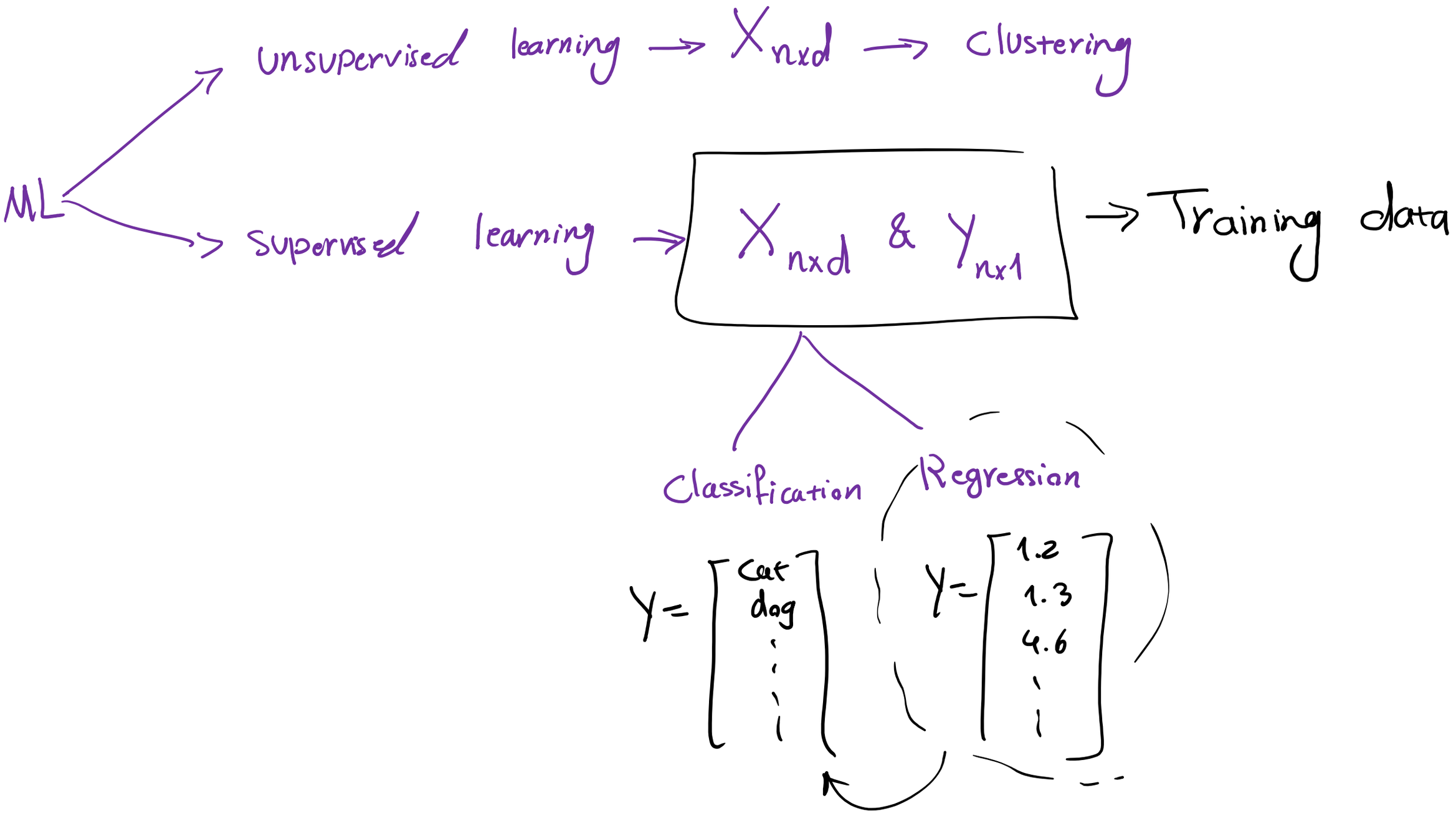

ML
Unsupervised learning
X_{nxd}
Clustering
Supervised learning
X_{nxd} & Y_{nx1}
Training data
Classification
Regression
Y=
cat
dog
Y=
1.2
1.3
4.6

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

Training data X & y ⇒ 80% → Training data 20% → Testing data

Supervised Learning: Overview

70% training data 10% validation 20% testing

Functions $\mathcal{F}$

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

Training data

$$\{(x^{\{i\}}, y^{\{i\}}) \in \underbrace{\mathcal{X}}_{n \times d} \times \underbrace{\mathcal{Y}}_{n \times 1}\}$$

X =

y=

LEARNING

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

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

actual

$\hat{y}_p$

Learning machine

PREDICTION

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

New data

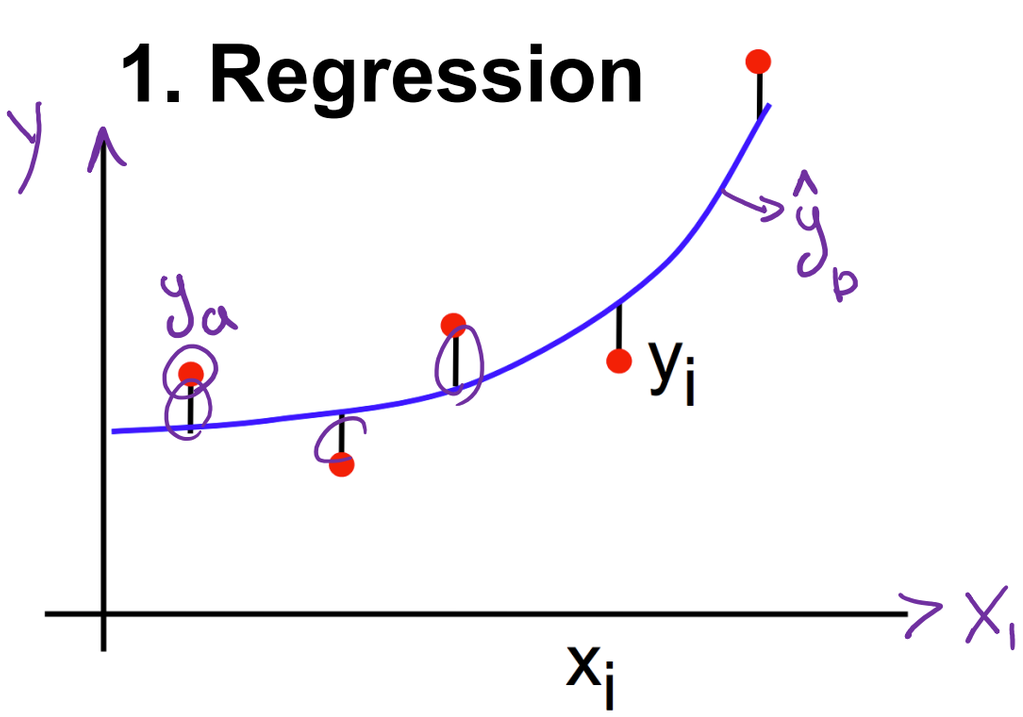
x

Supervised Learning: Two Types of Tasks

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

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

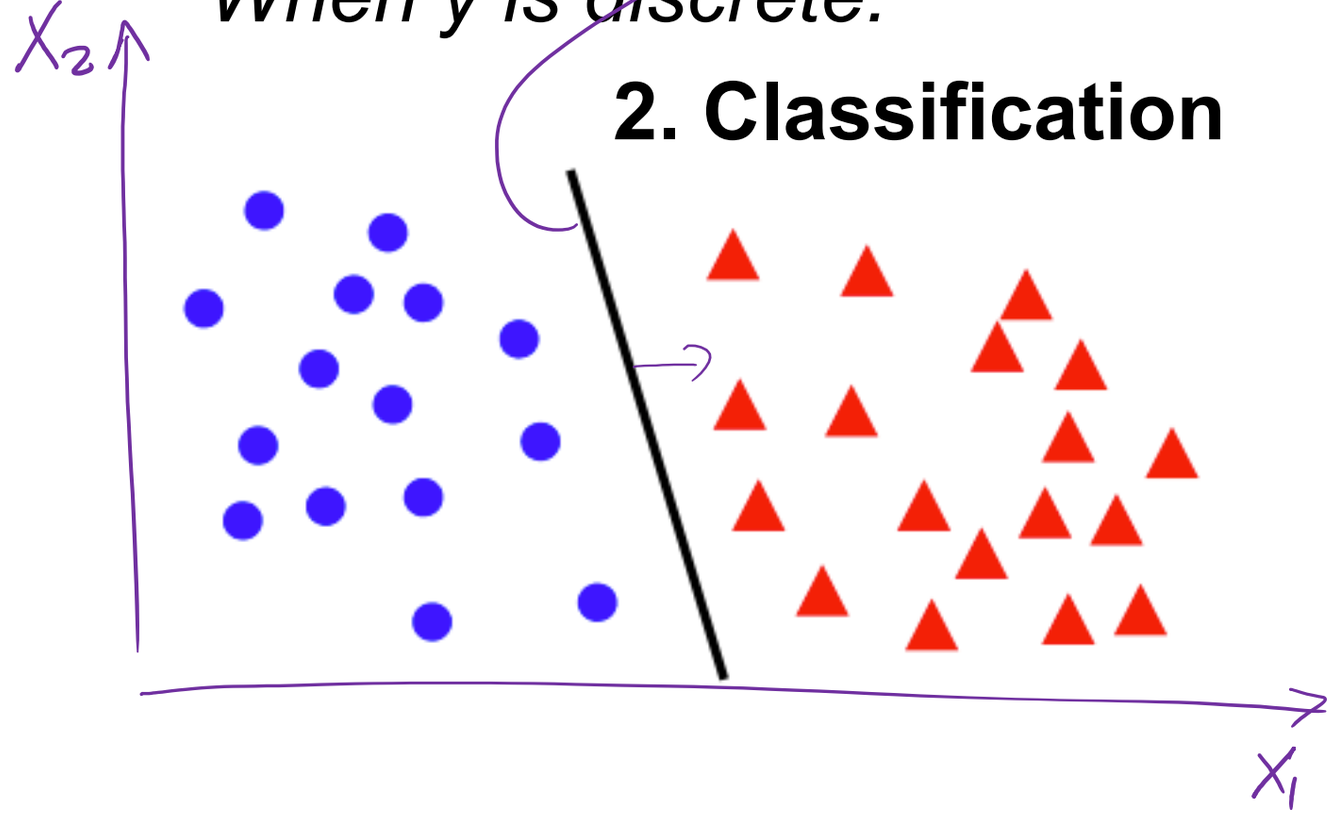
When y is continuous:



Curve fitting

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

When y is discrete:

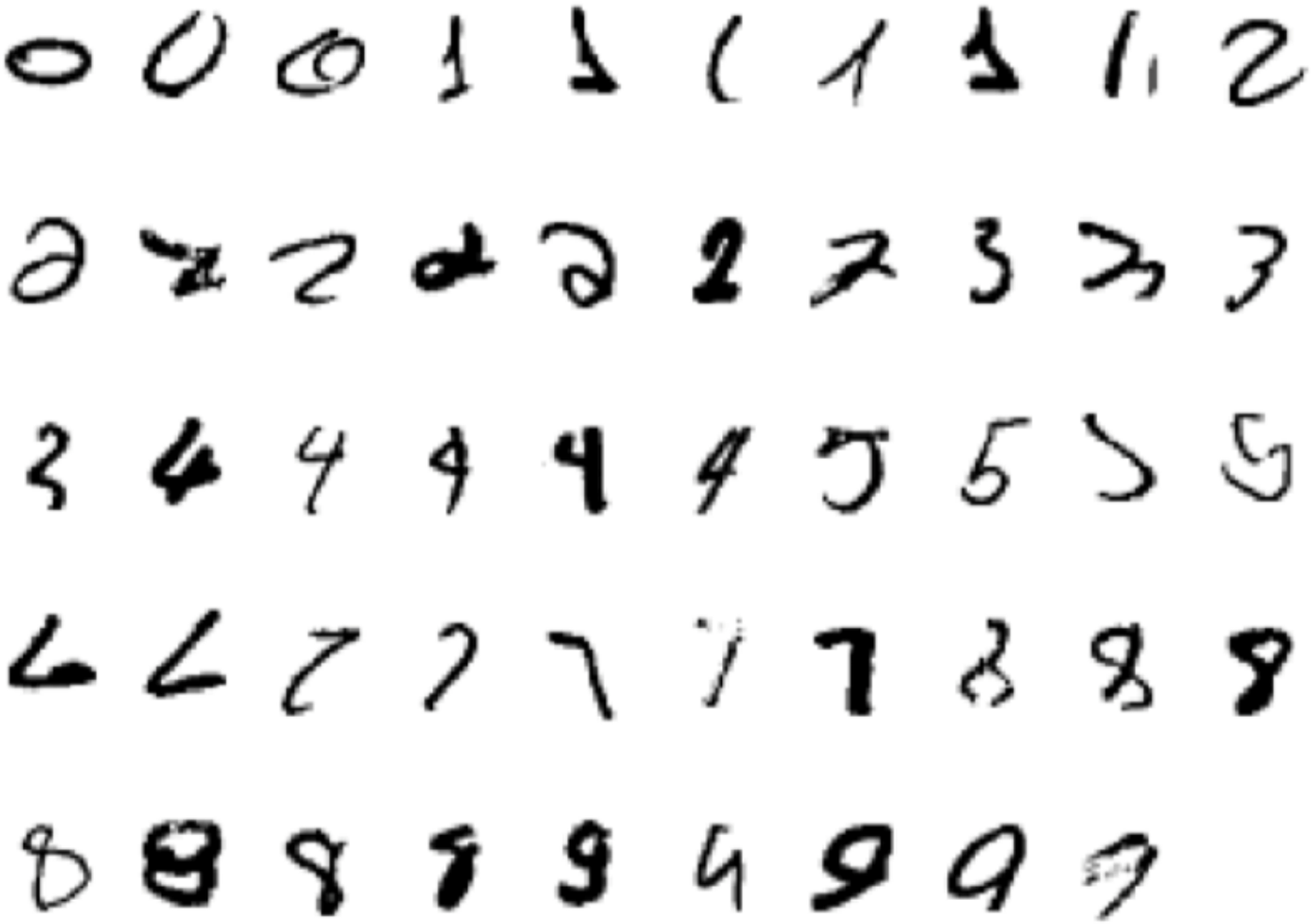


Class estimation

Classification Example 1: Handwritten digit recognition

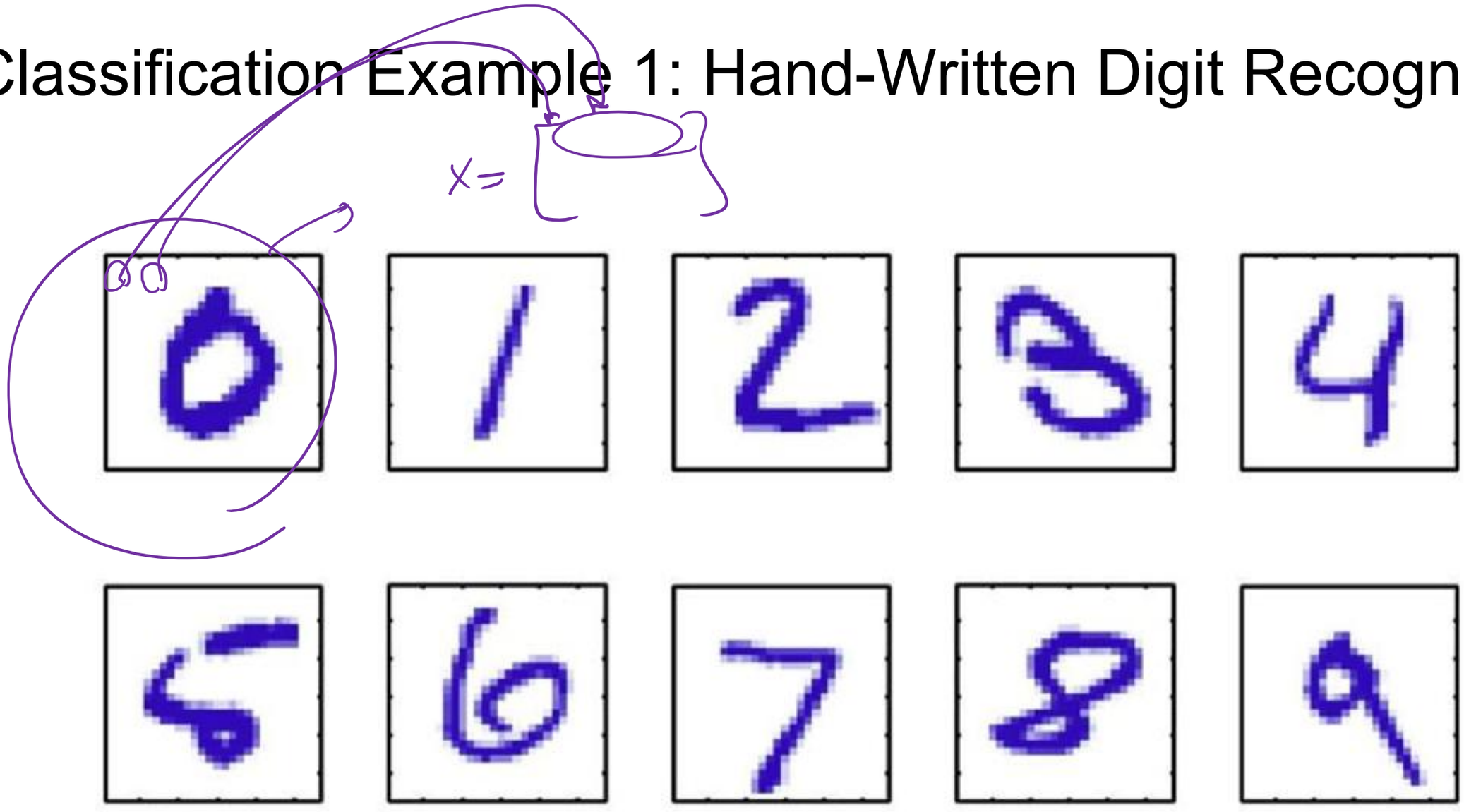
As a supervised classification problem

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



Images are 28 x 28 pixels

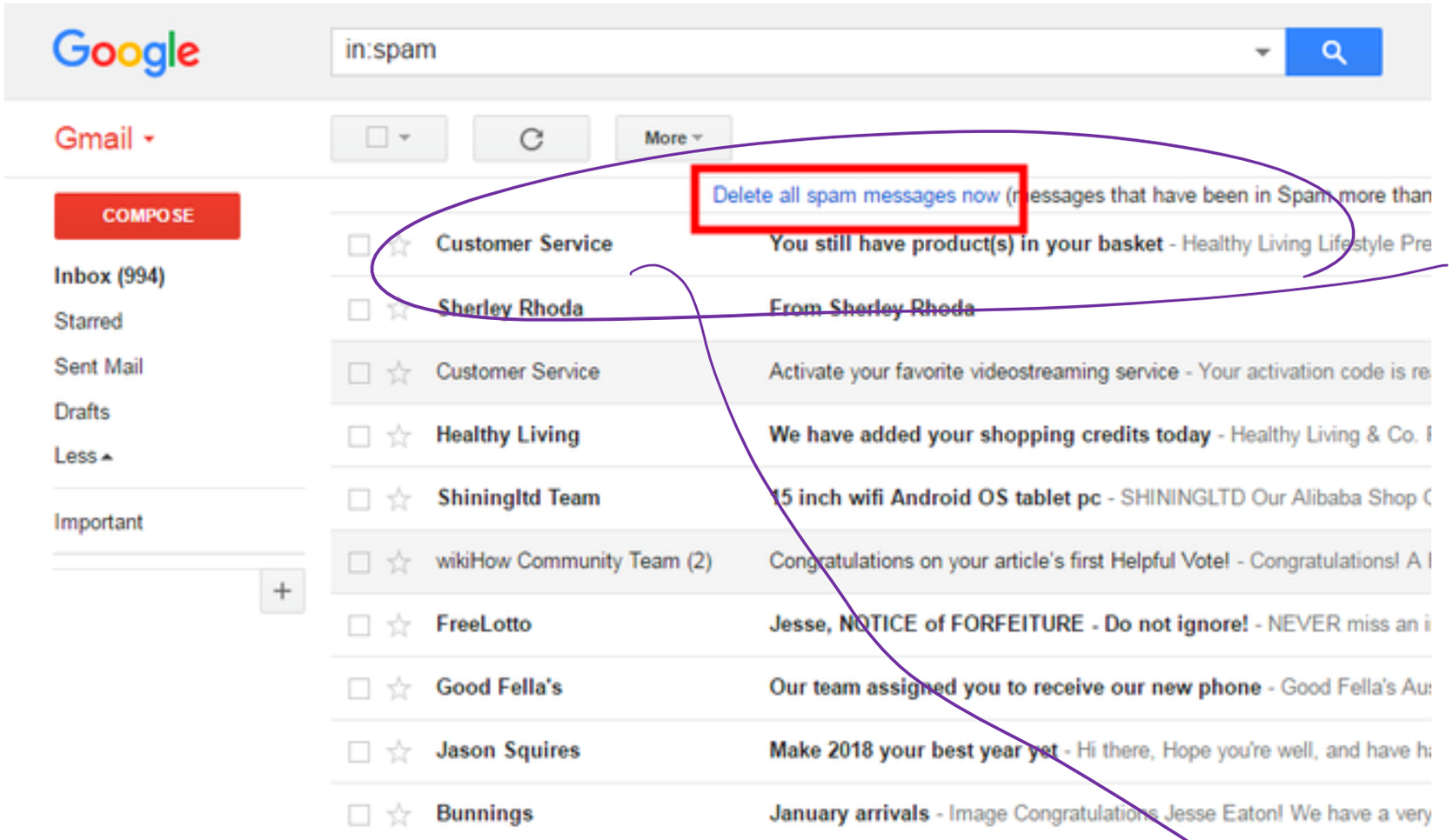
A classification problem

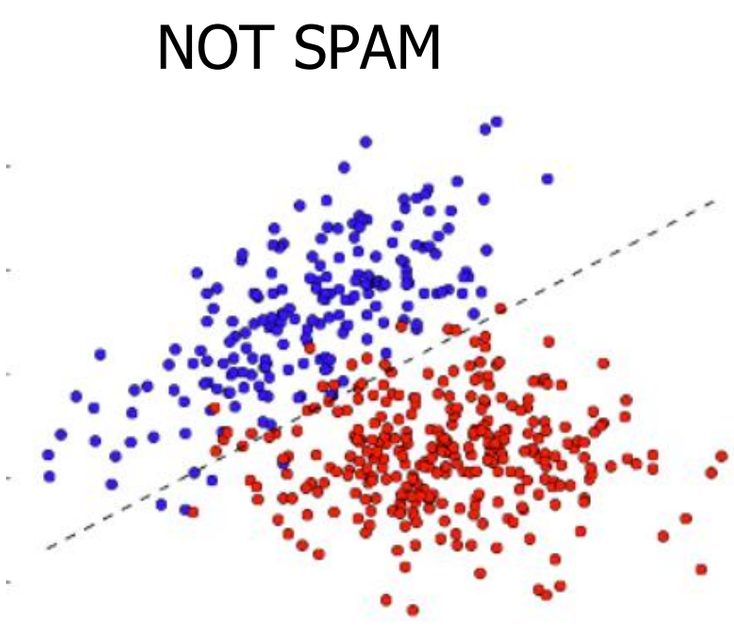
Represent input image as a vector $\mathbf{x} \in \mathbb{R}^{784}$

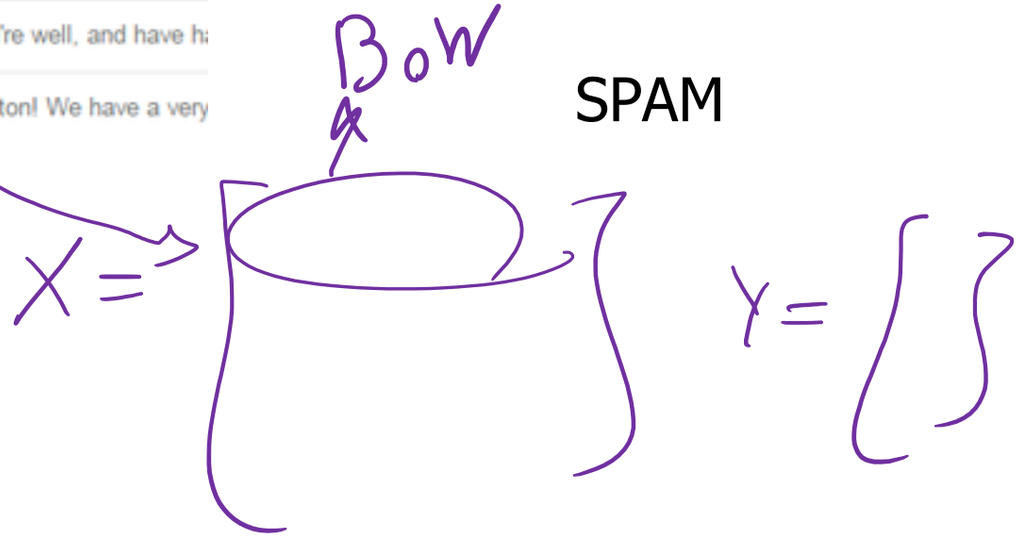
Learn a classifier $f(\mathbf{x})$ such that,

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

Classification Example 2: Spam Detection







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

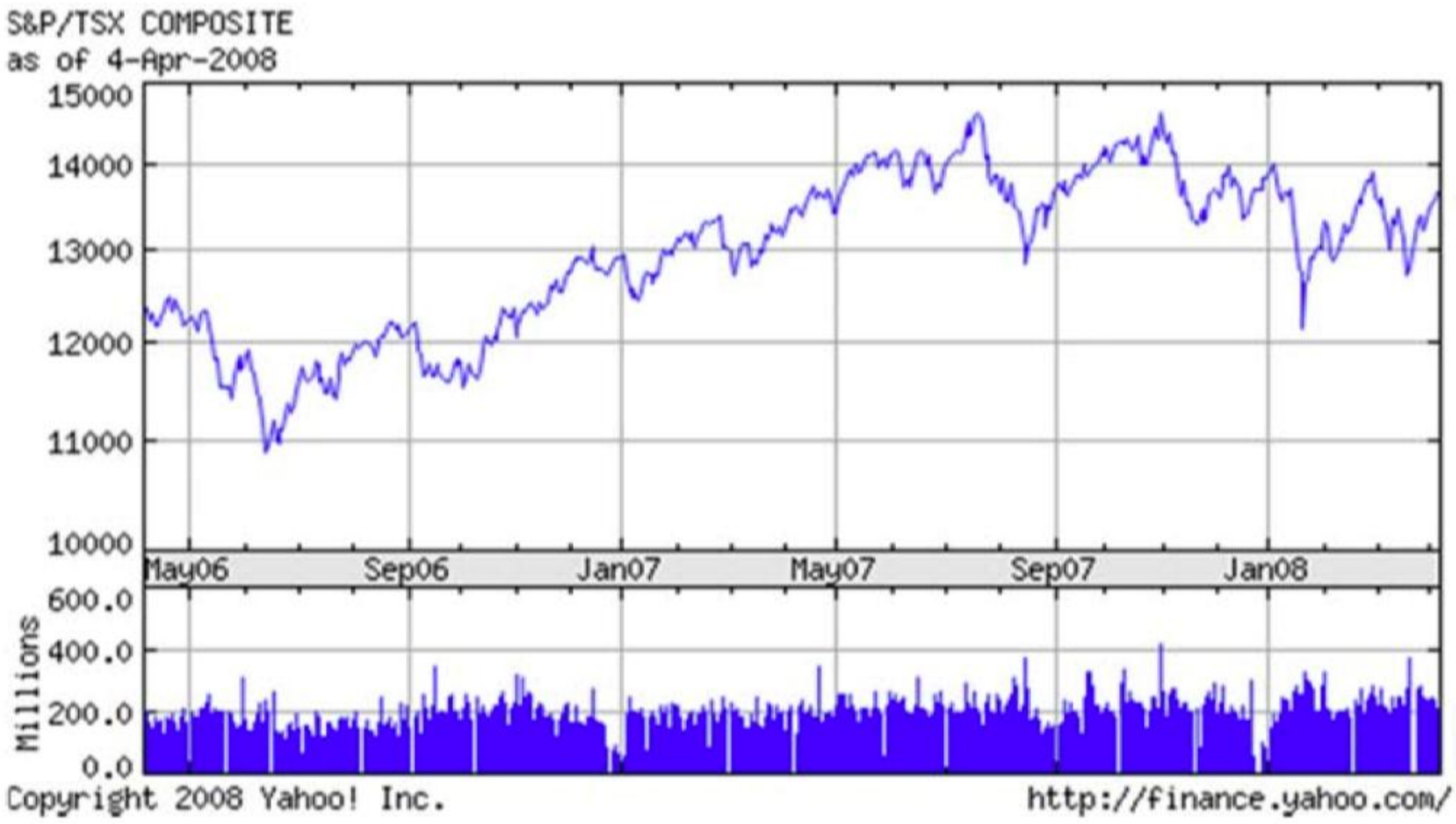
- Suppose you are to move to Atlanta
- And you want to find the **most reasonably priced** apartment satisfying your **needs:**

  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	?

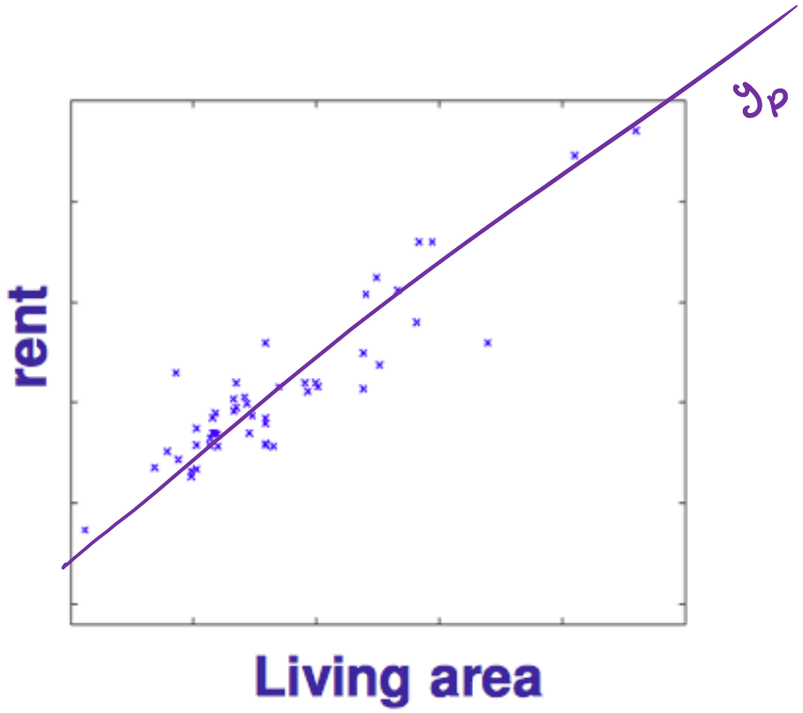
Regression Example 2: Stock Price Prediction



- Task is to predict stock price at future date

A regression problem

$f(x^{[i]}) = m x^{[i]} + b$



$f(x^{[i]}) = \underbrace{\theta_0}_{\text{bias term}} + \underbrace{\theta_1}_{\text{parameter}} x_1$ → the parameter that defines the behavior of feature x_1

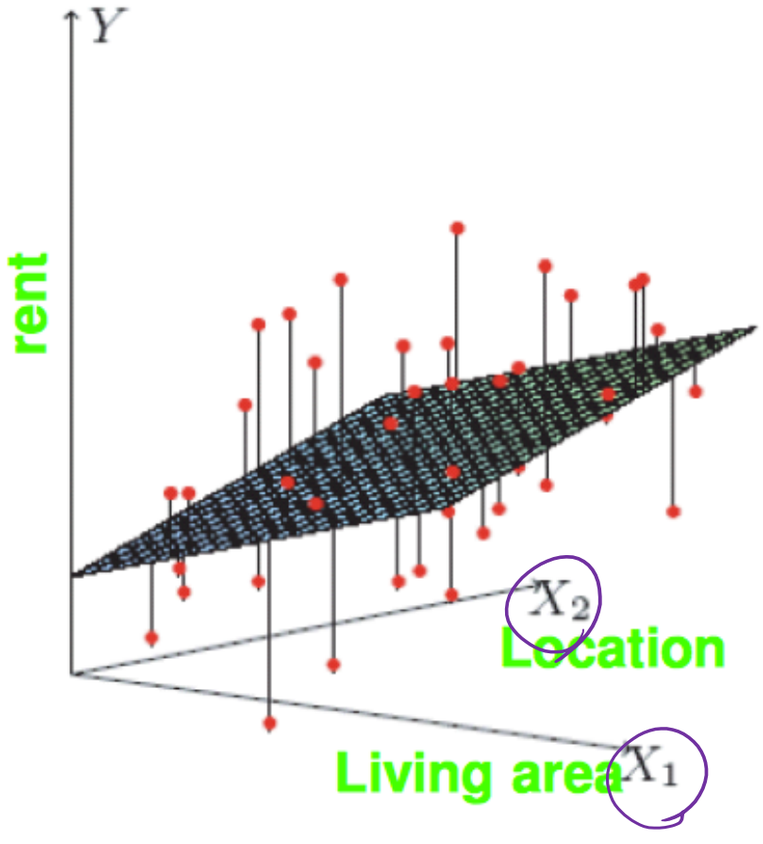
- **Features:**
  - Living area, distance to campus, # bedroom …
  - Denote as $x = (x_1, x_2, \ldots, x_d)$

$x^{[i]} = [1, x_1^{[i]}, x_2^{[i]}, \ldots, x_d^{[i]}] \cdot \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}_{d+1 \times 1}$

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

$f(x^{[i]}) = x^{[i]} \cdot \theta$

$f(x) = \hat{y}_p = x \cdot \theta$ → LCF



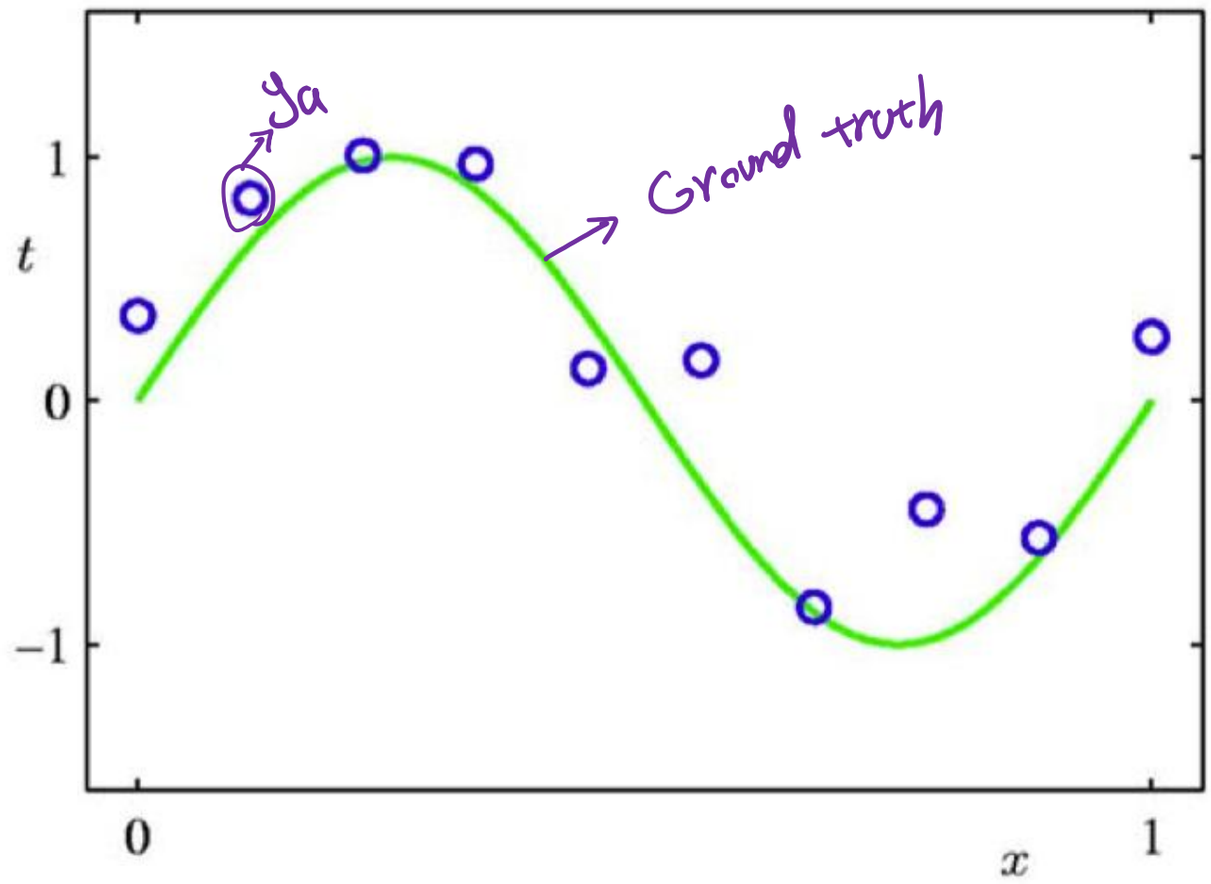
- **Training set:**

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

$f(x^{[i]}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

LCF → linear combination of feature

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

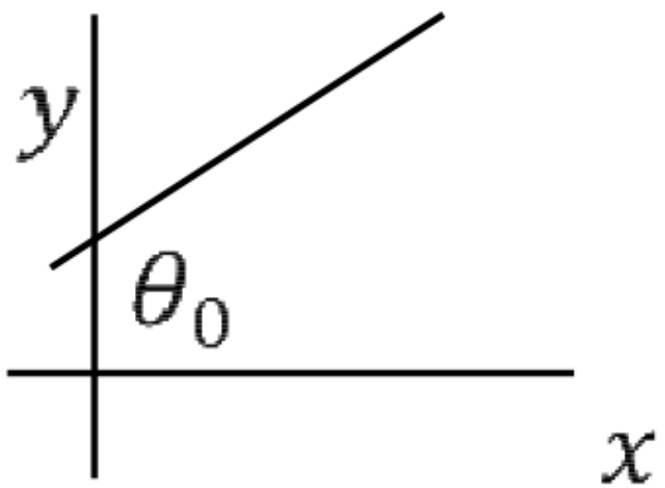
$$y = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d + \epsilon$$

- 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

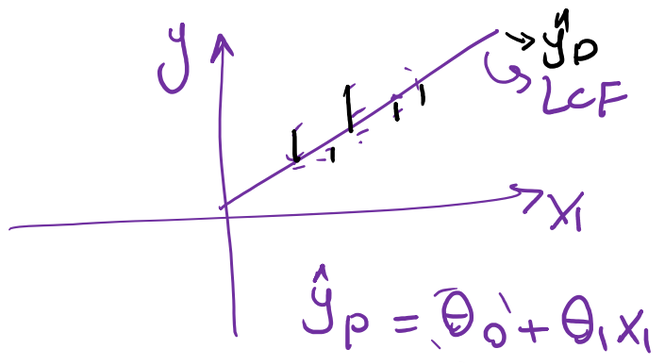
$f(x) = \hat{y}_p = x\theta \rightarrow$ LCF

- Then $\hat{y}_p = x\theta + \epsilon$



LCF $\quad f(x^{\{i\}}) = \hat{y}_p = \underset{\sim}{\theta_0} + \underset{\sim}{\theta_1} x_1^{\{i\}} + \cdots + \theta_d x_d^{\{i\}} \Rightarrow \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \\ \\ \theta_d \end{bmatrix}_{d+1 * 1}$

$f(x) = \hat{y}_p = X.\theta \rightsquigarrow$ LCF



Objective function $\quad L(\theta) = E(\theta) = \frac{1}{N} \left\{ \sum_{i=1}^{N} \left(\underset{\sim}{y_a^{\{i\}}} - \underset{\sim}{\hat{y}_p^{\{i\}}} \right)^2 \right.$

$\underset{\sim}{\hat{y}_p^{\{i\}}} = X^{\{i\}} . \theta$

$$\frac{\partial L(\theta)}{\partial \theta} = 0$$

Least Mean Square Method

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

Training

$$\hat{\theta}_{1x1} = argmin_{\theta} L(\theta) = \frac{1}{n}\sum_{i=1}^{N}\left(y^{\{i\}} - x^{\{i\}}\theta\right)^2$$

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

$$\frac{\partial L(\theta)}{\partial \theta} = 0$$

GD

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

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

$\hat{y}_p = x^{\{i\}} \cdot \theta$

Matrix form

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

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

$\theta \in \mathbb{R}$

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

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

$\hat{y}_p^{\{n\}}$

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

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

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

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

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

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

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

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

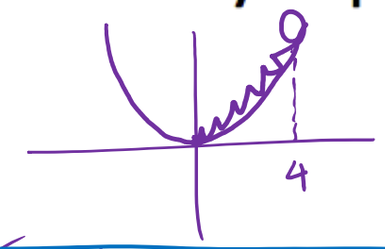
Not a big matrix because $n \gg d$This matrix is invertible most of the times. If we are VERY unlucky and columns of $\mathbf{X^T X}$ are not linearly independent (it's not a full rank matrix), then it is not invertible.

$f(x) = x^2$ $f(x) = 10x^2$ $f(x) = -x^2$

Alternative Way to Optimize

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

$f(x) = x^2$



$$\frac{\partial f(x)}{\partial x} = 2x$$

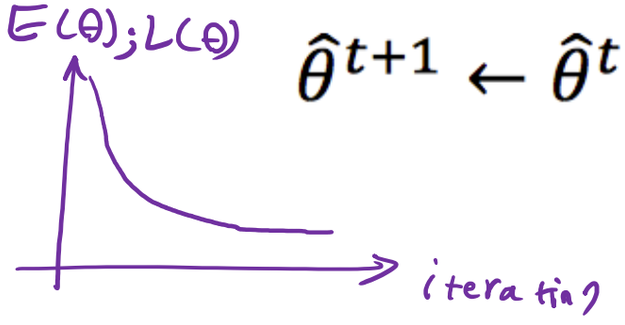
$$\frac{\partial L(\theta)}{\partial \theta} = -\frac{2}{n}\sum_{i=1}^{N}(x^{\{i\}})^T(y^{\{i\}} - x^{\{i\}}\theta)$$

$$x^{\{t+1\}} \leftarrow x^{\{t\}} - \alpha \frac{\partial f(x)}{\partial x} = (x^{\{t\}} - \alpha 2x)$$

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

- Gradient descent

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

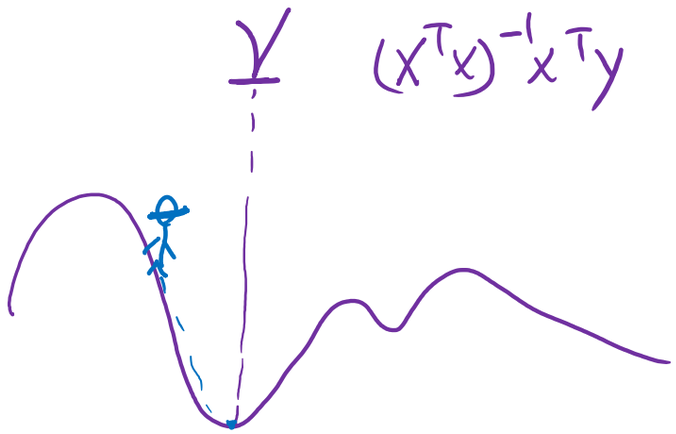


$$\hat{\theta}^{t+1} \leftarrow \hat{\theta}^t + \frac{\alpha}{n}\sum_{i=1}^{N}(x^{\{i\}})^T(y^{\{i\}} - x^{\{i\}}\theta)$$ "FGD"

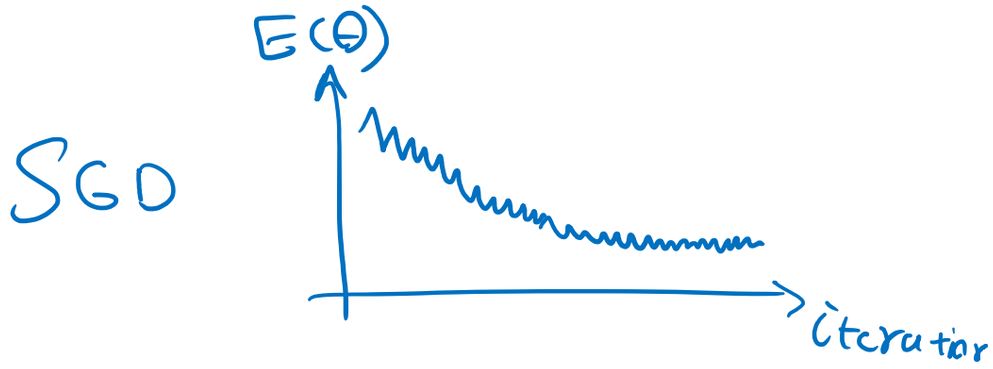
- Stochastic gradient descent (use one data point at a time)

SGD

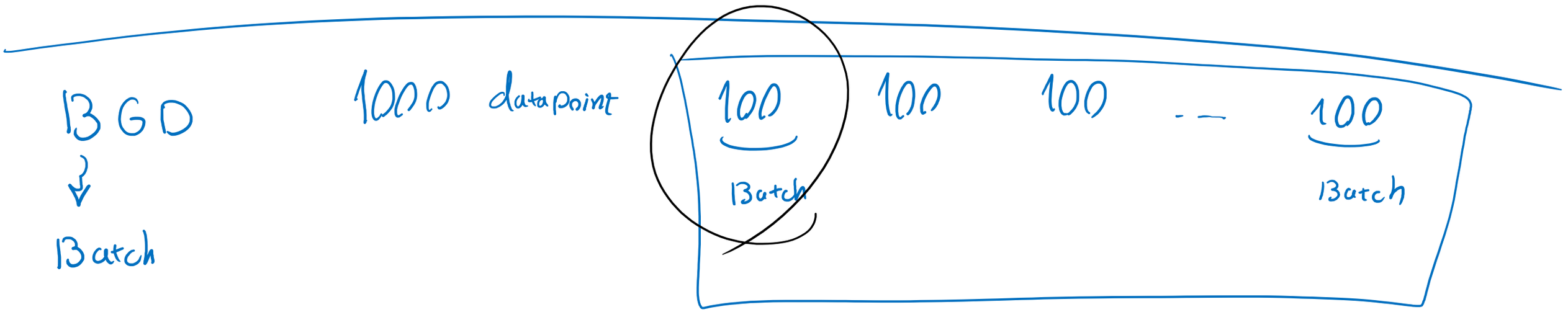
$$\hat{\theta}^{t+1} \leftarrow \hat{\theta}^t + \beta_t \times (x^{\{i\}})^T(y^{\{i\}} - x^{\{i\}}\theta)$$



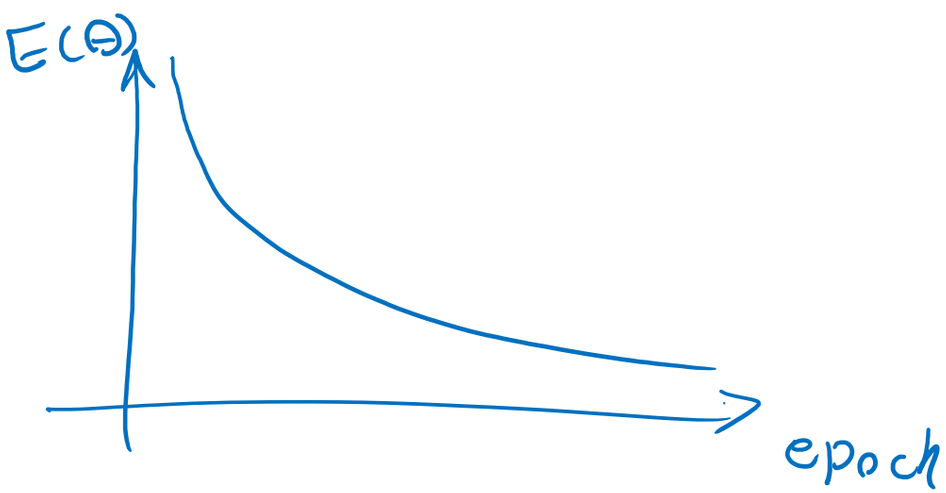
FGD



SGD



Batch



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

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

BGD

- Gradient descent

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

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 ..)

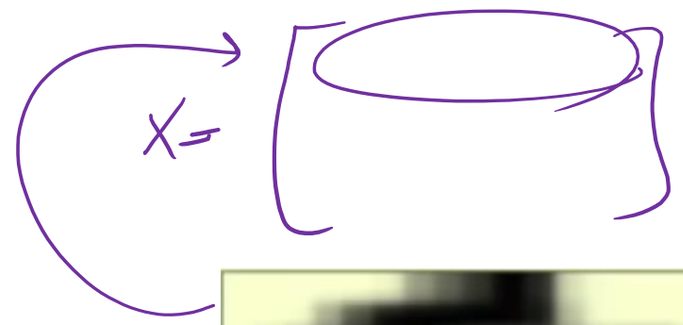
Linear regression for classification

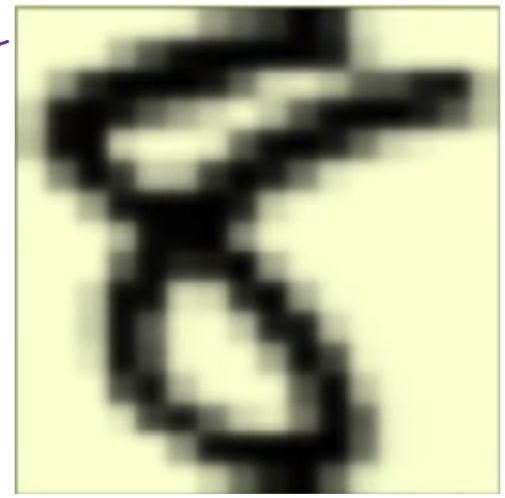
Raw Input $x = (1, x_1, \dots, x_{256})$

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

feature engineering

Extract useful information





16

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

$\theta_0\ \theta_1\ \theta_2$

Sum up all the pixels = intensity

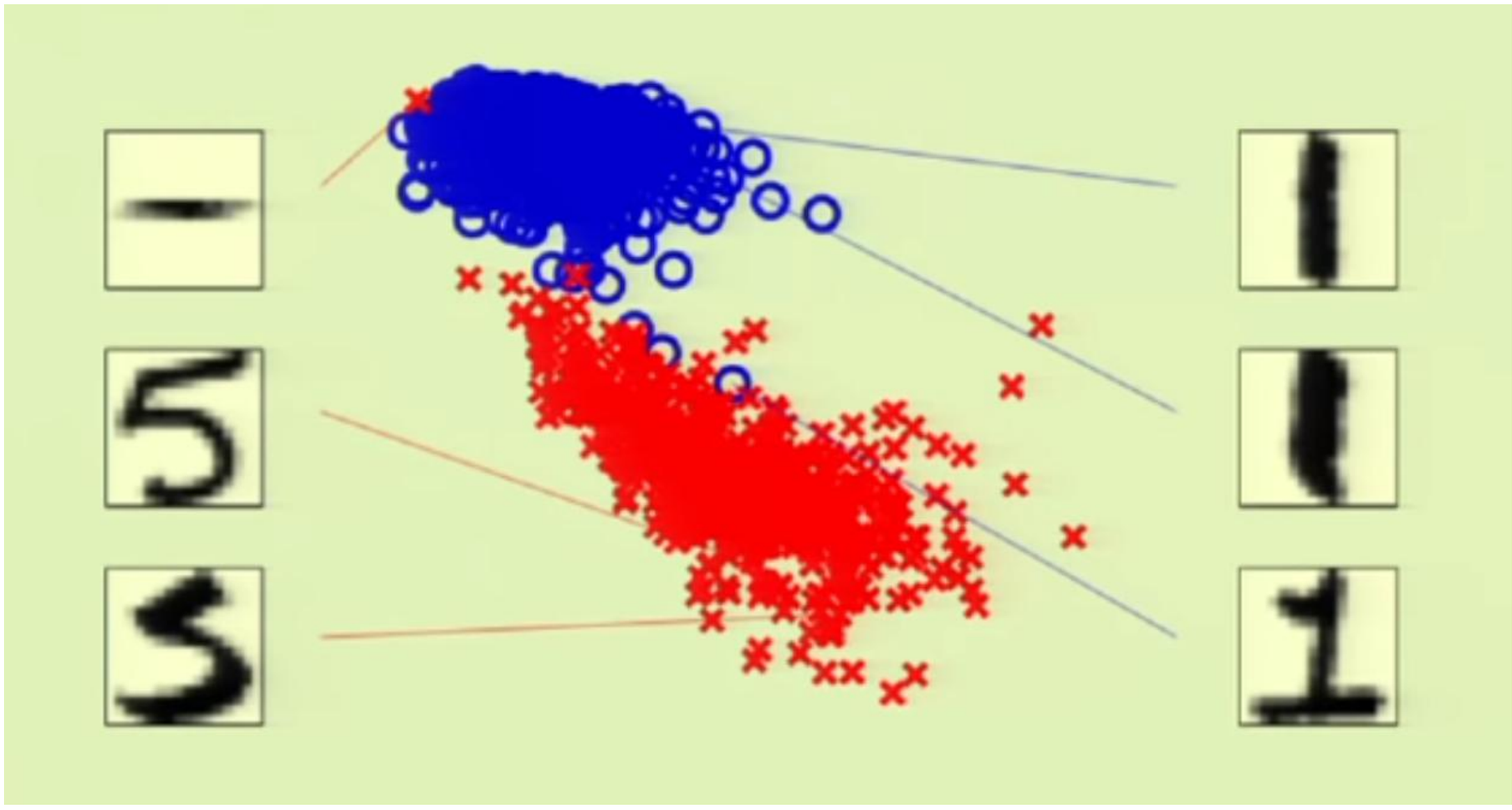
Symmetry = -(difference between flip version)

16

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

$$x_1 = intensity\ x_2 = symmetry$$

It is almost linearly separable



symmetry

intensity

Linear regression for classification

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

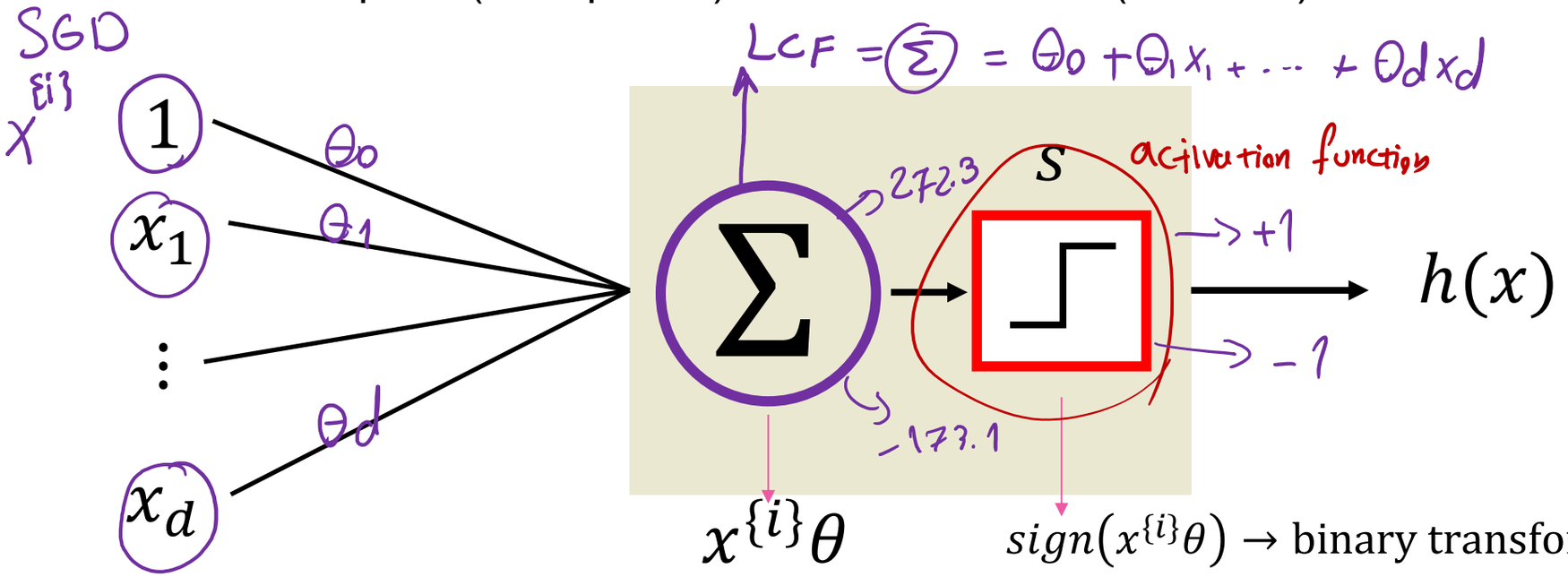
Use linear regression $x^{\{i\}}\theta \approx y^{\{i\}} = \pm 1$ $i =$ index of a data-point

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

decision line

a learning block

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



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

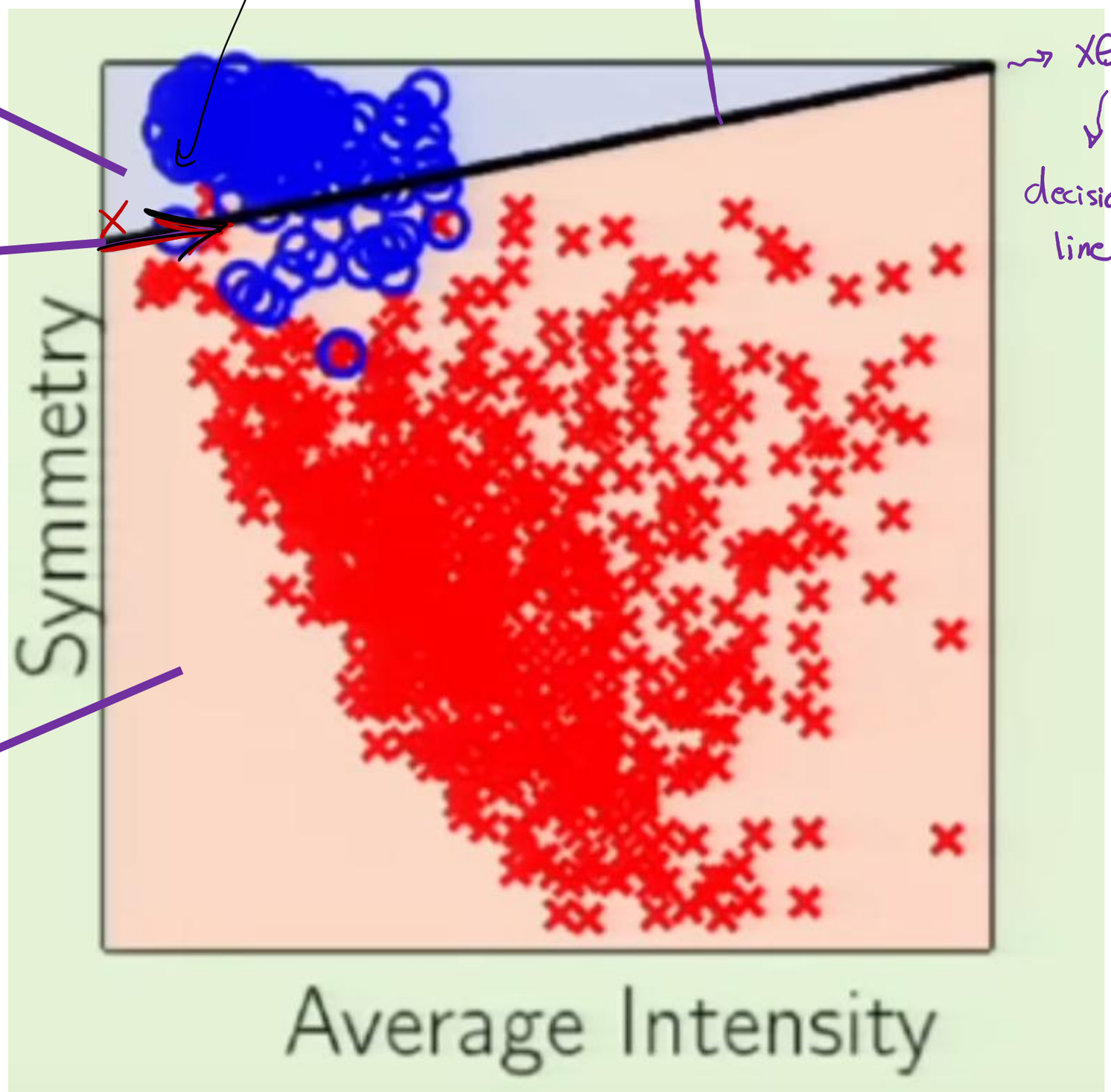
$(d+1 \times 1)$

+1

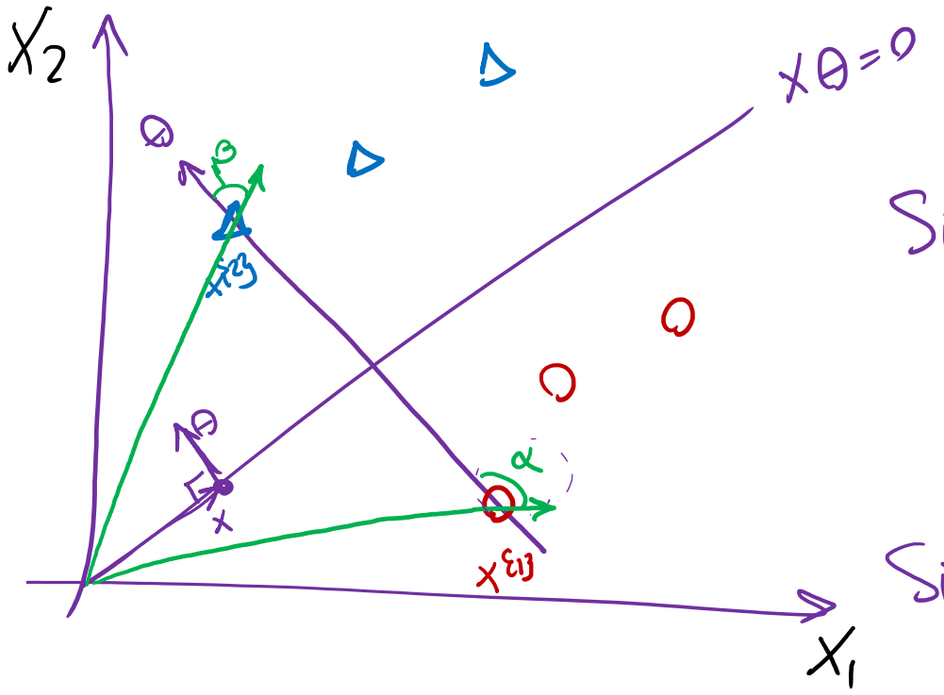
0

−1

$X\Theta = 0$



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



$LCF = x\theta = 0$

$$Sign\Big(LCF(x^{\{13\}}) = x^{\{1\}} \cdot \theta = |x^{\{13\}}|\,|\theta|\cos\alpha\Big) = -1$$

(+ + −)

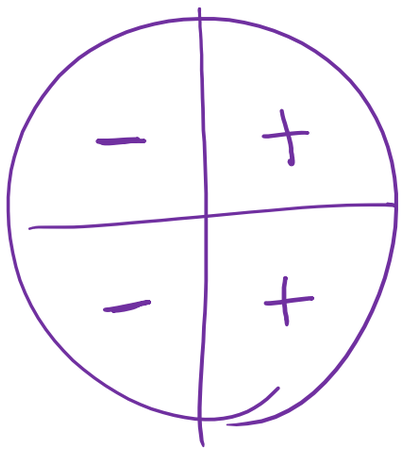
5

$$Sign\big(LCF(x^{\{23\}}) = x^{\{2\}} \cdot \theta = |x^{\{23\}}|\,|\theta|\cos\beta\big) = +1$$

(+ + +)

1

Cos



$LCF = f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

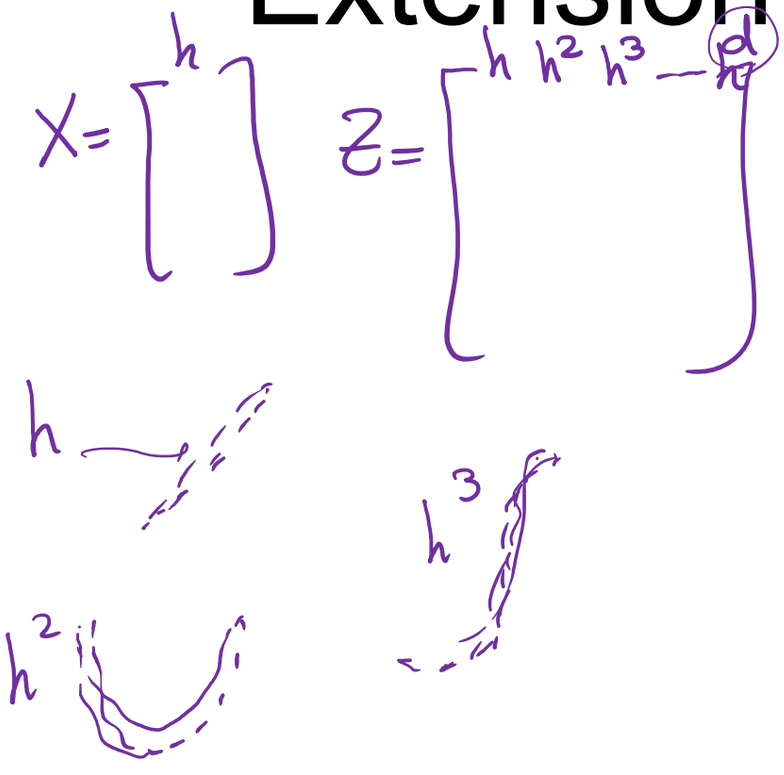
Plane

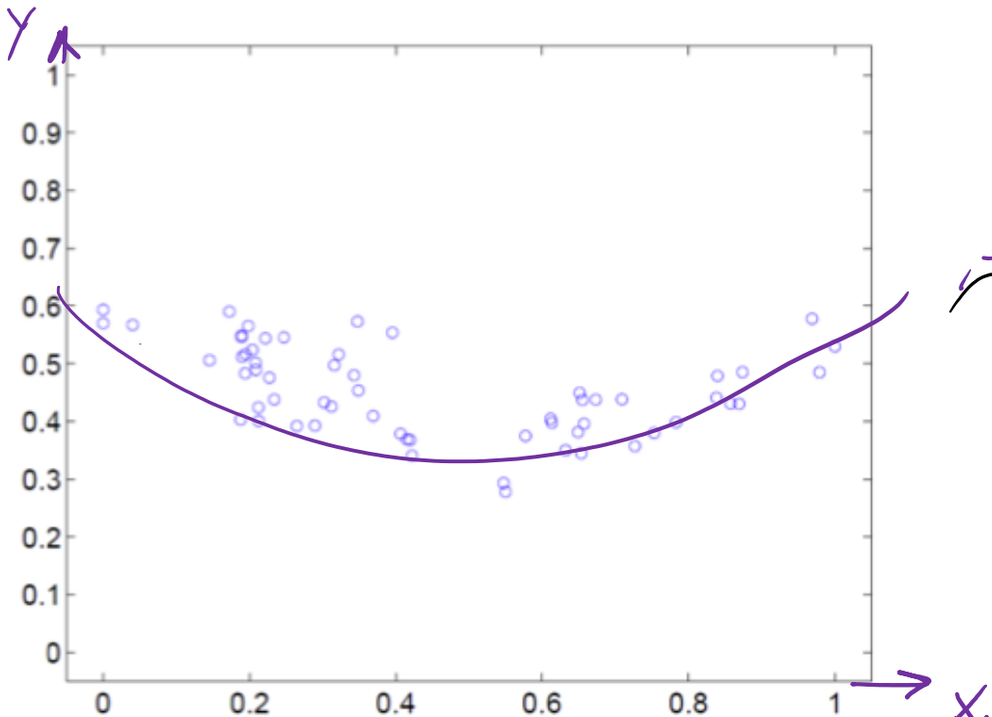
https://www.math3d.org/UPNV1IQvxt

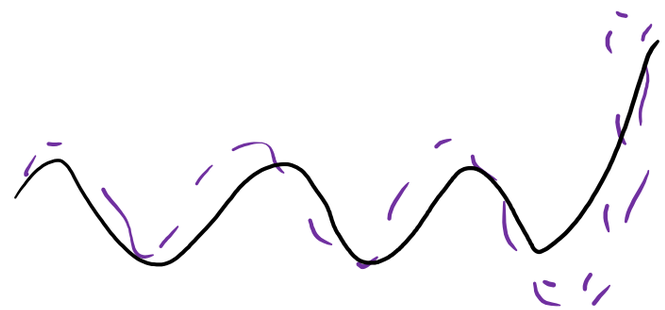
Outline

- Supervised Learning
- Linear Regression
- Extension

Extension to Higher-Order Regression







- Want to fit a polynomial regression model

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

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

LCF

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

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

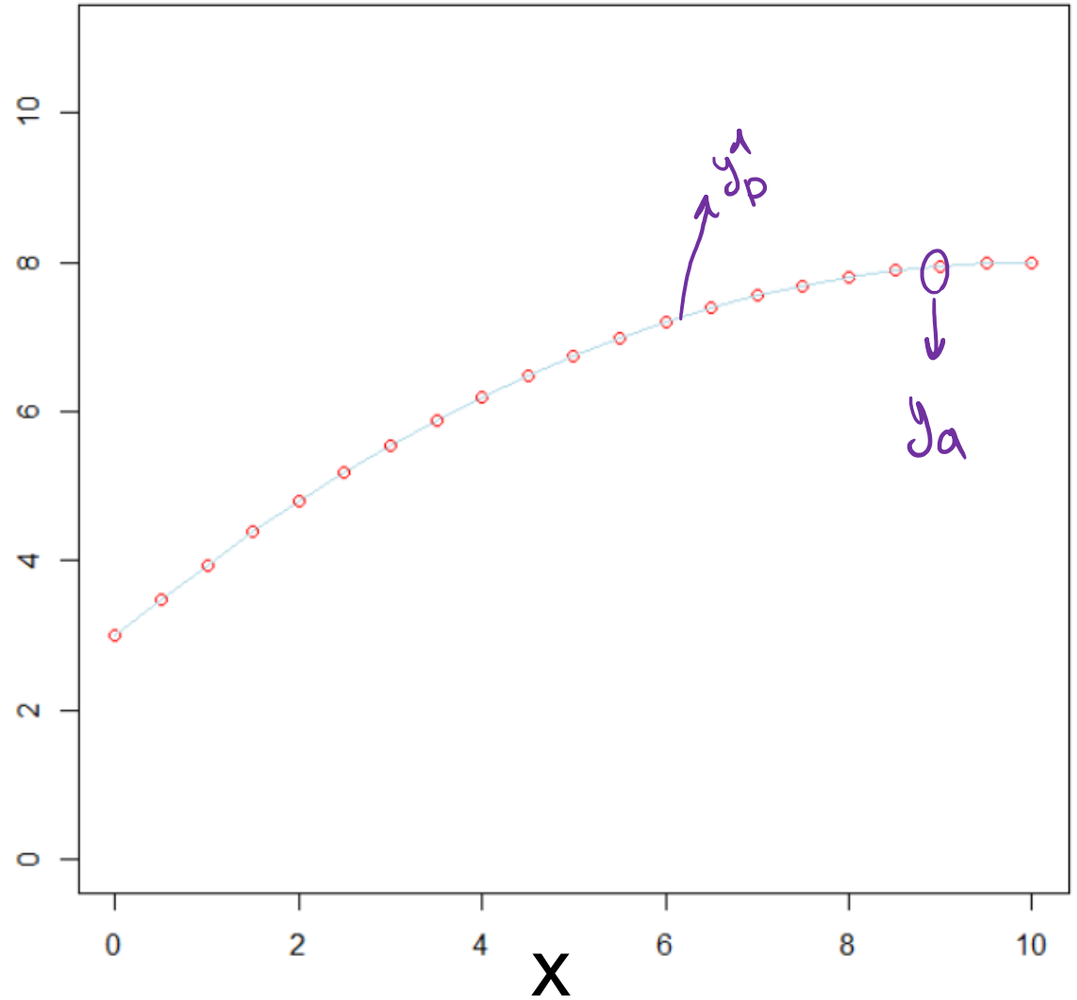
$$y = [3, 3.4875, 3.95, \ldots, 7.98, 8]$$

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

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

$Z = [\, x \quad x^2 \,]$

$X = [\, x \,]$

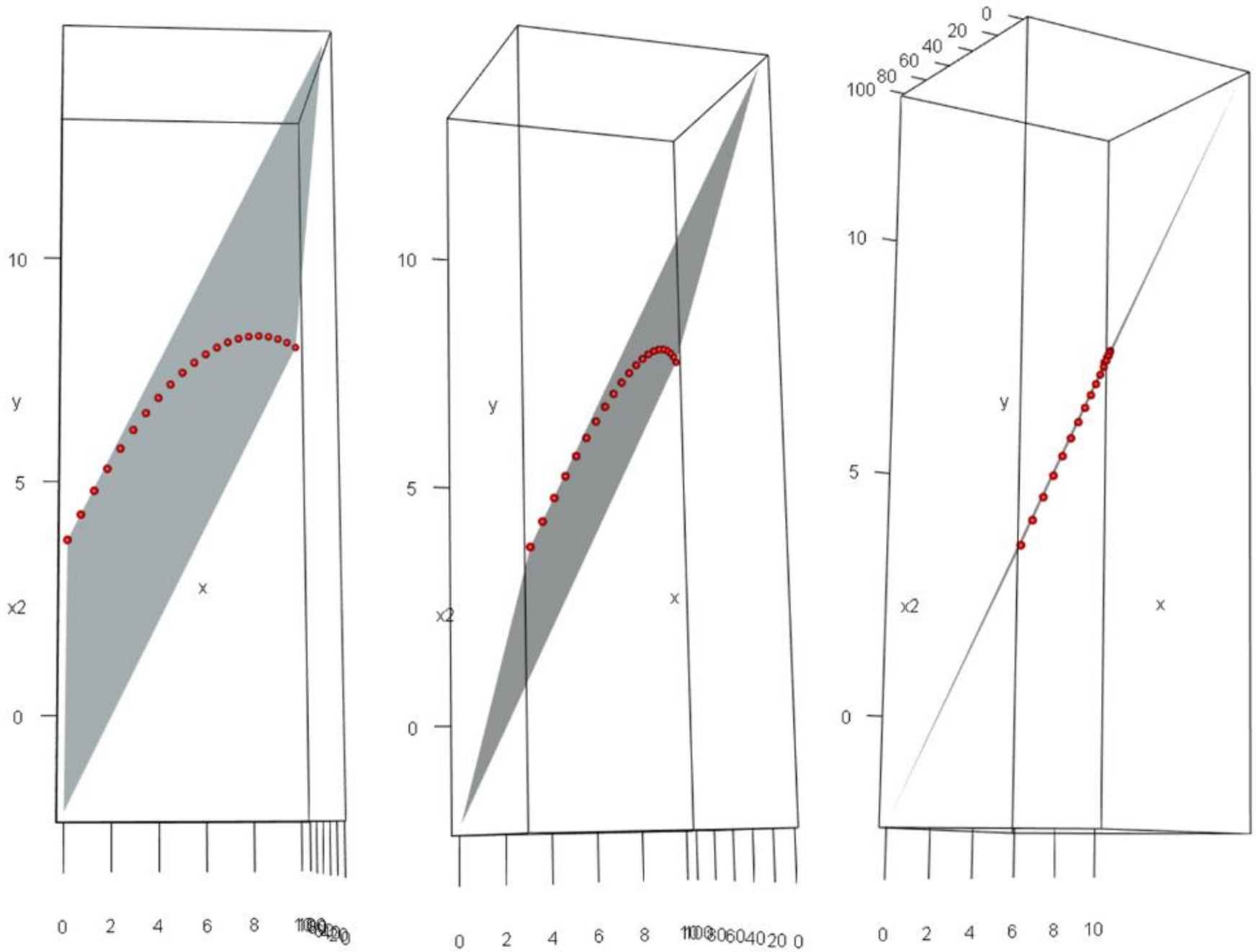


RMSE=0

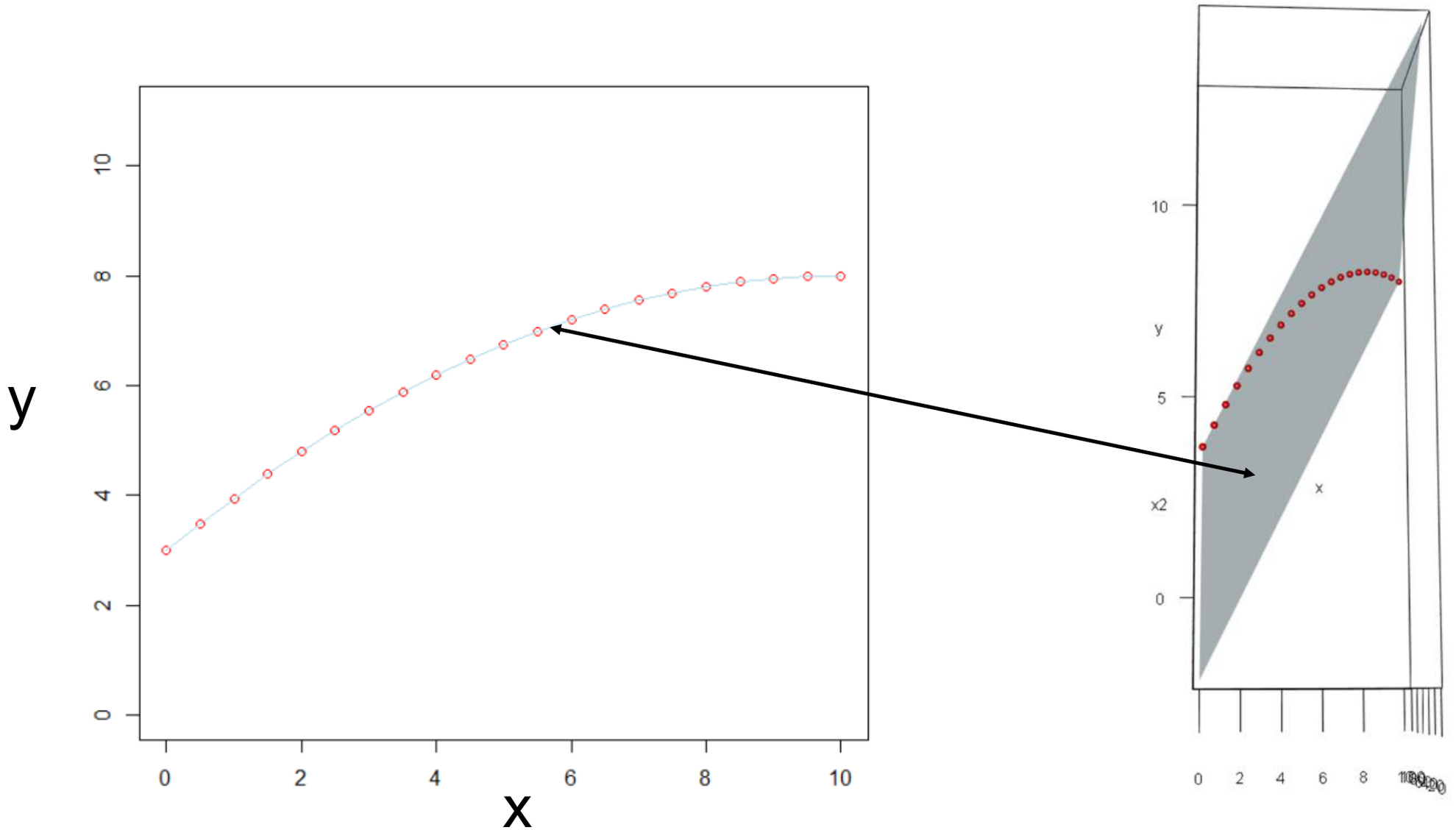
Let's add to the feature space

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

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



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.

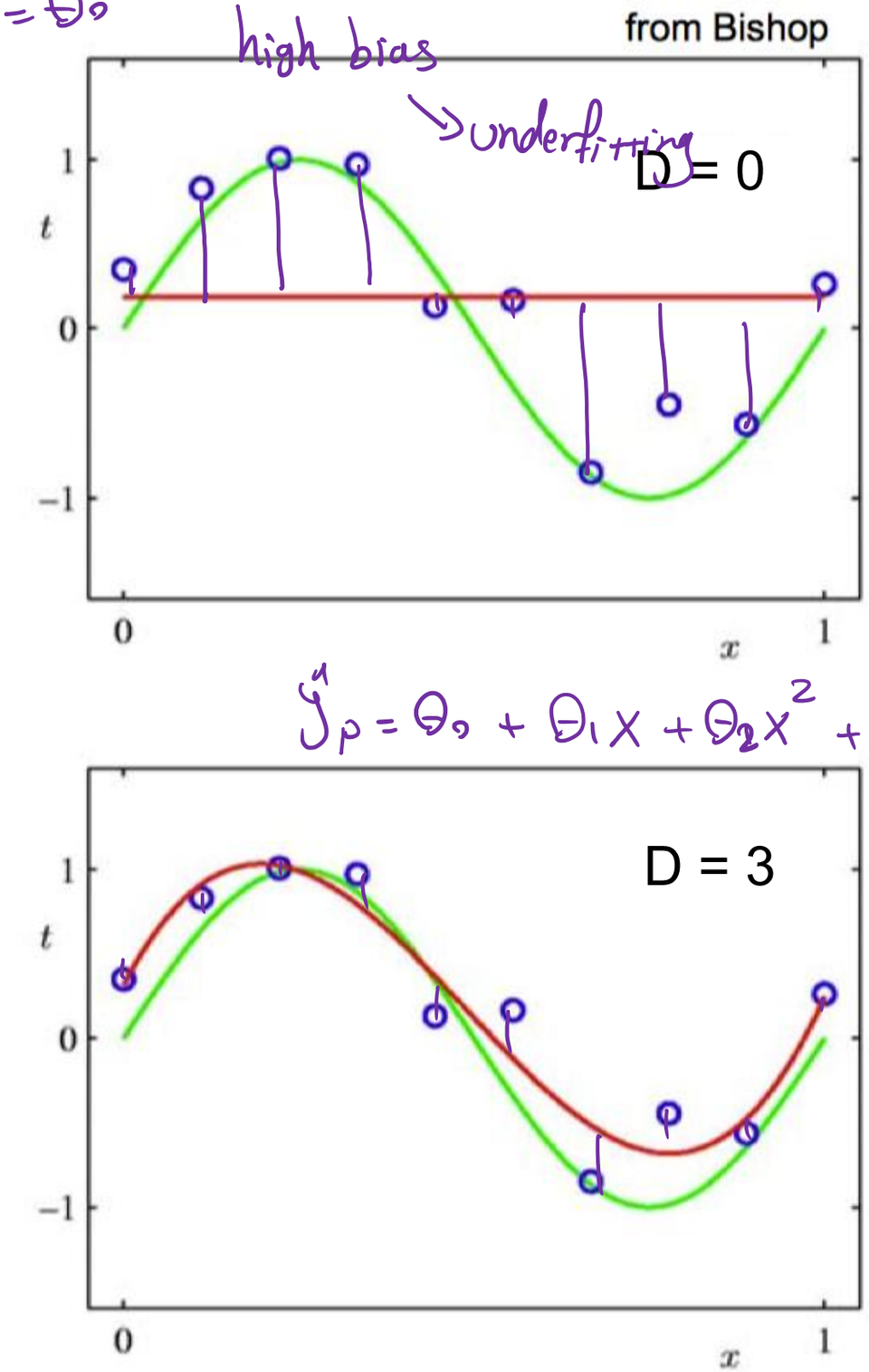


Increasing the Maximal Degree

$\hat{y}_p = \theta_0$

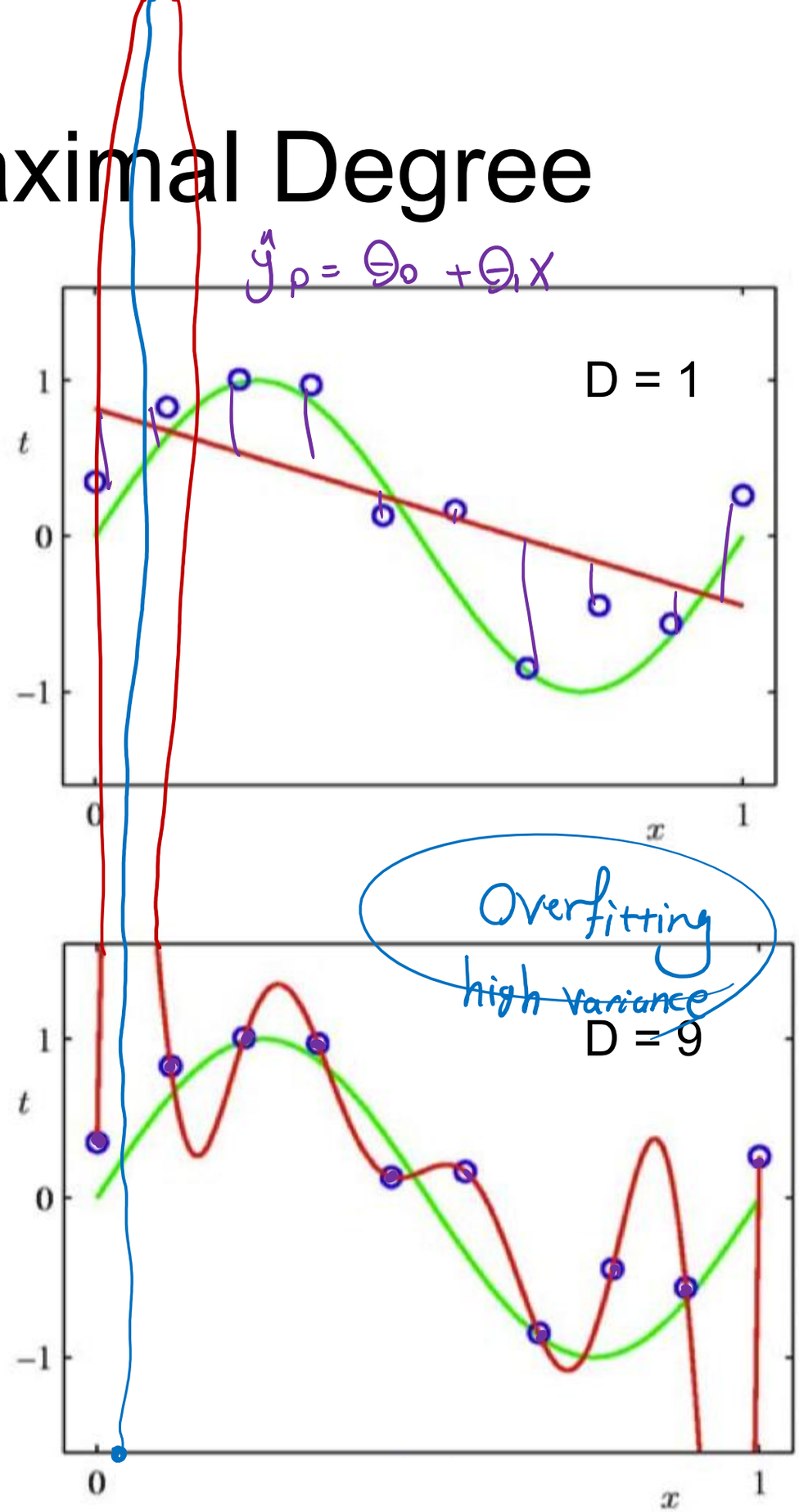
high bias

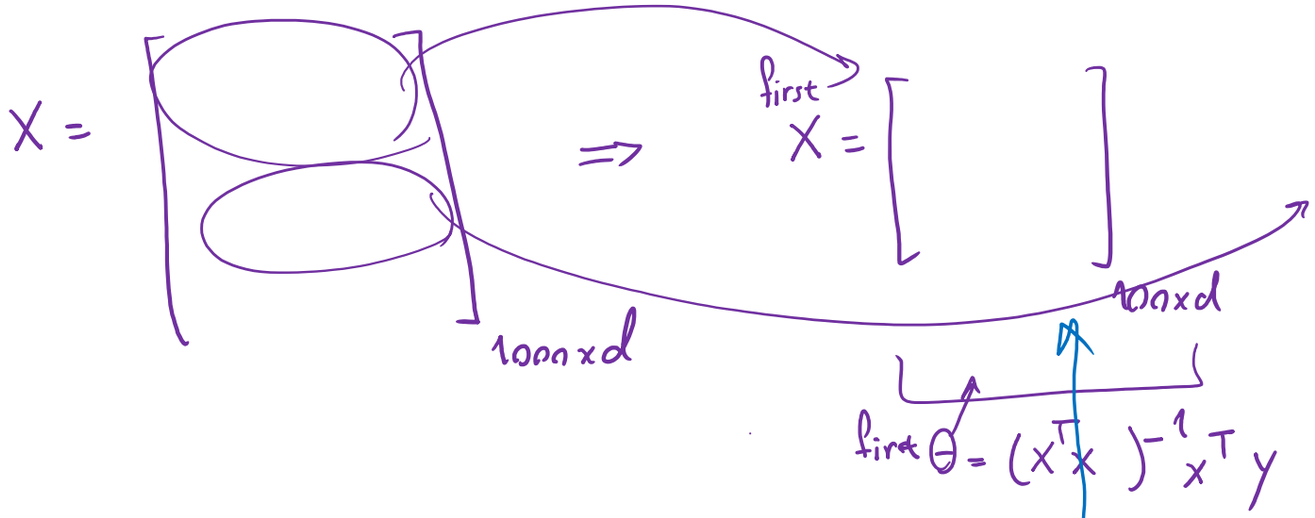
↘ underfitting

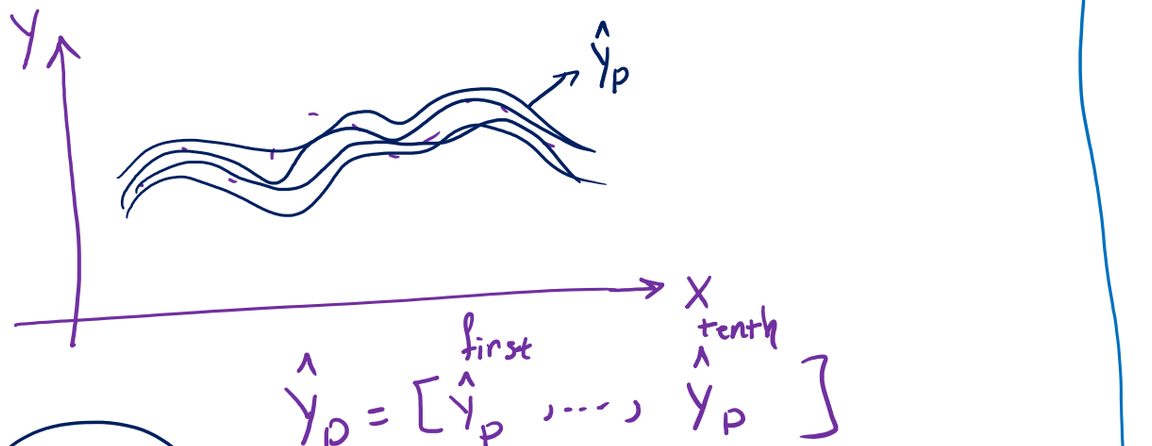


$\hat{y}_p = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$

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



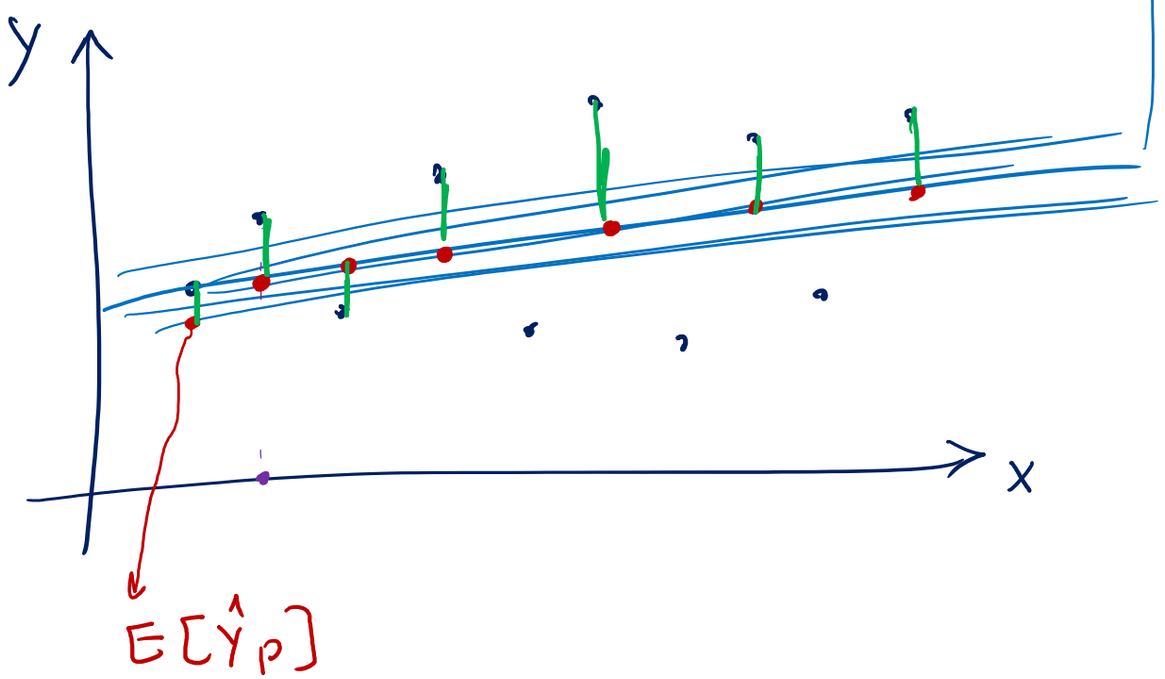




$$\frac{1}{N}\sum(\cdot) = E[\cdot]$$

$$E[\hat{y}_p] = \frac{1}{10}\sum_{j=1}^{10} \hat{y}_p \quad \in \mathbb{R} \qquad \{j^{th}\ \text{dataset}\}$$

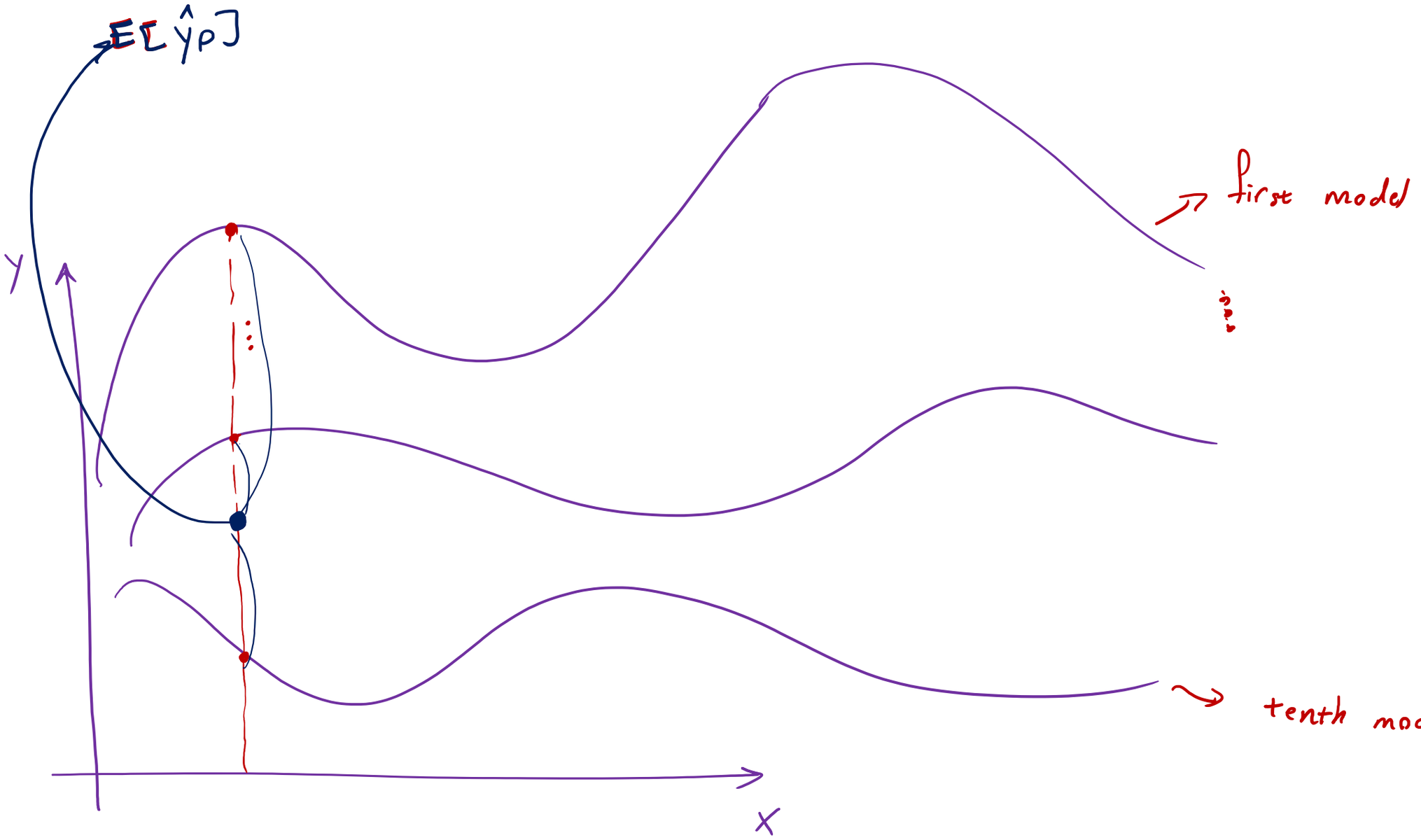
Bias



$$\text{Bias} = \hat{y}_a - E[\hat{y}_p]$$

$$E[E[\hat{y}_p]] = E[\hat{y}_p]$$

$E[\hat{y}_p] \in \mathbb{R}$



$$Var(x) = \frac{1}{N}\sum_{i=1}^{N}\left(x^{\{i\}} - \mu\right)^2 = E\left[\left(x^{\{i\}} - \mu\right)^2\right] = E\left[\left(x^{\{i\}} - E[x]\right)^2\right]$$

$$\text{Variance} = \frac{1}{10}\sum_{j=1}^{N}\left(\hat{y}_p^{\{j^{th}\ model\}} - E[\hat{y}_p]\right)^2 = E\left[\left(\hat{y}_p^{\{j^{th}\ model\}} - E[\hat{y}_p]\right)^2\right]$$

Bias-Variance Trade off

[Animation]

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

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

actual, predicted

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

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

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

y_a: Scalar, $E[y_p]$: Scalar

$E[E[y_p]] - E[y_p]$

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

$$\left(y_a - E[y_p]\right)^2 + E\left[\left(E[y_p] - y_p\right)^2\right] + 0 = \text{bias}^2 + \text{Variance}$$

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 $E\left[y_a - E\left[y_p\right]\right] = y_a - E\left[y_p\right]$

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

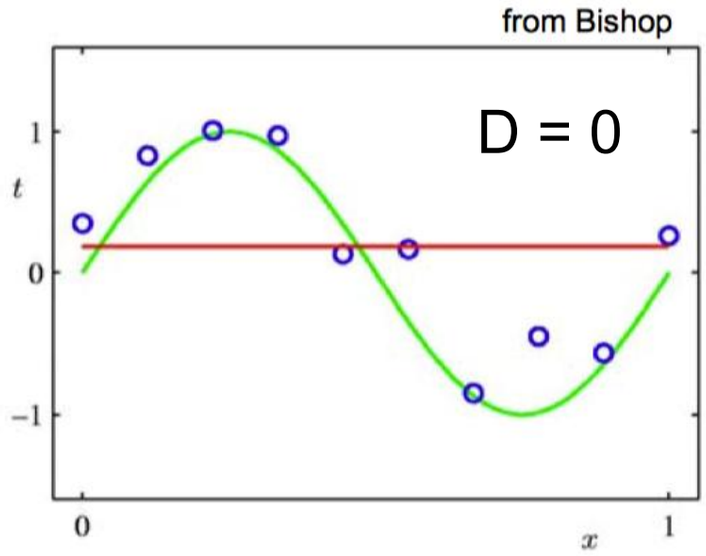
$$= 2\left(y_a - E\left[y_p\right]\right)E\left[E\left[y_p\right] - y_p\right]$$

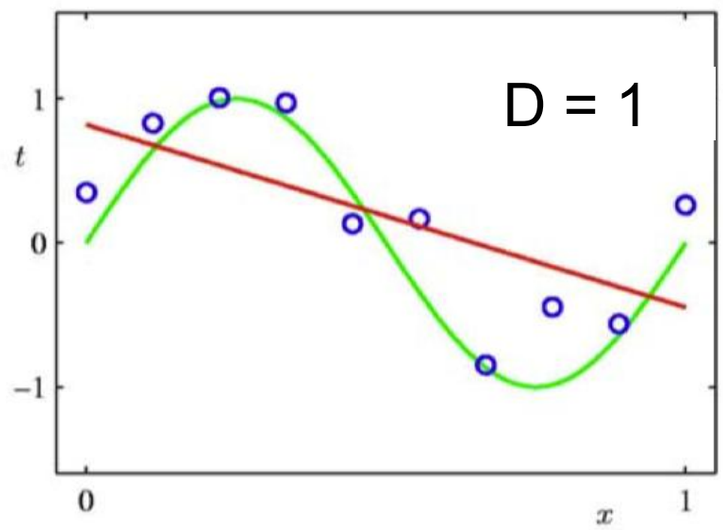
$$= 2\left(y_a - E\left[y_p\right]\right)\left(E\left[E\left[y_p\right]\right] - E\left[y_p\right]\right)$$

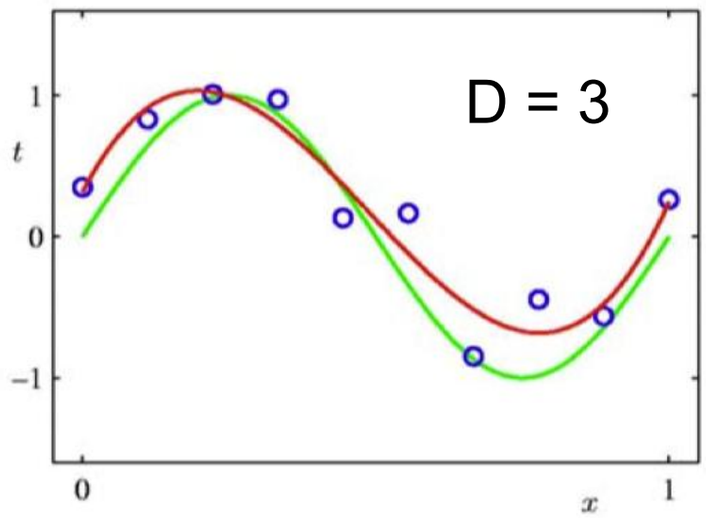
$$= 2\left(y_a - E\left[y_p\right]\right)\left(E\left[y_p\right] - E\left[y_p\right]\right) = 0$$

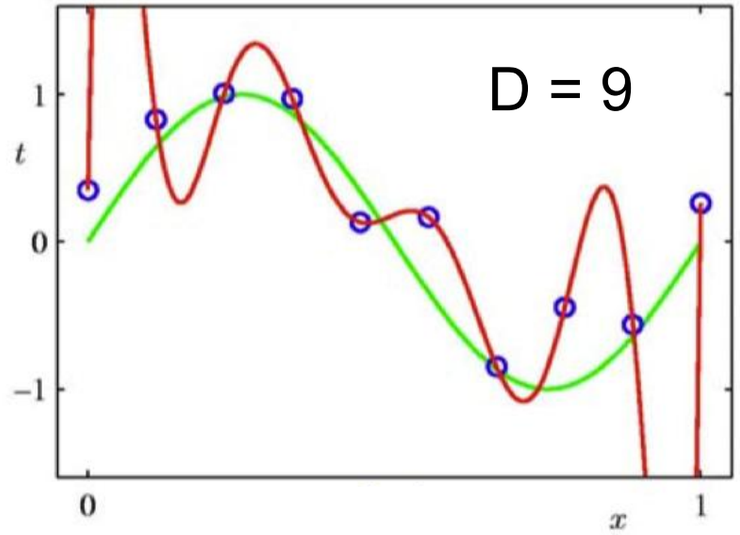
Which One is Better?

from Bishop









- 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