

# Announcements

- Midterm Assessment next Wednesday
  - Practice materials: Class slides/videos, Homeworks, weekly quizzes, weekly quiz practice test, assessment practice test
  - Don't forget your laptop, camera (if needed), cheatsheet (1 page only), buzzcard and pen/pencil. Scratch paper will be provided if needed.
  - Please be in class on time. The quiz will unlock on Canvas at 3:30 PM
  - The exam duration is 1 hour but you may need some time to log in using Honorlock
  - Everything covered upto clustering evaluation is on the syllabus
- HW3 is out
- Mid semester Anonymous Survey Out on Canvas. Due next Sunday

# The Moment



© Yongqing Bao

Incoming ML HW3

You; after HW2



# dimensionality reduction

## Recap

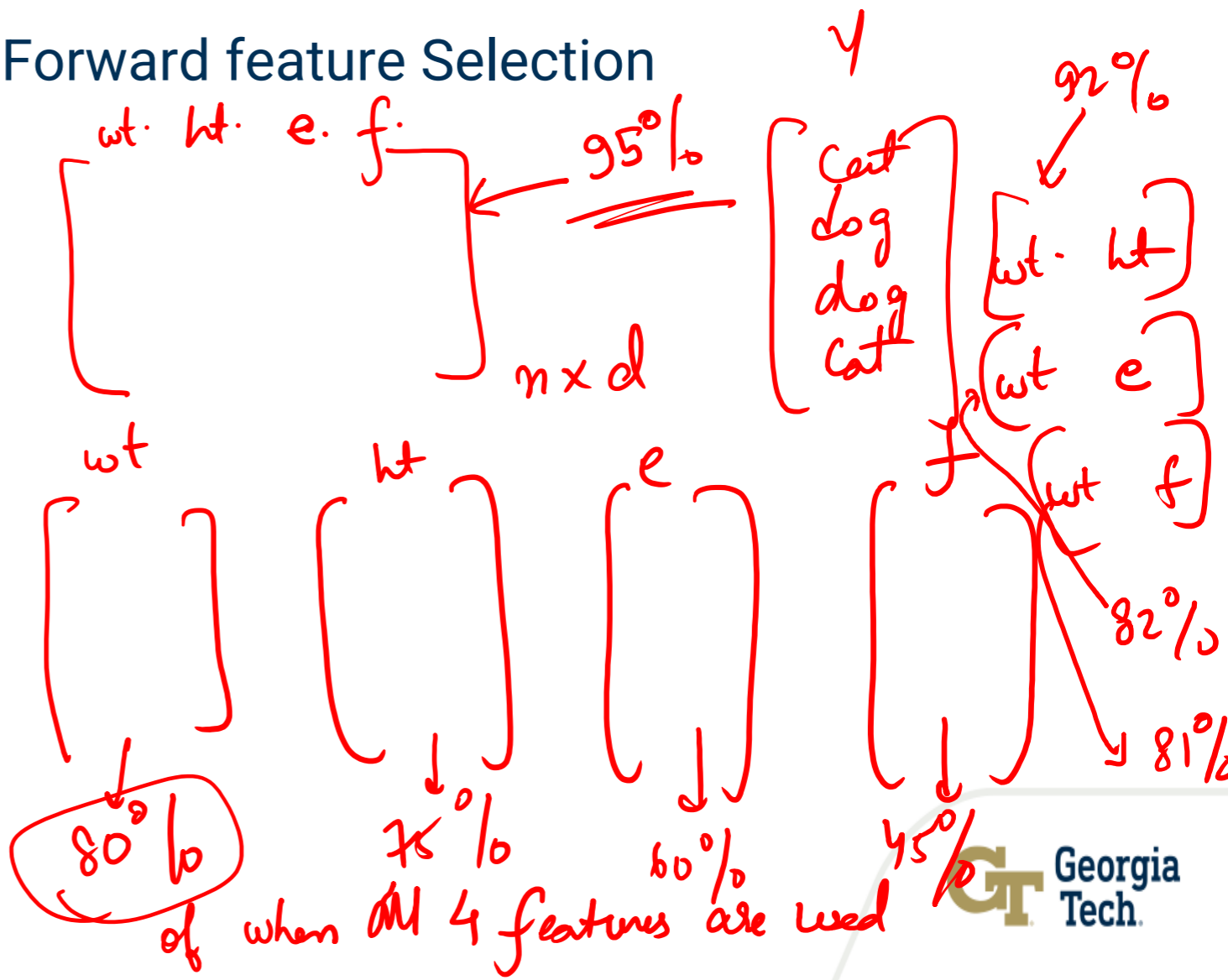
- PCA used for?
- PCA supervised or unsupervised?



$x$  dimension  $\rightarrow$  2 dimension

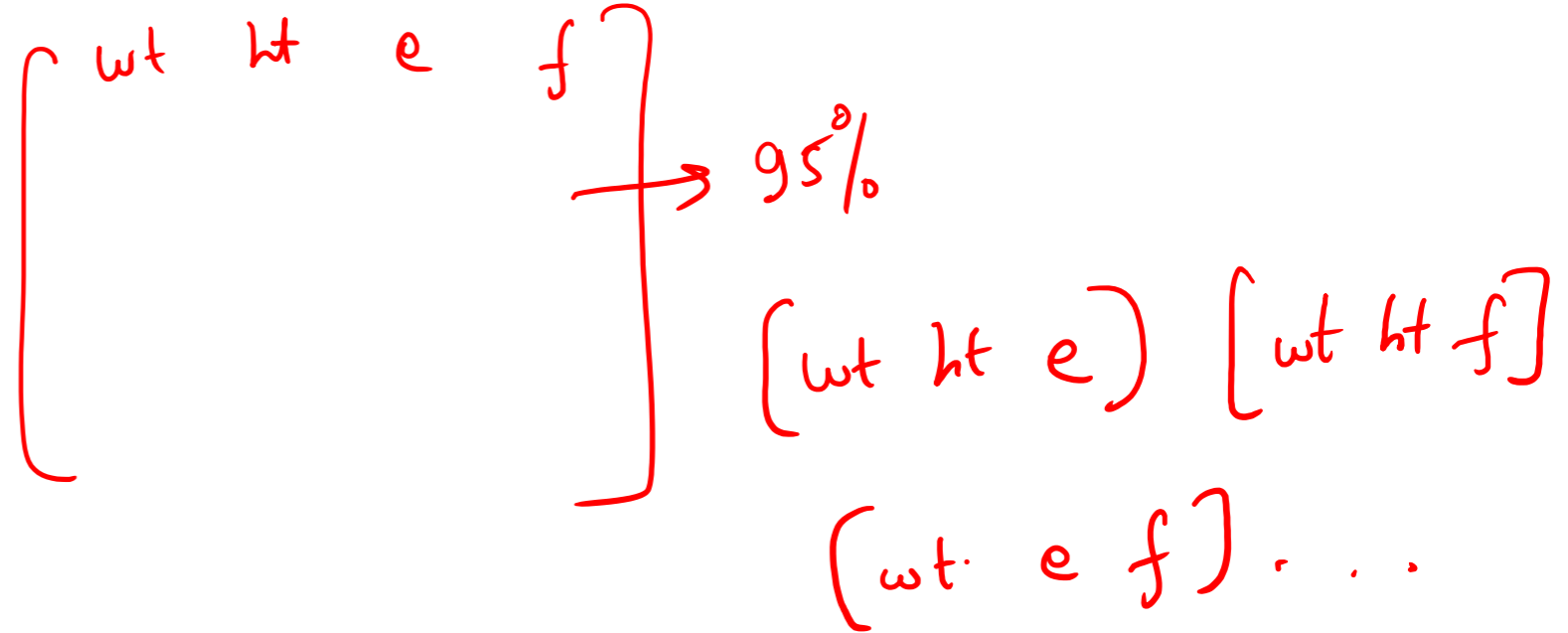
## Supervised dimensionality reduction - Forward feature Selection

- Start with **no features**.
- Add features **one at a time** — at each step, choose the feature that gives the biggest improvement to model performance (e.g., accuracy,  $R^2$ , AIC).
- Stop when adding new features no longer improves performance beyond threshold.



## Backward Feature Selection

- Start with **all features**.
- Remove features **one at a time** — at each step, drop the feature whose removal least reduces model performance.
- Stop when removing any more features harms performance too much.



Both forward and backward feature selection are quite computationally expensive

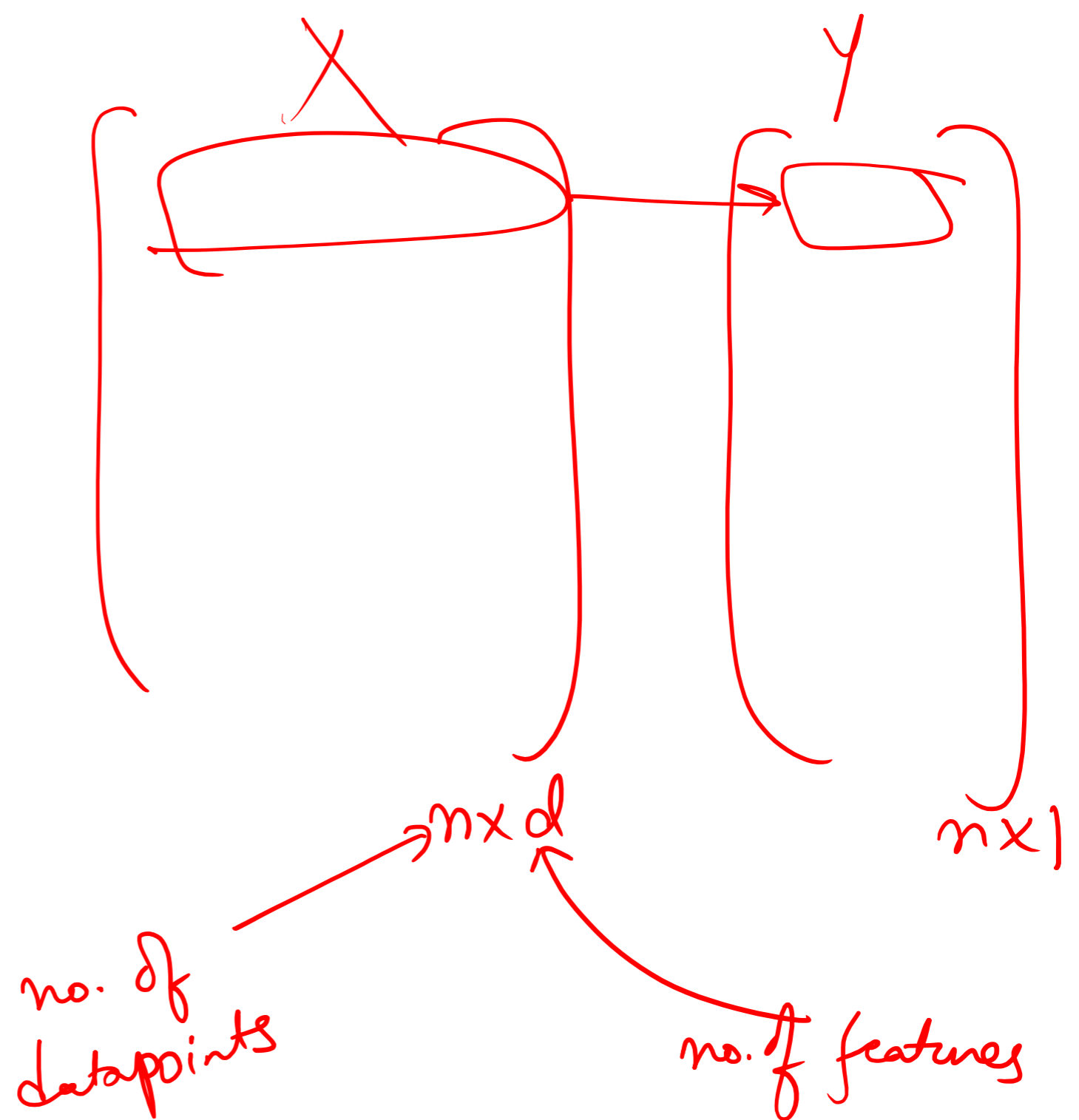
*↳ more computationally expensive*

# Linear Regression

Nimisha Roy  
Georgia Tech

# Outline

- Supervised Learning ←
- Linear Regression
- Extension



# Supervised Learning: Overview

Functions  $\mathcal{F}$

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

Training data

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

$y_a$  ← actual target  
 $\hat{y}_p$  ← prediction  
 $f(x^{(i)})$

→ find  $f$  that makes  $\hat{y}_p \approx y_a$

LEARNING

find  $\hat{f} \in \mathcal{F}$   
s.t.  $y_a \approx f(x^{(i)})$

linear function



Learning machine

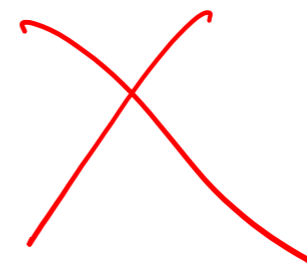
PREDICTION

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

New data

$$x$$

# Supervised Learning: Split of data



*k-means* →  $k$   
*GMM* →  $\mu, \sigma, k$   
*DBSCAN* →  $\epsilon, \text{minpts.}$   
 $10^0$  /  $0$  *HC* →  $X$

70%

## Training Set

- **Purpose:** Used to **fit** the model — the algorithm learns patterns, relationships, and weights from this data.
- **What happens here:** The model adjusts its internal parameters to minimize error.
- **Analogy:** Like studying the material before an exam.

## Validation Set

- **Purpose:** Used to **tune hyperparameters** and make design decisions (e.g., how many layers, what learning rate, etc.).
- **What happens here:** The model is not trained on this data; instead, it helps to evaluate how changes affect performance, preventing overfitting.
- **Analogy:** Like taking practice tests to decide which study strategy works best.

20%

## Test Set

- **Purpose:** Used to **assess the final performance** of the model after all training and tuning.
- **What happens here:** No further learning or tweaking — it provides an **unbiased estimate** of the model's generalization ability.
- **Analogy:** Like the final exam — measures how well you really learned the material.

# Supervised Learning: Two Types of Tasks

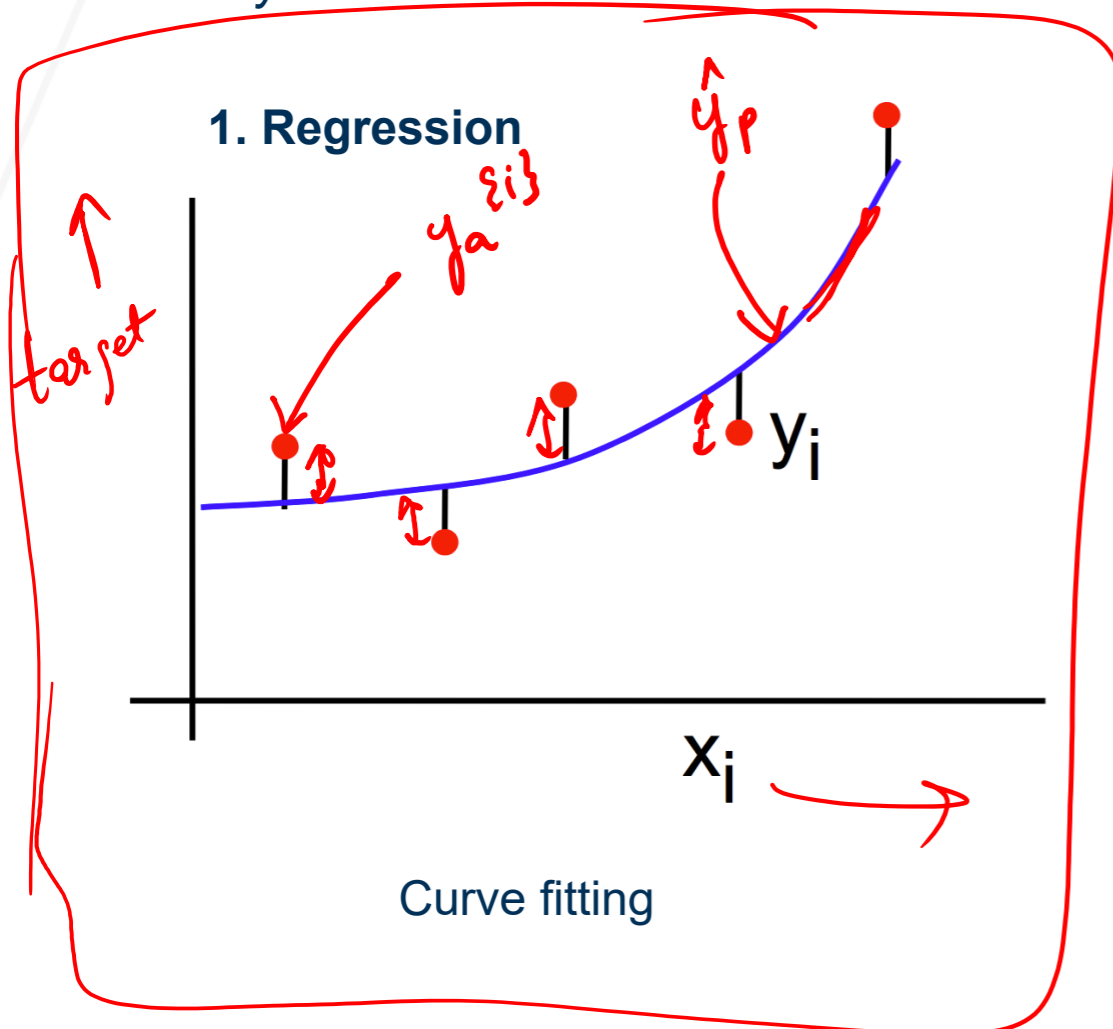
Given: training data

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

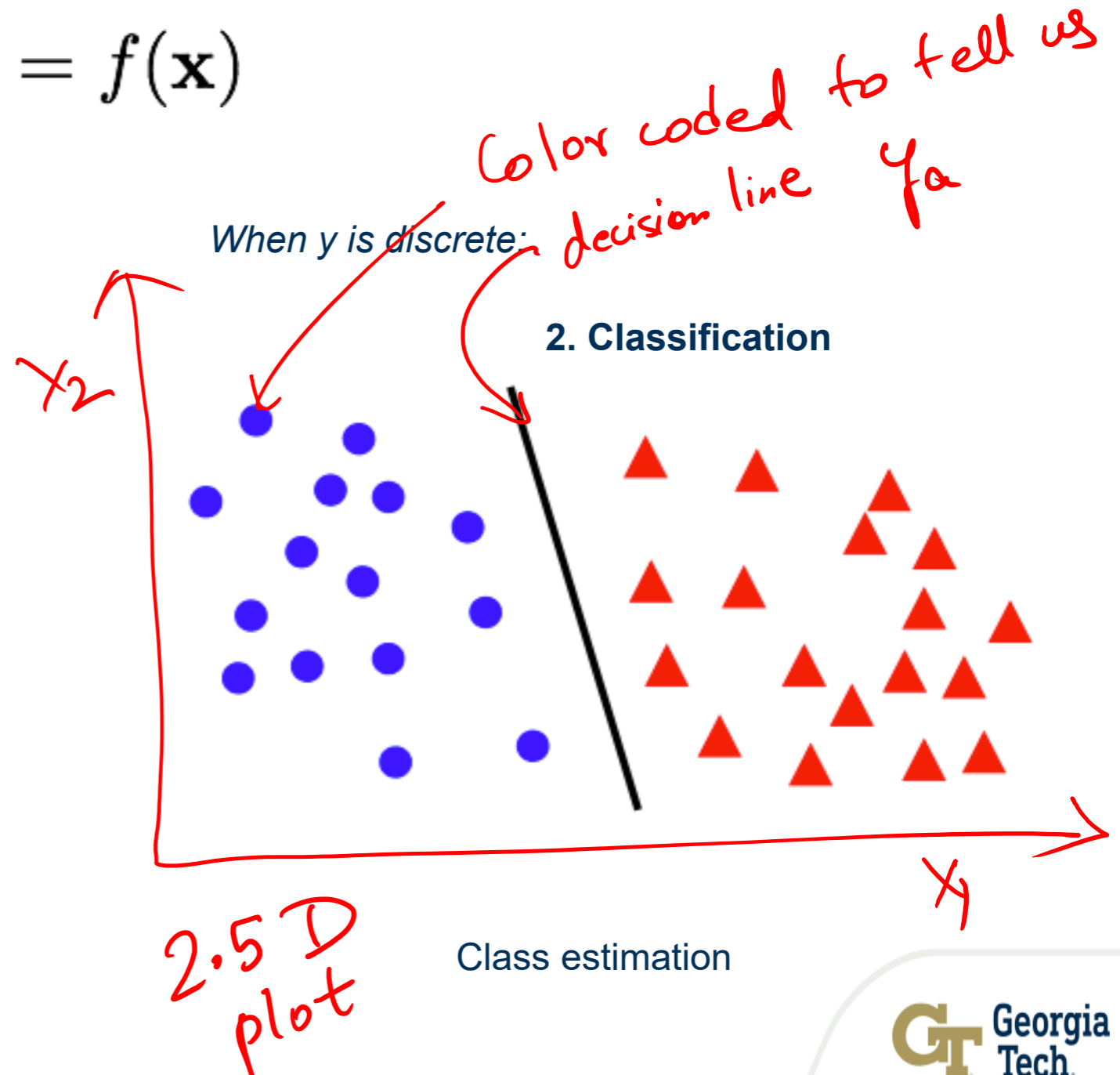
Learn: a function

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

When  $y$  is continuous:



When  $y$  is discrete:

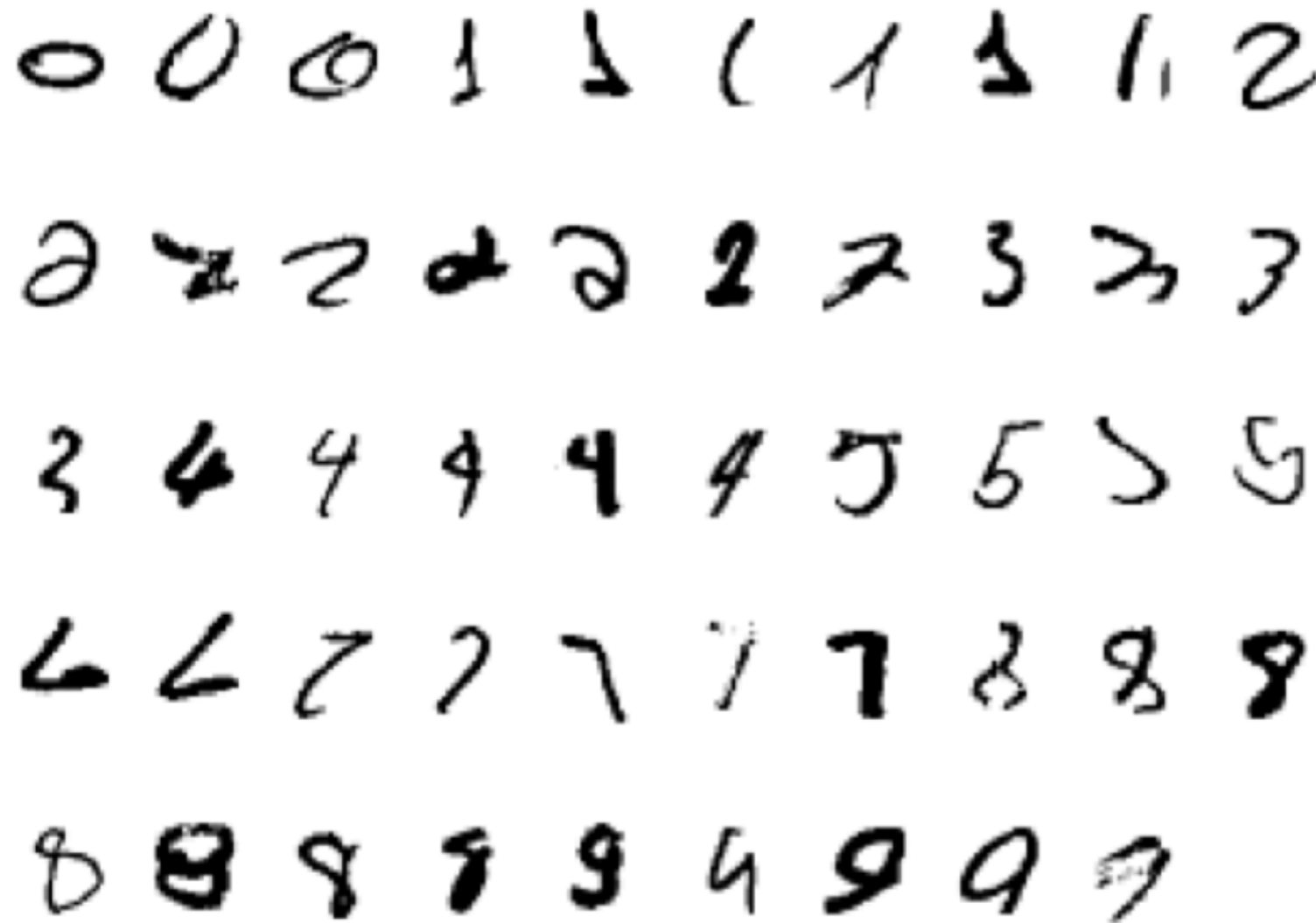


# Classification Example 1: Handwritten digit recognition

Classes → 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

As a supervised classification problem

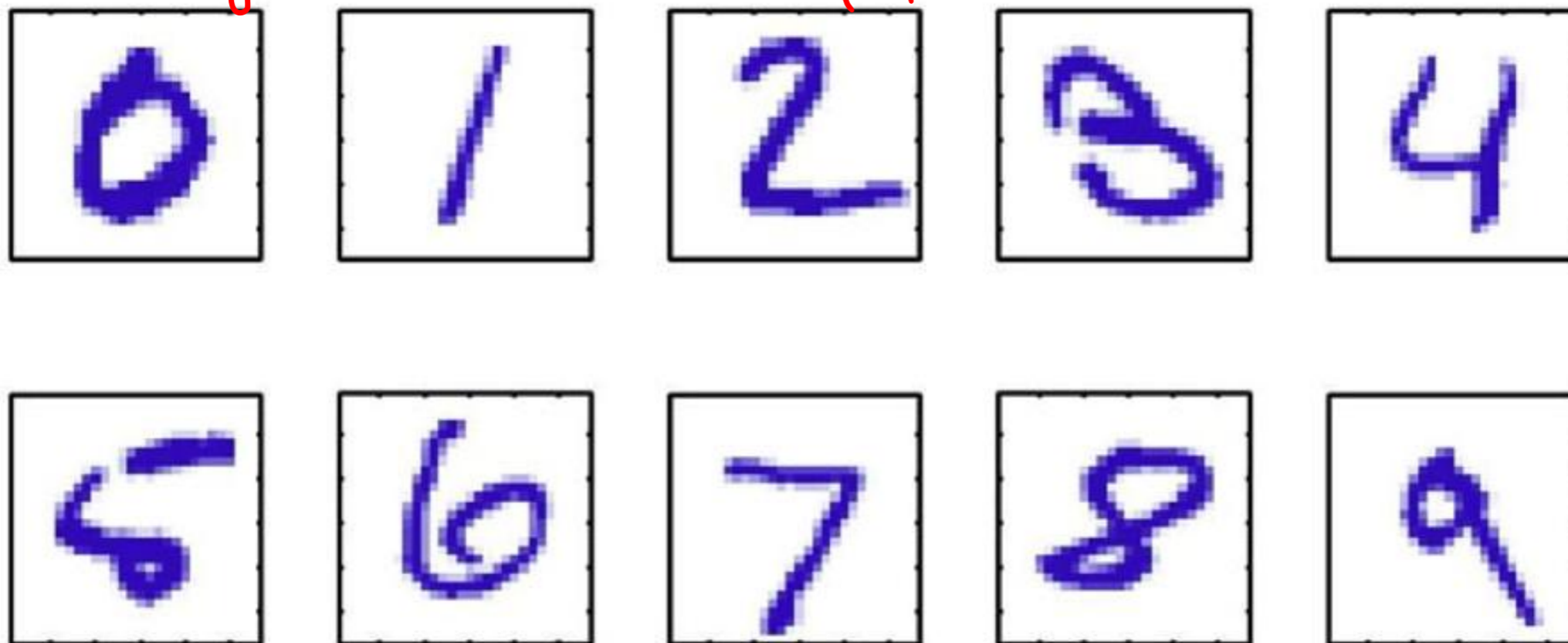
Start with training data, e.g. 6000 examples of each digit



- 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

① Preprocessing / Feature Engineering / Convolution  
② Dimensionality Reduction / Forward Feature Selection / Backward Feature Selection



Images are 28 x 28 pixels

A classification problem

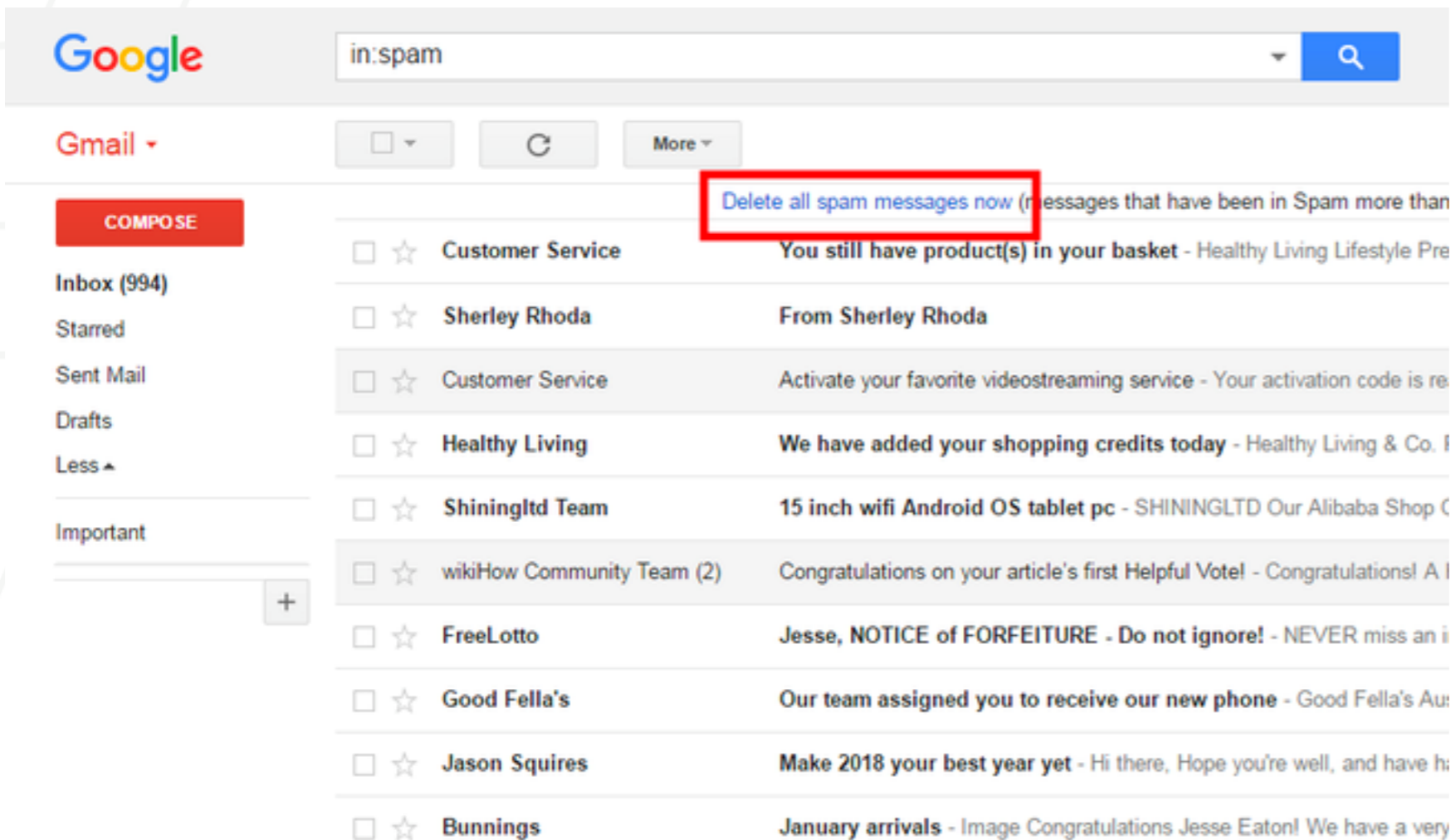
784

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

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

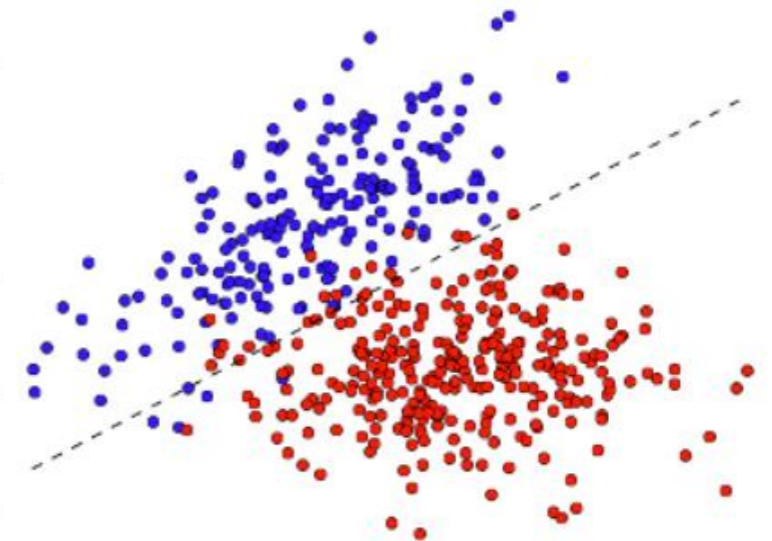
$$f : \mathbf{x} \rightarrow \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

# Classification Example 2: Spam Detection



- ① feature engineering
- ② remove stop words / lemmatization

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 <sup>2</sup> )	# bedroom	Rent (\$)
230	1	600
506	2	1000
433	2	1100
109	1	500
...		
150	1	?
270	1.5	?

*training data*

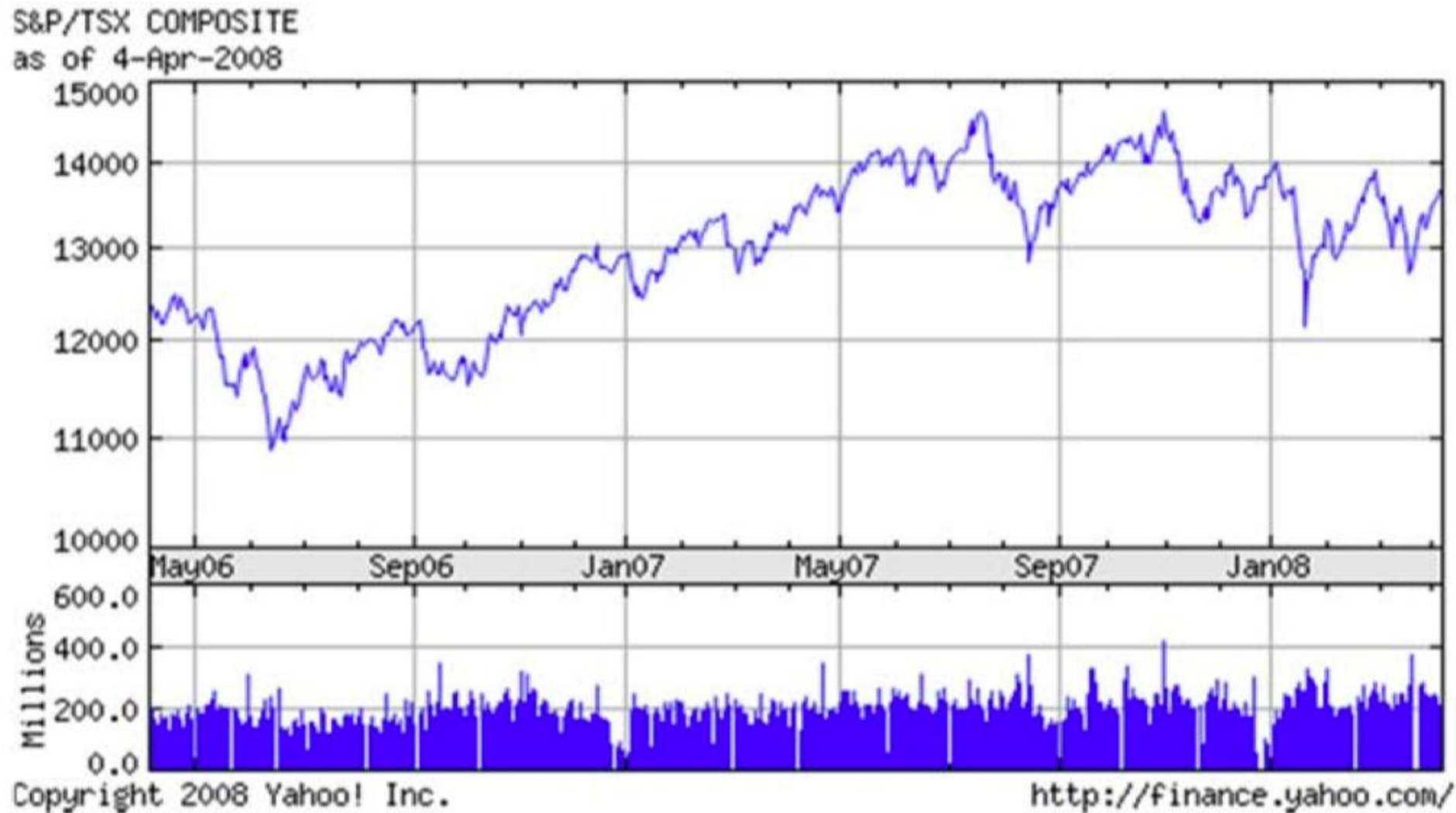
*ya*

*y-hat*

*test data*

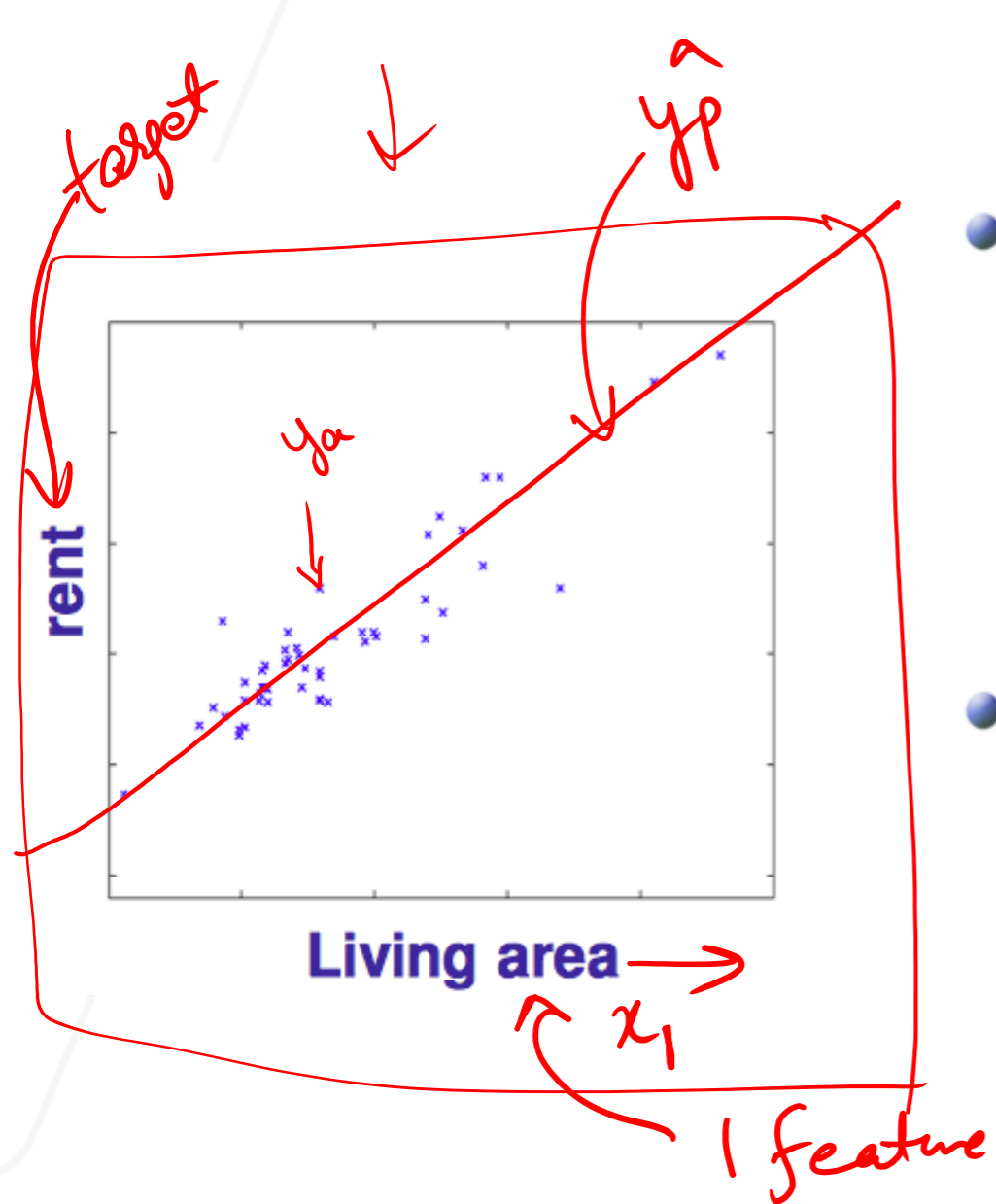
# Regression Example 2: Stock Price Prediction

Classification  
↑ or ↓  
next day



- Task is to predict stock price at future date

A regression problem



- Features:

- Living area, distance to campus, # bedroom ...
- Denote as  $x = (x_1, x_2, \dots, x_d)$

- Target:

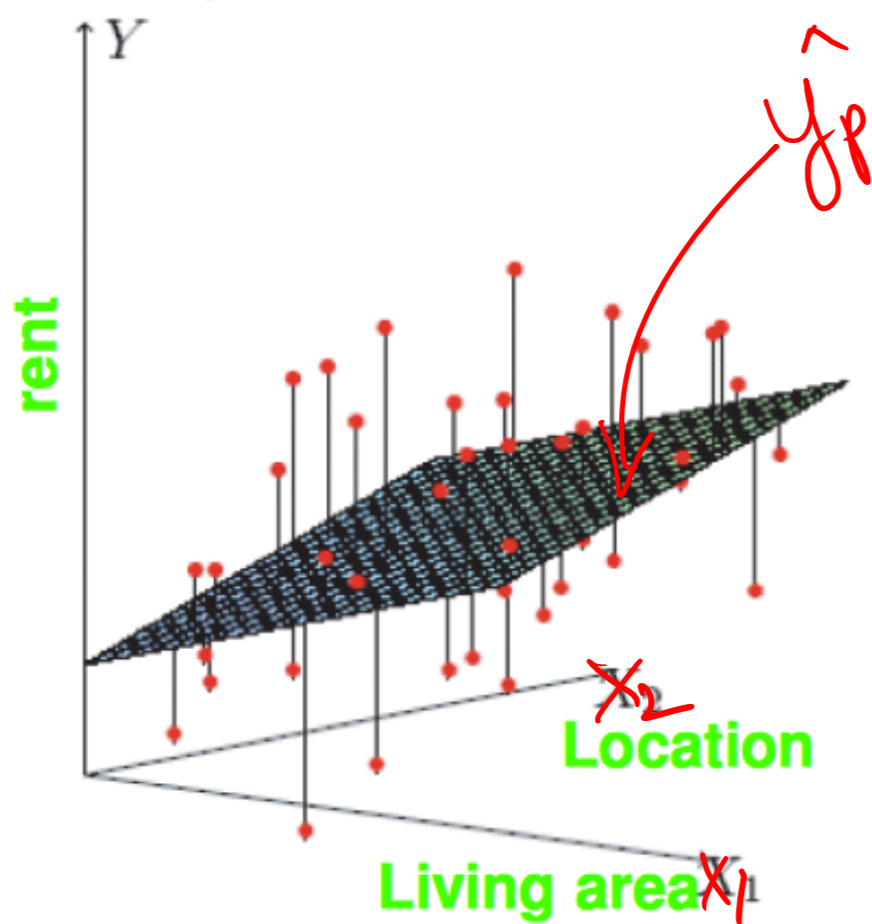
- Rent
- Denoted as  $y$

$$y = mx + b$$

Annotations for the equation above: 'slope' points to 'm', 'intercept' points to 'b', and 'rotation' points to the '+' sign.

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

Annotations for the equation above: 'bias term' points to  $\theta_0$ ; 'gateway to  $x_1$ ' points to  $\theta_1$ ; and 'feature 1' points to  $x_1$ .



IT IS  
LCF

$$\hat{y}_p = \theta_0 + \theta_1 x_1^2 + \theta_2 x_2^3 + \dots$$

feature engineering

● Training set:

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

~~not an LCF~~

$$\hat{y}_p = \theta_0^2 + \theta_1^2 x_1 + \theta_2^3 x_2 + \dots$$

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

$$x^{(i)} = \begin{bmatrix} 1 & x_1 & x_2 & \dots & x_d \end{bmatrix} \quad x^{(d+1)}$$

Parameters that the model needs to learn

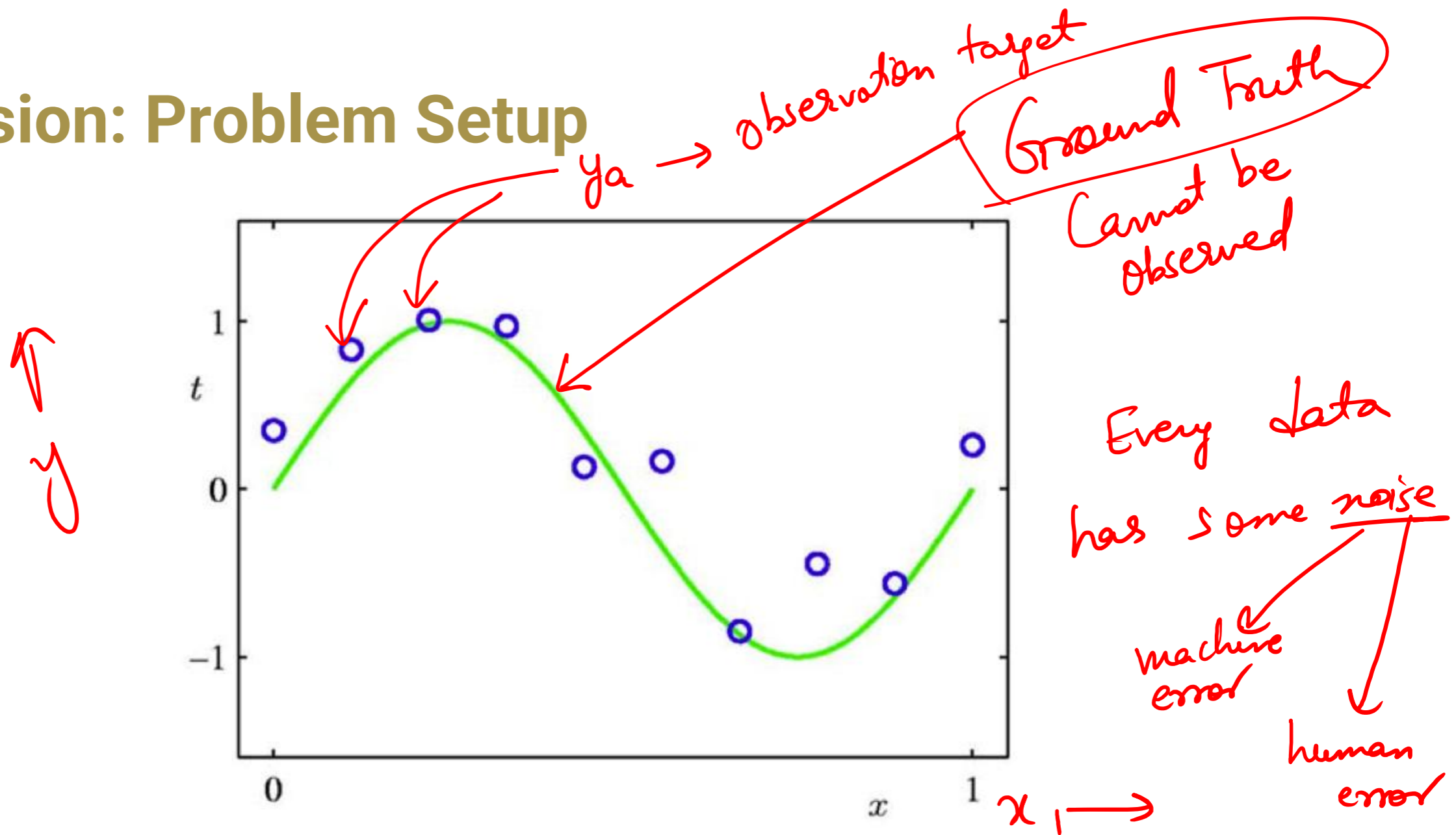
$$\theta =$$

$$\begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \quad (d+1 \times 1)$$

$$\hat{y}_p = X \cdot \theta = 1 \times \theta_0 + x_1 \times \theta_1 + \dots + x_d \times \theta_d$$

LCF: 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

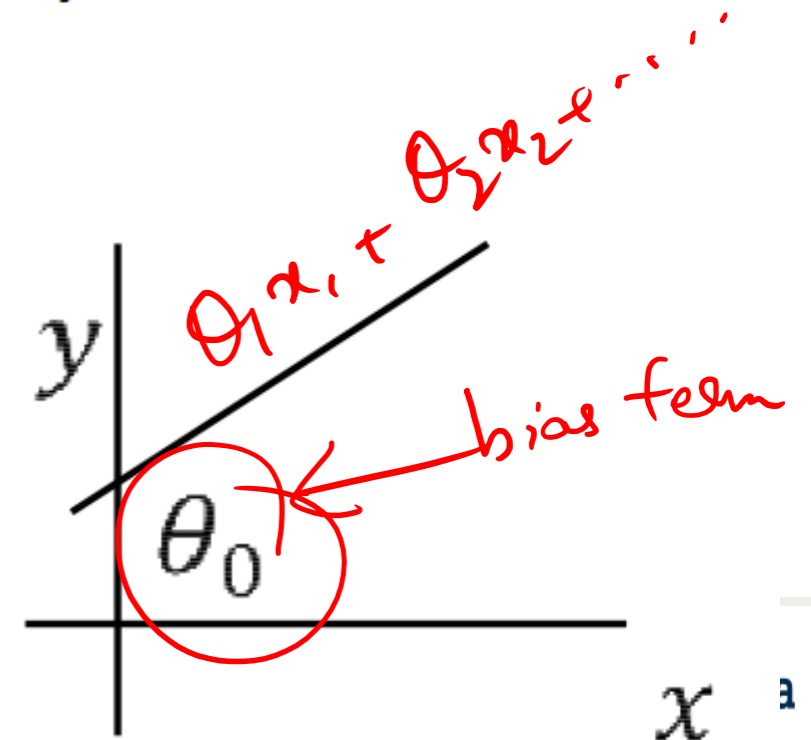
- Assume  $y$  is a linear function of  $x$  (features) plus noise  $\epsilon$

$$\hat{y}_p = \theta_0 + \theta_1 x_1 + \dots + \theta_d x_d + \epsilon$$

*noise*

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



# Least Mean Square Method

$E[\theta] \leftarrow$  Expectation

$\int y^{\{i\}} - x^{\{i\}} \theta$

- Given  $n$  data points, find  $\theta$  that minimizes the mean square error

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

Error  $\rightarrow E(\theta)$

$y_a$   $\hat{y}_p$   $\hat{\epsilon}_i$

- Why square?
- Differentiability
  - Global optimum convex
  - Penalization

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

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

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

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

for loop

## Matrix form

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

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

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

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

$n \times 1$

# Matrix Version and Optimization

$$y = X\theta$$

$$\theta = X^{-1} \cdot y$$

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

Closed Form Sol.  
Global sol.

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

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

or for a rectangular matrix

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

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

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

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

$X_{n \times d}$

$n = \text{instances}$

$d = \text{dimension}$

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

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.

# Recap

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

• Why good?

①

Closed Form Sol.

②

Global Sol.

• Why bad?

①

Computationally Expensive  $X^T X$

②

$X^T X \rightarrow$  may not have an inverse

Computationally expensive inverse

# Alternative Way to Optimize



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

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

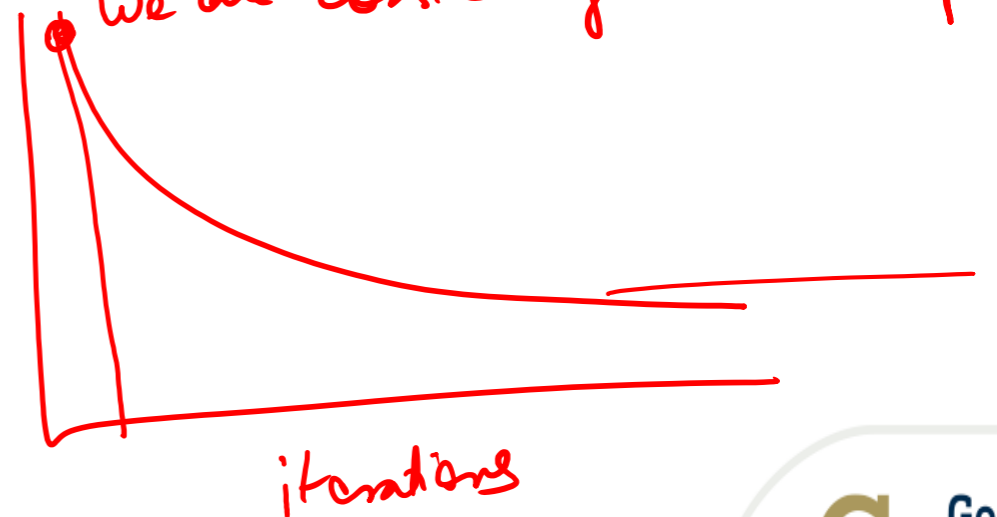
$$\theta^{t+1} \leftarrow \theta^t - \alpha \frac{\partial L}{\partial \theta}$$

- Gradient descent**

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

We are considering all datapoints

$E(\theta)$   
 $L(\theta)$



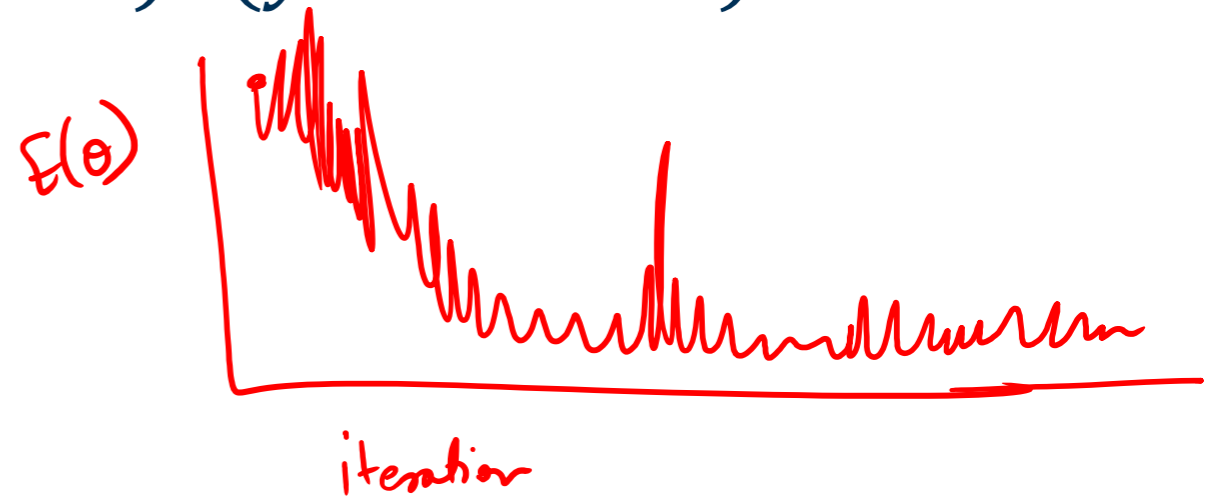
# Alternative Way to Optimize

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

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

*only 1 data point*

*Batch Gradient Descent*



# Recap

- Stochastic gradient update rule

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

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

- Gradient descent

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

- 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 ..)

# Quick Knowledge Check

1. What is the main difference between regression and classification tasks? A. Regression predicts discrete labels, classification predicts continuous outputs. B. Regression predicts continuous values, classification predicts discrete labels. C. Both predict continuous values. D. Both predict categories
2. Which of the following is an example of a regression task? A. Detecting spam emails. B. Predicting a car's fuel efficiency. C. Recognizing handwritten digits. D. Detecting positive vs. negative sentiments in reviews
3. In **backward feature selection**, what happens at each step? A. The feature that least improves performance is added. B. The feature that least reduces performance is removed. C. The feature with the lowest correlation is removed. D. The most important feature is added
4. In the linear regression equation  $\hat{y} = w_1x_1 + w_2x_2 + \dots + b$ , what does **b** represent? A. The slope of the model. B. The learning rate. C. The bias or intercept term. D. The regularization penalty
5. What is the goal of the least mean square (LMS) method? A. Minimize classification errors. B. Maximize model complexity. C. Minimize the sum of squared residuals between predictions and actual values. D. Minimize correlation between features
6. Why is it important to include a bias term in linear regression? A. To allow the model to fit data that doesn't pass through the origin. B. To normalize input features. C. To regularize the model. D. To reduce computational cost

# Recap

Linear Combination of Features  
 $y_p^{\text{scalar}} = \theta \cdot X^{\text{features}}$   
 $X^{\text{features}} = [1 \ x_1 \ x_2 \ x_3 \ \dots \ x_d]_{1 \times (d+1)}$   
 $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}_{(d+1) \times 1}$   
 $y_a$

• LCF

• Objective function

$$L(\theta) / E(\theta) = \frac{1}{N} \sum_{i=1}^N (y_a^{\text{features}} - \theta \cdot X^{\text{features}})^2$$

• Global solution

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

• Why good?

→ Global sol.

→ Penalizes error more → could be dangerous for outliers

→ Differentiable

→ Closed form Sol.

• Why bad?

- Compute power  $X^T X$
- $(X^T X)^{-1}$  sometimes may not exist
- Outliers

# GD vs SGD vs BGD

## Numerical Approaches

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

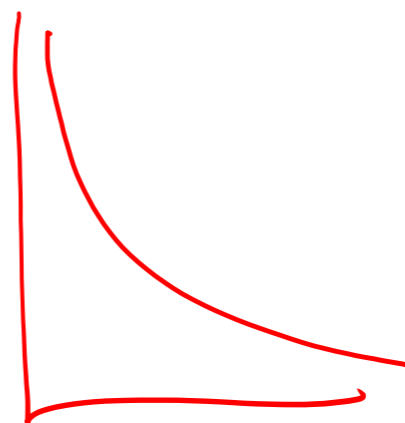
Stochastic Batch

looks at the entire dataset for 1 iteration  $\rightarrow$  high memory consumption

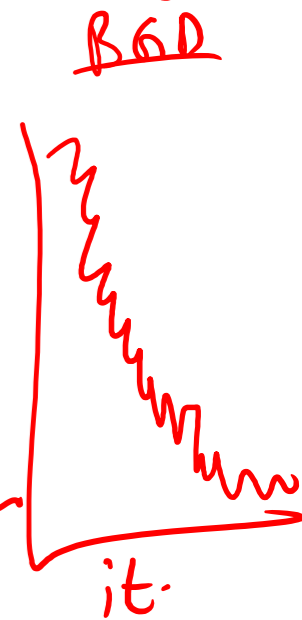
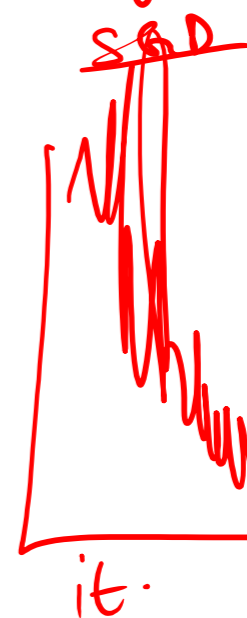
look at 1 datapoint at a time  $\rightarrow$  Too long to converge

1 batch at a time.

$E(\theta)$



iteration = epoch



Epoch

1 epoch means we have covered all datapoints at least once

# Linear regression for classification

Its not good for  
classification  
tasks

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

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

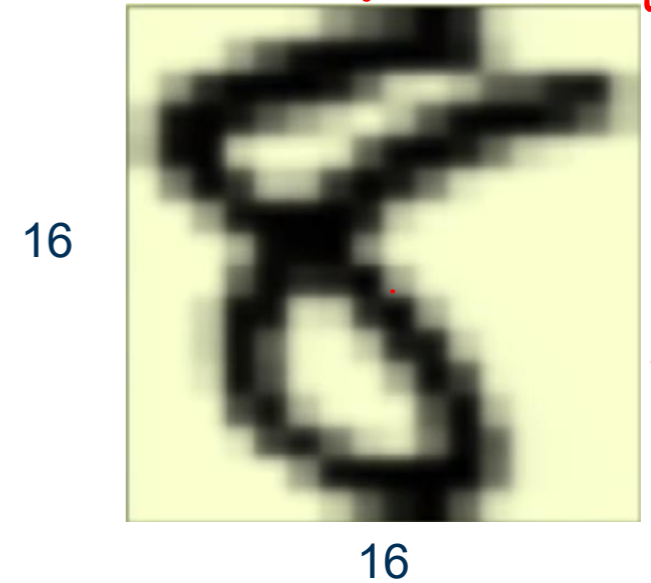
Extract useful information

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

*Sum up all the pixels = intensity*

*Symmetry = -(difference between flip version)*

Problem Statement: Classify  
5 vs 1 in handwritten  
digits using linear regression



$$\theta = (1, x_1, x_2)$$

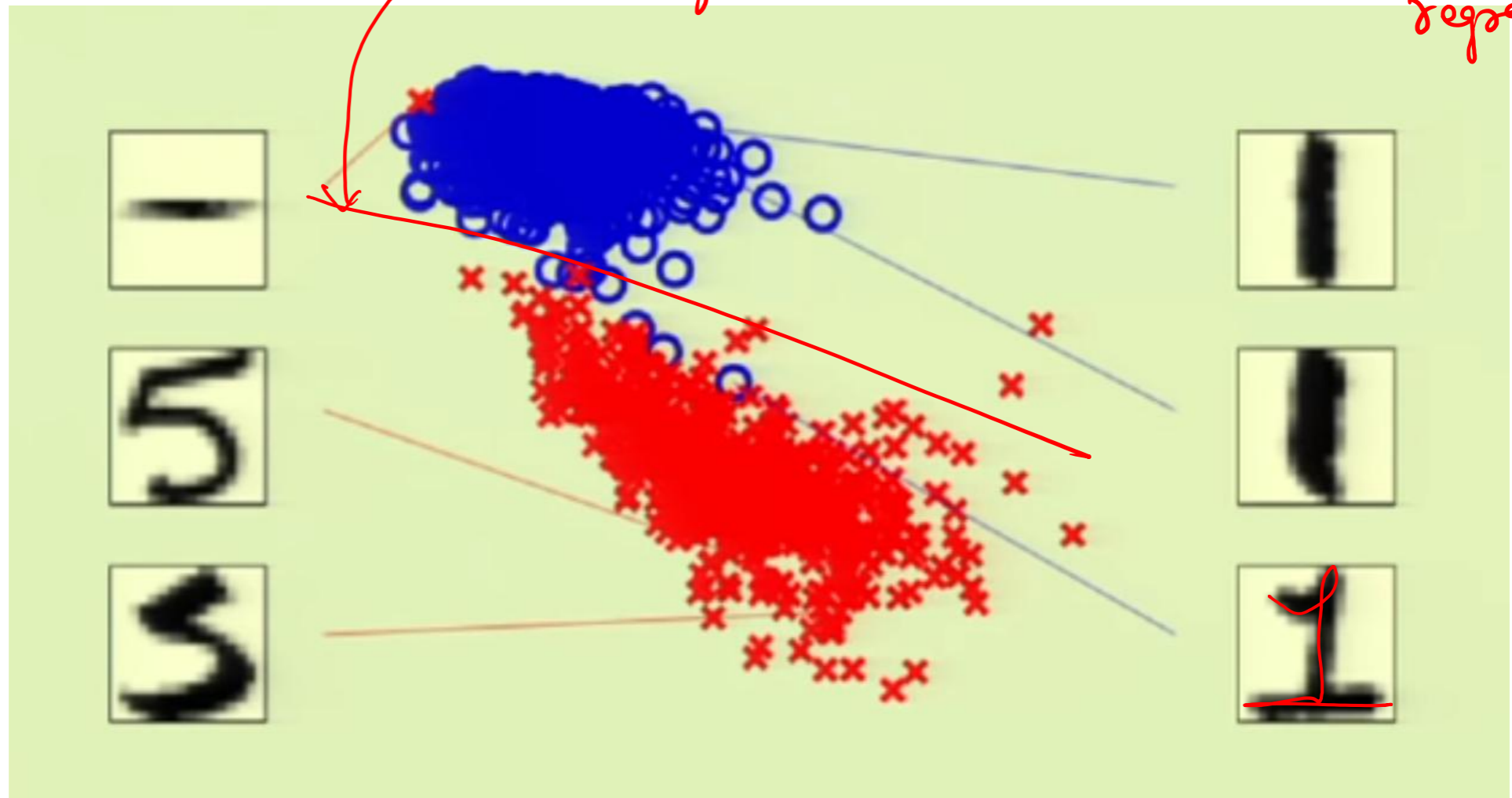
$x_1 = \text{intensity}$   $x_2 = \text{symmetry}$

It is almost linearly separable

How to get the decision line using linear regression?

2.5D

symmetry



intensity

# Linear regression for classification

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

$-\infty < \text{value} < \infty$

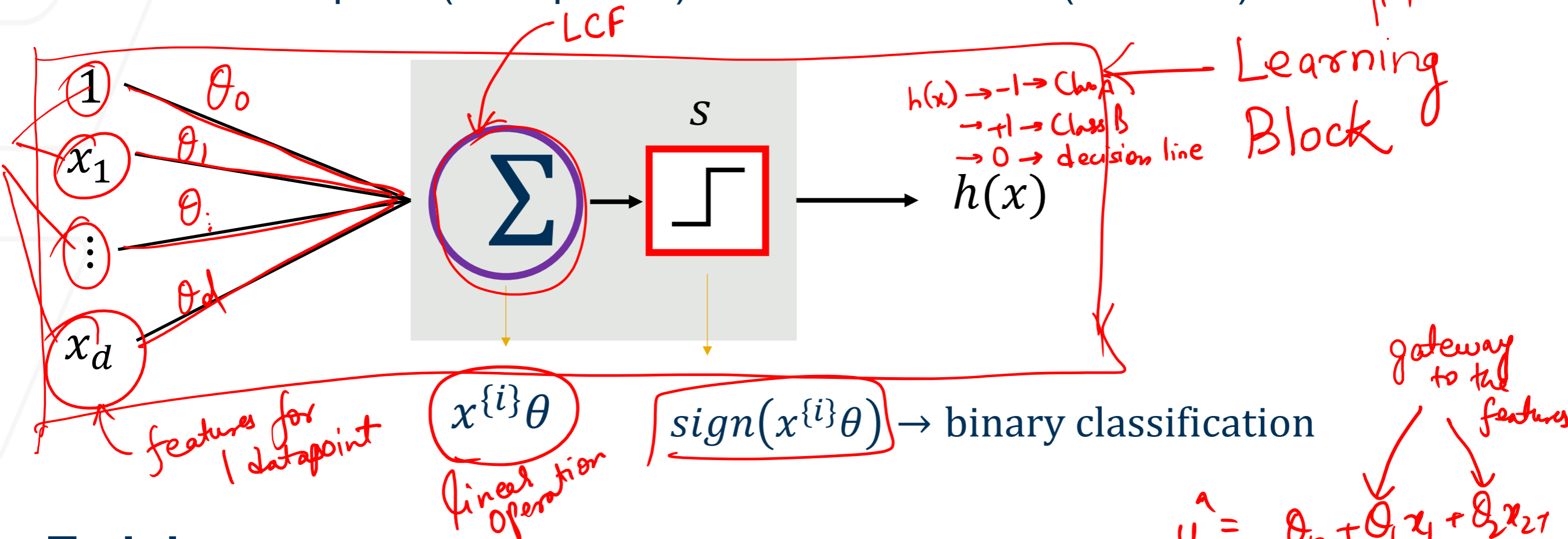
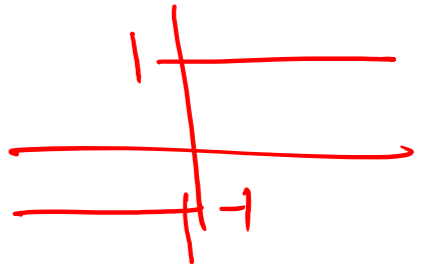
Binary-valued functions are also real-valued  $\pm 1 \in R$

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

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

# Linear regression for classification: Model Structure

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



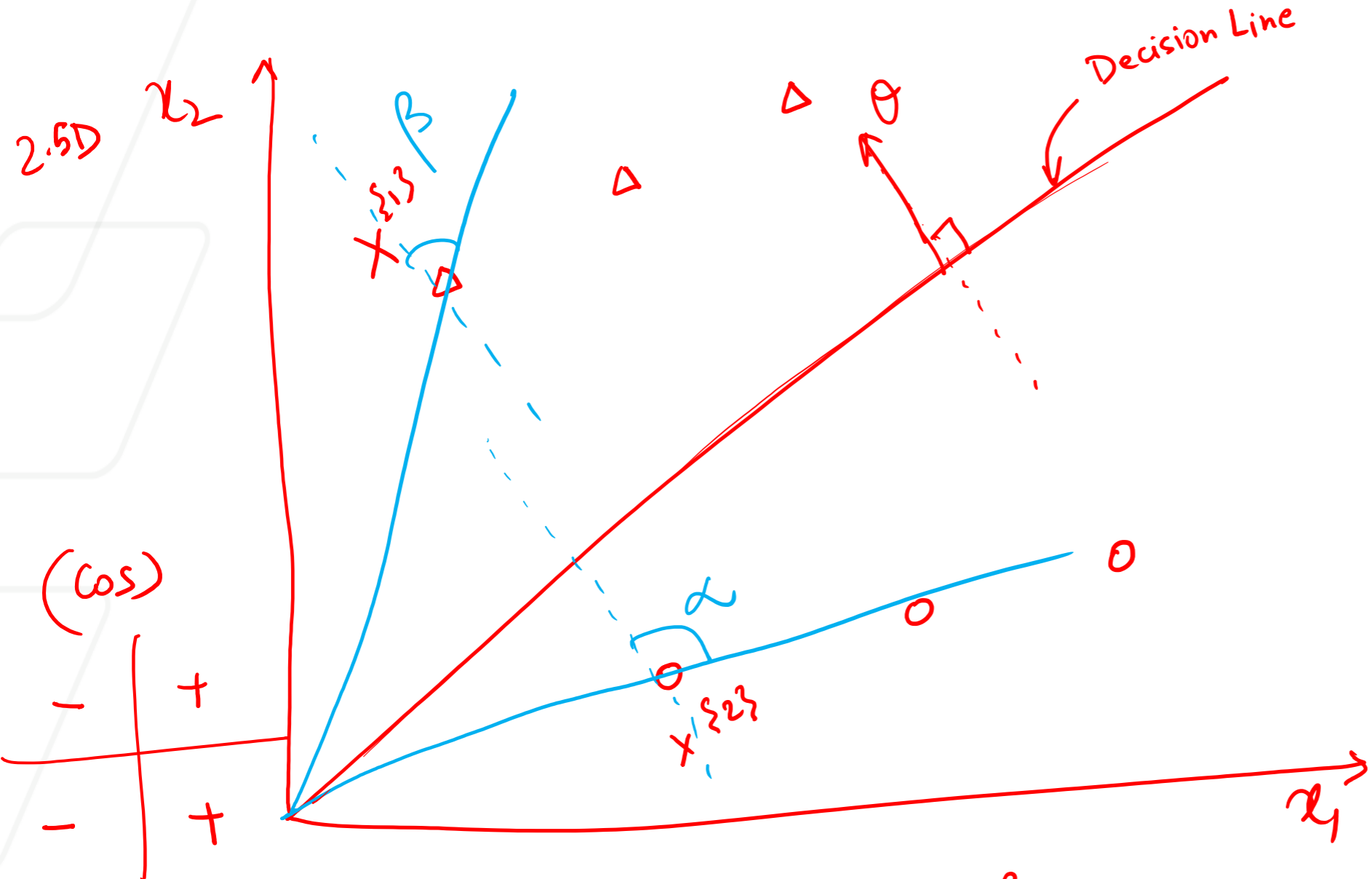
## Training:

1. Take a **training example**  $x^{(i)}, y^{(i)}$
2. Compute  $s = x^{(i)}\theta$
3. Compute the **error**
4. Update parameters

$$\theta \leftarrow \theta - \eta \nabla L$$

## Testing:

1. Take a **test datapoint**  $x^{(i)}, y^{(i)}$
2. Compute  $s = x^{(i)}\theta$
3. Compute  $\text{sign}(S)$
4. Decide classification



$$x^{(1)} \cdot \theta \text{ on the decision line} = 0$$

$$\begin{aligned} \hat{y}_p^{(1)} &= x^{(1)} \cdot \theta \\ &= |x^{(1)}| |\theta| \cos \beta \\ &\quad \uparrow \quad \uparrow \quad \uparrow \\ &\quad \text{true} \quad \text{true} \quad \text{true} \\ &= +1 \rightarrow \text{Class A} \end{aligned}$$

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

$$\begin{aligned} \hat{y}_p^{(2)} &= x^{(2)} \cdot \theta \\ &= |x^{(2)}| \cdot |\theta| \cdot \cos \alpha \\ &\quad \uparrow \quad \uparrow \quad \uparrow \\ &\quad \text{true} \quad \text{true} \quad \text{-ve} \\ &= -1 \leftarrow \text{Class B} \end{aligned}$$

<https://www.math3d.org/UPNV1IQvxt>

1. Use regression to compute

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

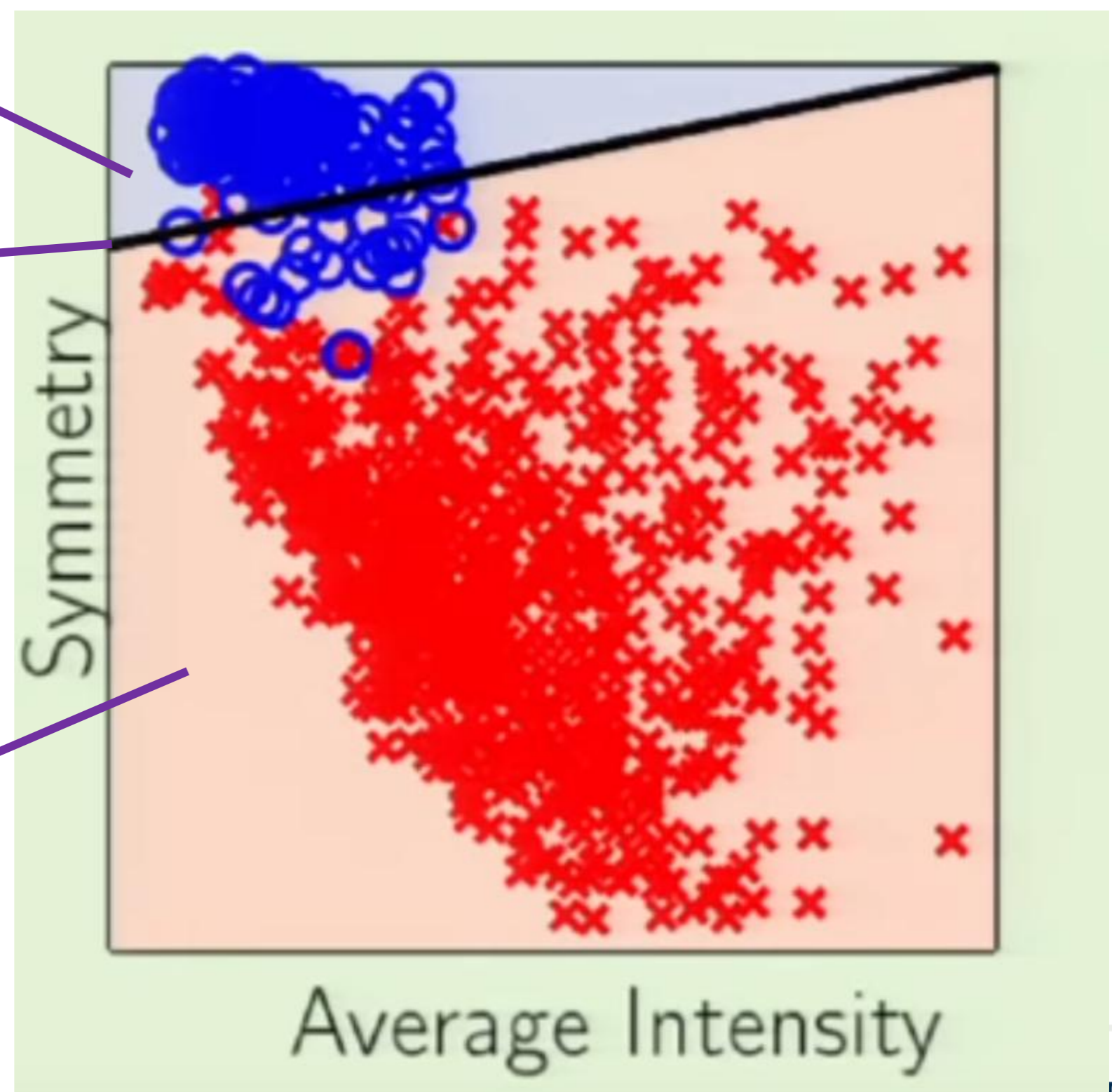
2. That gives a **specific vector**  $\theta$ .

3. Then the decision boundary is uniquely determined on  $y=0$  plane

+1

0

-1



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

# Why Linear Regression is not good for classification

## 1. Output Range Problem

Linear regression produces continuous outputs from  $-\infty$  to  $+\infty$ . But in classification (e.g., predicting 0 or 1), you need probabilities – values between 0 and 1. If you try to interpret the linear output as a probability, you'll often get nonsensical values (like 1.3 or -0.5).

## 2. Poor Decision Boundary for Binary Classes

Suppose you define a rule like “if prediction  $> 0.5 \rightarrow$  class 1, else  $\rightarrow$  class 0.” Linear regression will fit a line minimizing squared error, not a boundary that best separates classes. That means outliers or overlapping regions can drastically skew the boundary. In short: it's not optimizing for classification accuracy, but for minimizing distance between predicted and actual numeric labels (0 or 1).

## 3. Sensitivity to Outliers

Because linear regression uses mean squared error (MSE), large errors (caused by mislabeled or extreme data points) have a big impact. This distorts the model's predictions and decision line.

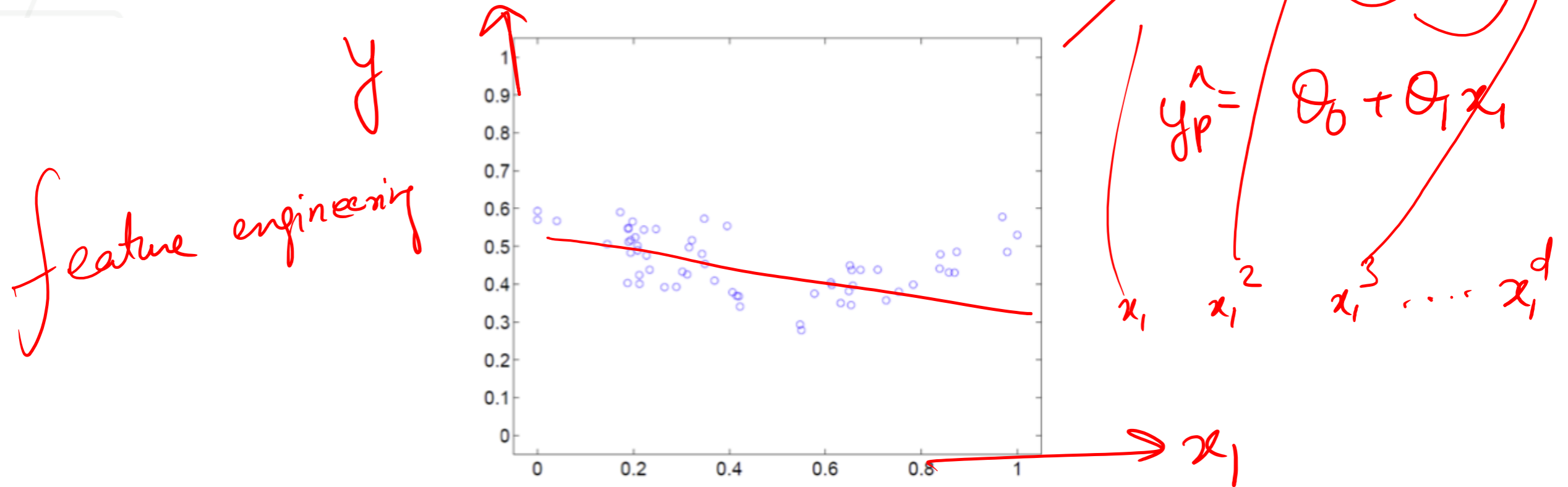
## 4. Non-linearity of Class Boundaries

Many classification problems have non-linear separations between classes. Linear regression can only produce a straight-line boundary, so it fails when data isn't linearly separable.

# Outline

- Supervised Learning
- Linear Regression
- Extension 

# Extension to Higher-Order Regression



- Want to fit a polynomial regression model

*Creating features*

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

- $z = \{1, x_1, x_1^2, \dots, x_1^d\} \in R^d$  and  $\theta = (\theta_0, \theta_1, \theta_2, \dots, \theta_d)^T$

$$y = z\theta$$

# Least Mean Square Still Works the Same

$$x = [1 \quad x_1]$$
$$Z = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_1^d & \dots & \dots & \dots \end{bmatrix}$$

- Given  $n$  data points, find  $\theta$  that minimizes the mean square error

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

- 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

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

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

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

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

# What is happening in polynomial regression?

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

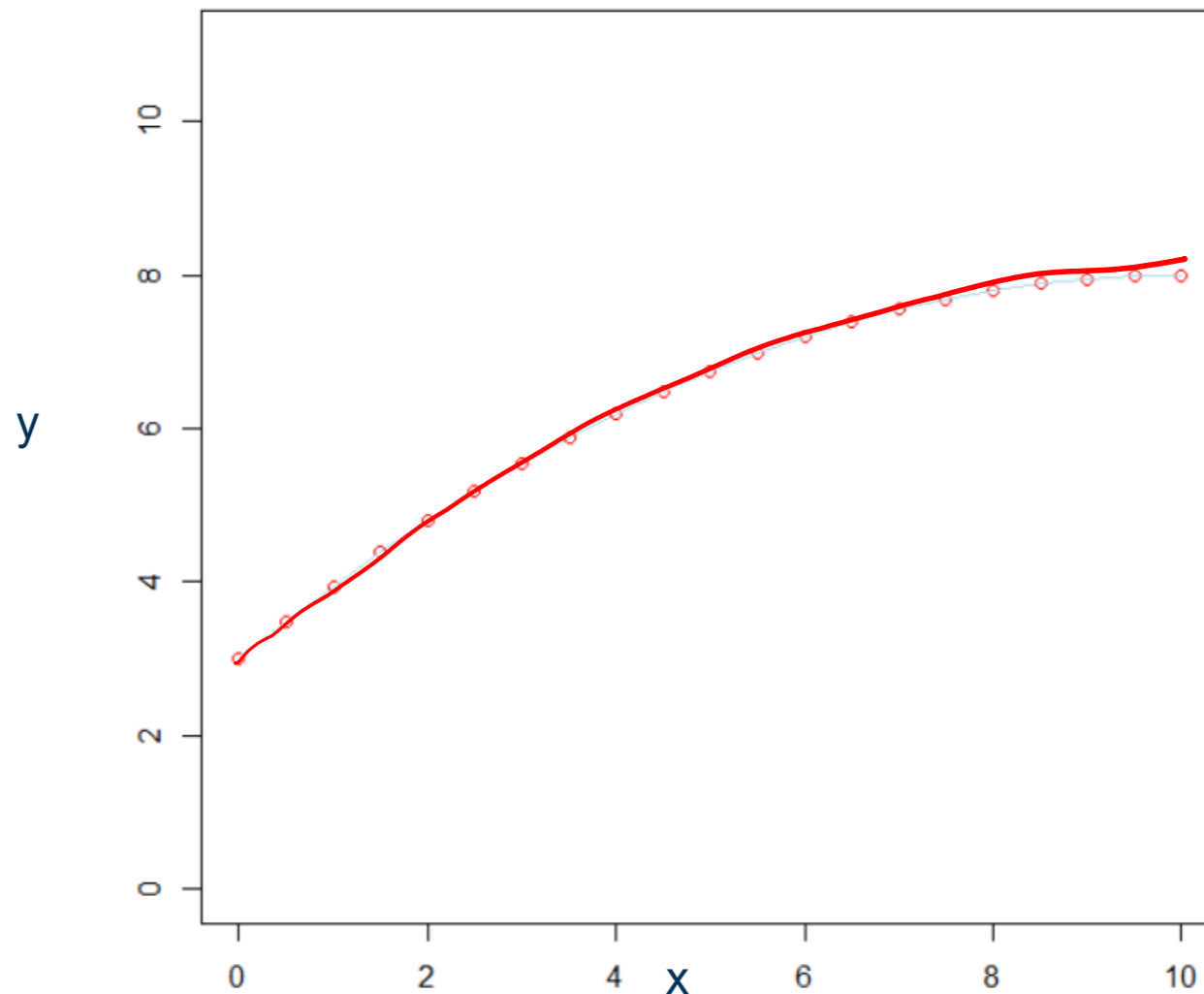
$$y = [3, 3.4875, 3.95, \dots, 7.98, 8]$$

*a*

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

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

*1 additional dimension here*

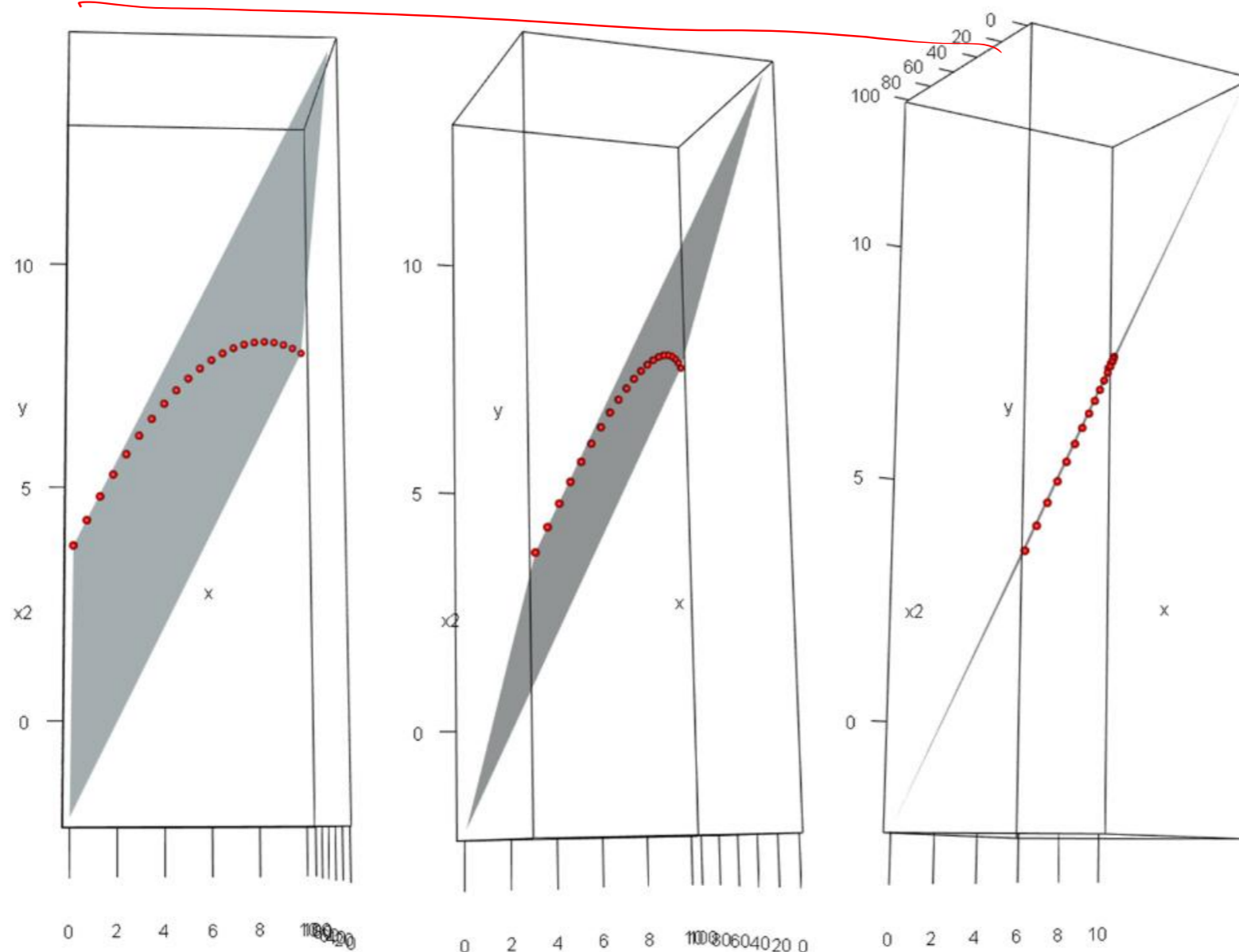


RMSE=0

# Let's add to the feature space

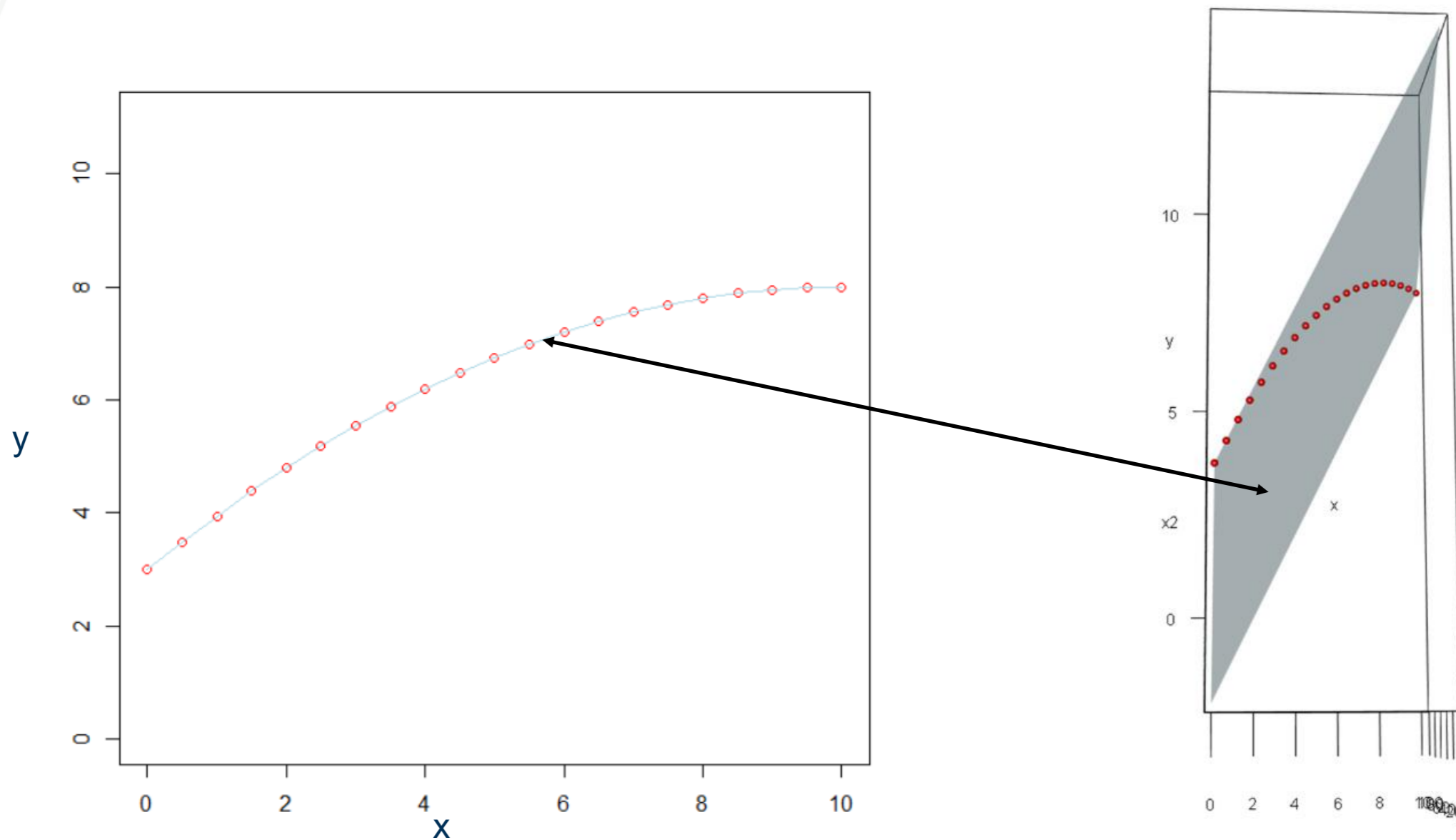
$$x_1 = [0, 0.5, 1, \dots, 9.5, 10] \quad x_1^2 = [0, 0.25, 1, \dots, 90.25, 100]$$

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



A model can look nonlinear in the original input space but still be linear in the parameters.

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



This explains why we can model complex nonlinear relationships using linear regression with transformed features.

$X = [1 \quad x_1]$  1 feature in  $x$  space

$Z = [1 \quad x_1 \quad x_1^2 \quad x_1^3 \quad \dots \quad x_1^d]$   $d$  features in  $z$  space

As we move from  $x$  to  $z$  space

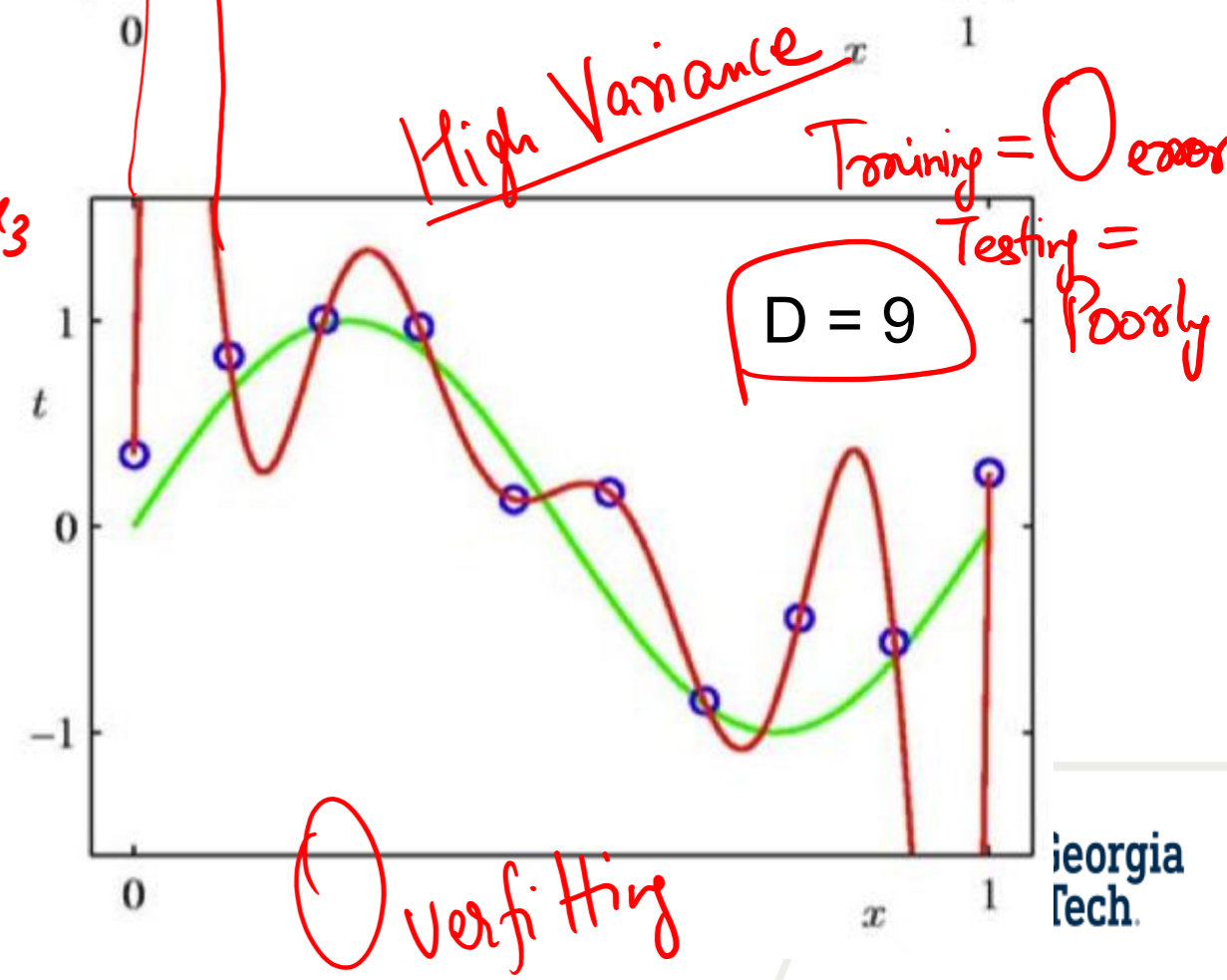
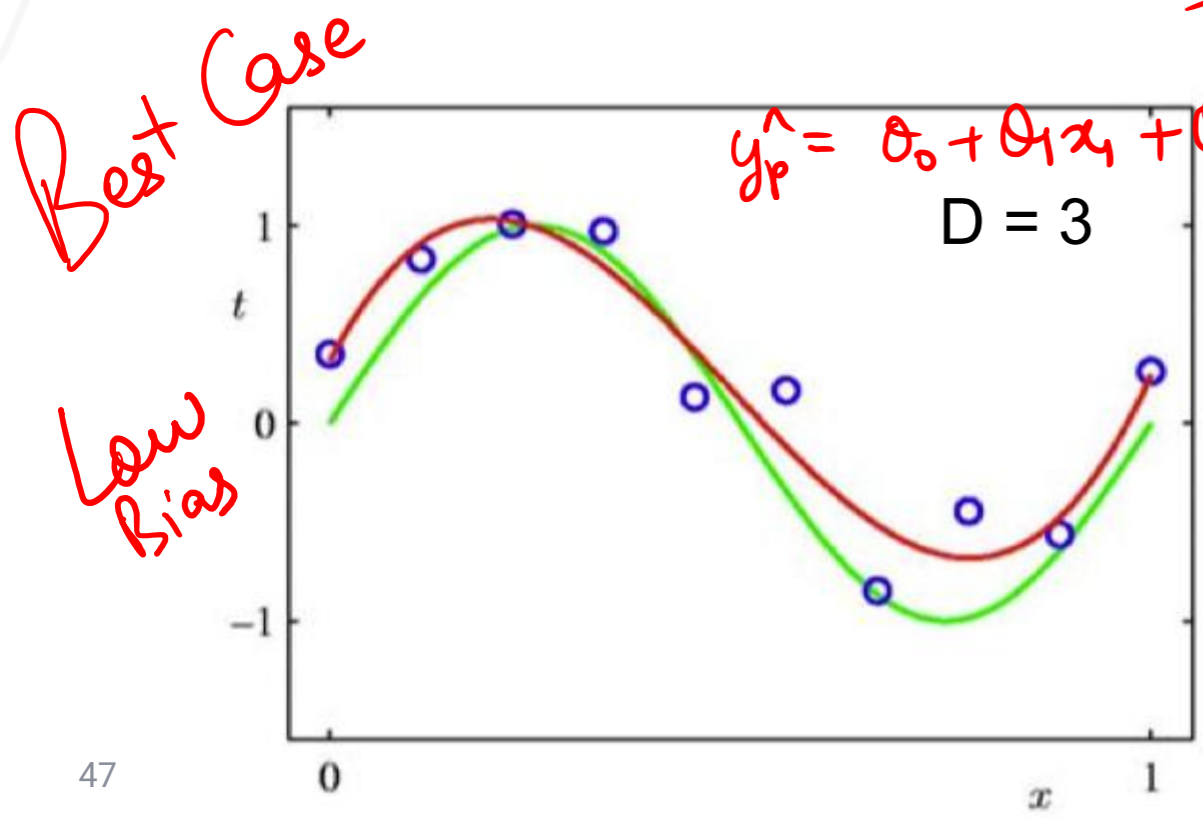
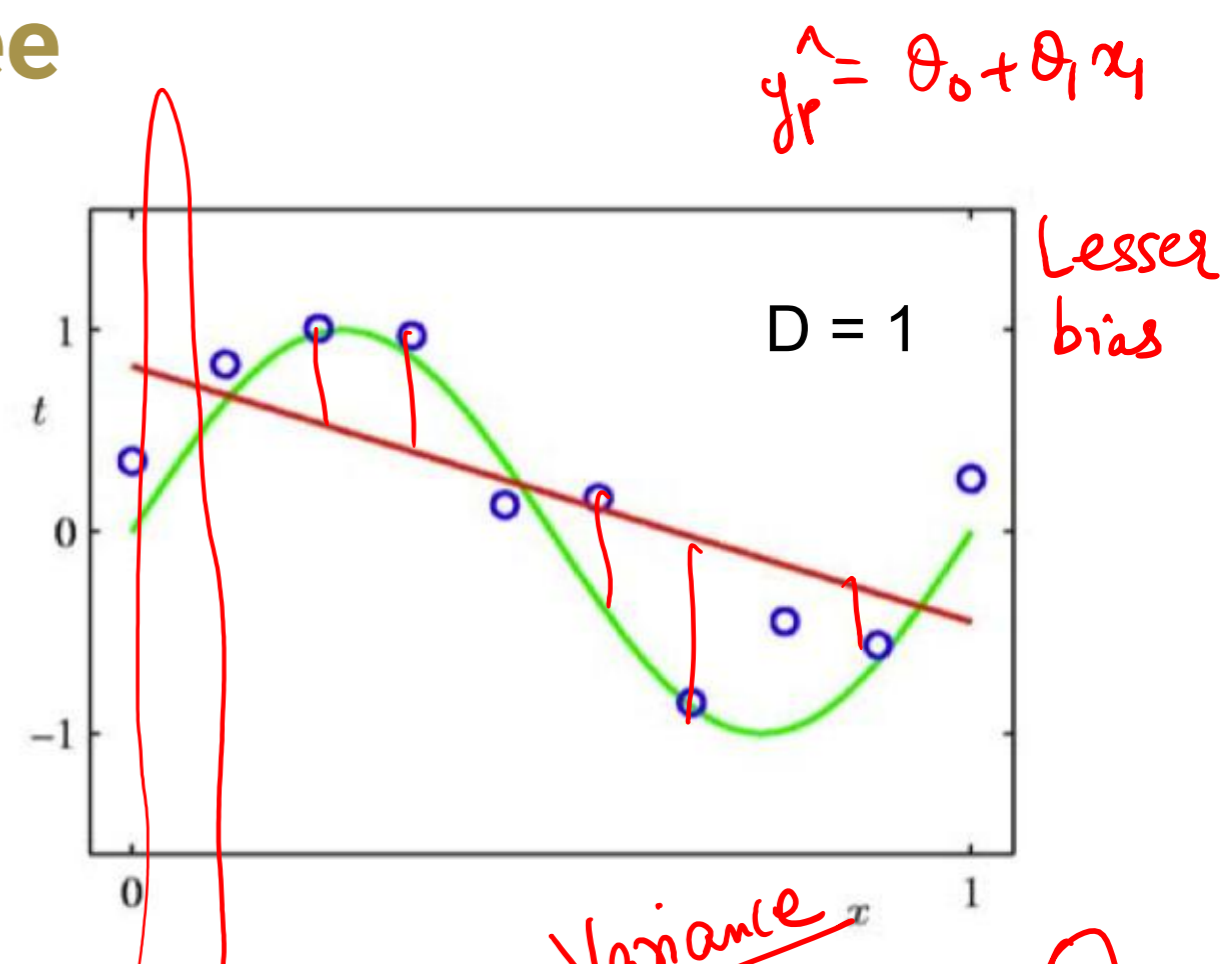
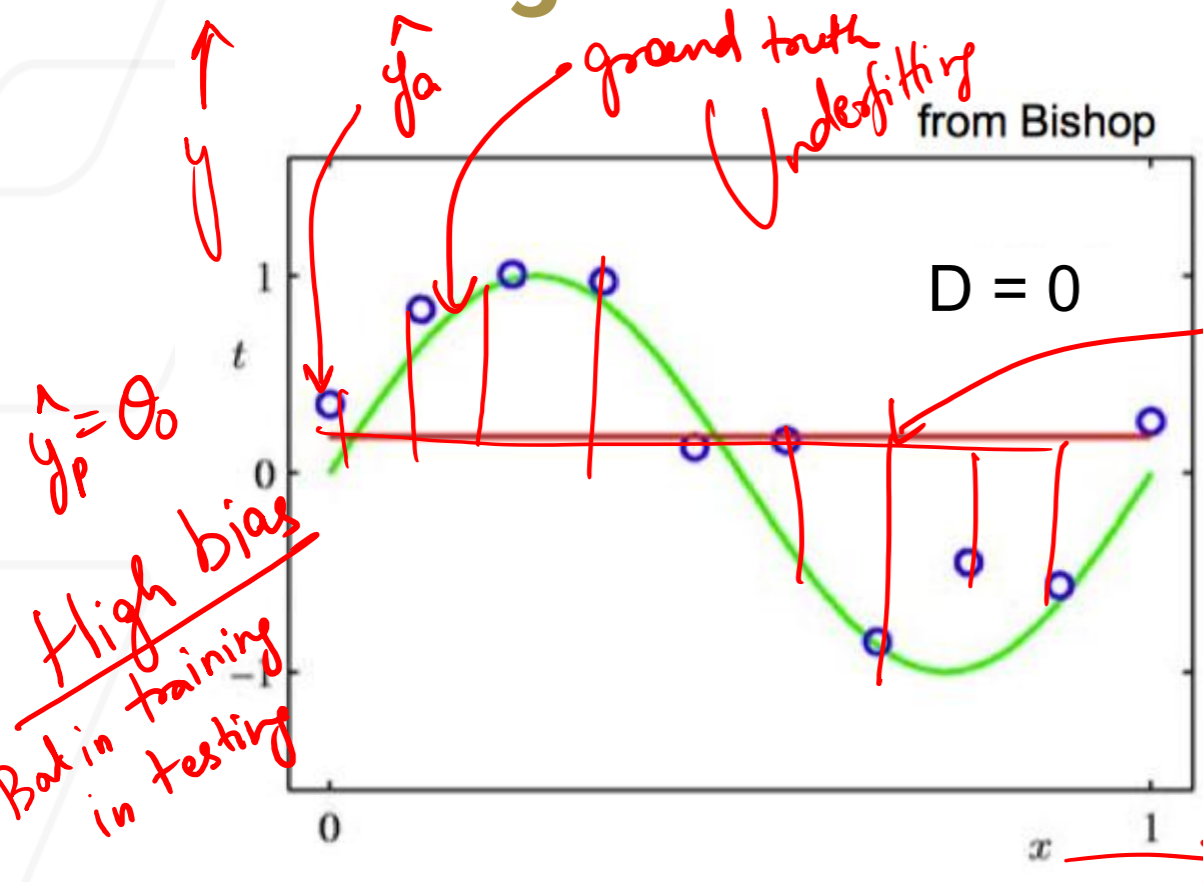
inc. in features

inc. in parameters

inc. in complexity of model

inc. in chances of  
Overfitting

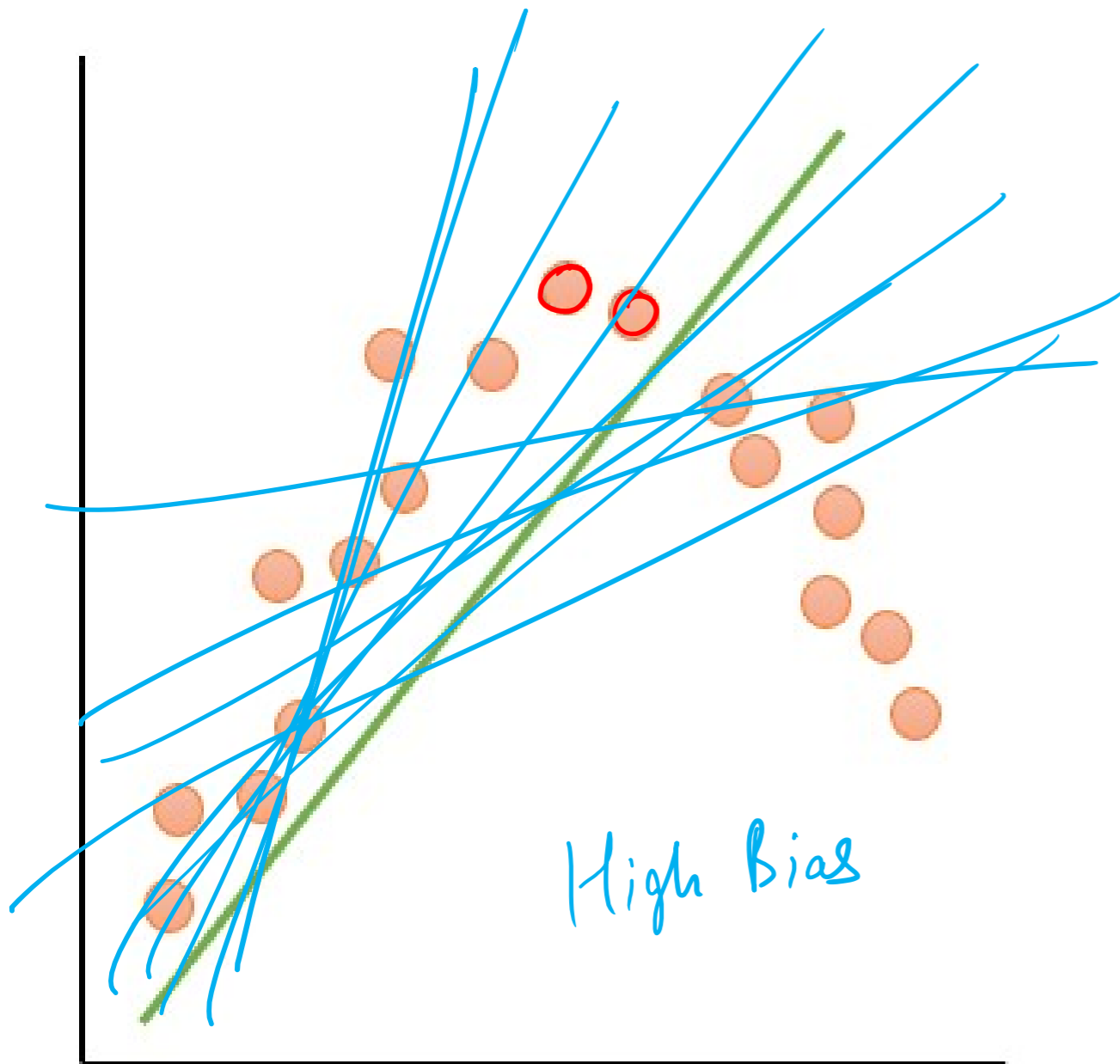
# Increasing the Maximal Degree



We create 10 datasets from training dataset

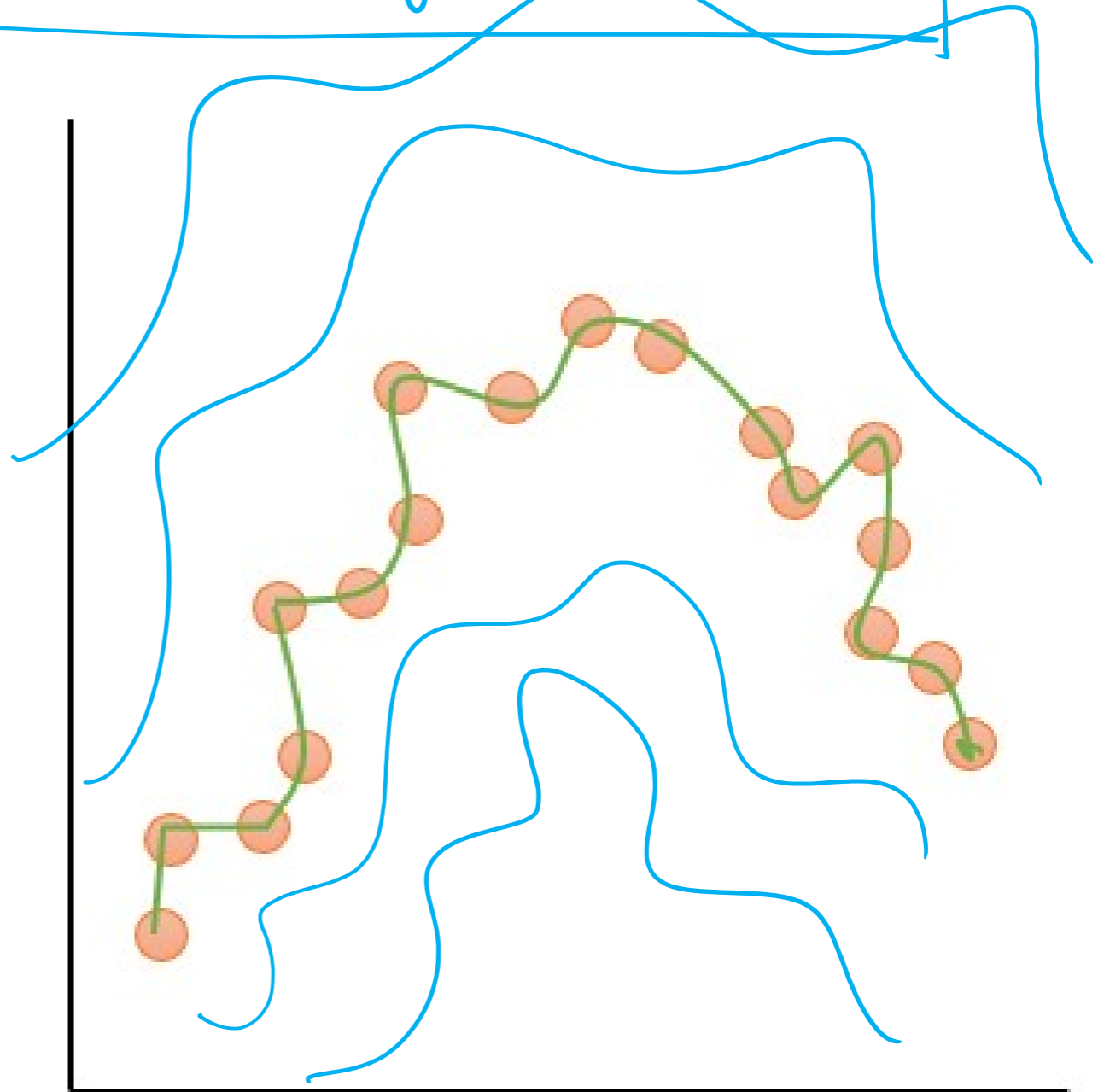
# Concept of Bias

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



High Bias

Low variance



Low bias  
High Variance

# Concept of Variance

- **Variance** measures how much the model's predictions **change** if you train it on a different sample of same data.
- A **high-variance** model is *very sensitive* to small changes in the training set.
  - Even a slightly different dataset produces a very different fitted curve.
- Overfitting with high maximal degree leads to high variance

# Bias-Variance Trade off

[Animation](#)

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

Why bias & variance both should be low?

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

$$(y_a - y_p)^2 = \underbrace{(y_a - E[y_p])}_a + \underbrace{(E[y_p] - y_p)}_b$$

$$= \underbrace{(y_a - E[y_p])^2}_{a^2} + \underbrace{(E[y_p] - y_p)^2}_{b^2} + \underbrace{2(y_a - E[y_p])(E[y_p] - y_p)}_{2ab}$$

Annotations:   
 -  $(y_a - E[y_p])$  is labeled "scalar"   
 -  $E[y_p]$  is labeled "scalar"   
 -  $(E[y_p] - y_p)$  is labeled "vector"   
 -  $2(y_a - E[y_p])(E[y_p] - y_p)$  is labeled "0" and "scalar"   
 -  $E[y_p]$  is labeled "scalar"   
 -  $y_p$  is labeled "vector"

$$E[\text{scalar}] = \text{scalar}$$

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

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

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

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

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

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

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

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

# Bias-Variance Trade off

[Animation](#)

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

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

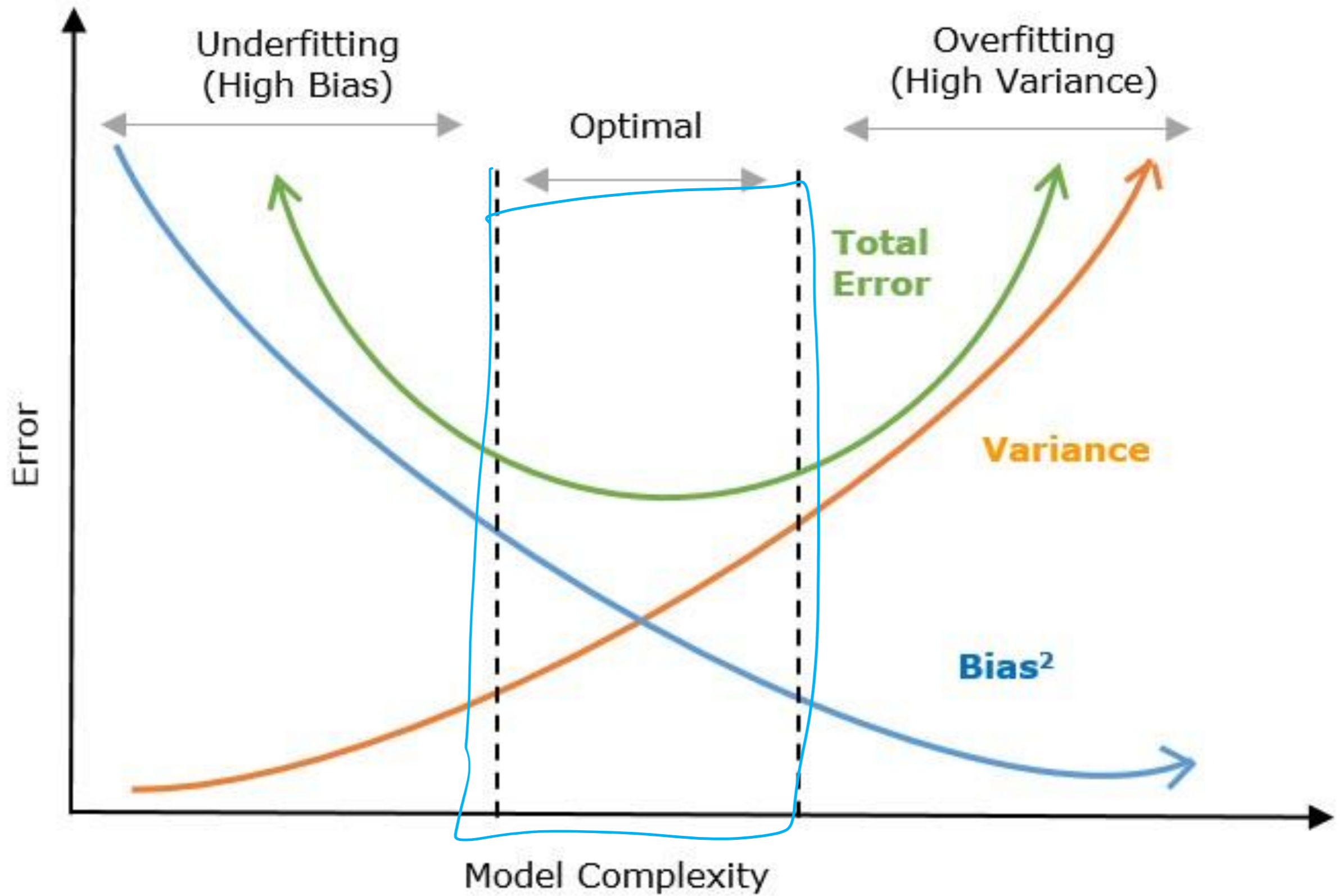
$$(y_a - y_p)^2 = (y_a - E[y_p] + E[y_p] - y_p)^2$$

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

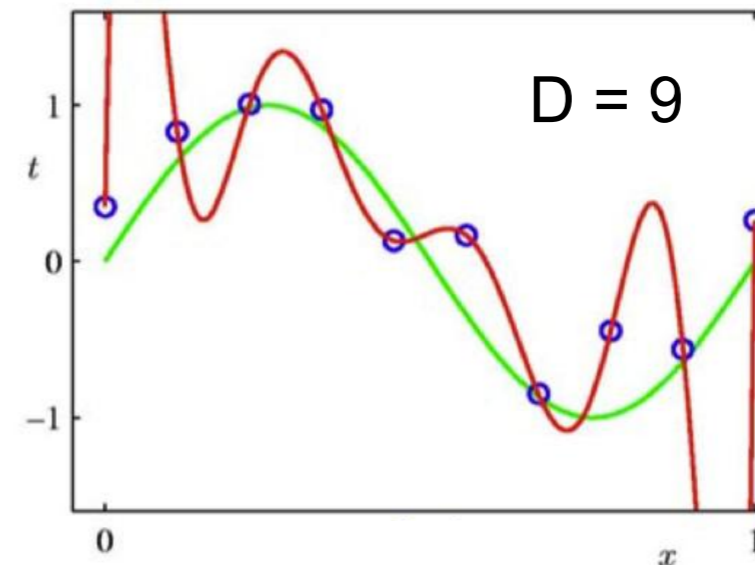
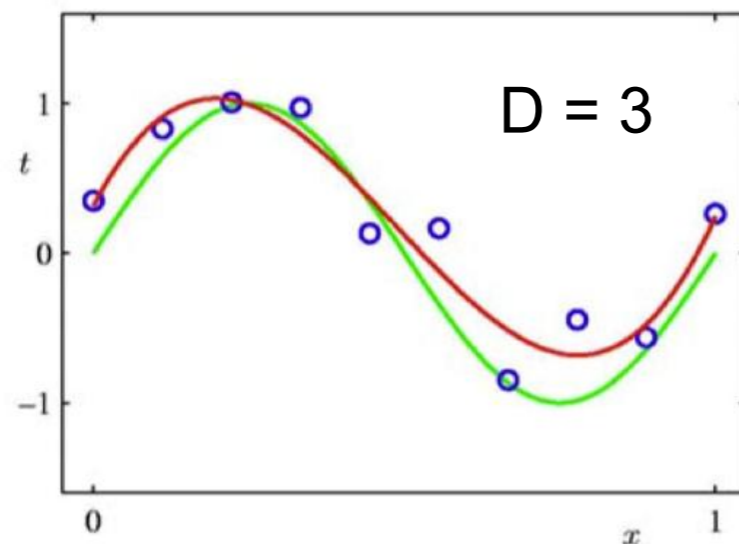
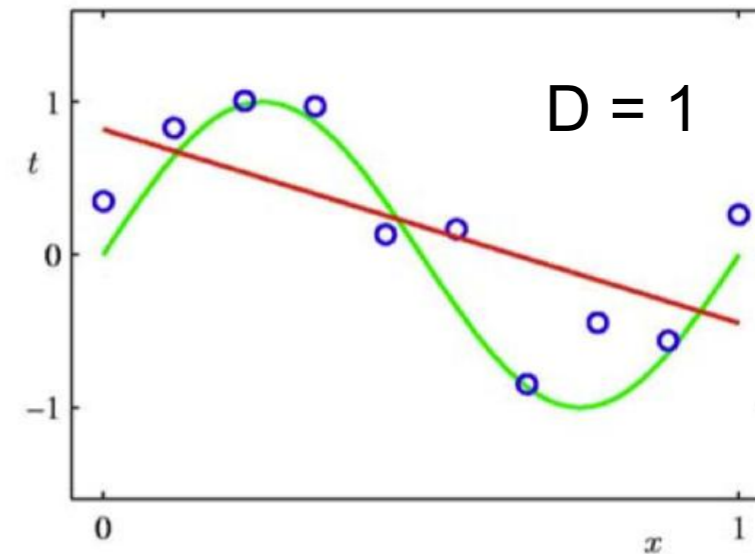
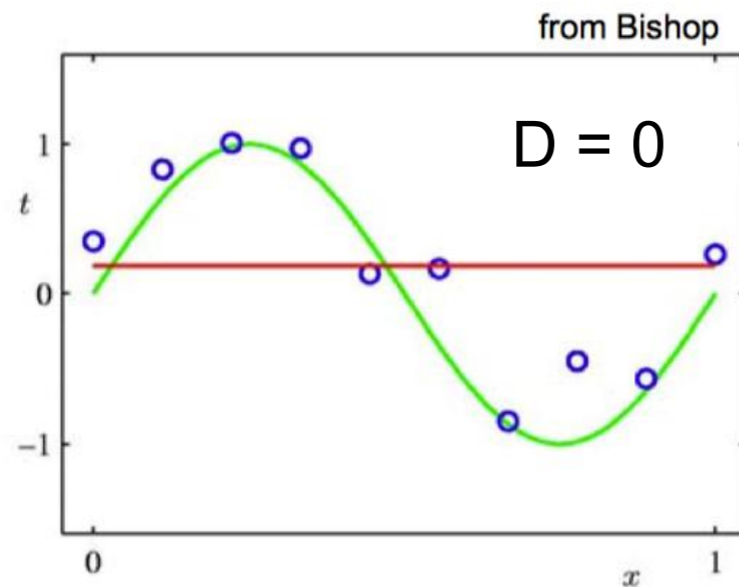
$$E[(y_a - y_p)^2] = \underbrace{(y_a - E[y_p])^2}_{\text{Bias}} + \underbrace{E[(E[y_p] - y_p)^2]}_{\text{Variance}}$$

$$= [\mathbf{Bias}]^2 + \mathbf{Variance}$$

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



# Which One is Better?



- Can we increase the maximal polynomial degree to very large, such that the curve passes through all training points?

# Take-Home Messages

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

# Quick Knowledge Check

1. Which of the following is added when extending linear regression to higher-order polynomial regression?  
A. More data points   B. Higher-degree features   C. More regularization   D. Less variance
2. In polynomial regression, what happens when we increase the maximal degree  $d$ ?  
A. The model becomes simpler   B. The model becomes more flexible  
C. The model underfits   D. The MSE always increases
3. A high-degree polynomial that fits all training points perfectly likely suffers from:  
A. Underfitting   B. Overfitting   C. Low variance   D. Low flexibility
4. Why is overfitting considered a high-variance problem?  
A. The model changes significantly with new data   B. It ignores noise  
C. It has high bias   D. It's too simple to capture the trend
5. Which of the following best describes **bias** in the bias–variance tradeoff?  
A. Variability across datasets   B. Sensitivity to noise   C. Systematic error due to model simplicity   D. Random error due to data splits

# Quick Knowledge Check

1. What does high variance typically lead to?  
A. Stable predictions  B. Poor generalization to new data  C. Low flexibility  D. Low training accuracy
2. In the bias–variance tradeoff, increasing model complexity generally:  
A. Increases bias and decreases variance  B. Decreases bias and increases variance  C. Decreases both bias and variance  D. Increases both bias and variance
3. Which scenario indicates underfitting?  
A. High training error, high test error  B. Low training error, high test error  C. Low training error, low test error  D. High training accuracy, low test accuracy

Good!