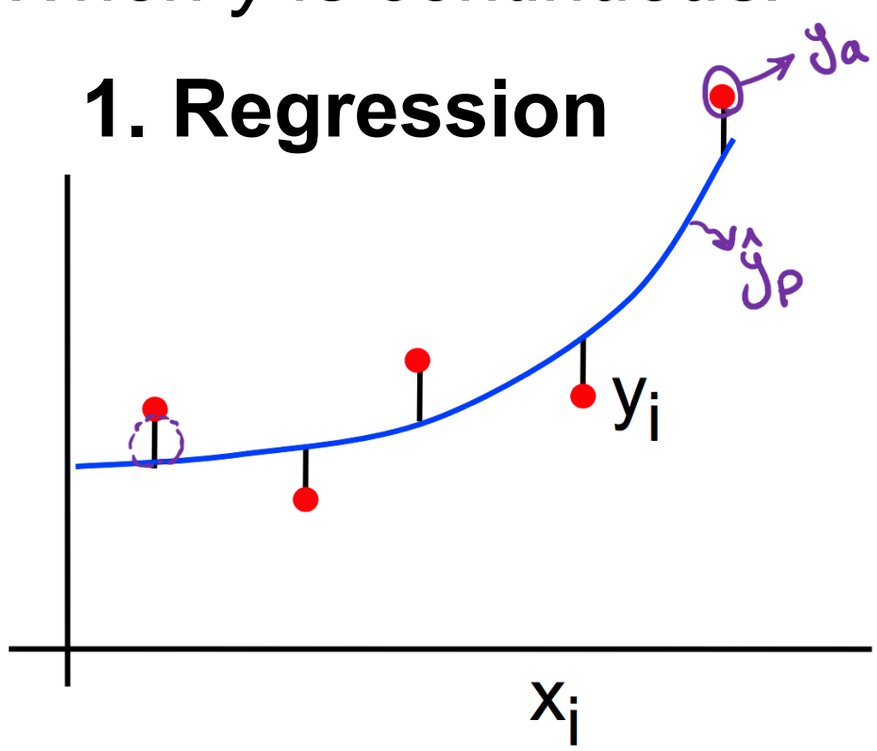
Supervised Learning: Two Types of Tasks

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

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

When y is continuous:

1. Regression

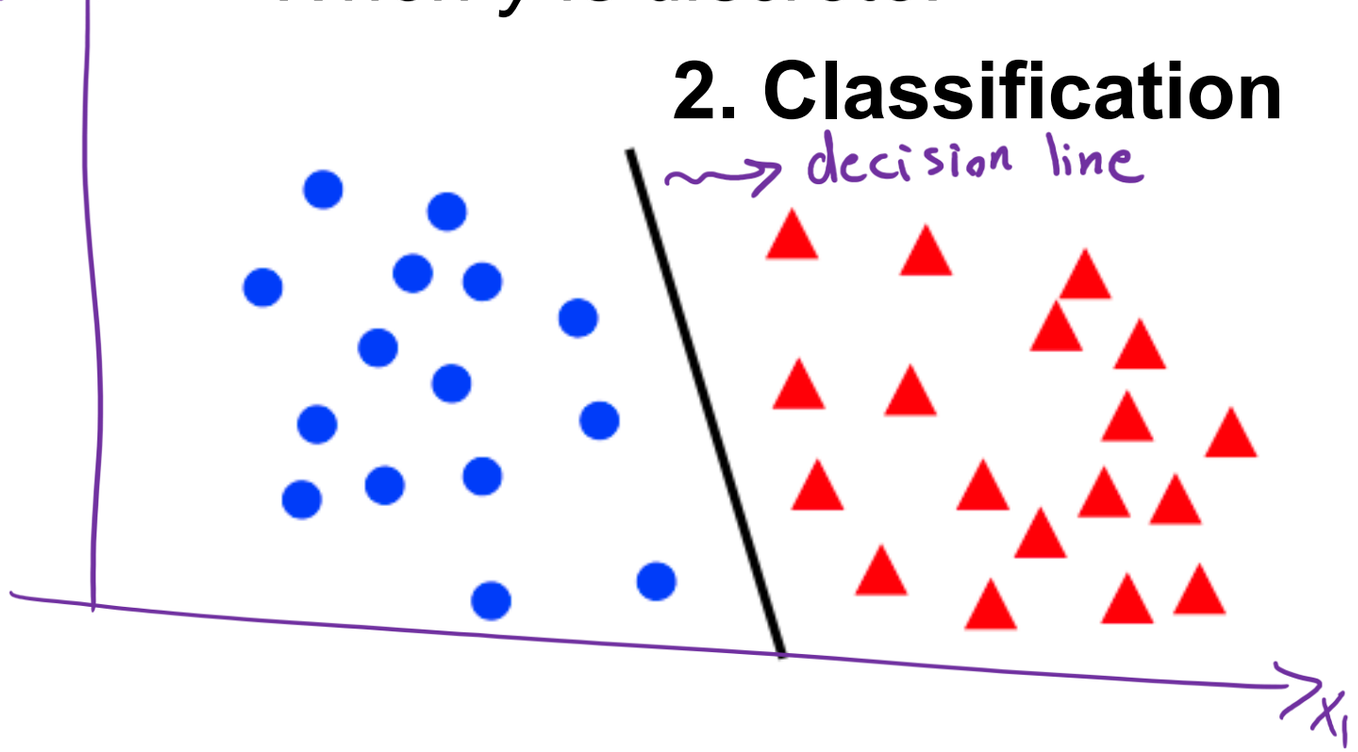


Curve fitting

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

When y is discrete:

2. Classification

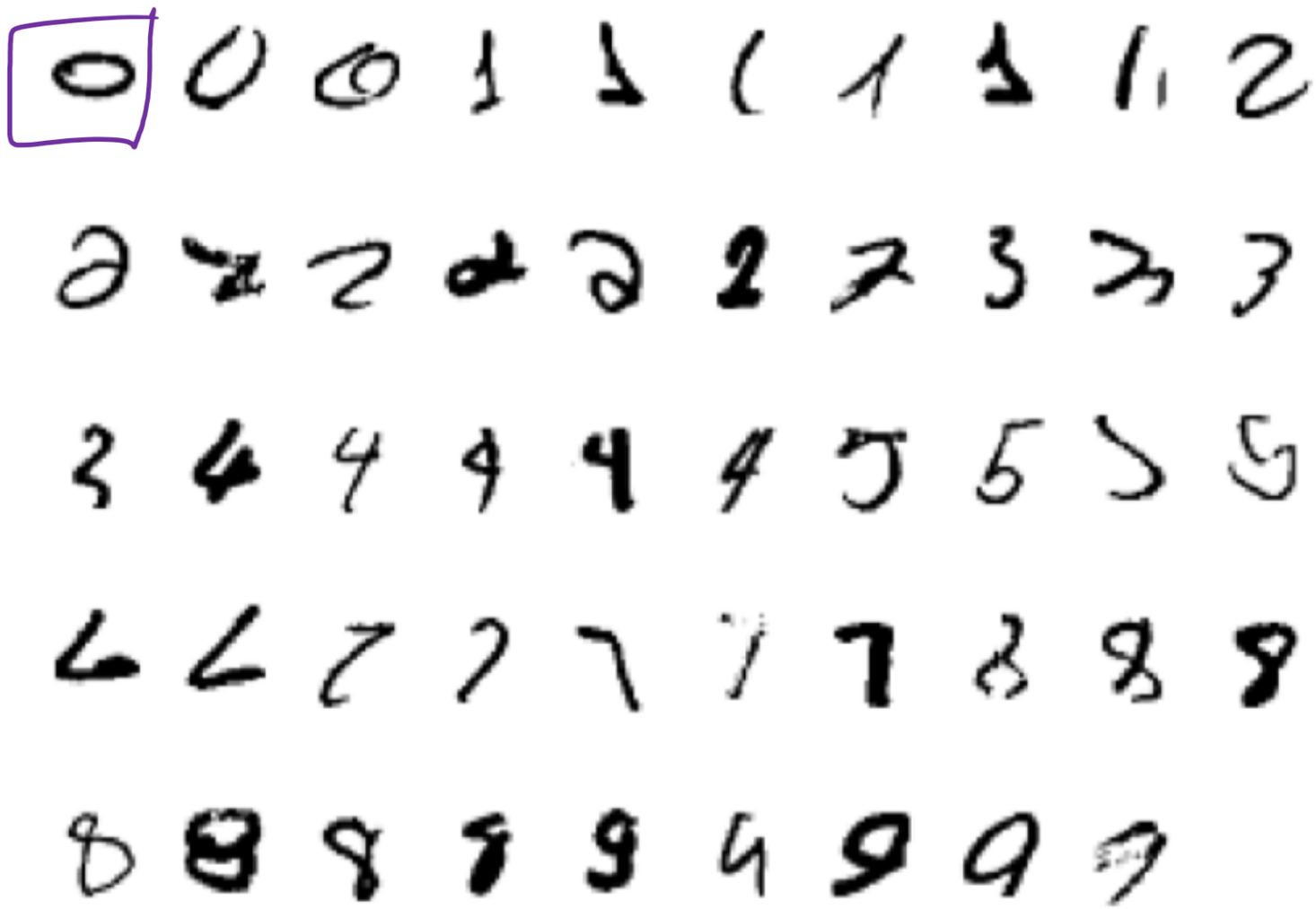


Class estimation

Classification Example 1: Handwritten digit recognition

As a supervised classification problem

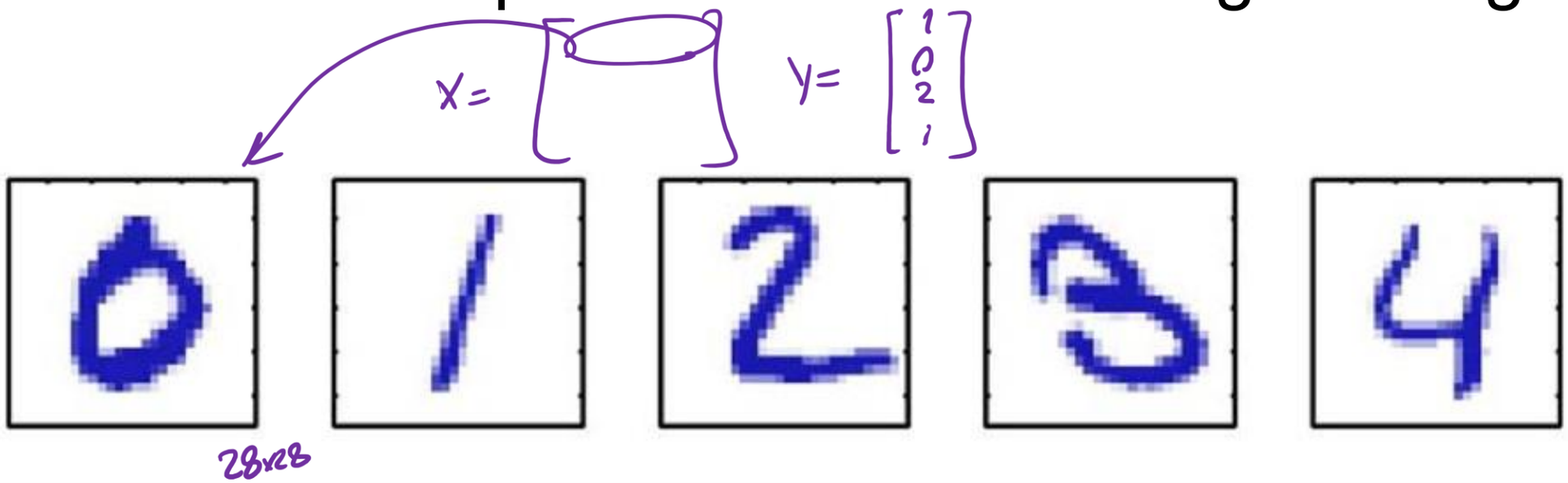
Start with training data, e.g. 6000 examples of each digit

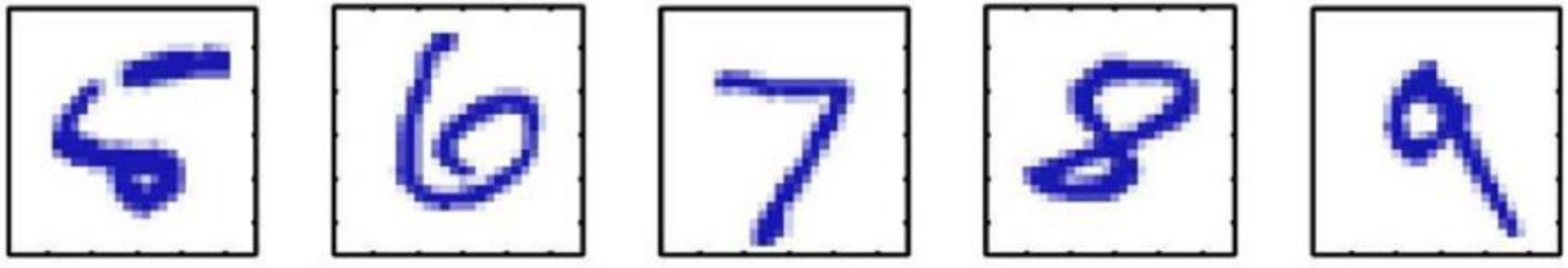


Multi class Classification

- 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

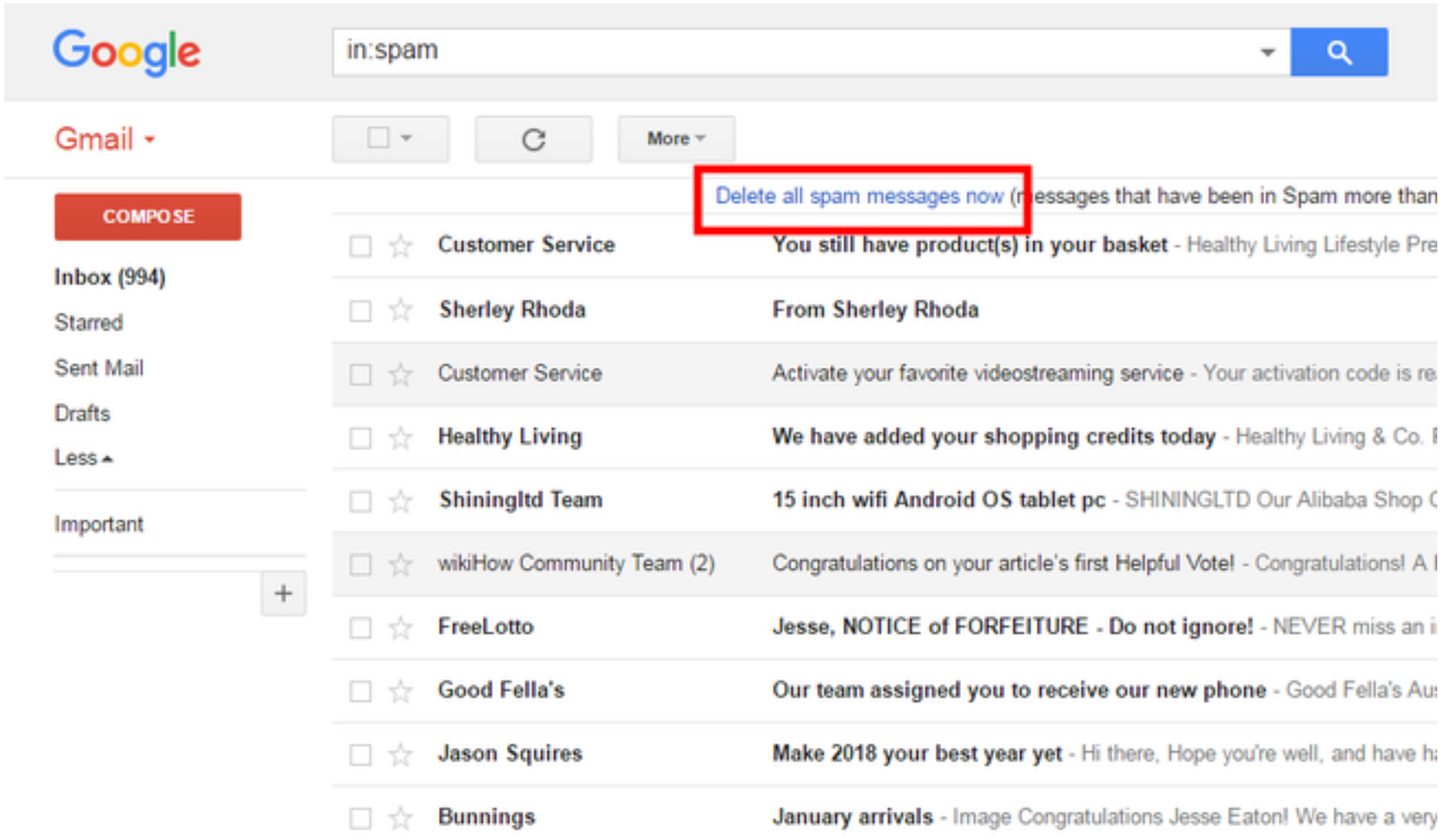
Represent input image as a vector $\mathbf{x} \in \mathbb{R}^{784}$

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

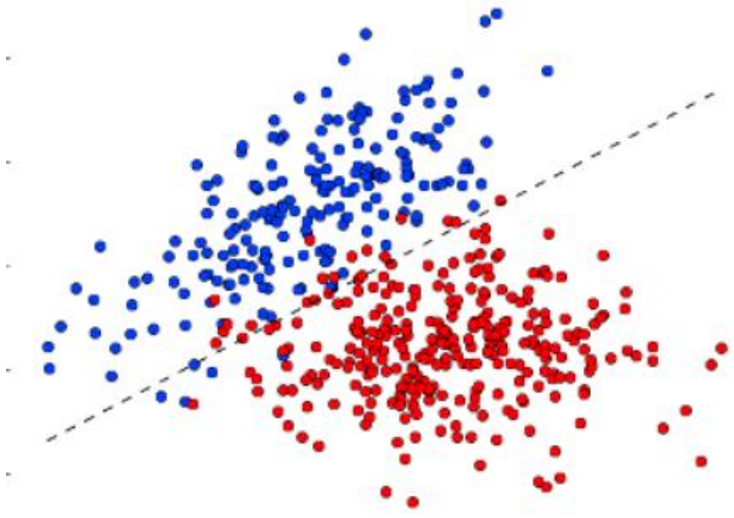
$$f : \mathbf{x} \rightarrow \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

Classification Example 2: Spam Detection

Binary classification



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

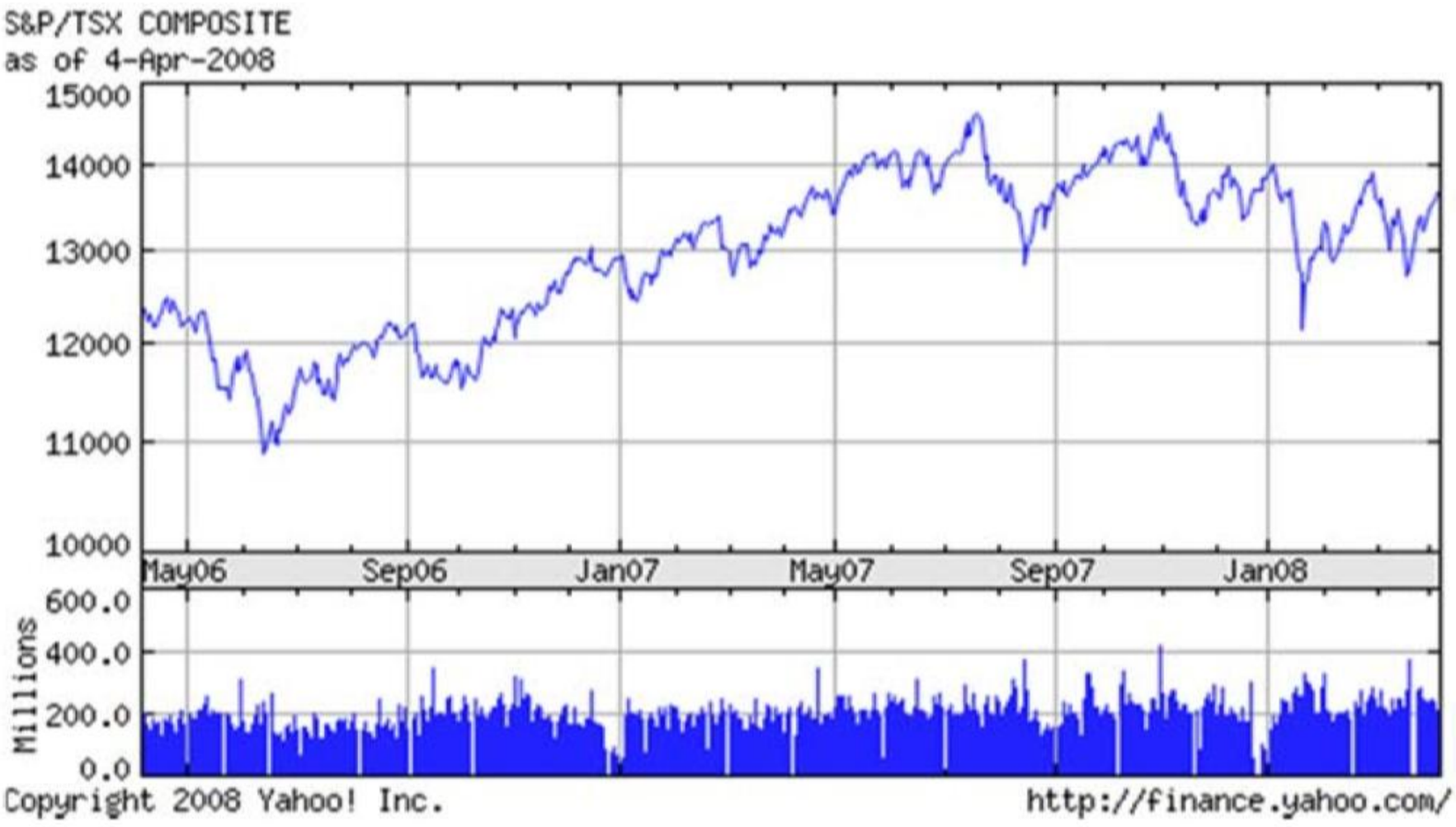
- Suppose you are to move to Atlanta
- And you want to find the **most reasonably priced** apartment satisfying your **needs:**

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

A regression problem

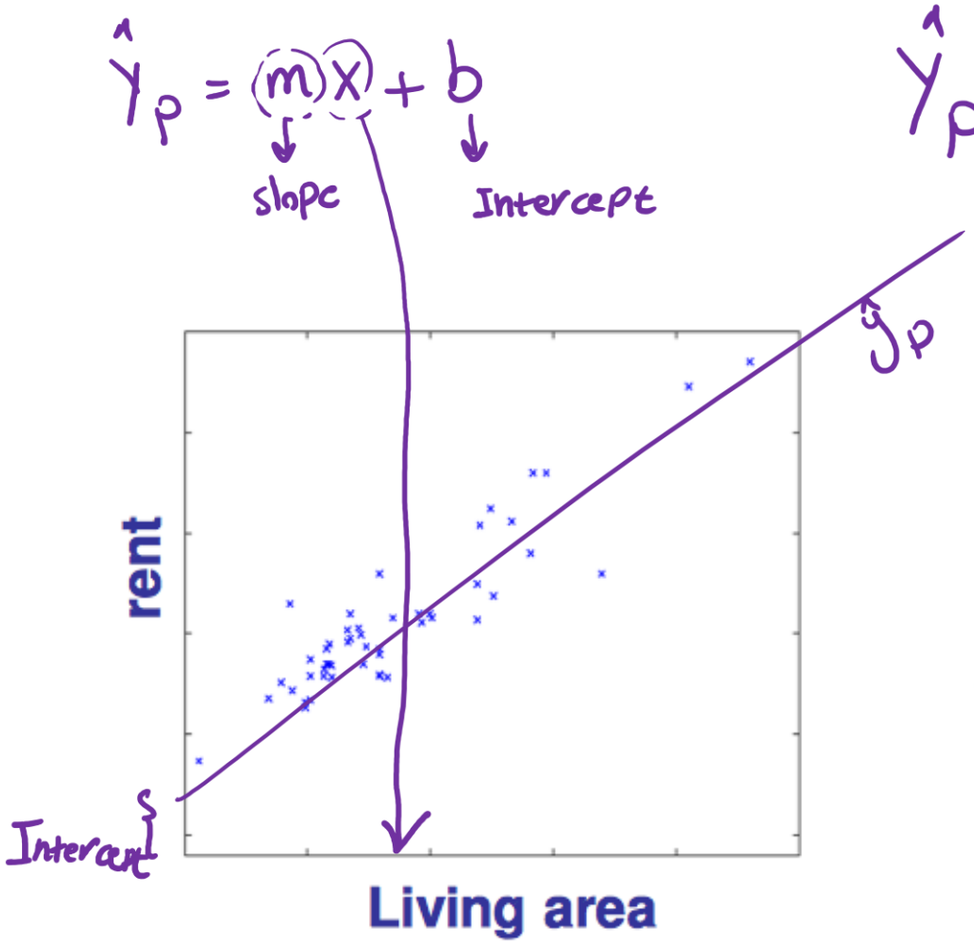
Living area (ft^2)	# bedroom	Rent ($)
230	1	600
506	2	1000
433	2	1100
109	1	500
…		
150	1	?
270	1.5	?

Regression Example 2: Stock Price Prediction



- Task is to predict stock price at future date

A regression problem



$\hat{y}_p = \theta_0 + \theta_1 x_1$

θ_0: bias term

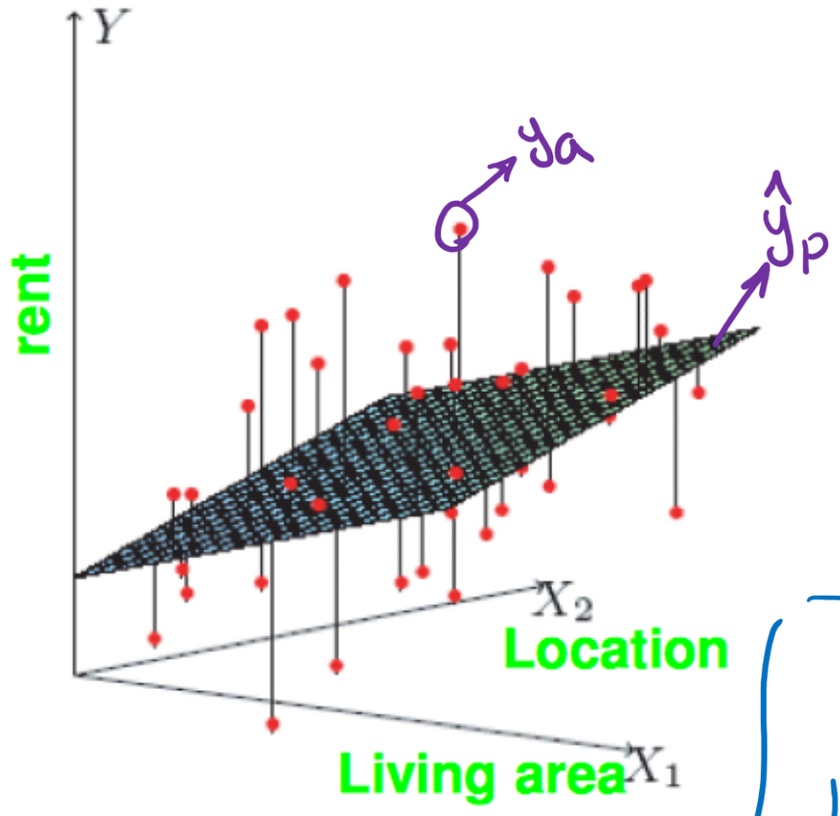
θ_1: The parameter that defines contribution of feature x_1

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

$$X = \begin{bmatrix} 1 & x_1 & x_2 & \cdots & x_d \\ 1 & & & & \\ 1 & & & & \\ 1 & & & & \\ \vdots & & & & \end{bmatrix}$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{bmatrix}$$

$X_{1\times d}^{[i]} \cdot \theta_{d\times 1} = \hat{y}_{p\ 1\times 1} \in \mathbb{R}$



- Training set:

$$x = \{x^{\{1\}}, x^{\{2\}}, \ldots, x^{\{n\}}\} \in R^d$$

$$y = \{y^{\{1\}}, y^{\{2\}}, \ldots, y^{\{n\}}\}$$

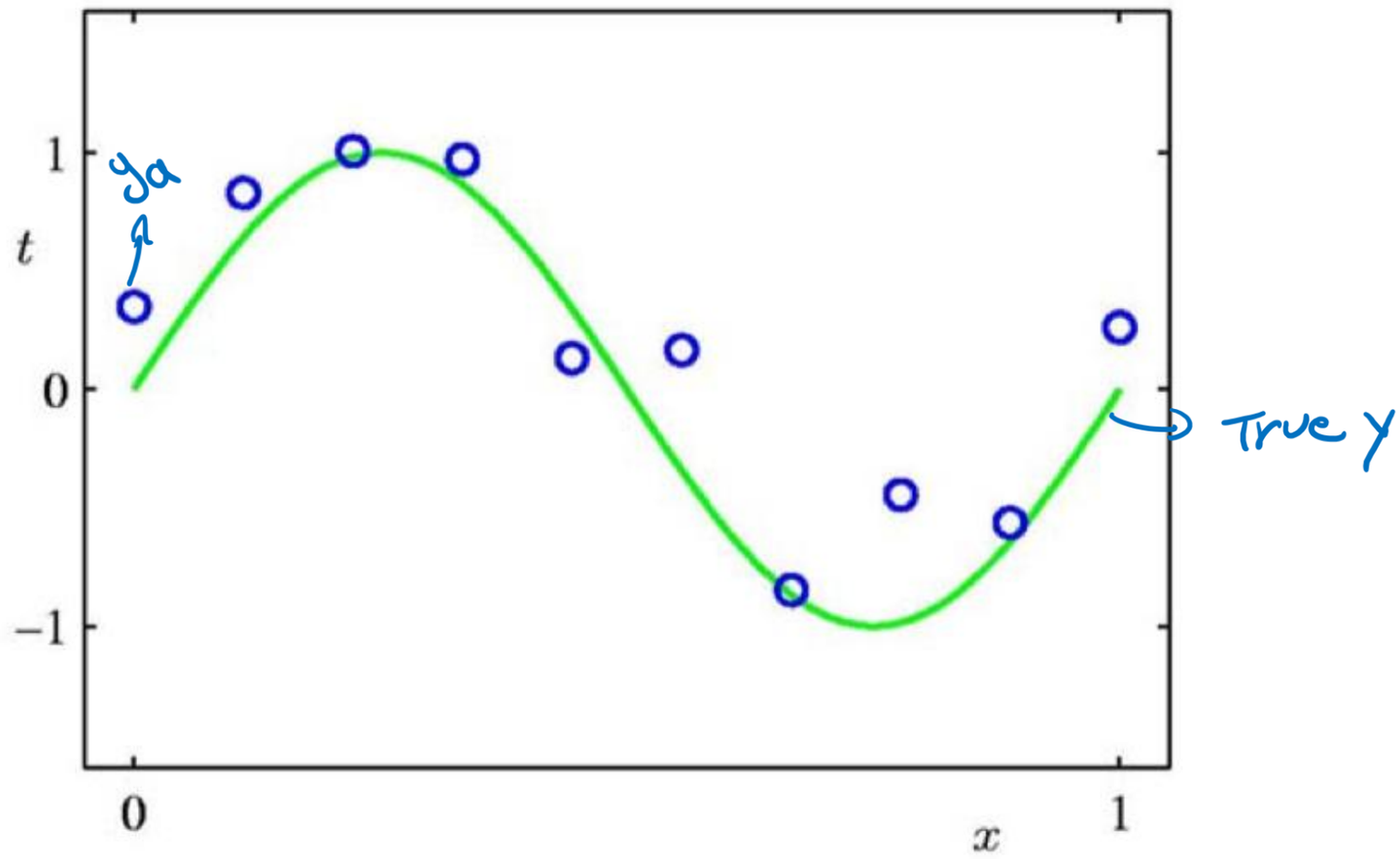
$\hat{y}_p = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

$\hat{y}_p = \theta_0 + \theta_1 x_1 + \theta_2^2 x_2$ LCF

$\hat{y}_p = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d = X\theta$

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

Linear Regression

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

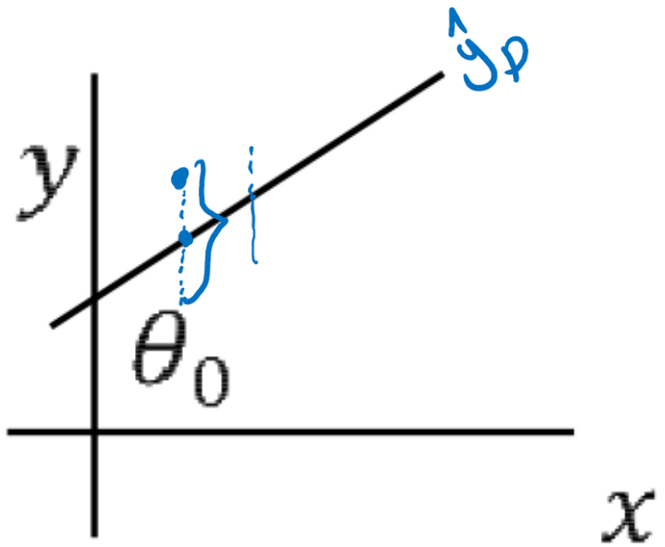
$$y = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d + \epsilon = X\theta$$

LCF

- where ϵ is an error term of unmodeled effects or random noise

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

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



$E[\cdot]$ → expectation

$E(\,)$ → error

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

Least Mean Square Method

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

Training $\hat{\theta} = argmin_{\theta} L(\theta) = \frac{1}{n}\sum_{i=1}^{N}\left(y^{\{i\}} - x^{\{i\}}\theta\right)^2$

loss, avg, Convex, $E(\theta)$ error, $\in \mathbb{R}$, y_a, $\hat{y}_p$

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

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

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

For Loop

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

$X\theta = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d$

Matrix form

Column Vector

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

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

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

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

$\hat{y}^{\{1\}}$ $\hat{y}^{\{2\}}$ $\hat{y}_p$

Matrix Version and Optimization

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

Let's rewrite it as:

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

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

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

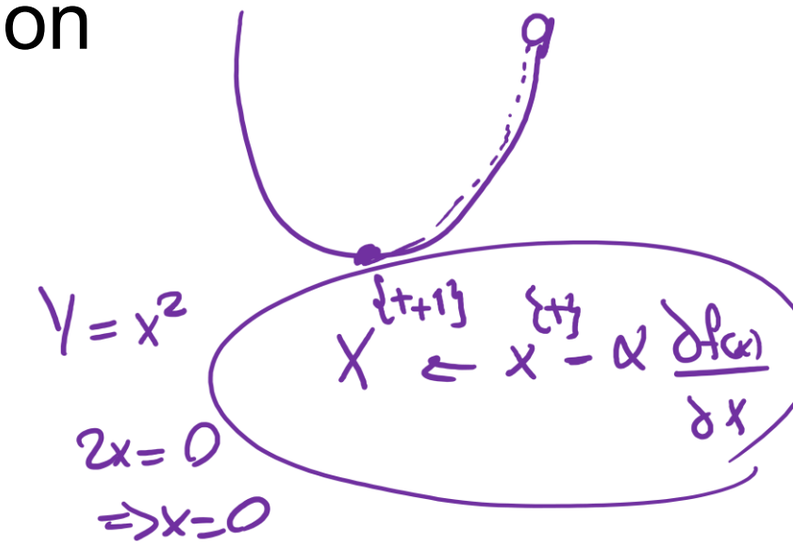
$$\Rightarrow \underset{d+1\times 1}{\theta} = \underbrace{(X^T X)^{-1}X^T}_{d+1\times n}\underbrace{y}_{n\times 1} = X^+ y$$

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

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

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

$X_{n \times d}$ $n =$ instances $d =$ 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 $\mathbf{X^T X}$ are not linearly independent (it's not a full rank matrix), then it is not invertible.

Alternative Way to Optimize

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

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

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

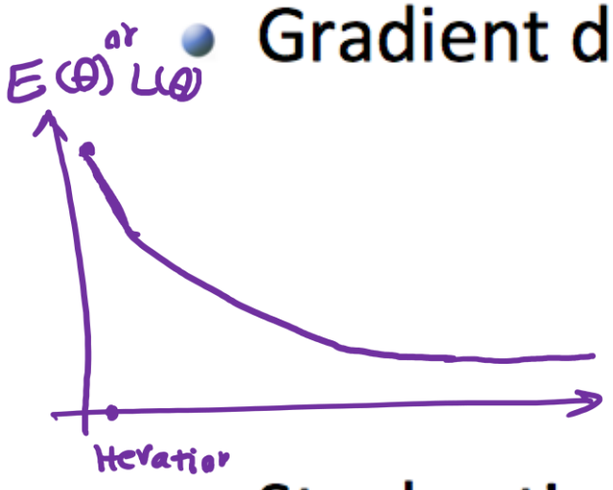
E(θ)

epoch

Full

FGD

- Gradient descent



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

epoch

BGD → Batch Gradient descent

100

20 20 20 20 20

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

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

Recap

- Stochastic gradient update rule

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

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

↓ BGD

↑

- Gradient descent

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

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

- Solve normal equations

$$\theta = (X^T X)^{-1} X^T y \rightsquigarrow \text{Closed form solution}$$

  - 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 ..)

$$\hat{y}_p = \underbrace{\theta_0}_{\cdots} + \underbrace{\theta_1}_{\cdots} x_1 + \cdots + \underbrace{\theta_d}_{\cdots} x_d = X\theta$$

LCF

$$\underset{1\times d+1}{X^{\{i\}}}\ \underset{d+1\times 1}{\theta} \in \mathbb{R}$$

θ → column vector $\begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}$

Convex optimization

$$L(\theta) = E(\theta) = \underbrace{\frac{1}{N}\sum_{i=1}^{N}}_{\text{avg}} (y_a - \hat{y}_p)^2 = E\left[(y_a - y_p)^2\right]$$

$\underline{X\theta} = \hat{y}^{\{i\}}$

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

$$\underset{d+1\times 1}{\theta} = \underbrace{\left(\underset{d+1\times n}{X^T}\ \underset{d+1\times d+1}{X}\right)^{-1}}\ \underset{d+1\times n}{X^T}\ \underset{n\times 1}{Y}$$

→ Closed Form Solution

FGD, SGD, BGD

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

Linear regression for classification

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

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

16

Extract useful information

$\theta_0\ \theta_1\ \theta_2$

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

16

Sum up all the pixels = intensity

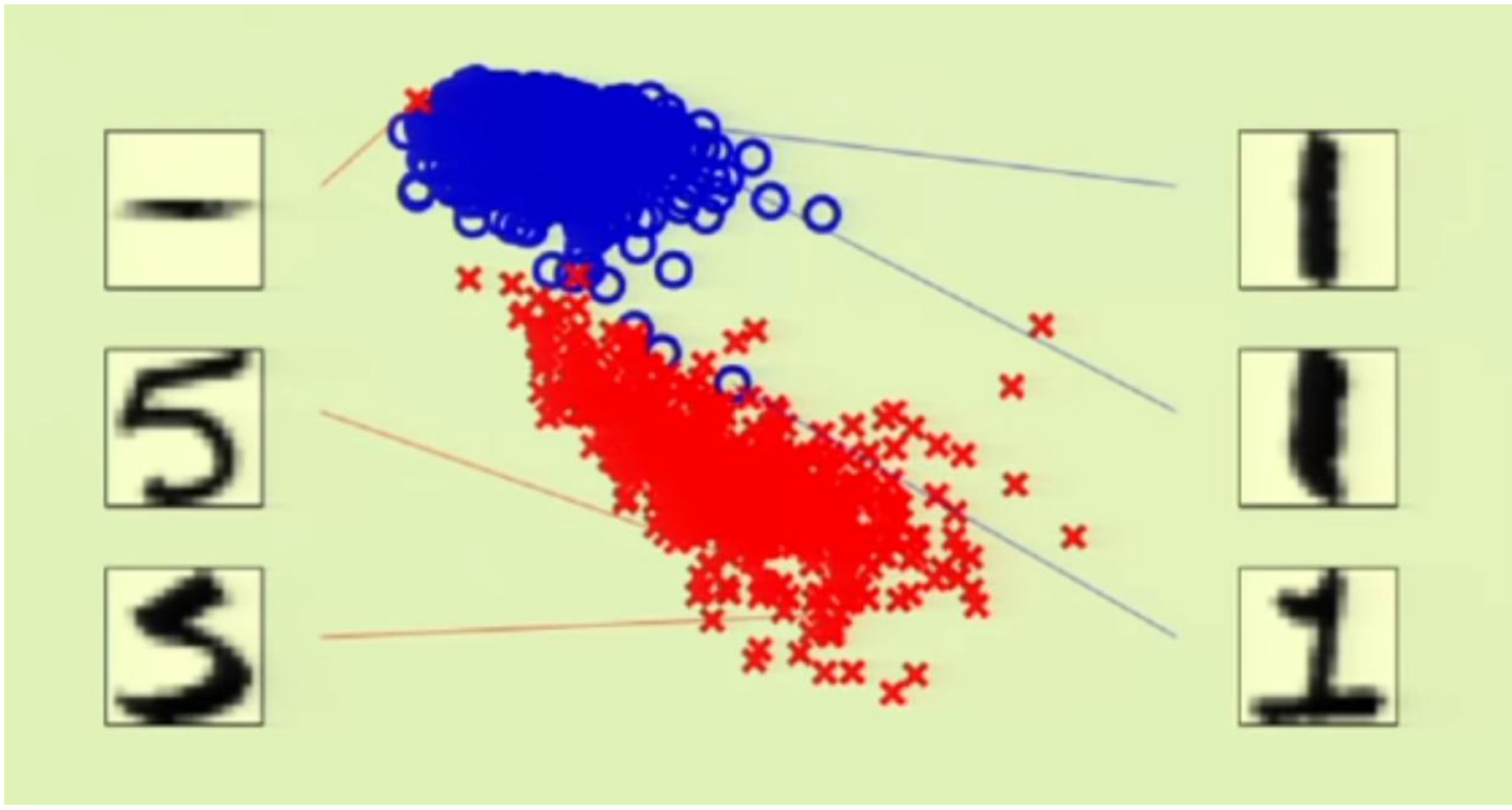
Symmetry = -(difference between flip version)

"Feature Engineering"

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

$$x_1 = intensity\ x_2 = symmetry$$

It is almost linearly separable



symmetry

intensity

Linear regression for classification

$x^{\{i\}}\theta \in \mathbb{R}$

$y_{reg} = [\ ]$ continous $\rightarrow$ $y_{class} = [\ ]$ discrete

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

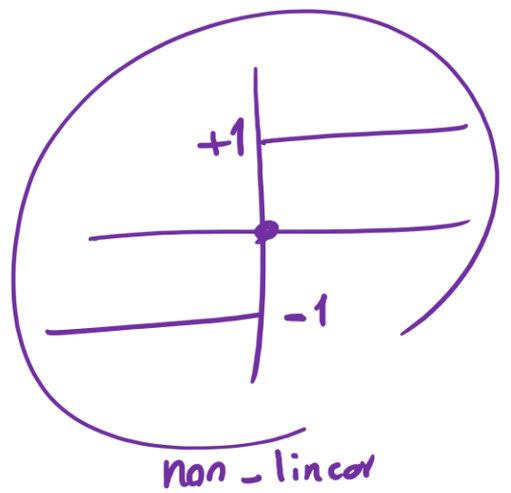
Use linear regression $x^{\{i\}}\theta \approx y^{\{i\}} = \pm 1$ $i =$ index of a data-point

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

5

decision line

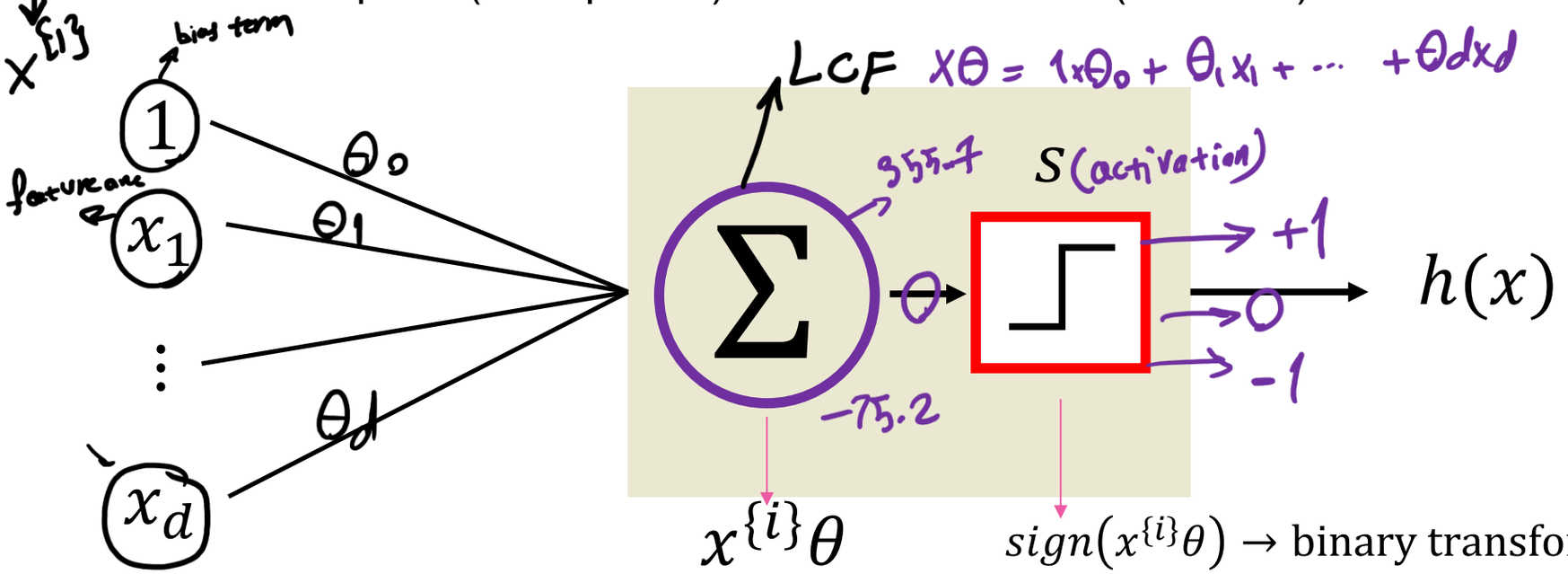
1



SGD

$x^{\{i\}}$

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



a learning block

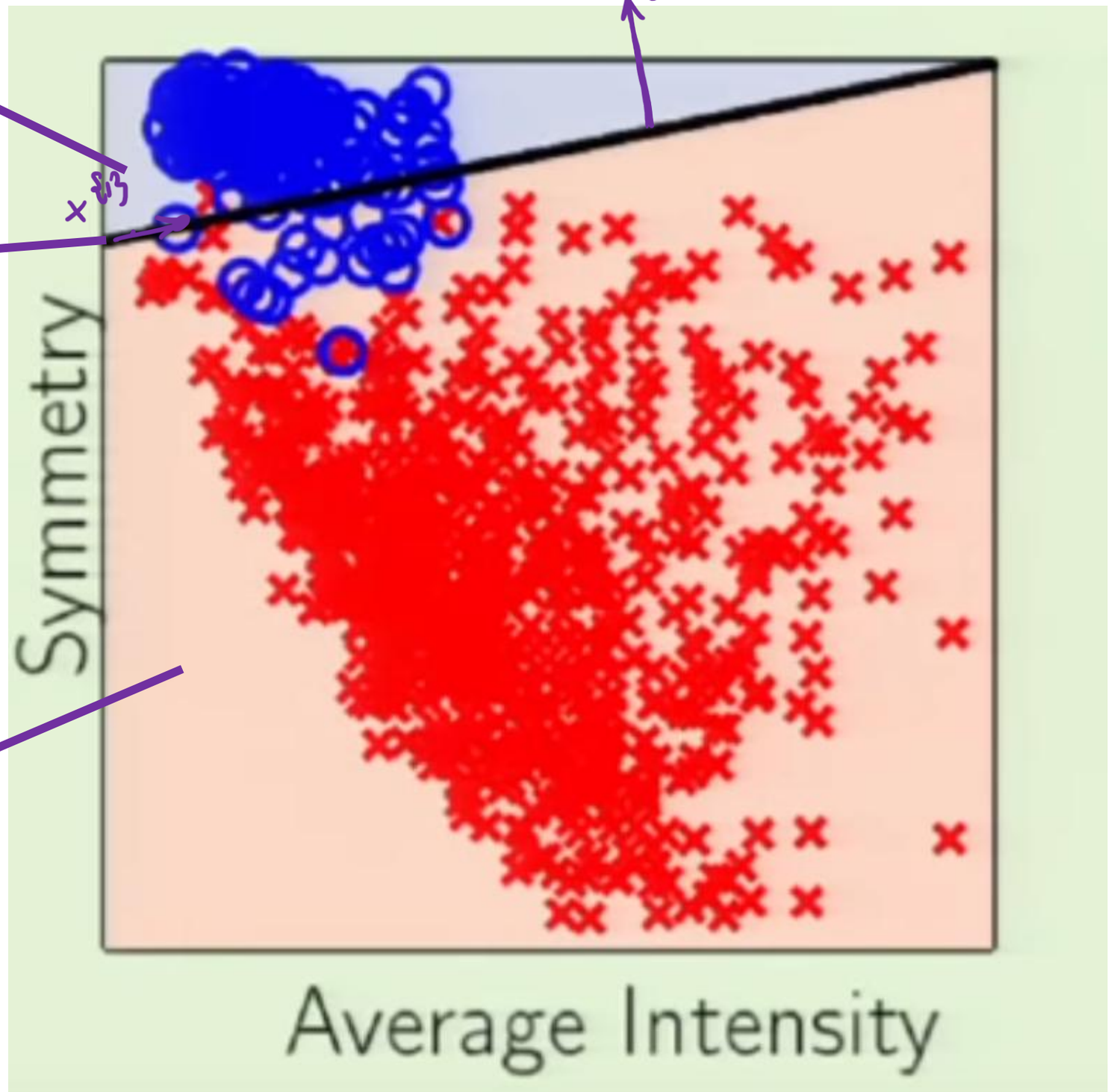
+1

0

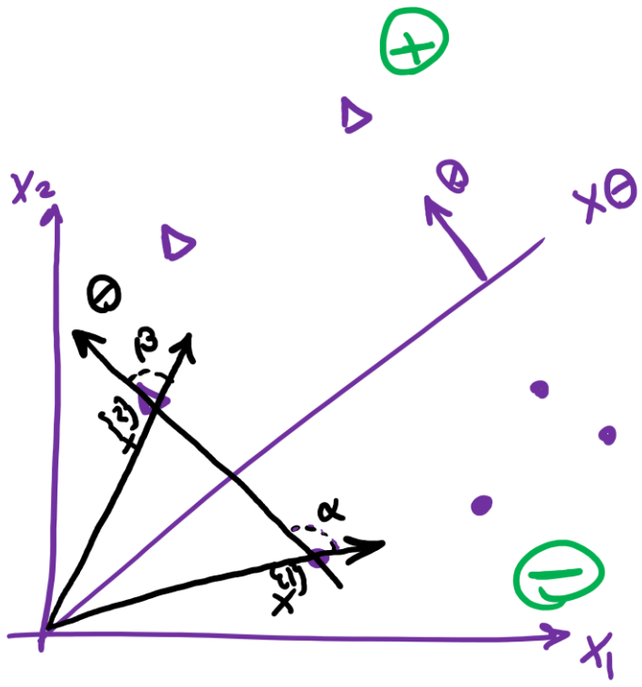
$X\theta = y$

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

−1



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



$$x^{[1]} \cdot \theta = \underbrace{|x^{[1]}|}_{+} \cdot \underbrace{|\theta|}_{+} \underbrace{\cos\alpha}_{-} < 0$$

$$\rightarrow -1$$

$$x^{[2]} \cdot \theta = \underbrace{|x^{[2]}|}_{+} \cdot \underbrace{|\theta|}_{+} \underbrace{\cos\beta}_{+} > 0$$

$$\rightarrow +1$$

$\rightarrow$ plane

$$\hat{y}_p = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

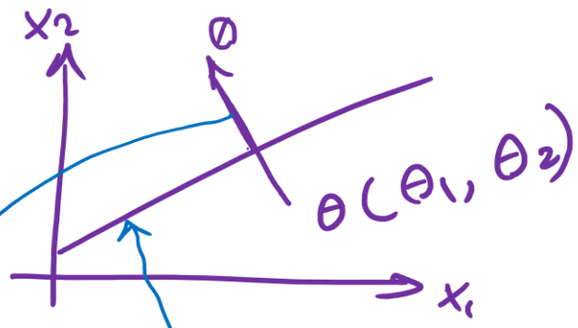
https://www.math3d.org/UPNV1IQvxt

$$\hat{y}_p = \theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$$

$$\theta_2 x_2 = -(\theta_0 + \theta_1 x_1)$$

$$x_2 = -\frac{\theta_0}{\theta_2} - \frac{\theta_1}{\theta_2} x_1$$

$y = mx + b$



$$\rightsquigarrow \left(1, -\frac{\theta_1}{\theta_2}\right)$$

$$\left((\theta_1, \theta_2)\right) \cdot (\theta) = \theta_1 - \theta_2 \frac{\theta_1}{\theta_2} = 0$$

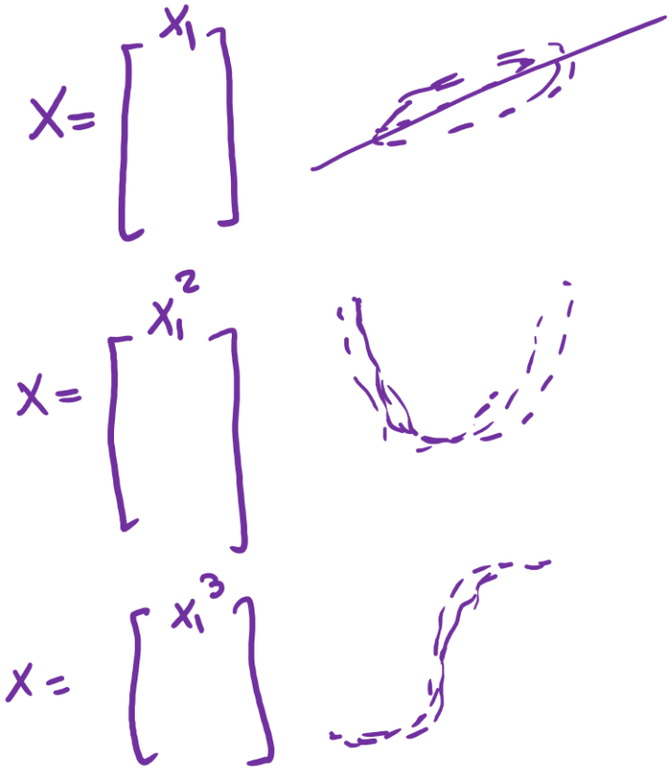
$$X\theta = 0$$

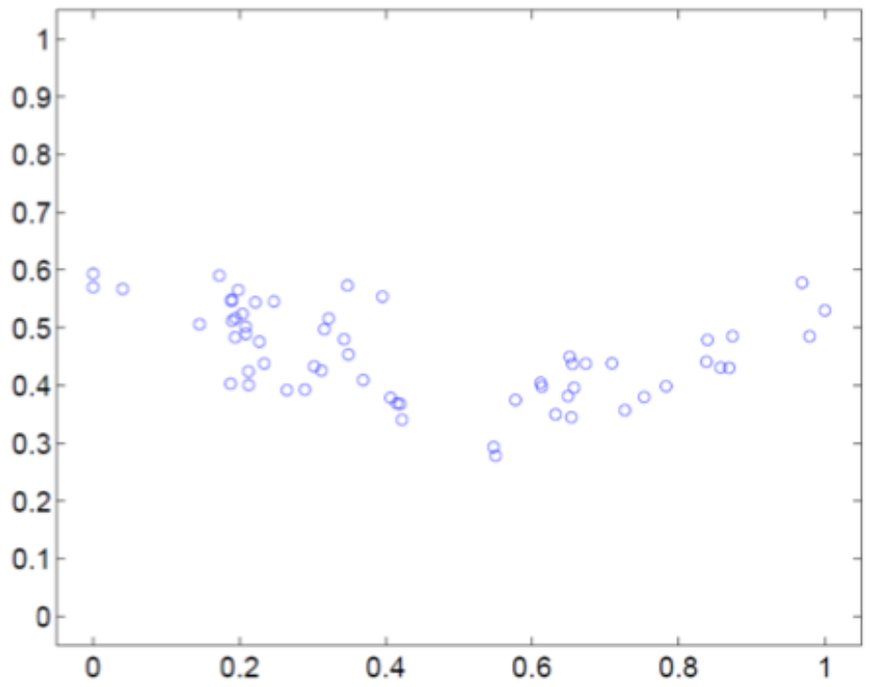
Outline

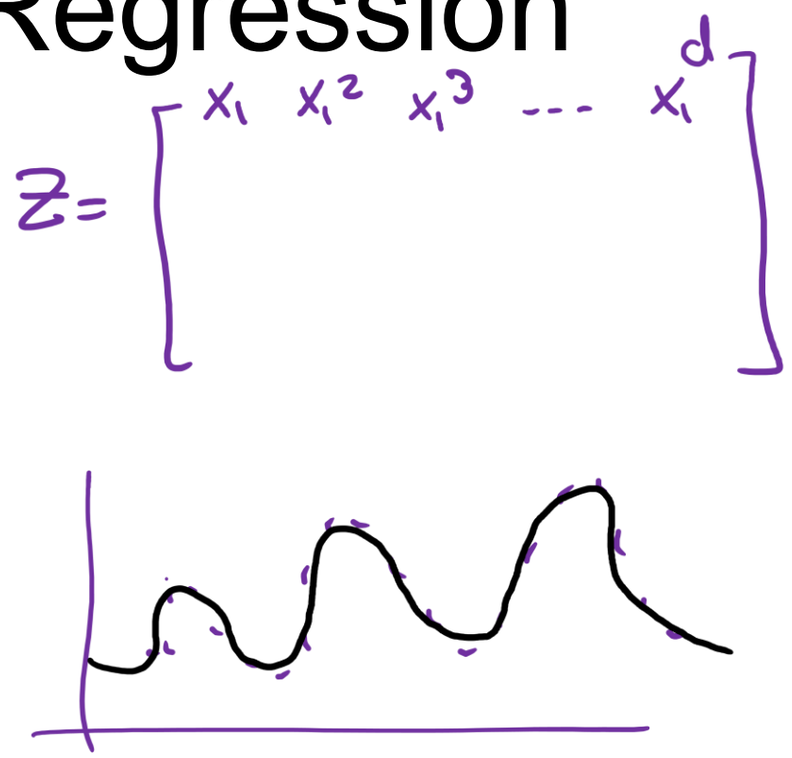
- Supervised Learning
- Linear Regression
- Extension 

Polynomial Feature Engineering

Extension to Higher-Order Regression







- Want to fit a polynomial regression model

LCF

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

$\theta_0 + \theta_1 z_1 + \theta_2 z_2 + \cdots + \theta_d z_d$

- $z = \{1, x, x^2, \ldots, x^d\} \in R^d$ 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

$$\hat{\theta} = argmin_{\theta} L(\theta) = \frac{1}{n}\sum_{i=1}^{N}\left(y^{\{i\}} - z^{\{i\}}\theta\right)^2$$

- 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

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

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

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

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

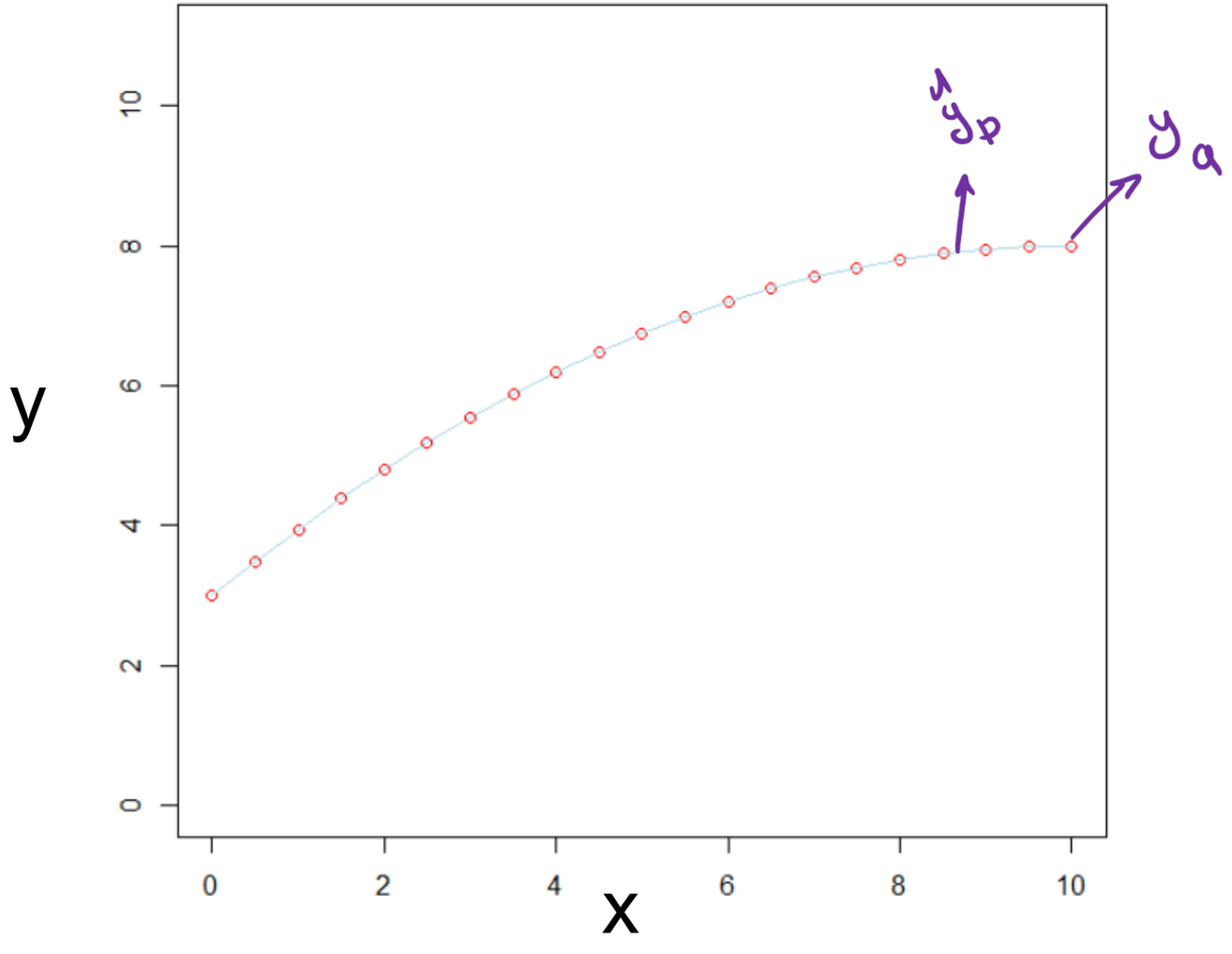
What is happening in polynomial regression?

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

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

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

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



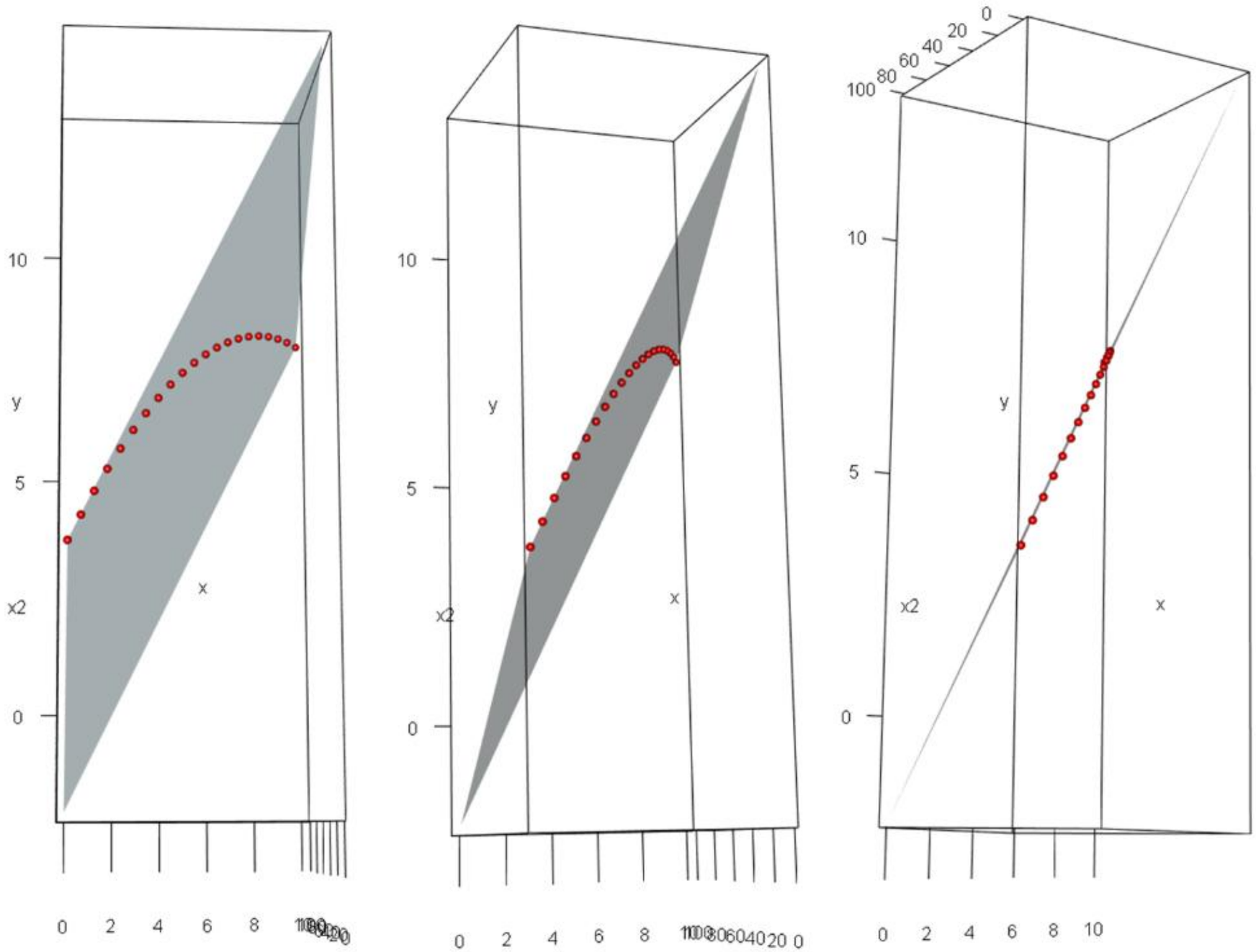
RMSE=0

MSE=0

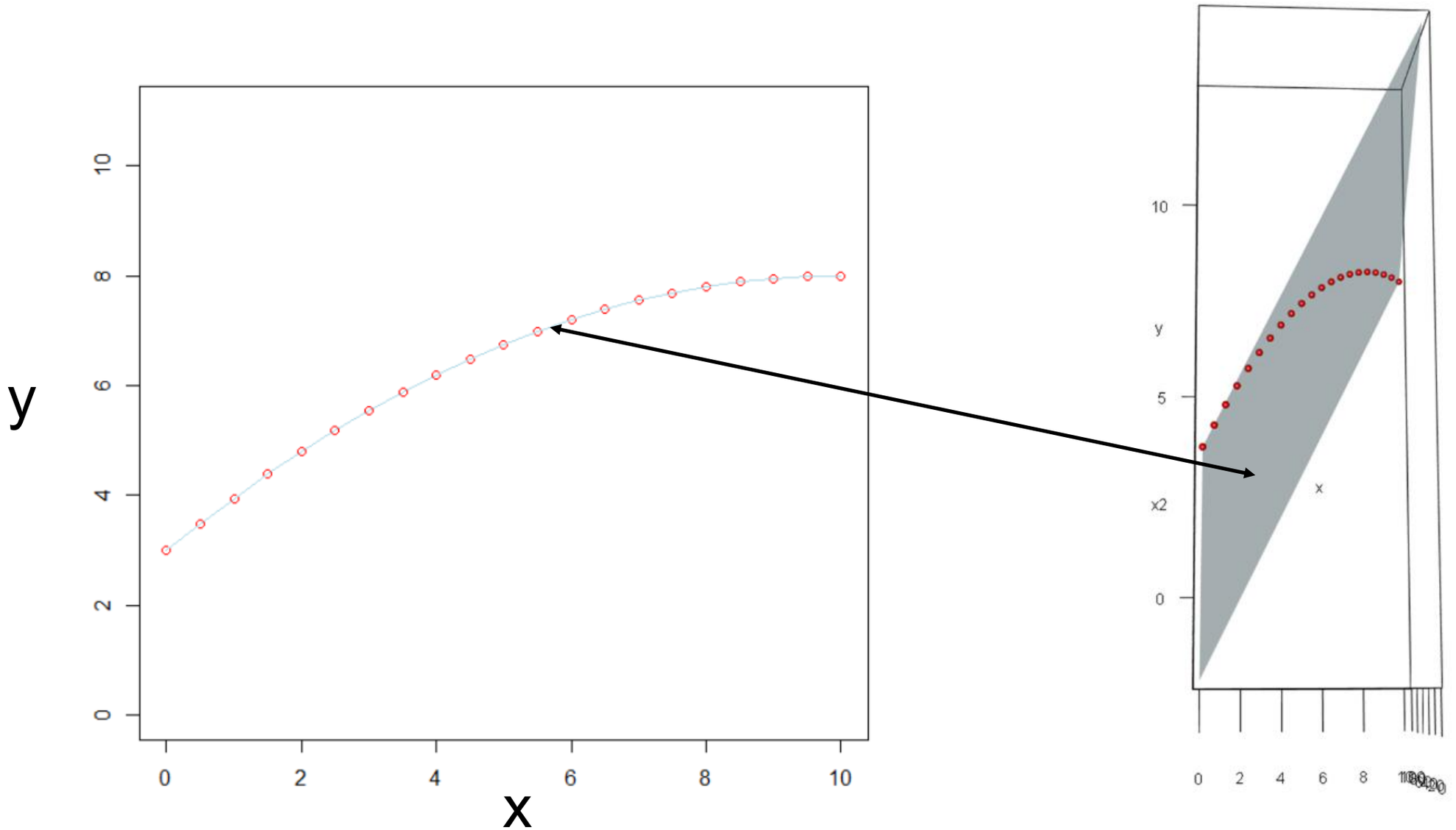
Let's add to the feature space

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

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



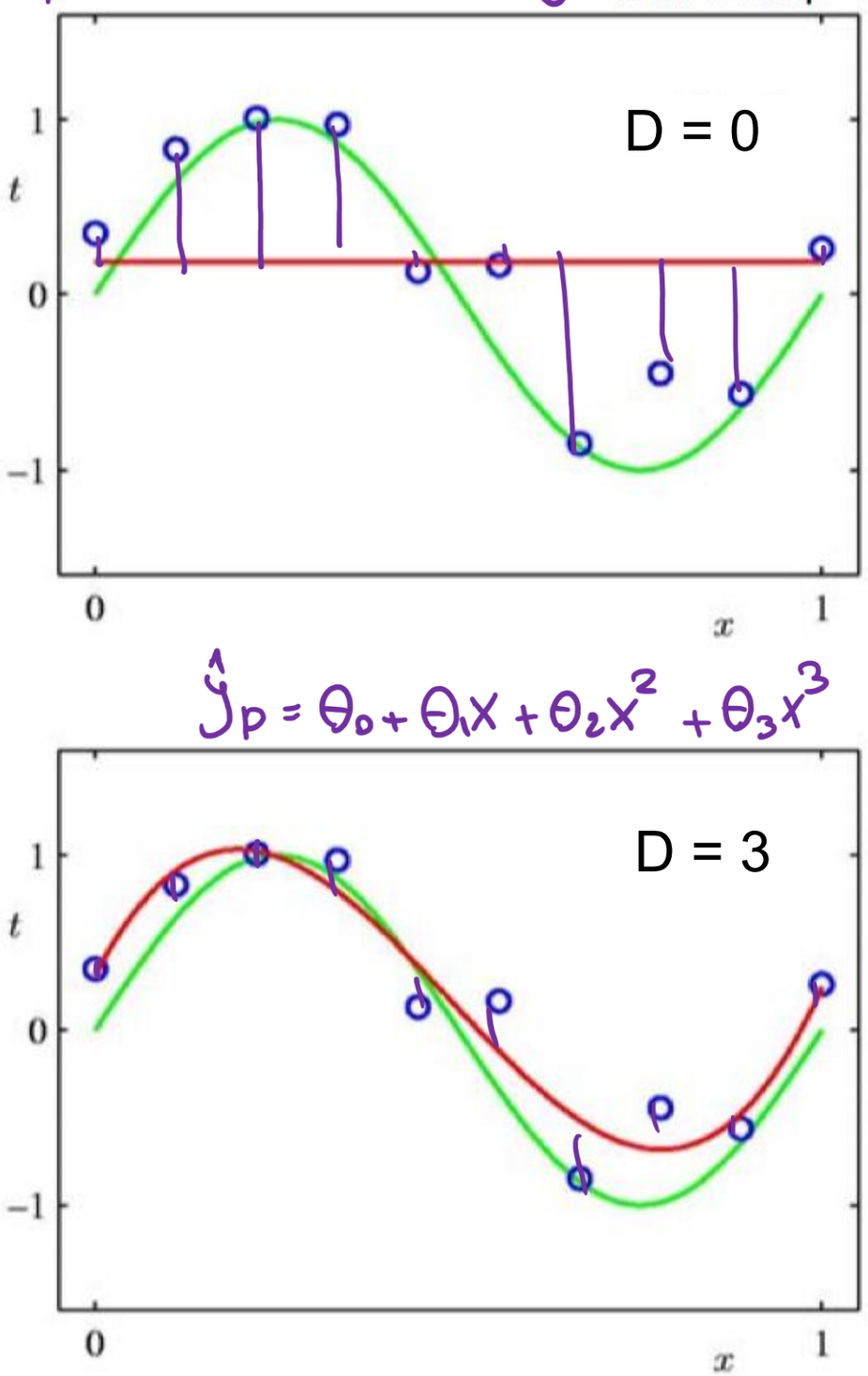
We are fitting a D-dimensional hyperplane in a D+1 dimensional hyperspace (in above example a 2D plane in a 3D space). That hyperplane really is 'flat' / 'linear' in 3D. It can be seen a non-linear regression (a curvy line) in our 2D example in fact it is a flat surface in 3D. So the fact that it is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



Increasing the Maximal Degree

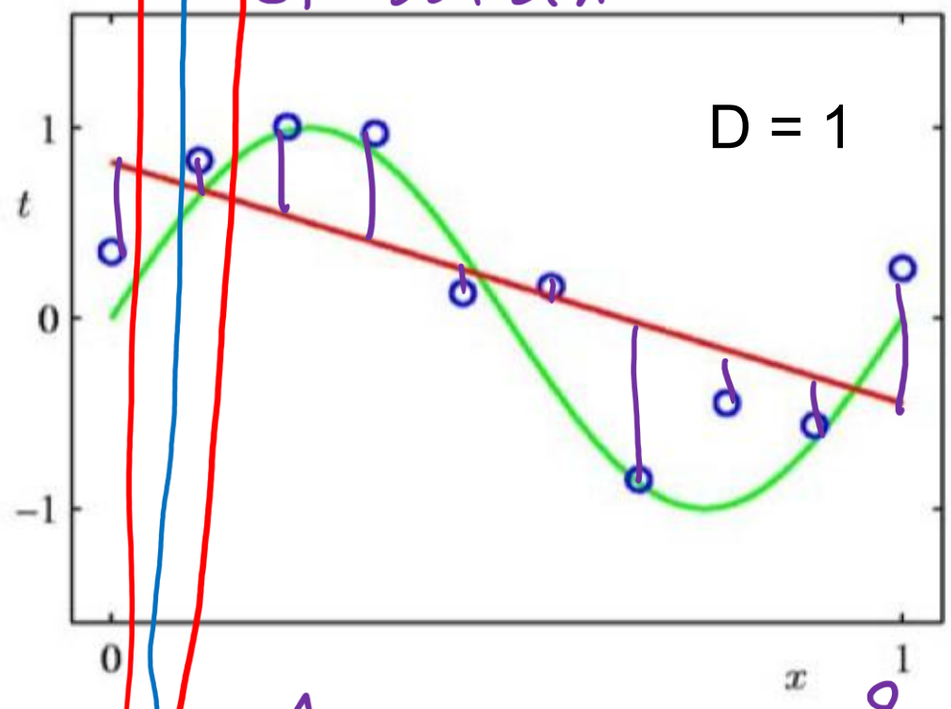
$\hat{y}_p = \theta_0$ underfitting

from Bishop



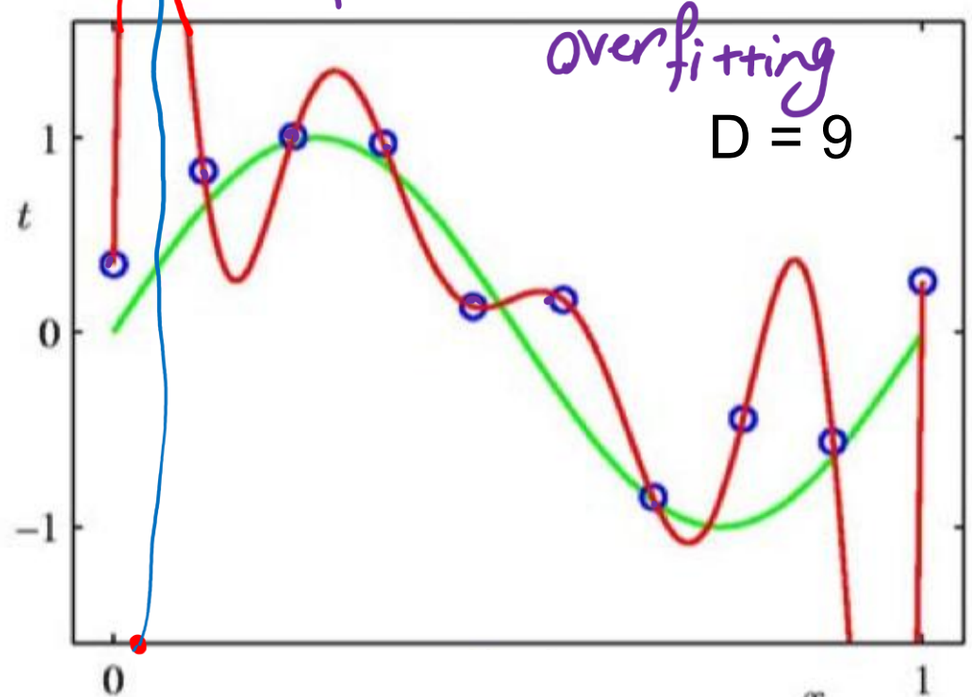
$\hat{y}_p = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$

$\hat{y}_p = \theta_0 + \theta_1 x$



$\hat{y}_p = \theta_0 + \theta_1 x + \dots + \theta_9 x^9$

overfitting

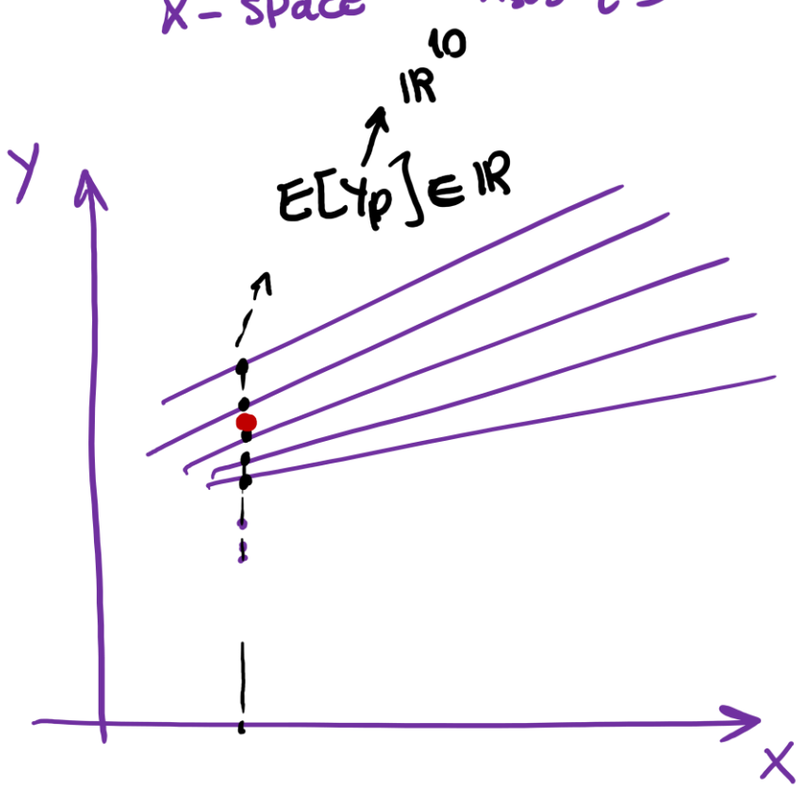


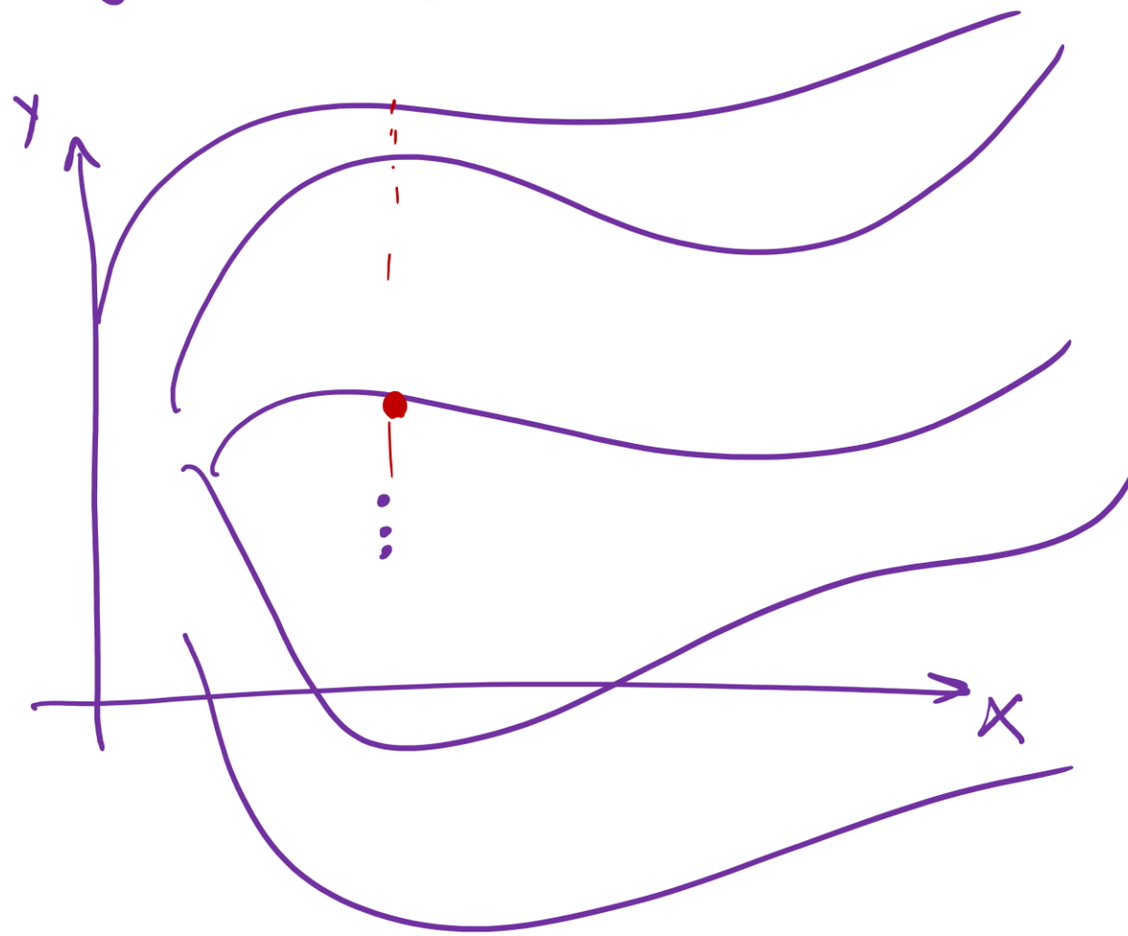
$X = \begin{bmatrix} x \\ \vdots \end{bmatrix}_{1000\times 1} \rightsquigarrow X_{sub_{(1)}} = \begin{bmatrix} \\ \\ \end{bmatrix}_{100\times 1}$

$X_{sub_{(2)}} = \begin{bmatrix} \\ \\ \end{bmatrix}$ ---- $X_{sub_{(10)}} = \begin{bmatrix} \\ \\ \end{bmatrix}$

$Z_{sub} = \begin{bmatrix} x & x^2 & x^3 & - \\ & & & \end{bmatrix}$ Z - space

X - space $X_{sub} = \begin{bmatrix} x \\ \end{bmatrix}$





Bias-Variance Trade off

[Animation]

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

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

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

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

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

$$\left(y_a - E[y_p]\right)^2 + \frac{1}{N}\sum_{i=1}^{N}\left(E[y_p] - y_p\right)^2 + \cdots\cdots\left(E[E[y_p]] - E[y_p]\right)$$

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

Bias-Variance Trade off

Animation

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

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

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

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

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

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

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

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

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

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

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

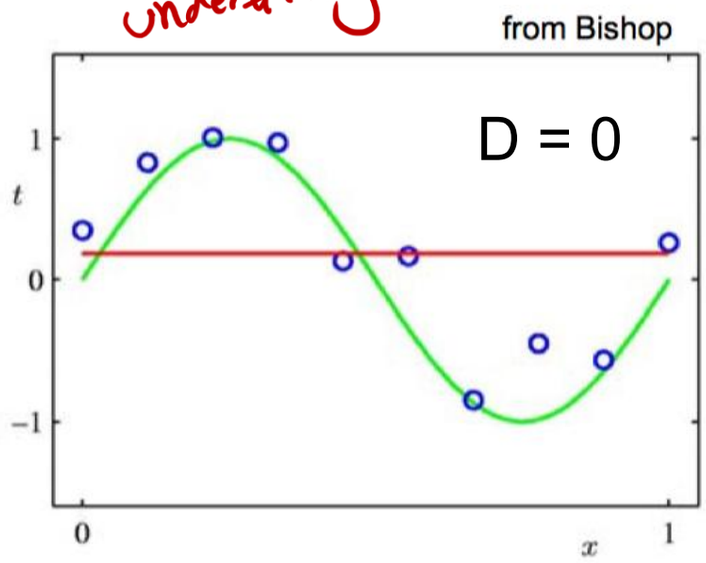
$$= 2\left(y_a - E\left[y_p\right]\right)\left(E\left[E\left[y_p\right]\right] - E\left[y_p\right]\right)$$

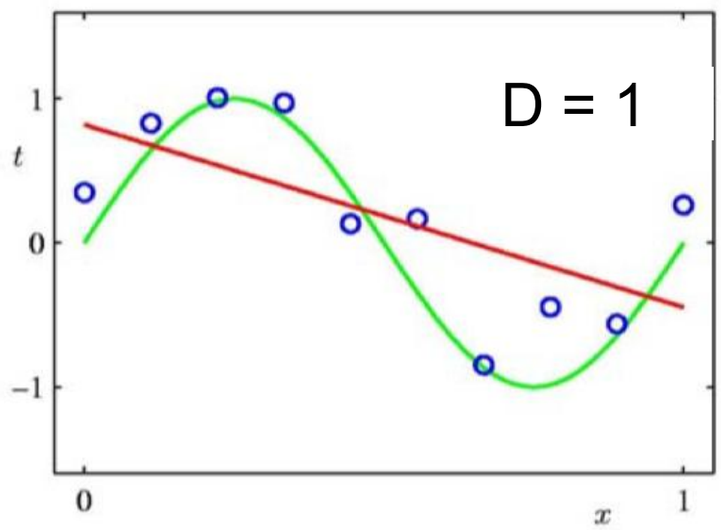
$$= 2\left(y_a - E\left[y_p\right]\right)\left(E\left[y_p\right] - E\left[y_p\right]\right) = 0$$

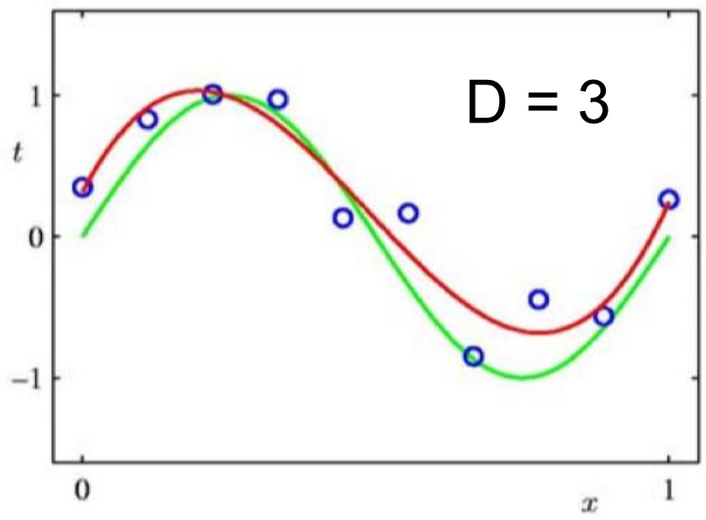
Which One is Better?

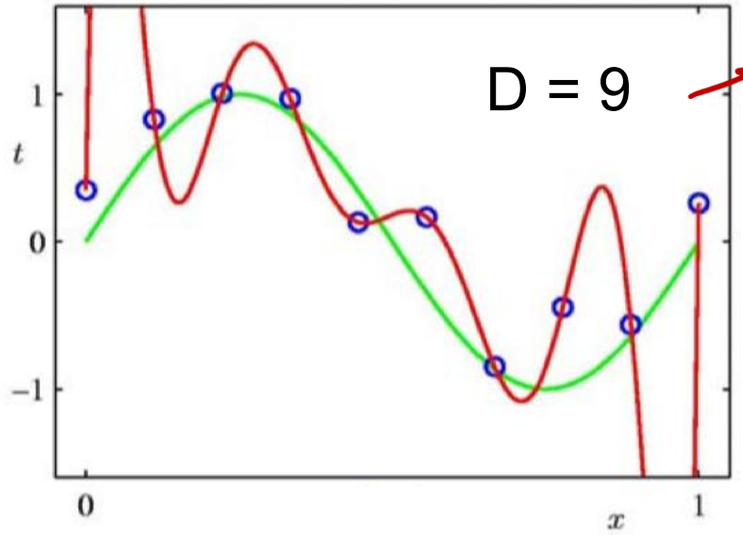
underfitting

from Bishop









overfitting

- 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