EE2211

Learning is any process by which a system improves performance from experience.

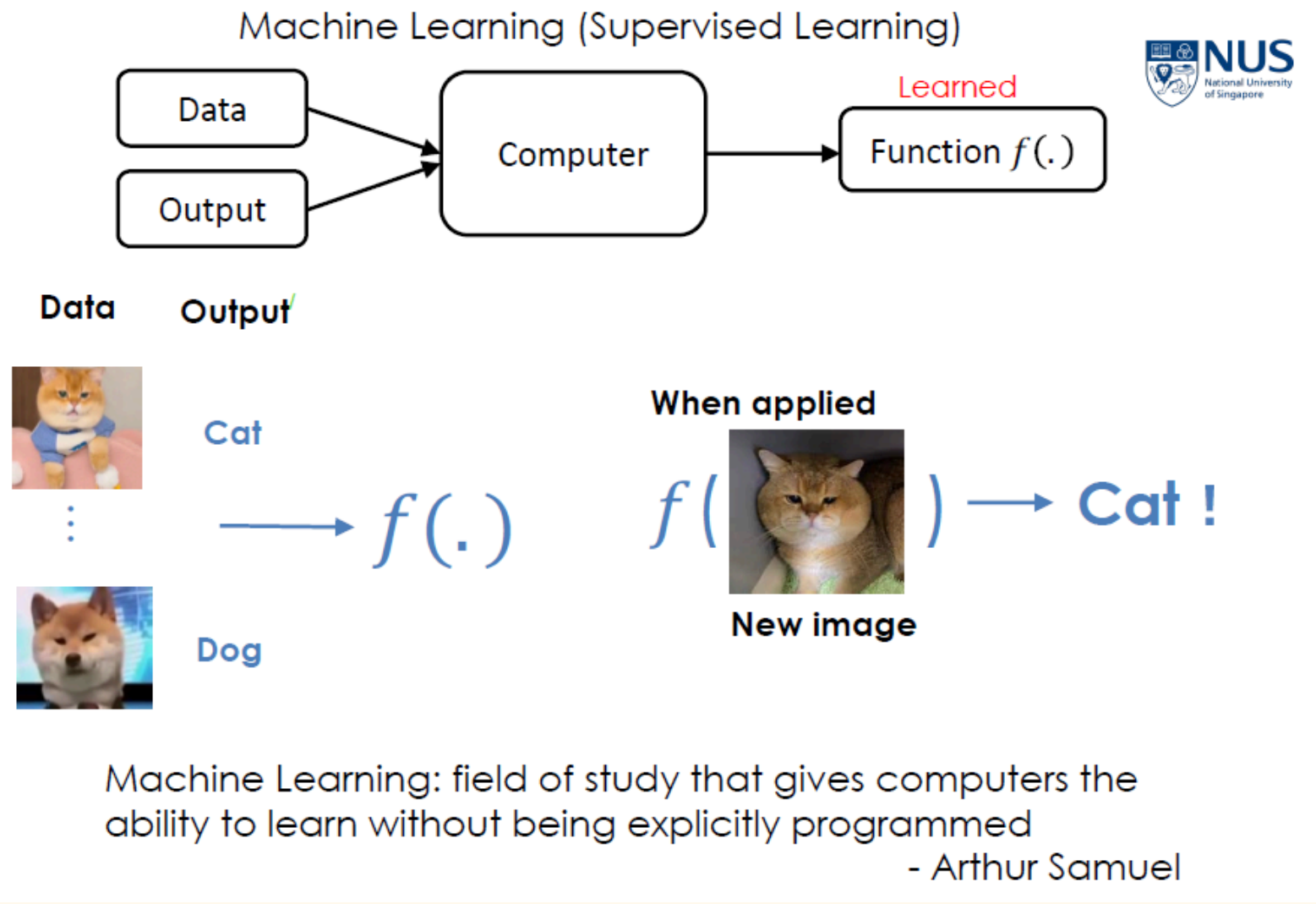
- Herbert Simon

A computer program is said to learn

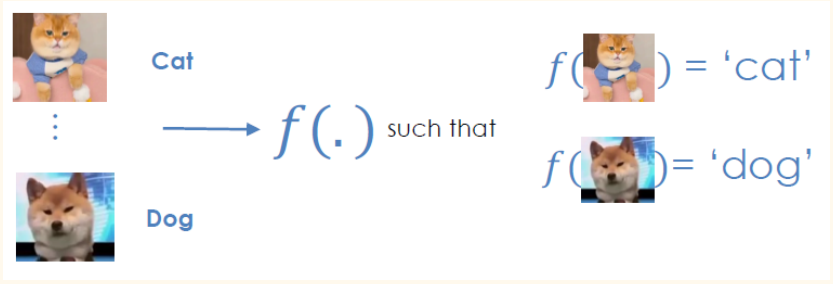
- from ***experience E***
- with respect to some class of ***tasks T***
- and ***performance measure P***,

if its performance at tasks in T, as measured by P, improves with experience E.

- Tom Mitchell

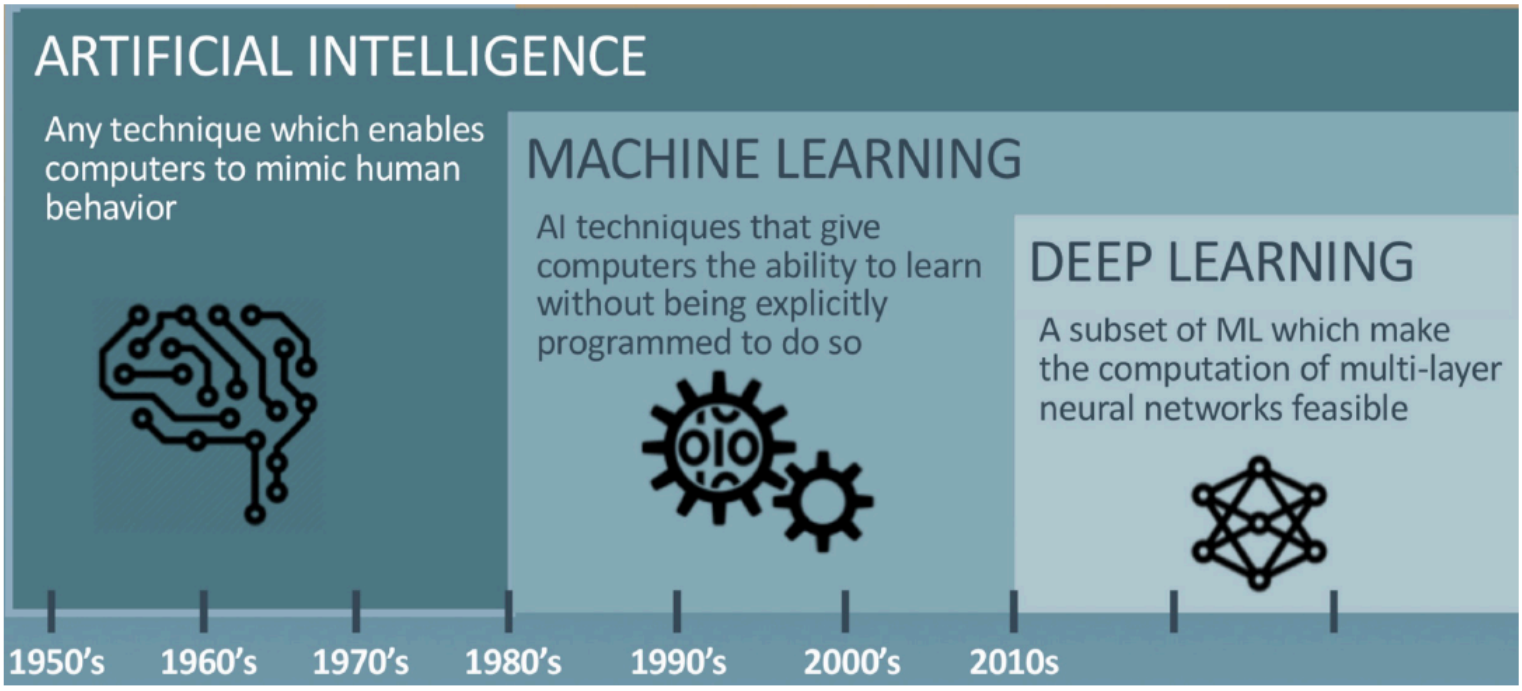


- Computer with machine learning algorithm
- Output also called label
- Data output pair
- the function is said to be learned



  -
  - f(data) = output
- Can predict after learnt

AI, Machine Learning, and Deep Learning



Example of AI but not ML: Deductive Reasoning

NUS is in Singapore, Singapore is in Asia -> NUS is in Asia

- ML can be used to achieve AI
- Deep learning is a subset of ML with specific architecture called neural network
- Deductive reasoning
    - If given A is in B and B is in C, then can reason that A is in C

When do we need machine learning?

Lack of human expertise
(Navigating on Mars)

Involves huge amount of data
(Genomics)

Learning is not always useful:

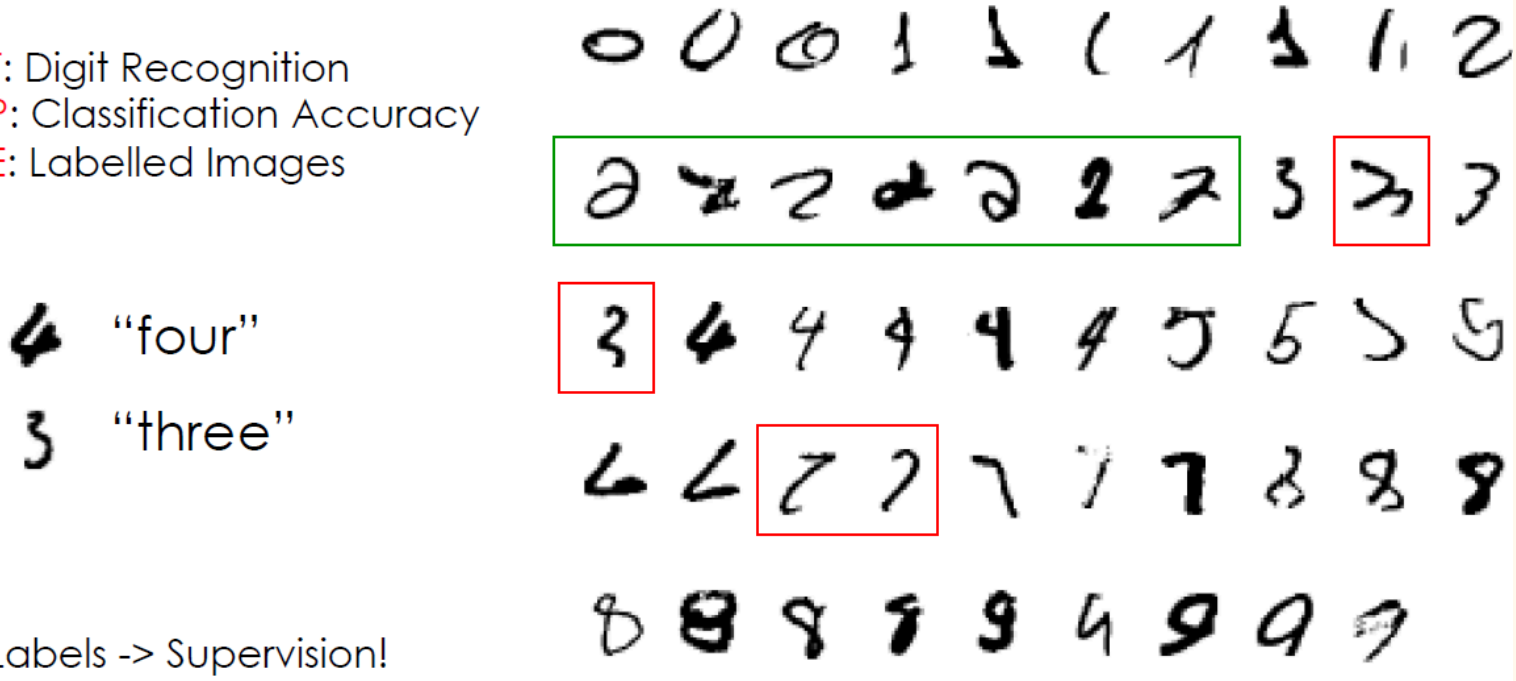
No need to "learn" to calculate payroll!

My Salary = Days_of_work * Daily Salary + Bonus

Application of Machine Learning

Task T, Performance P, Experience E

T: Digit Recognition
P: Classification Accuracy
E: Labelled Images



- used by USPS, United States Postal Service
- By giving the labelled images the supervision is given to the model

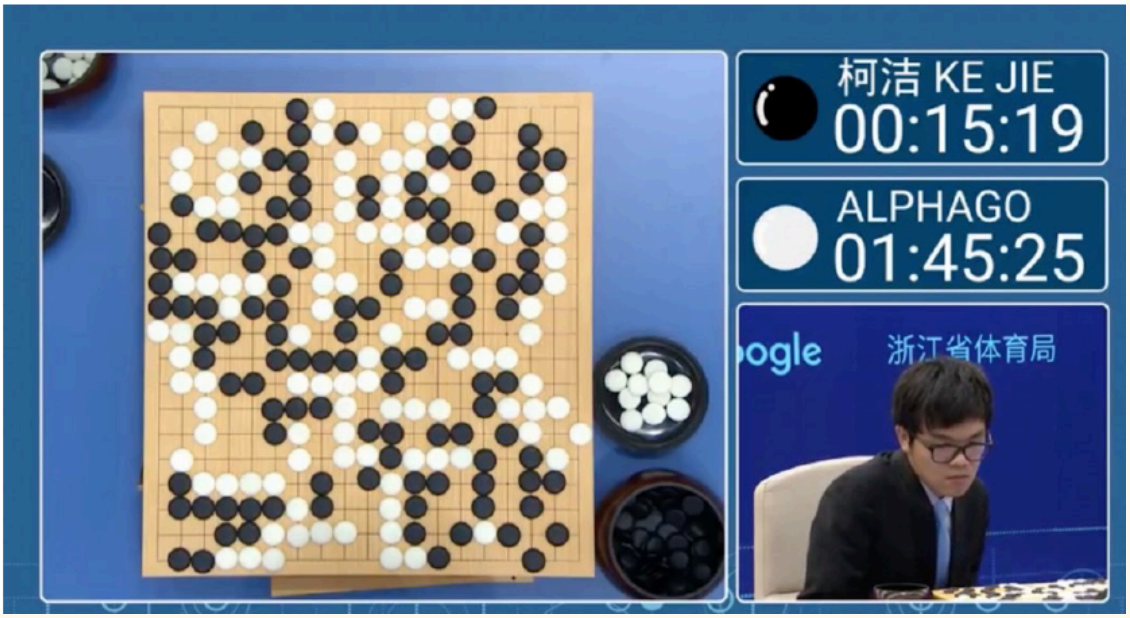
Task T, Performance P, Experience E

T: Email Categorization
P: Classification Accuracy
E: Email Data, Some Labelled

- Only some email data is labelled, user does not label all mails only some

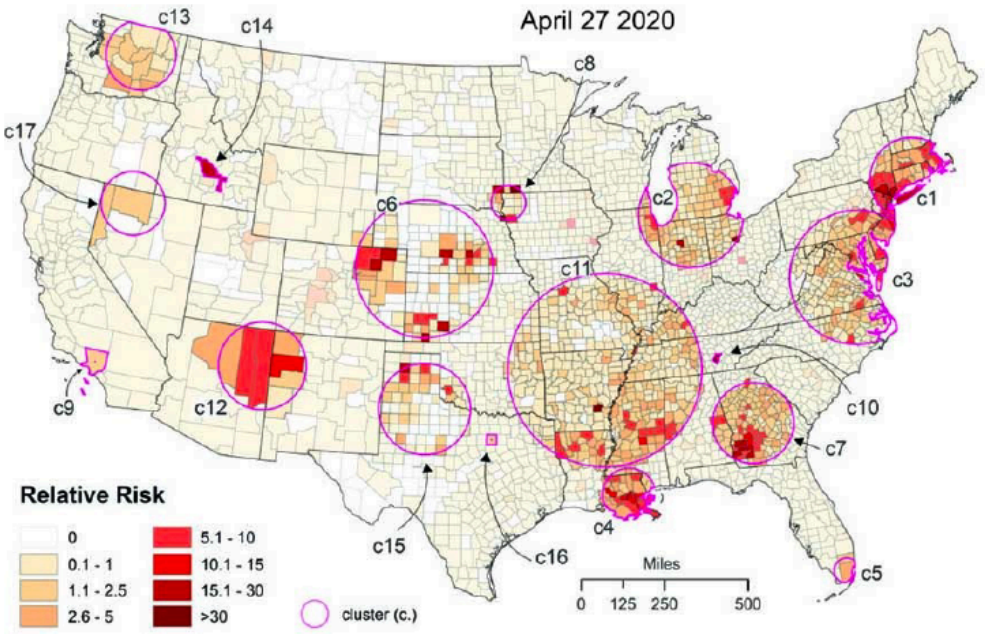
Task T, Performance P, Experience E

T: Playing Go Game
P: Chances of Winning
E: Records of Past Games



Task T, Performance P, Experience E

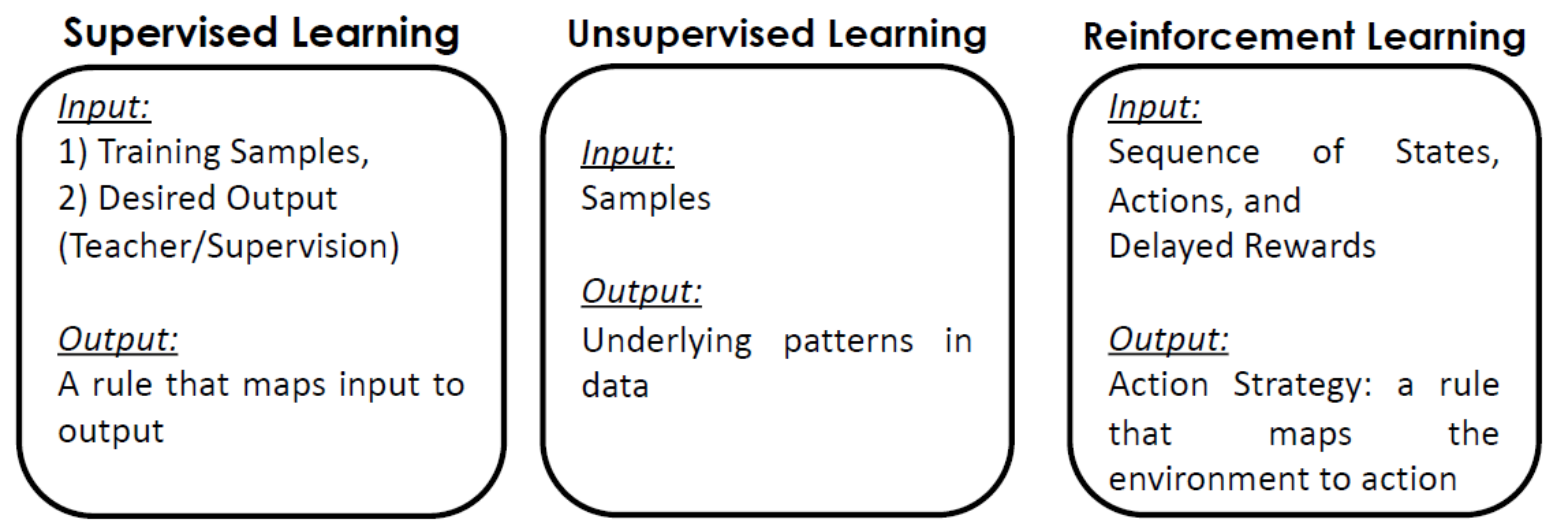
T: Identifying Covid-19 Clusters
P: Small Internal Distances
Larger External Distances
E: Records of Patients



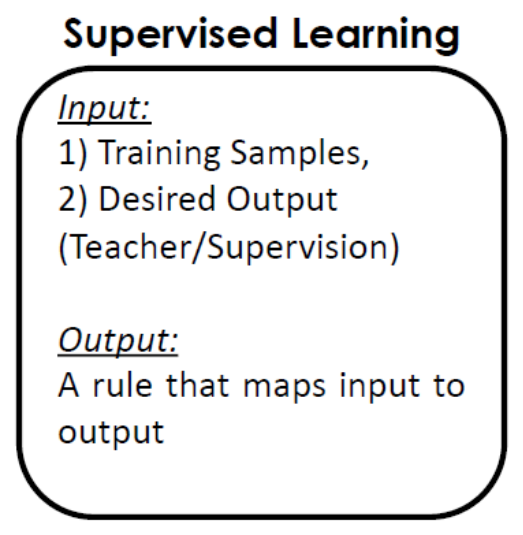
- Since model is finding out clusters

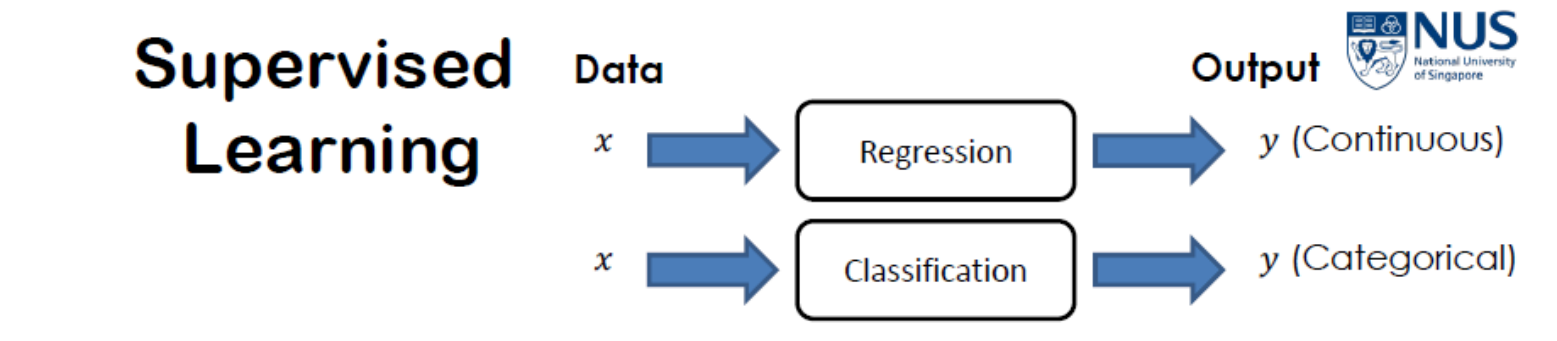
    - Between cluster should be big, picking two counties far apart from each other should belong in different cluster
        - small internal distance
    - But between neighbouring county should be in one cluster, so picking two counties close to each other they should belong in the same cluster
        - larger external distance
- Cannot be supervised as only have records of patients not cluster

Types of Machine Learning



Supervised Learning





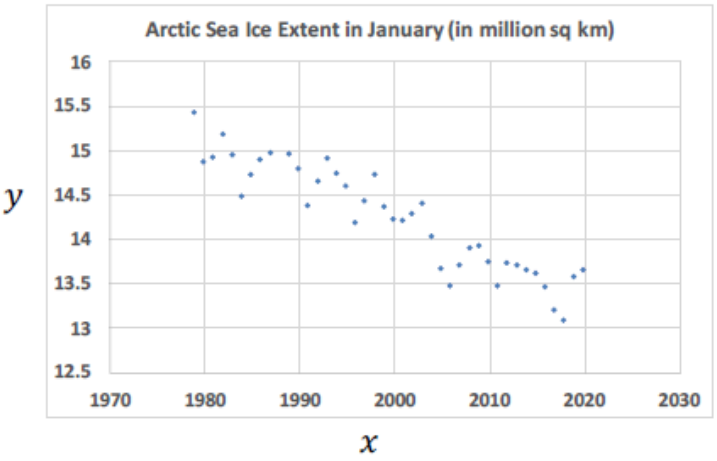
- If y is continuous, the type of learning is called regression

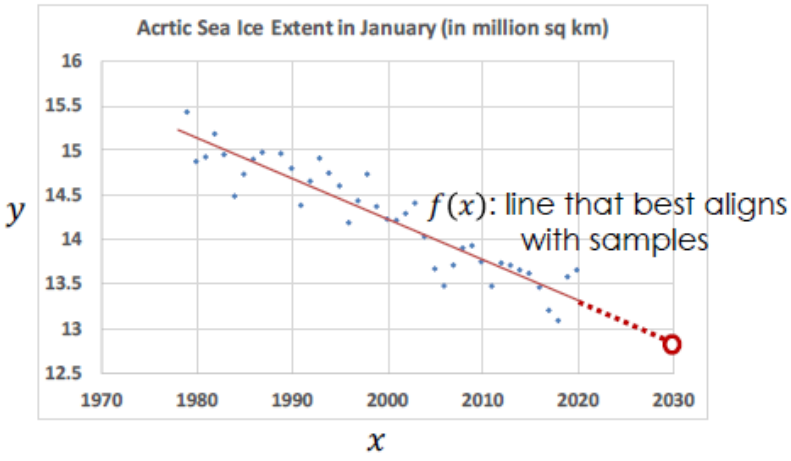
- if y is categorical, the type of learning is called classification

Regression

Regression

- Given $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$
- Learn a function $f(\mathbf{x})$ to predict real-valued y given $\mathbf{x}$





- if it's a line, then linear regression. If not line, regression but not linear
- to predict to the future
- Regression also belongs to the class of supervised learning methods.

Regression is just like classification except that y_i are no longer restricted to belong to a finite set. Rather it can take on uncountably[1] many values. In the case of regression, we typically do not say that y_i is the label.

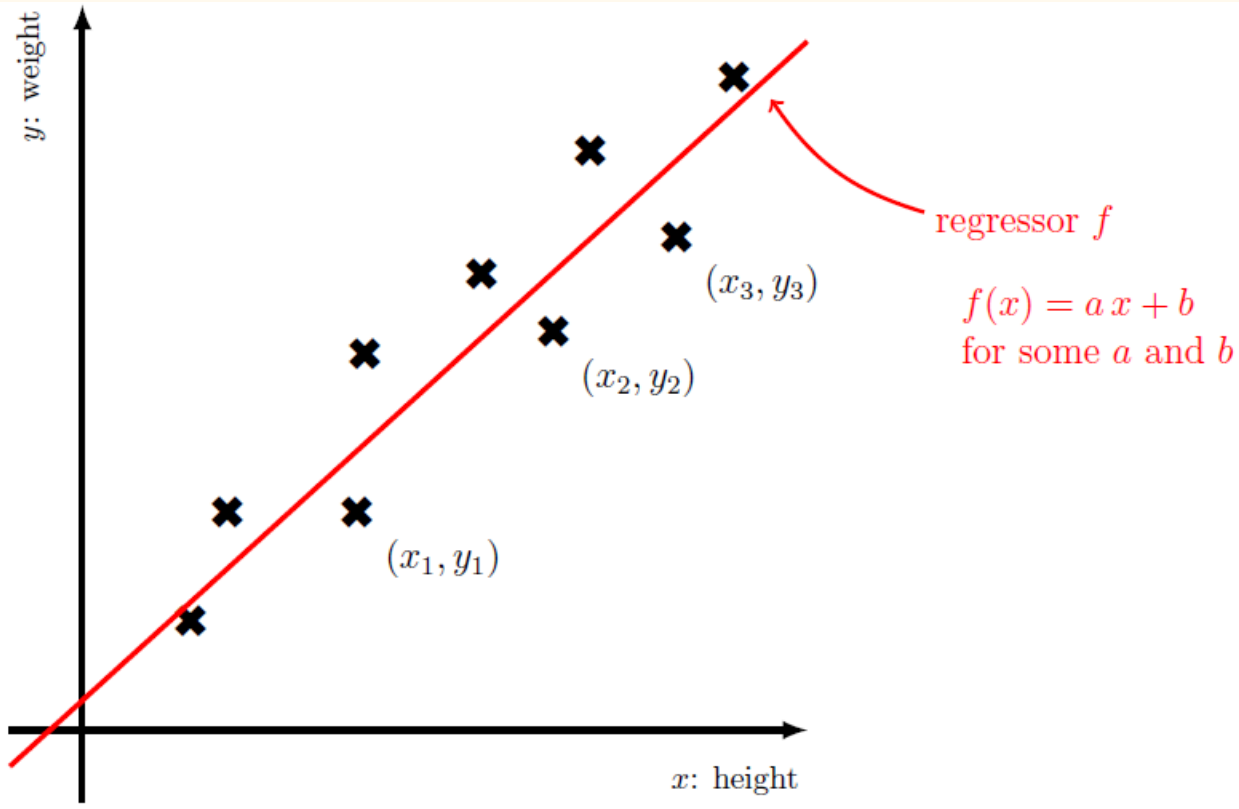


Figure 1.2: Height and weight regression example

Rather we use the term *target variable* or *outcome variable* or *dependent variable* to refer to the y_i's. For instance y_i could take any value in the interval $[0, 1]$ (which is uncountable) or it could take values in $\mathbb{R}$. In either case, the data examples in the dataset $\mathcal{D}$, denoted as $(\mathbf{x}_i, y_i)$ belong to $\mathbb{R}^d \times \mathbb{R}$, which means that each feature vector or training sample $\mathbf{x}_i \in \mathbb{R}^d$ (d-dimensional Euclidean space) and each target variable $y_i \in \mathbb{R}$ is a real number. We wish to learn a *regressor* $f : \mathbb{R}^d \to \mathbb{R}$ which is a function that brings us from the feature space to the set of real numbers. The regressor f should be designed such that given a new feature vector $\mathbf{x}'$, its corresponding target value y' is "well predicted", in some precise mathematical sense, by $f(\mathbf{x}')$.

- Say each feature vector $x_i \in \mathbb{R}$ is scalar (so $d = 1$) and represents the height of a student. The target variable y_i represents the weight of the same student. Given the data of $m = 600$ students in EE2211 captured in the dataset $\mathcal{D} = \{(x_i, y_i) : 1 \leq i \leq 600\}$, I am tasked to learn a regressor $f : \mathbb{R} \to \mathbb{R}$ so that I can use the height of Alice x' in a new class EE9999 to predict her weight $y' = f(x')$. See Fig. 1.2.
- Say each feature vector $\mathbf{x}_i \in \mathbb{R}^3$ is a 3-dimensional vector and $x_{i,1}$ represents the air pressure at location i, $x_{i,2}$ represents the amount of rainfall at location i and $x_{i,3}$ represents the "amount of greenery" at location i. Each y_i corresponds to the temperature at location i. The temperature can take on uncountably many values—it is a real number. Given the dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i) : 1 \leq i \leq m\}$, we would like to learn a regressor $f : \mathbb{R}^3 \to \mathbb{R}$ such that if I give you the feature vector of new location $\mathbf{x}'$, you can tell me the temperature $y' = f(\mathbf{x}')$.

Classification

The most canonical task is classification. This problem takes the following form. We have a *dataset* $\mathcal{D}$ which consists of a certain number m of *data examples* $(\mathbf{x}_i, y_i), i = 1, \ldots, m$. Each data example $(\mathbf{x}_i, y_i)$ consists of a *feature vector* (also called *training sample* or *training example*)

$$\mathbf{x}_i = \begin{bmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,d} \end{bmatrix} \quad \text{or} \quad \mathbf{x}_i = \begin{bmatrix} x_{i,1} & x_{i,2} & \ldots & x_{i,d} \end{bmatrix}^\top, \tag{1.1}$$

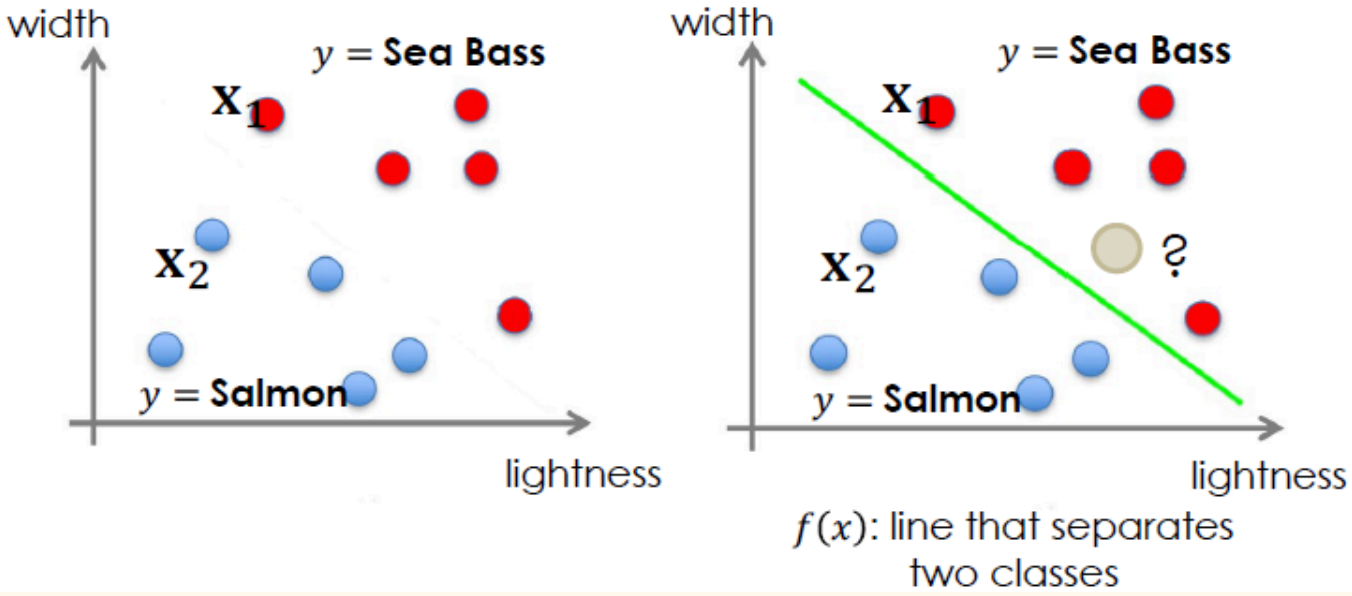
(usually an element of d-dimensional Euclidean space $\mathbb{R}^d$) and a *label* y_i. The crux of classification is that the label y_i can only take on *finitely many values* so $y_i \in \{1, 2, \ldots, c\}$ for some integer $c \geq 2$, where c is the number of classes. We do not allow y_i to take on infinitely many values.

The classification problem consists in finding a *classifier* which is a function $f : \mathbb{R}^d \to \{1, 2 \ldots, c\}$ such that it accurately predicts labels given new samples, called *test samples*. This function f is constructed or learned based on the dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i) : 1 \leq i \leq m\}$.

Classification

- Given $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$
- Learn a function $f(\mathbf{x})$ to predict categorical y given $\mathbf{x}$

Feature Space



- The two axis is the feature of the category, the plane is called feature space
- f(x) learnt is to separate the two category
- Classification belongs to the class of supervised learning methods.
- Example
  - We have records of $m = 100$ emails. Let $d = 2$. Thus, there are two components in each $\mathbf{x}_i$. Furthermore the first component of $\mathbf{x}_i$, denoted as $x_{i,1}$, counts the number of times the word "love" appears. The second component of $\mathbf{x}_i$, denoted as $x_{i,2}$, counts the number of times the word "money" appears. Each label $y_i \in \{0, 1\}$ where $y_i = 1$ means that the i^{th} email is spam while $y_i = 0$ means that the i^{th} email is non-spam. Based on the dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i) : 1 \leq i \leq 100\}$ of $m = 100$ emails, we are tasked to learn a classifier $f : \mathbb{R}^2 \to \{0, 1\}$ such that we can accurately predict whether the next email you receive (which is of course not in the training dataset) is spam or not. See Fig. 1.1. In this case in which there are only two classes, we refer to this as *binary* classification.

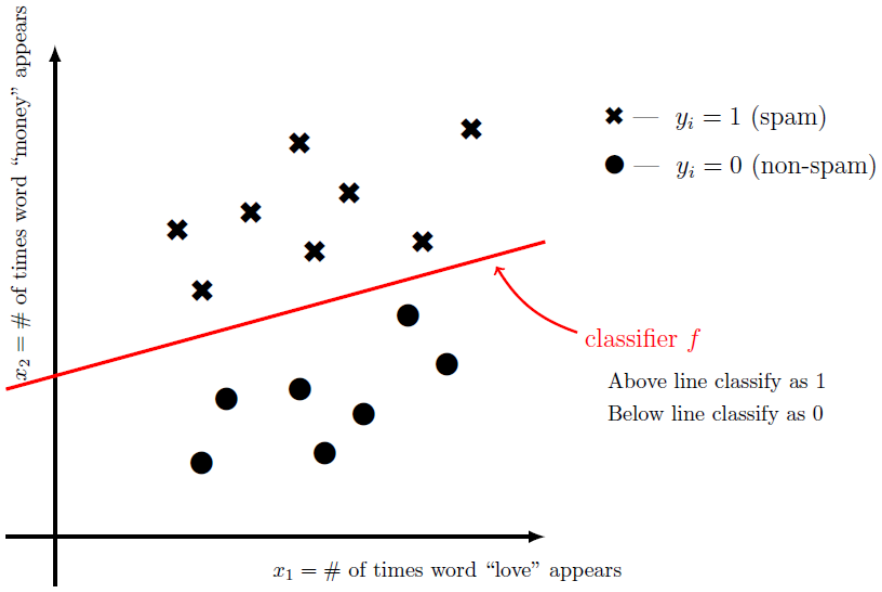
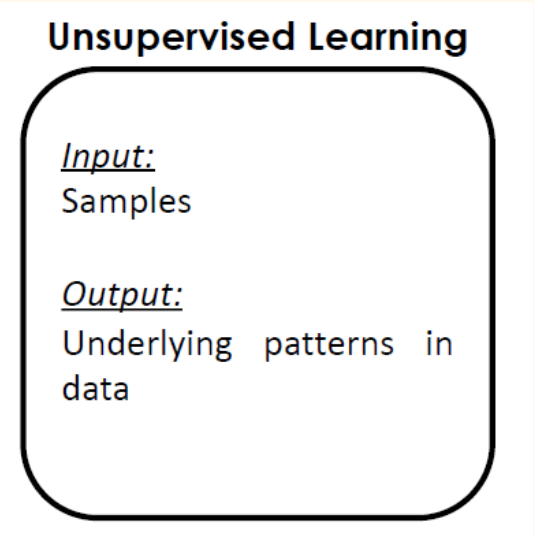


Figure 1.1: Spam classification

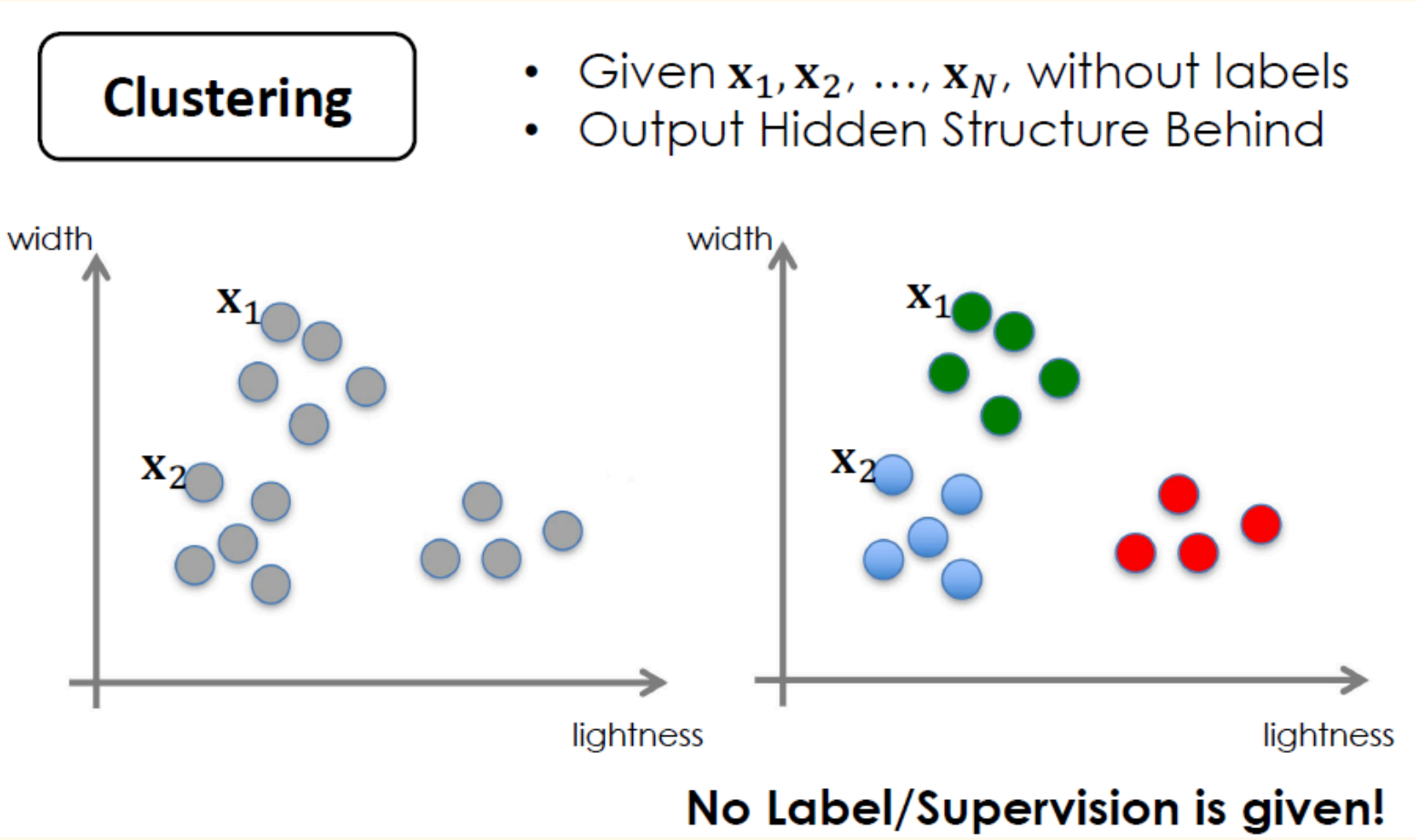
- Consider an image classification problem. We are given a dataset of size $m = 10^6$, namely, $\mathcal{D} = \{(\mathbf{x}_i, y_i) : 1 \leq i \leq 10^6\}$ where each $\mathbf{x}_i$ represents an image that is vectorized into a vector. For example, each image contains $5 \times 5 = 25$ pixels and for the sake of simplicity, each pixel value is 0 or 1. Thus $\mathbf{x}_i \in \{0,1\}^{5\times 5} \cong \{0,1\}^{25}$ and $d = 25$. Of course, we can have more complicated images that have multiple quantization levels (not restricted to binary) and colors as well, but the treatment will be the same. To each image, there is a label y_i which can take on 10 values so $c = 10$. These 10 values signify what object is in the image, e.g., $y_i \in \{\text{dog}, \text{cat}, \ldots, \text{snake}\}$. We would now like to design a image classifier $f : \{0,1\}^{25} \to \{\text{dog}, \text{cat}, \ldots, \text{snake}\}$ that "does well" (in some sense) on new images (or test images) that contain one of the 10 animals. Since there are more than two classes in this scenario, we refer to this as *multi-class* classification.

- Is the following a classification problem? There are a total of $M \geq 2$ football teams. Each football team $i \in \{1, 2, \ldots, M\}$ plays L games against every other football team j. Say there are no draws. The number of times i beats j is denoted as $b_{ij} \in \{0, 1, \ldots, L\}$ so necessarily $b_{ij} + b_{ji} = L$ (convince yourself that this is the case). We set $b_{ii} = 0$ for all i. Clearly, if one team wins all its games against other team, it is the best. When this is not case, which is common, it is not so clear what is the best strategy to *rank* the team. More precisely, given the matrix $\mathbf{B} = [b_{ij}]_{1\leq i\leq M, 1\leq j\leq M}$ we would like to rank the teams from top (best) to bottom (worst). This is a non-standard machine learning problem. However, notice that the total number or rankings is $M!$, which is finite, so in some sense this is like a classification problem. In what way is it not a classification problem? See [XFTF19] for some discussion of how we can use machine learning to learn the skills of tennis players.

Unsupervised Learning



Clustering



- Only clustering in syllabus for unsupervised
  - only categorical in syllabus, continuous exist
- Can only give x, cannot give y, there is no label/desire output
- Output hidden structure behind, cluster the groups
- small internal distance and larger external distance

In clustering, the labels or target variables y_i are no longer available and we *only* have access to the feature vectors $\mathcal{D} = \{\mathbf{x}_i : 1 \leq i \leq m\}$. There is often reason to believe that these feature vectors can be grouped or clustered into different clusters.

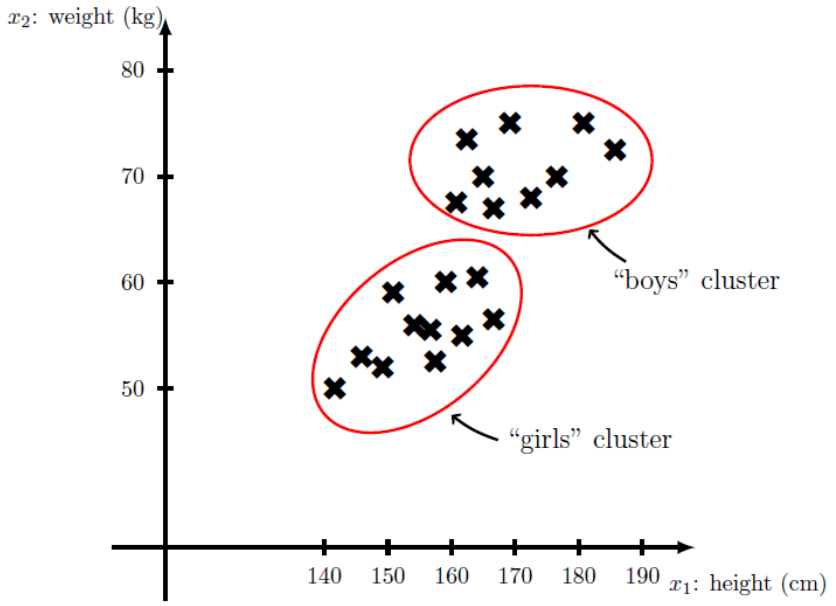


Figure 1.3: Height and weight clustering example

- Say the vector $\mathbf{x}_i = [x_{i,1}, x_{i,2}]^\top$ represents the height and weight of the i^{th} student in this semester's EE2211 cohort. There is reason to believe that girls are shorter and lighter than boys. So given $\mathcal{D} = \{\mathbf{x}_i : 1 \leq i \leq m\}$, can we assign each partition this set into 2 groups automatically? See Fig. 1.3. Note that while the clustering algorithm knows that there are 2 clusters, it does not and cannot assign semantic meanings (such as "boys" and "girls") to the clusters it discovers. It can only tell us that $\mathcal{D}$ is partitioned into two non-empty disjoint subsets $\mathcal{D}_1$ and $\mathcal{D}_2$ (i.e., $\mathcal{D}_1 \cap \mathcal{D}_2 = \emptyset$ and $\mathcal{D}_1 \cup \mathcal{D}_2 = \mathcal{D}$).
- Say each $\mathbf{x}_i = [x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4}]^\top$ and $x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4}$ respectively represent the number of times the words "winner", "victory", "stock", "prices" appeared in the i^{th} article in a bag of m news articles. There is reason to believe that sports articles would contain more of the first 2 words but finance articles would contain more of the last 2 words. Hence, it seems possible to design a clustering algorithm to group the m feature vectors into two clusters, one representing sports articles while the other representing finance articles.

Reinforcement Learning

Reinforcement Learning

Input:
Sequence of States, Actions, and Delayed Rewards

Output:
Action Strategy: a rule that maps the environment to action

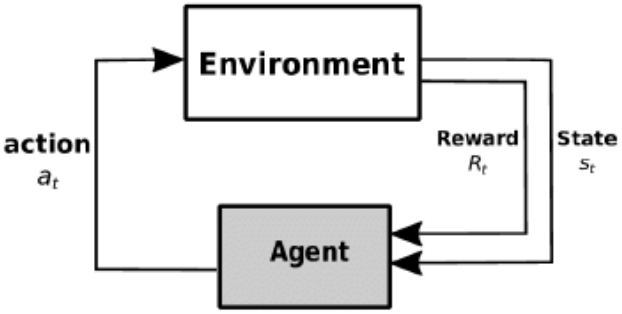
- When task requires a sequence of state or actions

Breakout Game

Initial Performance **Training 15 minutes** **Training 30 minutes**

- Its a gif of the game playing better after training

- Given sequence of states $\boldsymbol{S}$ and actions $\boldsymbol{A}$ with (delayed) rewards $\boldsymbol{R}$
- Output a policy $\pi(a, s)$, to guide us what action a to take in state s



$\boldsymbol{S}$: Ball Location, Paddle Location, Bricks

$\boldsymbol{A}$: left, right

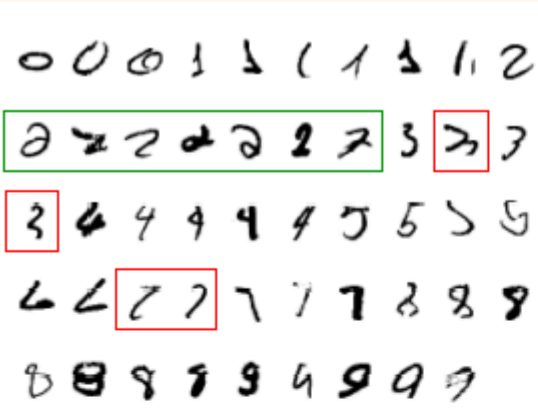
$\boldsymbol{R}$:
positive reward
Knocking a brick, clearing all bricks

negative reward
Missing the ball

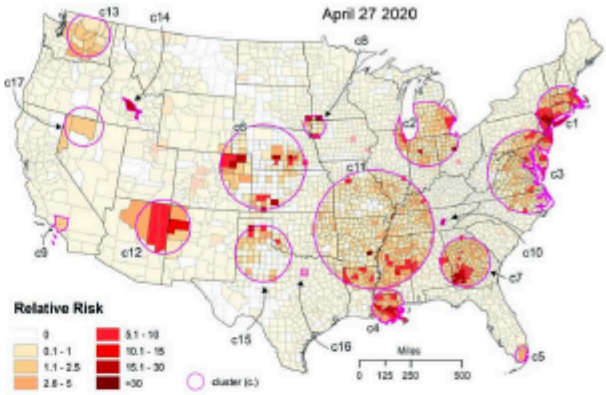
zero reward
Cases in between

- Intuitively, reinforcement learning with its actions and rewards is like supervision learning. But it's given a different label because the nature is so different

Example



Supervised

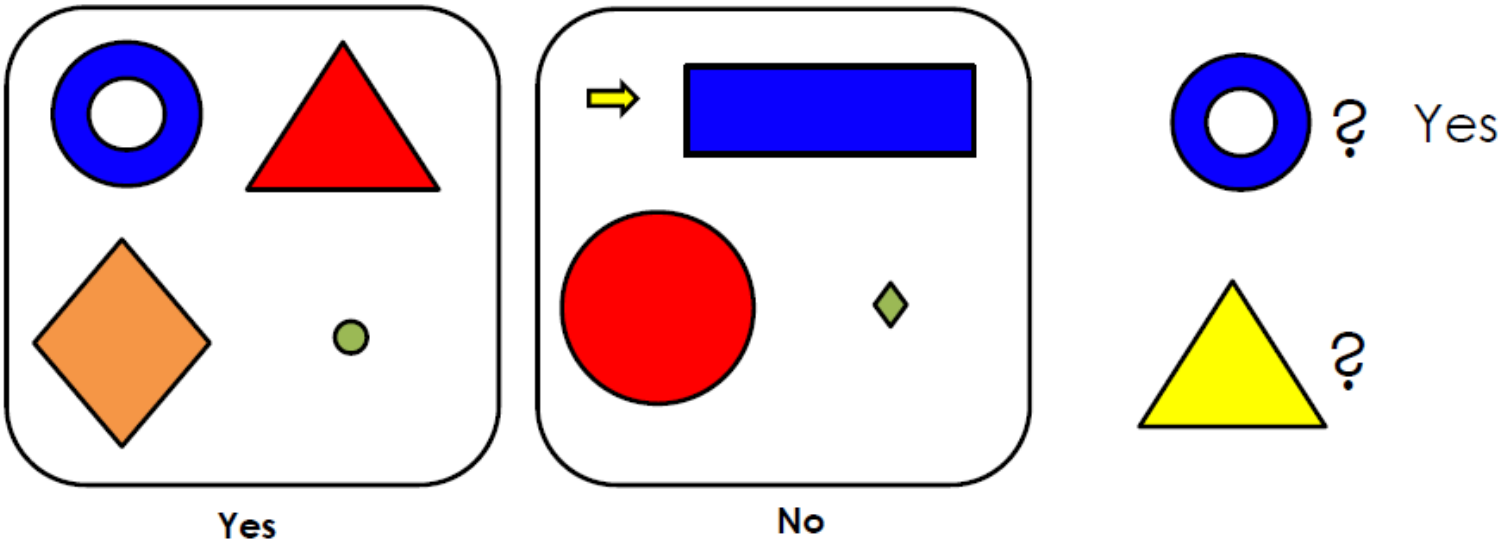


Unsupervised

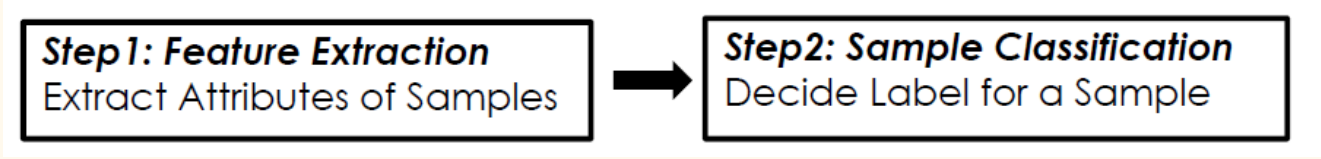
Supervised

Reinforcement

Walking Through A Toy Example: Token Classification

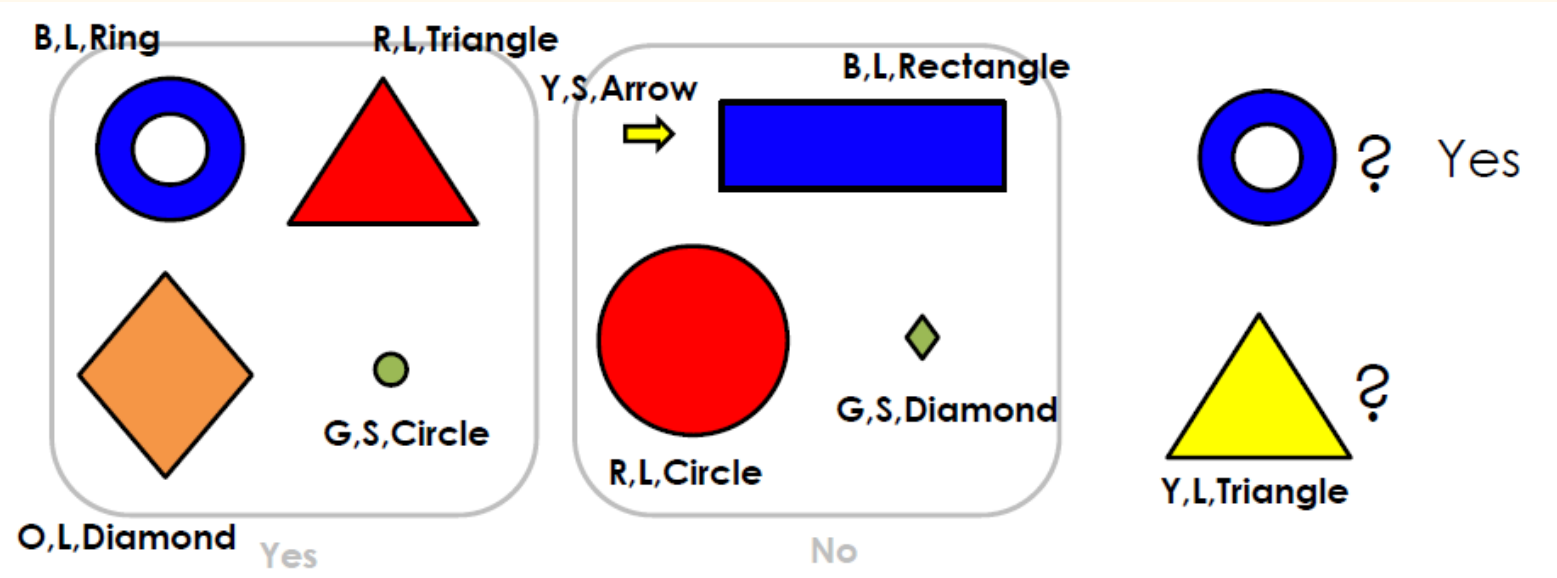


- Issue comes out when the testing sample (yellow triangle) does not overlap with the training sample (the set of all shapes with labels)
- Need to train a classification system



- requires a way to describe each sample, by extracting out their attributes
- based on the feature extracted, conduct the classification
- Sample classification can have many ways to do
  - Nearest Neighbour Classifier is just one way

Step 1: Feature Extraction



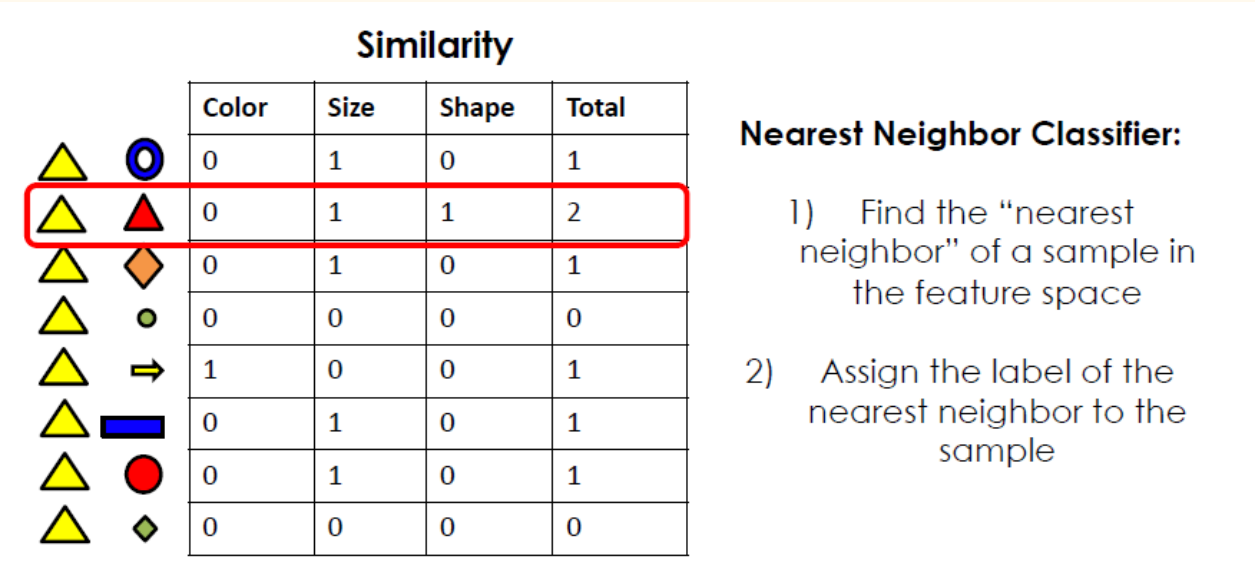
- Extract the feature of colour, size, shape (can have more)

Feature Extraction

	Color	Size	Shape	Label
	Blue	Large	Ring	Yes
	Red	Large	Triangle	Yes
	Orange	Large	Diamond	Yes
	Green	Small	Circle	Yes
	Yellow	Small	Arrow	No
	Blue	Large	Rectangle	No
	Red	Large	Circle	No
	Green	Small	Diamond	No
	Yellow	Large	Triangle	?

- To check the testing sample, check the testing sample against all the other training sample, and build up a Similarity table
- feature extraction = feature selection (for EE2211)

Step 2: Nearest Neighbour Classifier



Similarity

	Color	Size	Shape	Total
	0	1	0	1
	0	1	1	2
	0	1	0	1
	0	0	0	0
	1	0	0	1
	0	1	0	1
	0	1	0	1
	0	0	0	0

Nearest Neighbor Classifier:

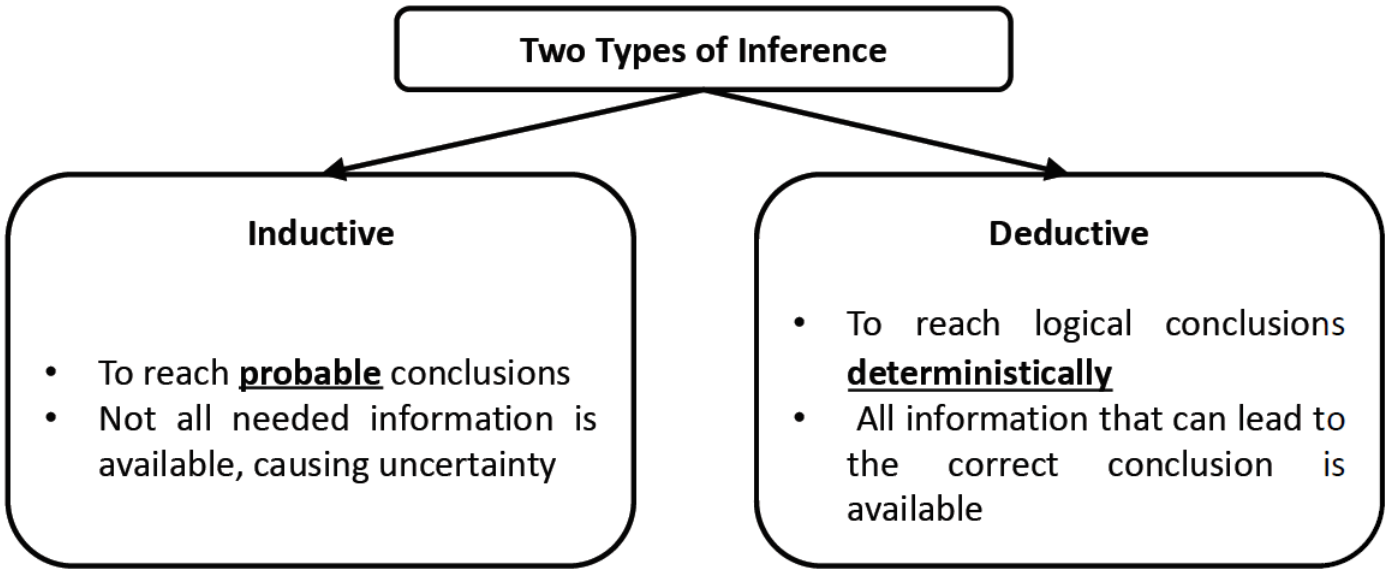
1) Find the "nearest neighbor" of a sample in the feature space

2) Assign the label of the nearest neighbor to the sample

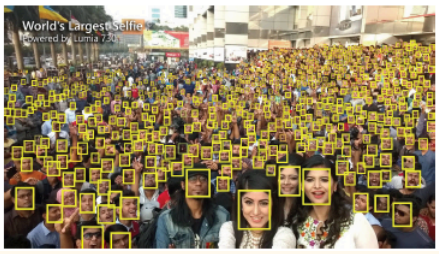
- Yellow big triangle is the most similar to the red big triangle, hence, given the label of red big triangle. Most similar in the feature space
- Nearest neighbour = more similar
- If attributes is changed the result may differ
    - need to extract as many feature as possible to avoid draw in similarity
    - if has draw choose any
- In EE2211, the newly labelled will not be learn, only the experience from the input/output pair given
    - done in advanced technique
- Nearest Neighbour Classifier does not requires training, it's just comparing against lookup table

Inductive vs. Deductive Reasoning

- Main Task of Machine Learning: to make inference



Probability and Statistics



Rule-based reasoning

NUS is in Singapore, Singapore is in Asia -> NUS is in Asia

- Inductive
  - Do not have all the information needed, can only give probable conclusion
  - Result have probability/confidence
  - Has uncertainty
  - If face is far away hard to tell
  - EE2211 ML: is inductive
  - Guessing population from sample, small picture to big picture
- Deductive
  - Have all the information needed
  - Conclusion is deduced from information and hence 100% confidence and certainty
  - Draw conclusion deterministically
  - With the big picture get the small picture

Summary and Practise Questions

Three Components in ML Definition

Task T, Performance P, Experience E

Two Types of Supervised Learning

Classification, Regression

Three Types of in ML

Supervised Learning
Unsupervised Learning
Reinforcement Learning

One Type of Unsupervised Learning

Clustering

Inductive and Deductive

Inductive: Probable
Deductive: Rule-based

Example of a Classifier Model

Nearest Neighbor Classifier

Which of the following statement is true?

A. Nearest Neighbor Classifier is an example of unsupervised learning
B. Nearest Neighbor Classifier is an example of deductive learning
C. Nearest Neighbor Classifier is an example of feature selection
D. None of the above is correct.

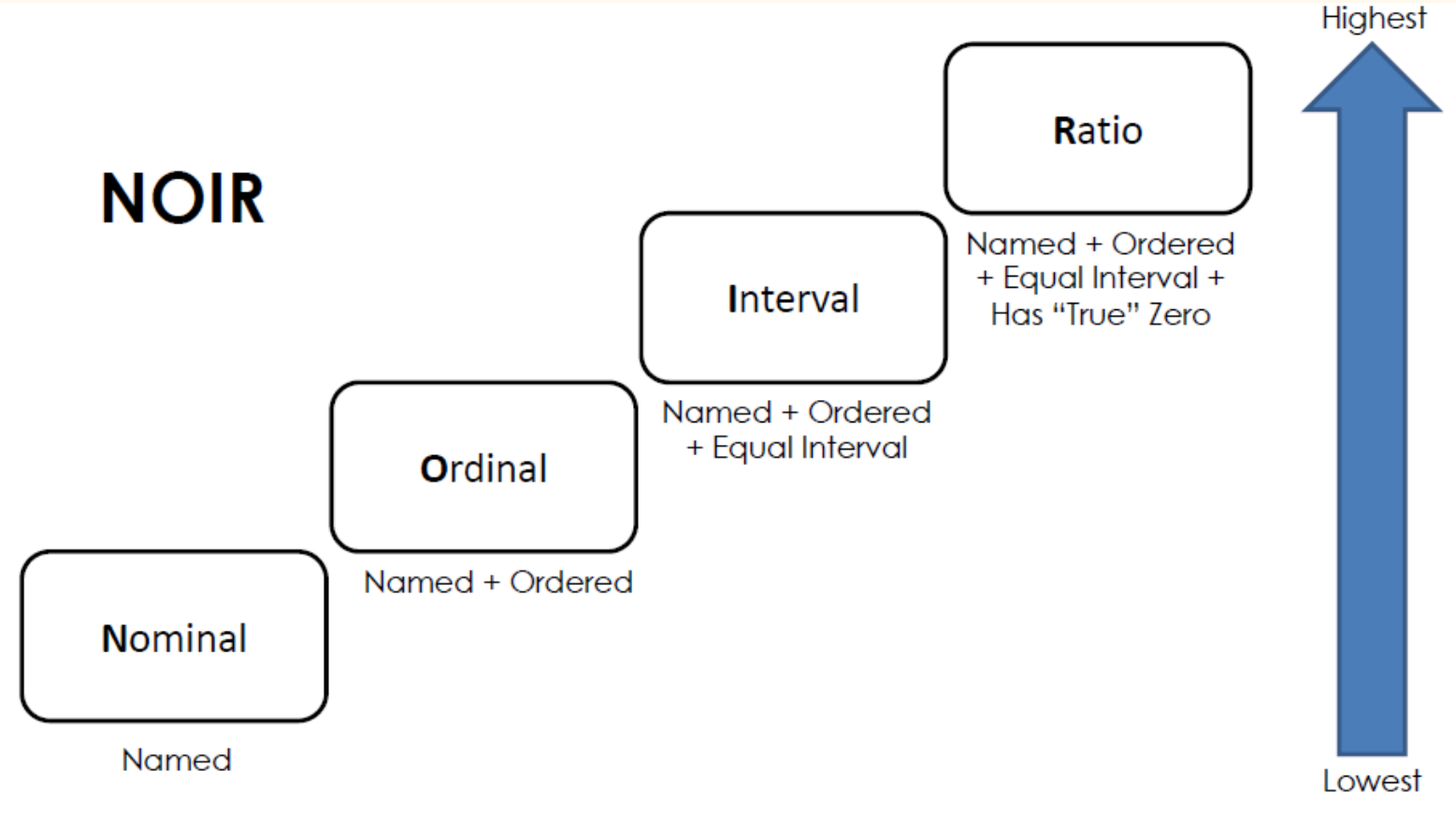
- A is wrong, nearest neighbour classifier is supervised
- B is wrong, nearest neighbour classifier is not with full certainty
- C is wrong, nearest neighbour classifier is the second step, the first step is feature extraction/selection
    - other classification can be used
    - feature extraction = feature selection (for EE2211)

Lec2: Data Engineering

Types of Data

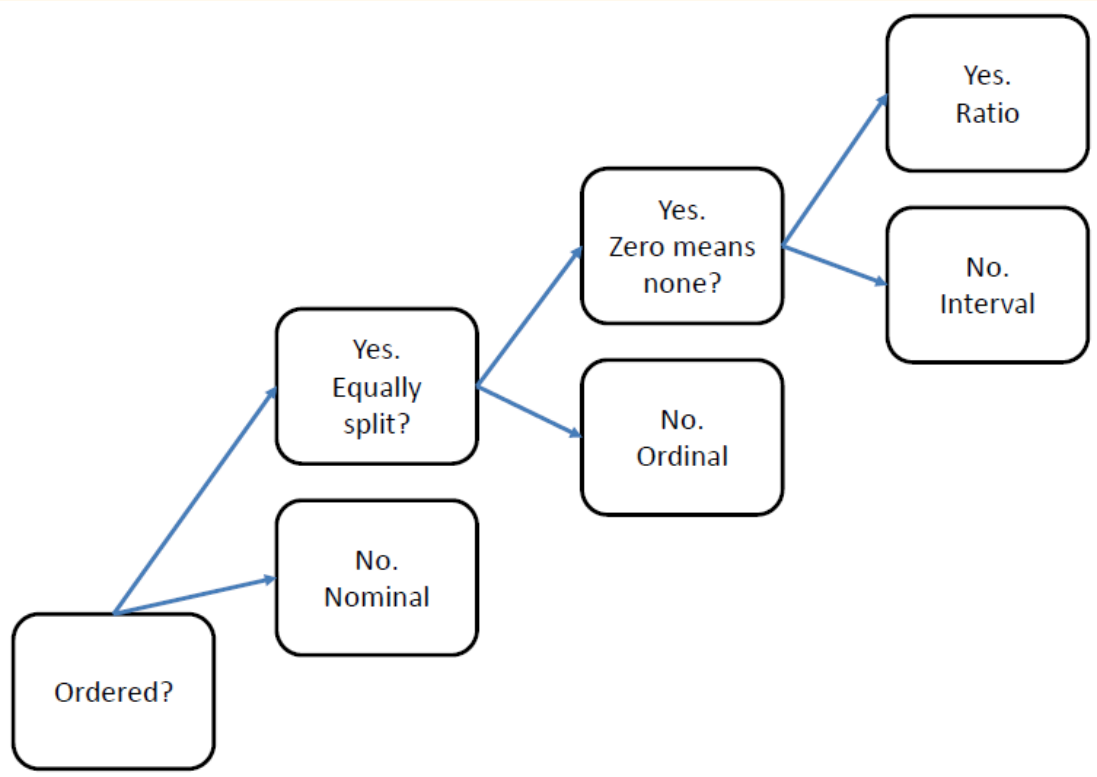
- Based on Levels/Scales of Measurement
  - Nominal Data
  - Ordinal Data
  - Interval Data
  - Ratio Data
- Based on Numerical/Categorical
  - Numerical, also known as Quantitative
  - Categorical, also known as Qualitative
- Other aspects
  - Available or Missing Data

Levels/Scales of Measurement



- Going up means, Ordinal is Named + Ordered

We can estimate	Nominal	Ordinal	Interval	Ratio
Frequency Distribution	Yes	Yes	Yes	Yes
Median	No	Yes	Yes	Yes
Add or subtract	No	No	Yes	Yes
Mean, standard deviation	No	No	Yes	Yes
Ratios	No	No	No	Yes



Nominal Data

- Lowest Level of Measurement
- Discrete Categories
- **NO** natural order
- Estimating a mean, median, or standard deviation, would be meaningless.
- Possible Measure: mode, frequency distribution

- Example:

Ordinal Data

- **Ordered** Categories
- Relative Ranking
- Unknown "distance" between categories: orders matter but not the difference between values
- Possible Measure: mode, frequency distribution + median

- Example:
  - Evaluate the difficulty level of an exam
    - 1: Very Easy, 2: Easy, 3: About Right, 4: Difficulty, 5: Very Difficulty

- can sort means can have median

Interval Data

- Ordered Categories
- Well-defined "**unit**" measurement:
  - Distances between points on the scale are measurable and well-defined
  - Can measure differences!
- **Equal Interval** (between two consecutive unit points)
- **Zero is arbitrary** (not absolute), in many cases human-defined
  - If the variable equals zero, it does not mean there is none of that variable
- **Ratio is meaningless**
- Possible Measure: mode, frequency distribution + median + mean, standard deviation, addition/subtraction
- Example:
  - Temperature measured in Celsius
    - For instance: 10 degrees C, 28 degrees C
  - Year of someone's birth
    - For instance: 1990, 2005, 2010, 2022

- because zero is arbitrary defined, the ratio of two value is meaningless
- 20 degC is not twice as energetic as 10degC

Ratio Data

- Most precise and **highest** level of measurement
- Ordered
- Equal Intervals
- **Natural Zeros**:
  - If the variable equals zero, it means there is none of that variable
  - Not arbitrary
- Possible Measure: mode, frequency distribution + median + mean, standard deviation, addition/subtraction + multiplication and division (ratio)

- Example:
  - Weights
    - 10 KG, 20 KG, 30 KG

  - Time
    - 10 Seconds, 1 Hour, 1 Day

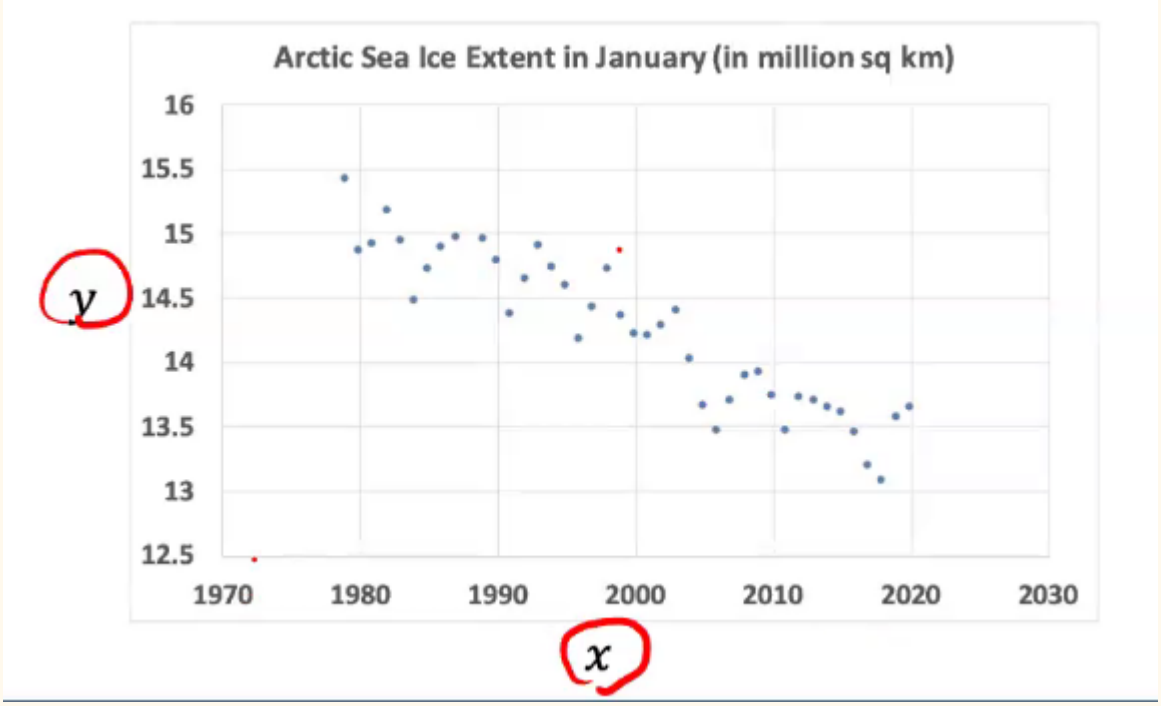
- well defined ratio, 20 kg twice as heavy as 10 kg

Example

1. **Favorite Restaurant**
   - Mcdonald's, Burger King, Subway, KFC, …
2. **Weight of luggage measured in KG**
3. **SAT Scores: note that, SAT ranges is [400, 1600]**
4. **Size of Packed Eggs in supermarkets**
   - Small, Medium, Large, Extra Large, …
5. **Military rank**
   - General, Major, Captain, …
6. **Number of people in a household**
   - 1, 2, 3, 4, 5, …
7. **Credit Score in United States: the range is [300, 850]**

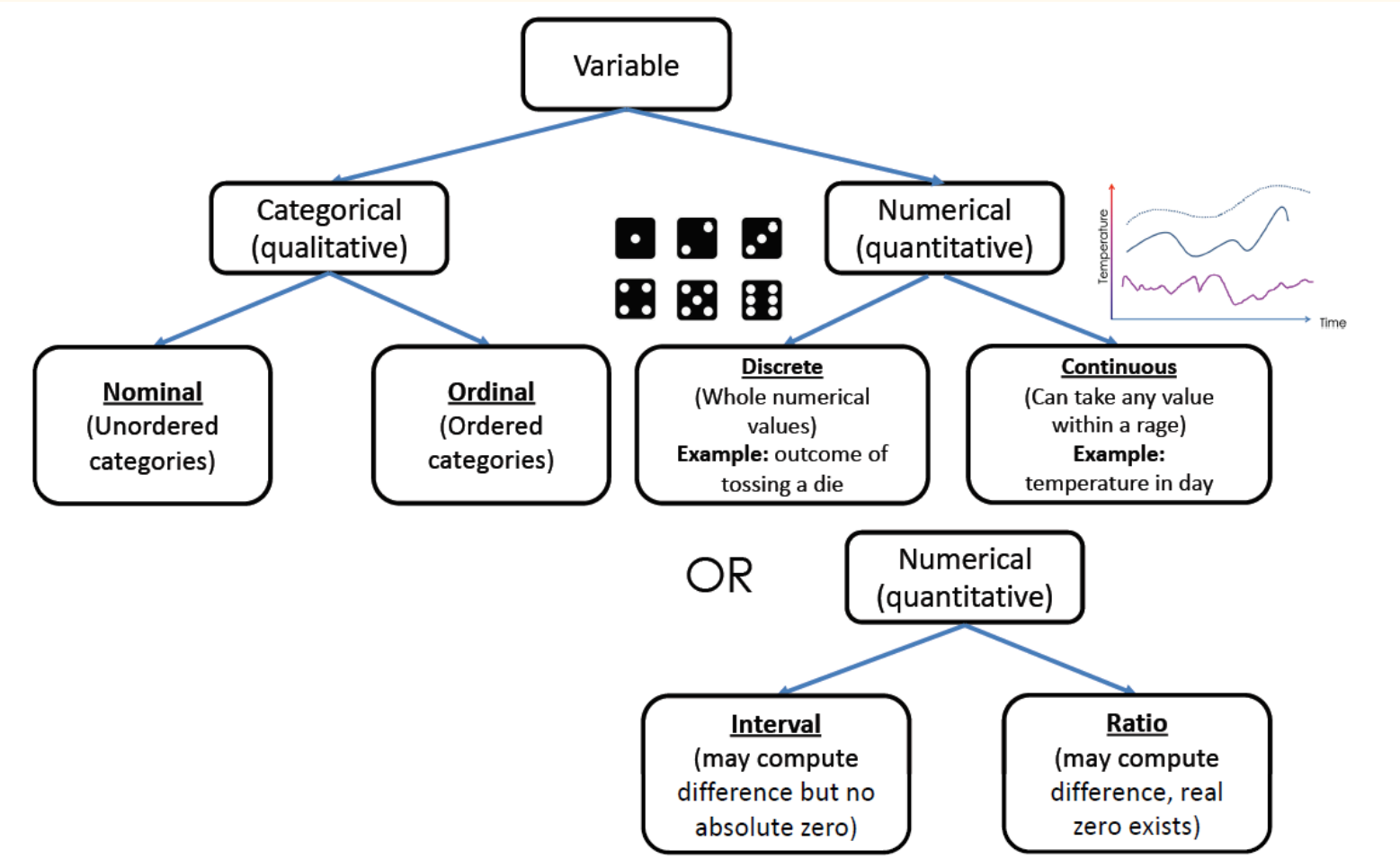
1. Nominal

2. Ratio
3. Interval
4. Ordinal
5. Ordinal
6. Ratio
7. Interval



- x is interval, y is ratio

Based on Numerical/Categorical



Other Aspects

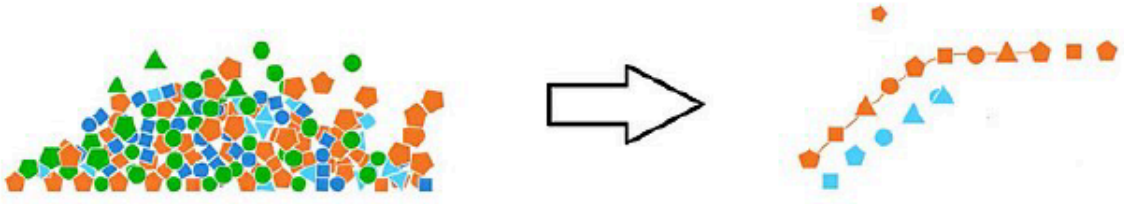
- Missing data: data that is missing and you do not know the mechanism.
  - You should use a single common code for all missing values (for example, "NA"), rather than leaving any entries blank.

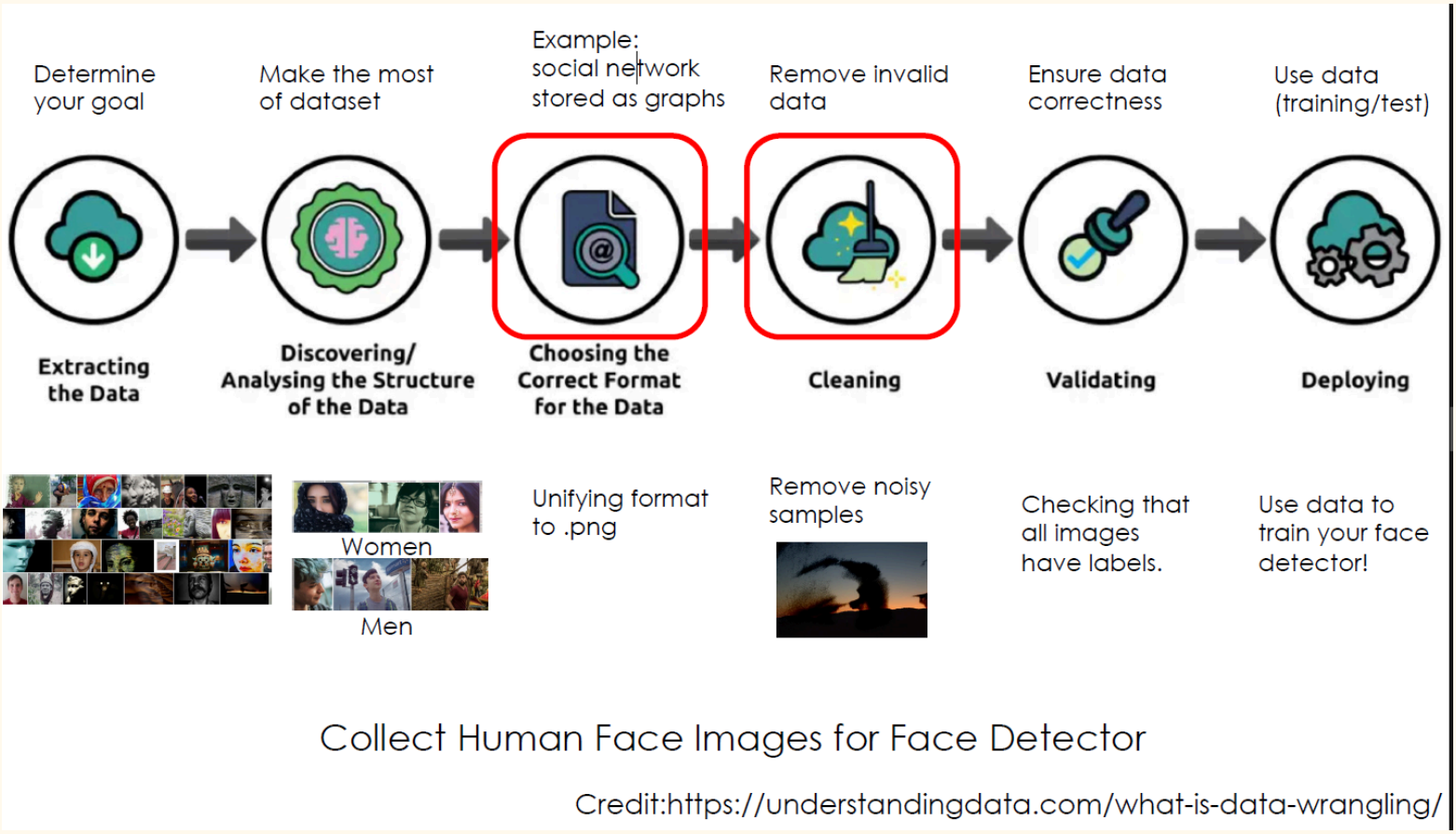
NUS student	Age	Country of birth
Olivia Tan	20	Singapore
Hendra Setiawan	19	Indonesia
John Smith	19	NA

Data Wrangling and Cleaning (Data Preprocessing)

- Prepare data for ML/feature extraction
- Cleaning is subset of wrangling

– The process of transforming and mapping data from one "raw" data form into another format with the intent of making it more appropriate and valuable for a variety of downstream purposes such as analytics.
– In short, transforms data to gain insight
– It is a general process!





- second step is to group the data

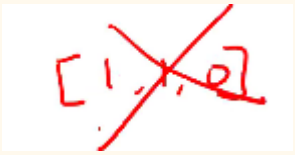
Formatting Data

Binary Coding

- **Binary Coding** to convert categories into binary form
  - One-hot encoding: unify several entities within one vector
    - Example: the color of a pixel can be red, yellow, or green
    - Very common in classification tasks!

$$\text{red} = [1, 0, 0]$$
$$\text{yellow} = [0, 1, 0]$$
$$\text{green} = [0, 0, 1]$$

- Format Labels/output
- One-hot encoding is one type of binary coding
  - Put stuff in one vector

- s!
$$\text{red} = [1, 0, 0]$$
$$\text{yellow} = [0, 1, 0]$$
$$\text{green} = [0, 0, 1]$$
  - One-hot memes, no inbetween, only one attributes can be present
    - 

Normalisation

Often we have feature vectors in which features are on different scales. We let m be the number of training samples and d be the number of features as usual. Say the feature vectors are

$$\mathbf{x}_1 = \begin{bmatrix} x_{1,1} \\ x_{1,2} \\ \vdots \\ x_{1,d} \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} x_{2,1} \\ x_{2,2} \\ \vdots \\ x_{2,d} \end{bmatrix}, \quad \ldots \quad \text{and} \quad \mathbf{x}_m = \begin{bmatrix} x_{m,1} \\ x_{m,2} \\ \vdots \\ x_{m,d} \end{bmatrix}. \tag{2.2}$$

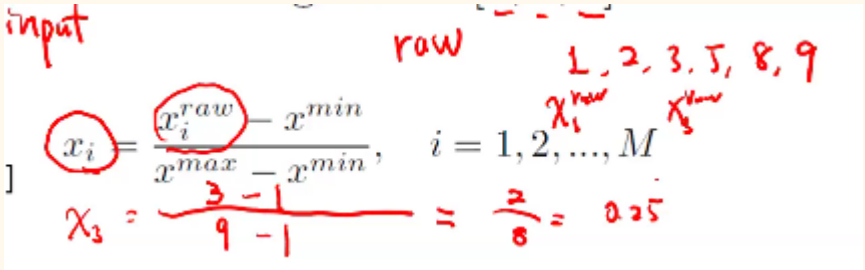
The first feature $x_{i,1}$ for $1 \leq i \leq m$ could be the height of students measured in centimeters so the dynamic range is $[x_{\min,1}, x_{\max,1}] = [140, 190]$. The second feature $x_{i,2}$ for $1 \leq i \leq m$ could measure the (American) shoe size of the students so it ranges from $[x_{\min,2}, x_{\max,2}] = [6, 13]$. So even if both features are deemed equally "important", unfortunately, any machine learning method would place more importance on the first feature because of its larger values, which is not ideal. Thus, we have to scale or normalize the features so that their dynamic ranges are roughly the same. We can use (at least) two methods to do so:

- **Normalization**
  - Linear Scaling: scale each variable to [0 1]

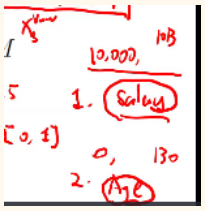
$$x_i = \frac{x_i^{raw} - x^{min}}{x^{max} - x^{min}}, \quad i = 1, 2, ..., M$$

  - Z-score standardization: each independent dimension of data is normally distributed

$$x_i = \frac{x_i^{raw} - E[X]}{\sigma(X)}, \quad i = 1, 2, ..., M.$$

- Format data/input
- Linear scaling
  - 

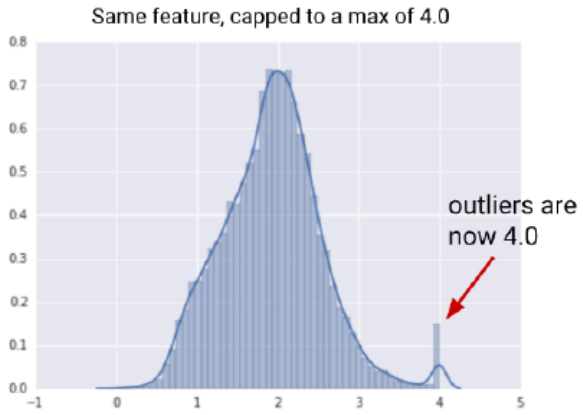
  - convert into 0-1

- any range to 0-1
- uses for converting two data of very different range into comparable range
- 
  - example is age and salary
- $$\bar{x}_{i,1} := \frac{x_{i,1} - x_{\min,1}}{x_{\max,1} - x_{\min,1}}$$

- Z-score
  - must be normally distributed
  - $$\hat{x}_1 = \frac{1}{m}\sum_{i=1}^{m} x_{i,1}, \quad \text{and} \quad \hat{\sigma}_1 = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(x_{i,1} - \hat{x}_1)^2}$$
  - $$\bar{x}_{i,1} := \frac{x_{i,1} - \hat{x}_1}{\hat{\sigma}_1}$$

Data Cleaning

- The process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database.
- Example:
  - Clipping outliers



  - Handling missing features

Students	Year of Birth	Gender	Height	GPA
Tan Ah Kow	1995	M	1.72	4.2
Ahmad Abdul	X NA	M	1.65	4.1
John Smith	1995	M	1.75	X NA
Chen Lulu	1995	F	X NA	4.0
Raj Kumar	1995	M	1.73	4.5
Li Xiuxiu	1994	F	1.70	3.8

- must make sure outlier is outlier before taking them out, otherwise it's still valuable data
- handling missing features has several ways

Data Cleaning: Handling missing features

1. Removing the examples with missing features from the dataset
   - Can be done if the dataset is big enough so we can sacrifice some training examples

2. Using a learning algorithm that can deal with missing feature values
   - Example: random forest

3. Using a data imputation technique

- Only can remove if dataset is big enough
- imputation means fill in the data

Data Cleaning: Handling missing features: Imputation

- Method 1. Replace the missing value of a feature by an average value of this feature in the dataset:

$$\hat{x}^{(j)} \leftarrow \frac{1}{N}\sum_{i=1}^{N} x_i^{(j)}$$

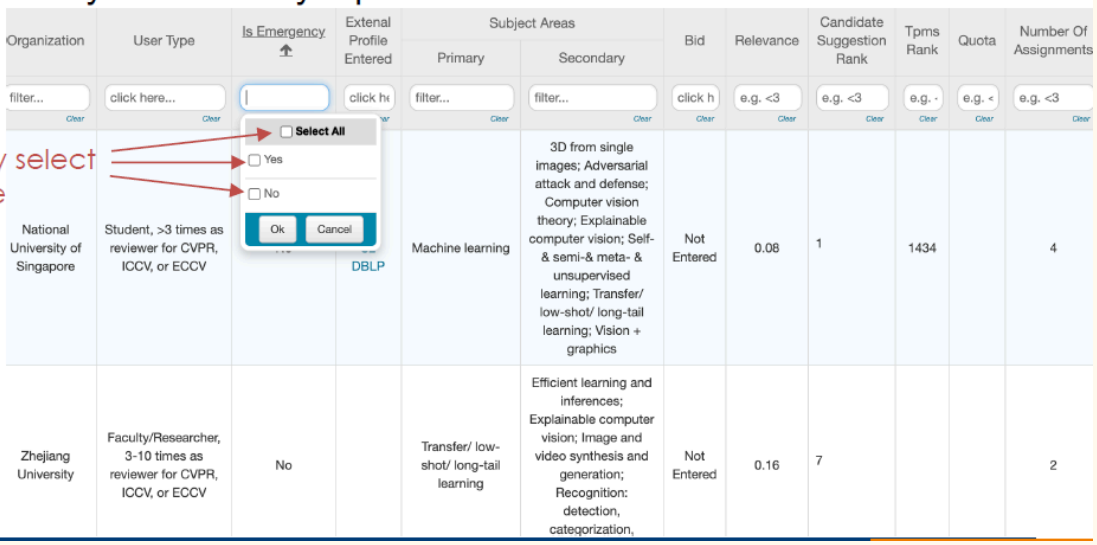
- Method 2. Highlight the missing value
  - Replace the missing value with a value outside the normal range of values.
  - For example, if the normal range is [0, 1], then you can set the missing value to −1.
  - Enforce the learning algorithm to learn what is best to do when the feature has a value significantly different from regular values.

- Method 1 is less commonly used, as it's not the best practice
- Method 2, is used to make the algorithm able to learn what to do when encounter -1 in the future
  - -1 is only for the algo, when finding mean and whatnot, ignore the -1

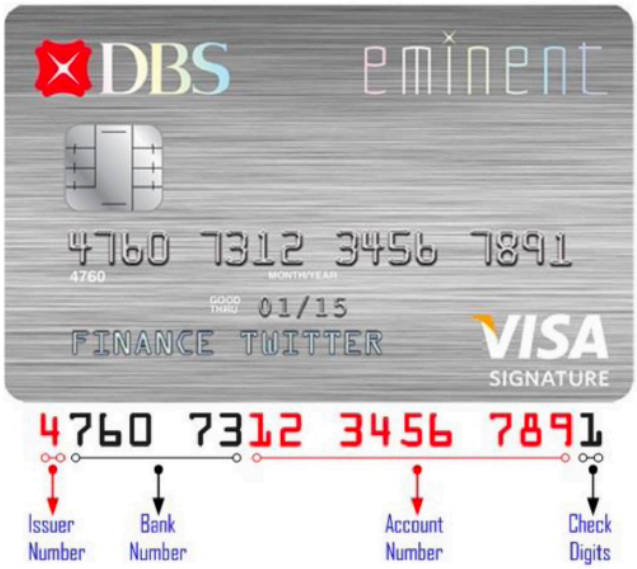
Data Integrity and Visualisation

Data Integrity

- Data integrity is the maintenance and the assurance of data accuracy and consistency;
  - A critical aspect to the design, implementation, and usage of any system that stores, processes, or retrieves data.
  - Very broad concept!
- Example:
  - In a dataset, numeric columns/cells should not accept alphabetic data.
  - A binary entry should only allow binary inputs

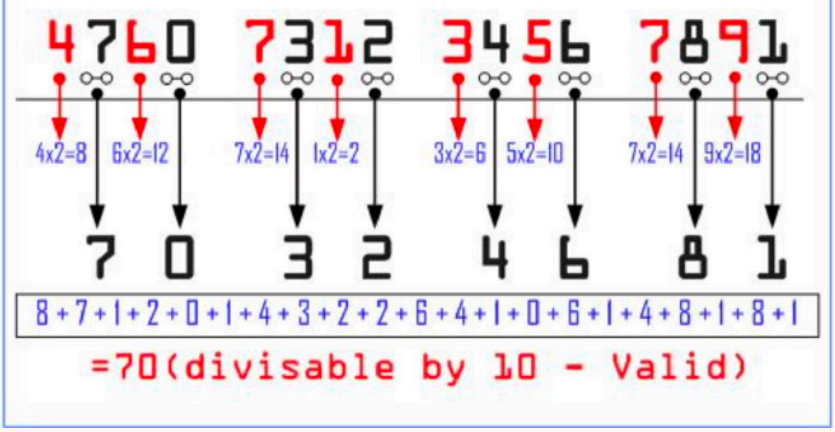


Organization	User Type	Is Emergency	Extenal Profile Entered	Subject Areas: Primary	Subject Areas: Secondary	Bid	Relevance	Candidate Suggestion Rank	Tpms Rank	Quota	Number Of Assignments
filter...	click here...		click h	filter...	filter...	click h	e.g. <3	e.g. <3	e.g. -	e.g. <	e.g. <3
National University of Singapore	Student, >3 times as reviewer for CVPR, ICCV, or ECCV		DBLP	Machine learning	3D from single images; Adversarial attack and defense; Computer vision theory; Explainable computer vision; Self- & semi-& meta- & unsupervised learning; Transfer/ low-shot/ long-tail learning; Vision + graphics	Not Entered	0.08	1	1434		4
Zhejiang University	Faculty/Researcher, 3-10 times as reviewer for CVPR, ICCV, or ECCV	No		Transfer/ low-shot/ long-tail learning	Efficient learning and inferences; Explainable computer vision; Image and video synthesis and generation; Recognition: detection, categorization,	Not Entered	0.16	7			2

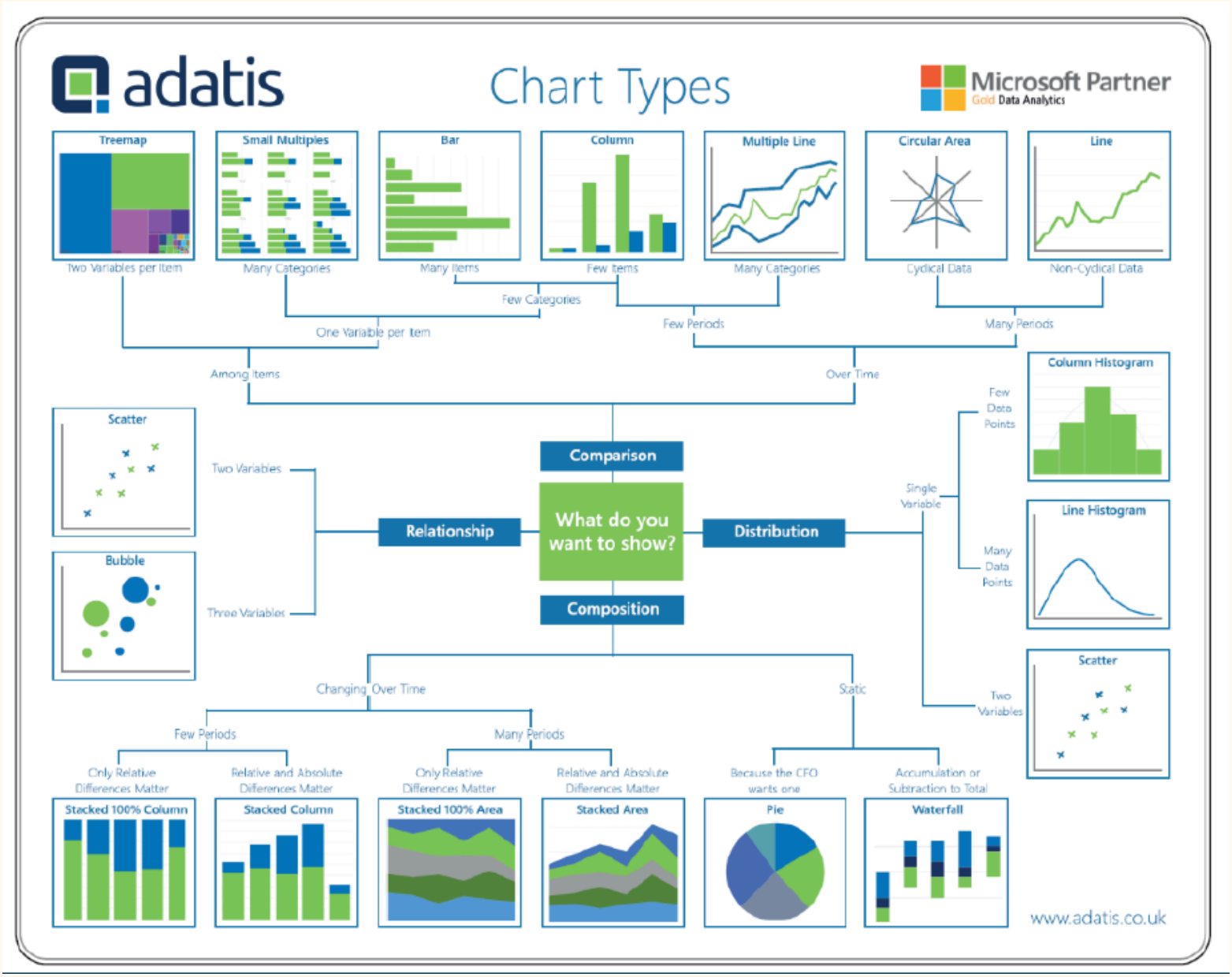

DBS
eminent
4760 7312 3456 7891
4760
MONTH/YEAR
GOOD THRU 01/15
FINANCE TWITTER
VISA
SIGNATURE

4760 7312 3456 7891
Issuer Number
Bank Number
Account Number
Check Digits

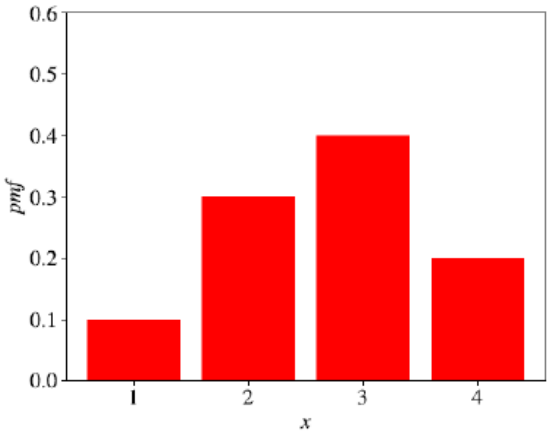

DBS
eminent
4760 7312 3456 7891
4760
MONTH/YEAR
GOOD THRU 01/15
FINANCE TWITTER
VISA
SIGNATURE


4760 7312 3456 7891
4x2=8
6x2=12
7x2=14
1x2=2
3x2=6
5x2=10
7x2=14
9x2=18
7 0 3 2 4 6 8 1
8+7+1+2+0+1+4+3+2+2+6+4+1+0+6+1+4+8+1+8+1
=70(divisable by 10 - Valid)

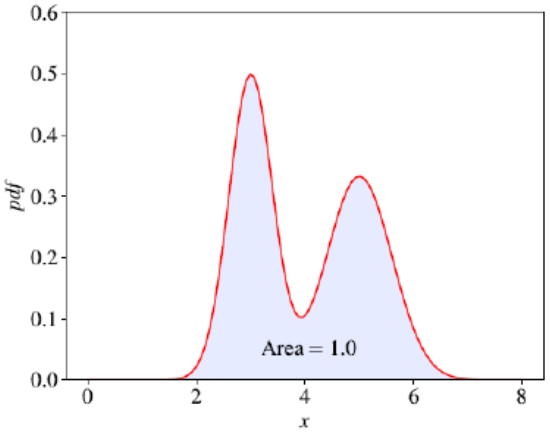
Data Visualization



Visualization: Distribution

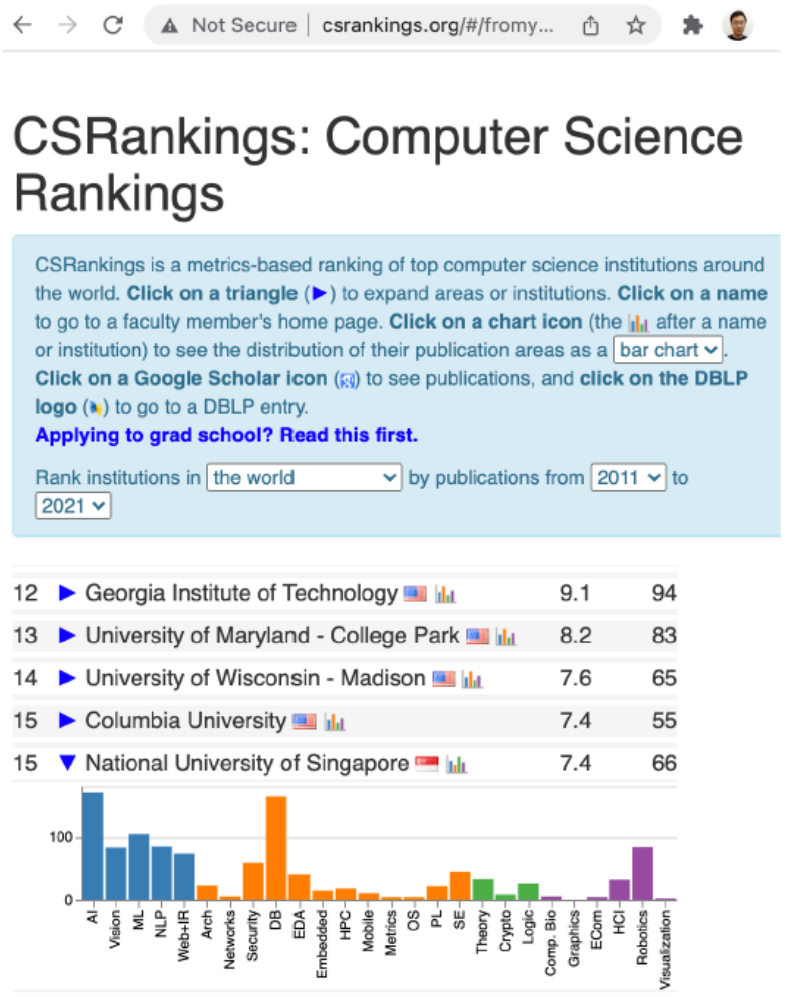


Probability Mass Function

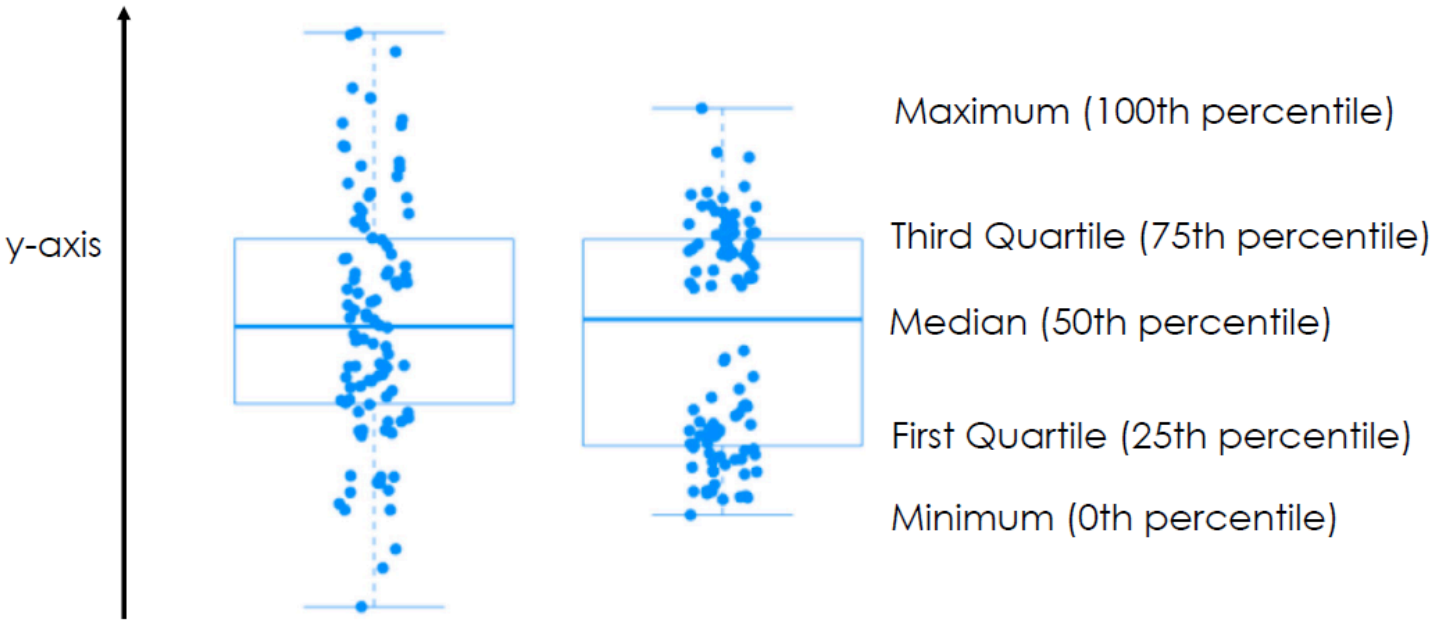


Probability Density Function

Visualization: Bars



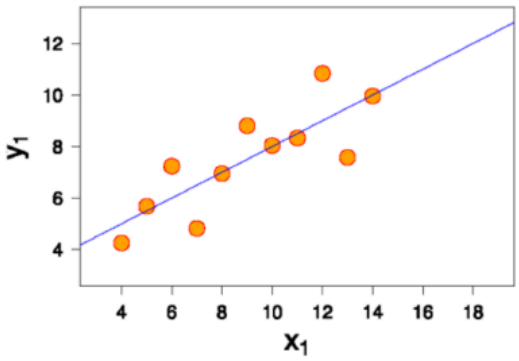
Visualization: Boxplots

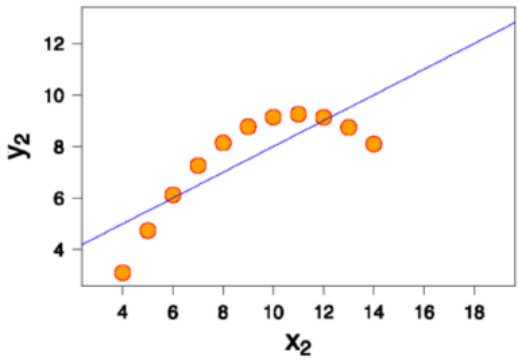


- The first quartile (Q_1) is defined as the middle number between the smallest number (i.e., Minimum) and the median of the data set.

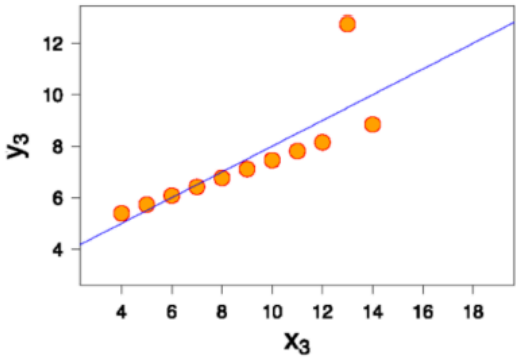
- The third quartile (Q3) is the middle number between the median and the highest value (i.e., Maximum) of the data set.

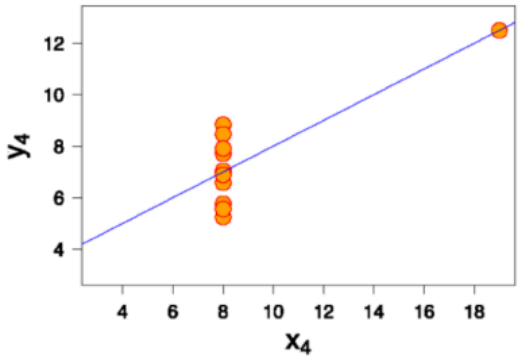
Why Visualization is Necessary





Four datasets with **identical** means, variances and regression lines!

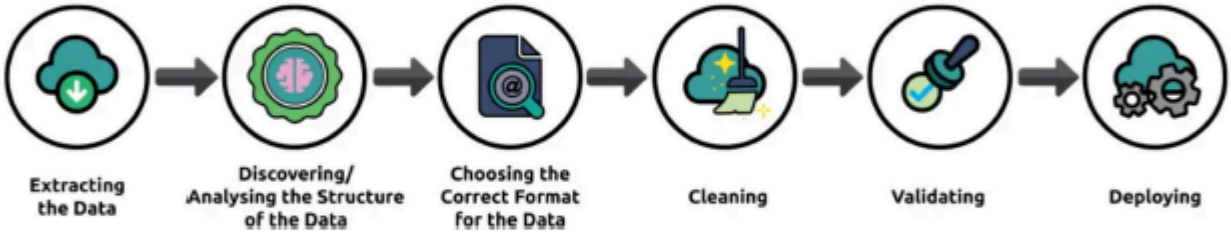




Hence, we need visualization to show their difference!

Summary and Practice Question

- Types of data
  - NOIR
- Data wrangling and cleaning



- Data integrity and visualization
  - Integrity: Design
  - Visualization: Graphical Representation

Color	Size	Shape
Blue	Large	Ring
Red	Large	Triangle
Orange	Large	Diamond
Green	Small	Circle
Yellow	Small	Arrow
Blue	Large	Rectangle
Red	Large	Circle
Green	Small	Diamond

What are the NOIR data types of *color*, *size*, and *shape* in the table?

- Colour - Nominal
- Size - Ordinal
- Shape - Nominal

Lec3: Introduction to Linear Algebra, Probability and Statistics

Introduction to Linear Algebra

Notations, Vectors, Matrices

- A **scalar** is a simple numerical value, like 15 or −3.25
  - Focus on real numbers
- **Variables** or **constants** that take scalar values are denoted by an *italic* letter, like *x* or *a*

- A **vector** is an ordered list of scalar values
  – Denoted by a **bold character**, e.g. **x** or **a**

- In many books, vectors are written column-wise:

$$\mathbf{a} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -2 \\ 5 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

- The three vectors above are two-dimensional, or have two elements

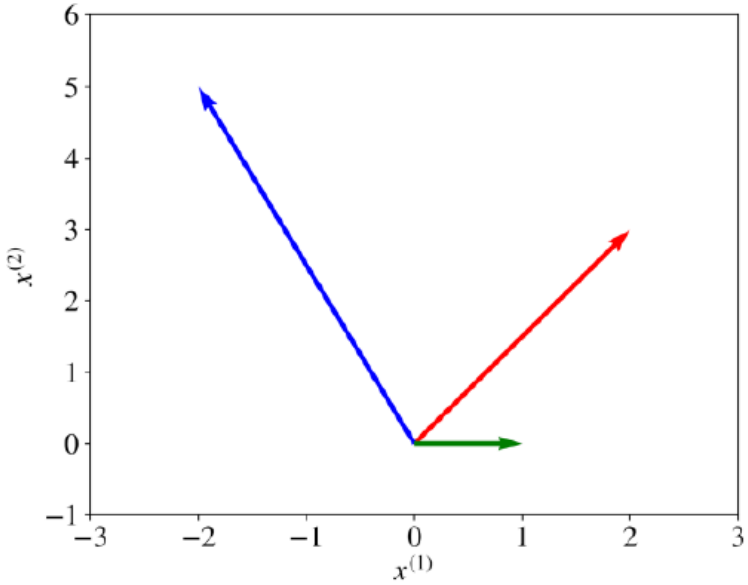
- We denote an entry or attribute of a vector as an italic value with an index, e.g. $a^{(j)}$ or $x^{(j)}$.
  – The index j denotes a specific dimension of the vector, the position of an attribute in the list

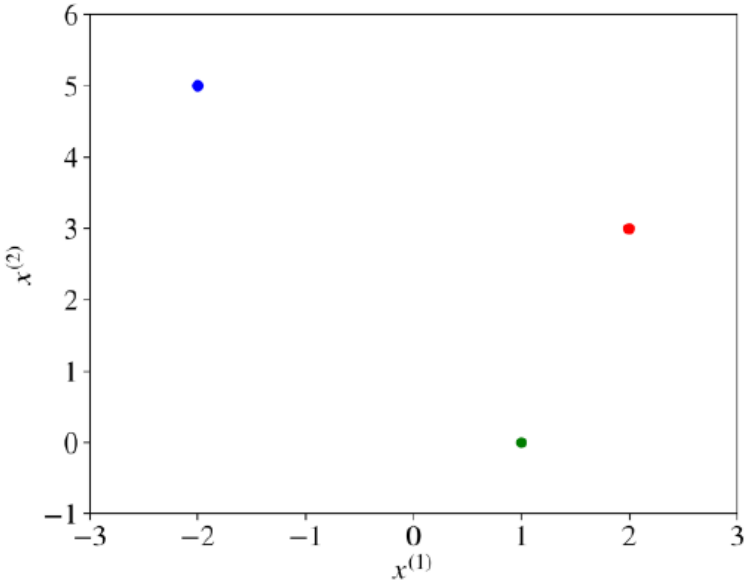
$$\mathbf{a} = \begin{bmatrix} a^{(1)} \\ a^{(2)} \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \quad \text{or more commonly} \quad \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

- Note:
  – $x^{(j)}$ is not to be confused with the power operation, e.g., x^2 (squared)
  – Square of an indexed attribute of a vector is denoted as $(x^{(j)})^2$.

- **Vectors** can be visualized as, in a multi-dimensional space,
  - arrows that point to some directions, or
  - points

Illustrations of three two-dimensional vectors, $\mathbf{a} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} -2 \\ 5 \end{bmatrix}$, and $\mathbf{c} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$





- A **matrix** is a rectangular array of numbers arranged in rows and columns
  - Denoted with bold capital letters, such as **X** or **W**
  - An example of a matrix with two rows and three columns: $\mathbf{X} = \begin{bmatrix} 2 & 4 & -3 \\ 21 & -6 & -1 \end{bmatrix}$

- A **set** is an unordered collection of unique elements
  - When an element x belongs to a set $\boldsymbol{S}$, we write $x \in \boldsymbol{S}$.
  - A special set denoted **R** includes all real numbers from minus infinity to plus infinity

- Note:
  - For elements in matrix **X**, we shall use the indexing $x_{1,1}$ where the first and second indices indicate the row and the column position.
  - Usually, for input data, rows represent samples and columns represent features

- set is unordered with unique elements
  - $S = \{3, 2\}$
- or **W** Feature 1 Feature 2 Feature 3

$$\mathbf{X} = \begin{bmatrix} 2 & 4 & -3 \\ 21 & -6 & -1 \end{bmatrix}$$ Sample 1, Sample 2

- **Capital Sigma**: the summation over a collection $\{x_1, x_2, x_3, x_4, \ldots, x_m\}$ is denoted by:

$$\sum_{i=1}^{m} x_i = x_1 + x_2 + \ldots + x_{m-1} + x_m$$

- **Capital Pi**: the product over a collection $\{x_1, x_2, x_3, x_4, \ldots, x_m\}$ is denoted by:

$$\prod_{i=1}^{m} x_i = x_1 \cdot x_2 \cdot \ldots \cdot x_{m-1} \cdot x_m$$

Linear dependence and independence

- A collection of d-vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$ (with $m \geq 1$) is called **linearly dependent** if

$$\beta_1 \mathbf{x}_1 + \cdots + \beta_m \mathbf{x}_m = 0$$

holds for some $\beta_1, \ldots, \beta_m$ that are **not all zero**.

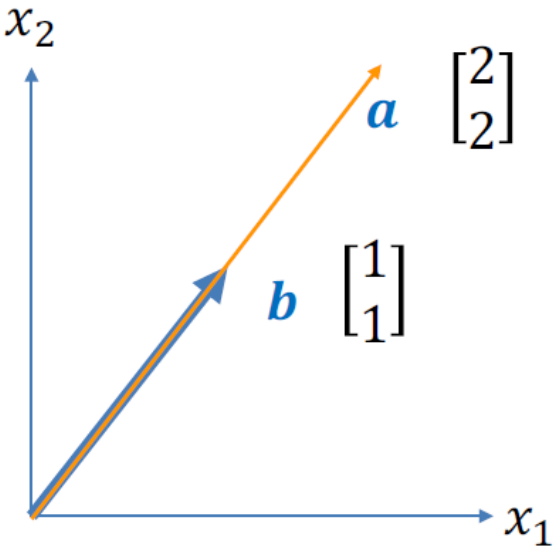
- A collection of d-vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$ (with $m \geq 1$) is called **linearly independent** if it is not linearly dependent, which means that

$$\beta_1 \mathbf{x}_1 + \cdots + \beta_m \mathbf{x}_m = 0$$

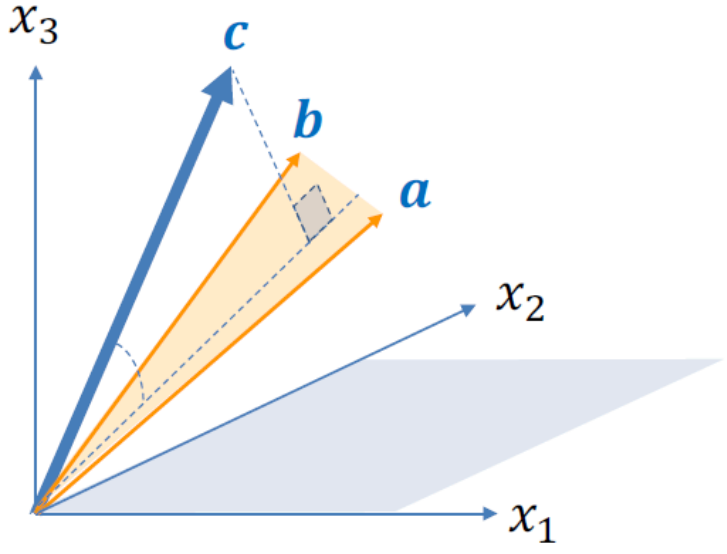
only holds for $\beta_1 = \cdots = \beta_m = 0$.

Note: If all rows or columns of a square matrix **X** are **linearly independent**, then **X** is **invertible.**

Geometry of dependency and independency



$$\beta_1 \boldsymbol{a} + \beta_2 \mathbf{b} = 0$$



$$\beta_1 \boldsymbol{a} + \beta_2 \boldsymbol{b} \neq \beta_3 \mathbf{c}$$

Systems of Linear Equations

These equations can be written compactly in matrix-vector notation:

$$\mathbf{X}\mathbf{w} = \mathbf{y}$$

Where

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,d} \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}.$$

Note:

- The data matrix $\mathbf{X} \in \mathcal{R}^{m \times d}$ and the target vector $\mathbf{y} \in \mathcal{R}^{m}$ are given
- The unknown vector of parameters $\mathbf{w} \in \mathcal{R}^{d}$ is to be learnt
- The rank($\mathbf{X}$) corresponds to the maximal number of linearly independent columns/rows of $\mathbf{X}$.

- Learn the solution to Xw = y, to map X to y, hence leaning w

- The principled way for computing rank is to do Echelon Form
  - https://stattrek.com/matrix-algebra/echelon-transform.aspx#MatrixA
- For small-size matrices, however, the rank is in many cases easy to estimate

- What is the rank of

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \\ 100 & 100 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -2 & 3 \\ 0 & -3 & 3 \\ 1 & 1 & 0 \end{bmatrix}$$

Causality

GEA1000 Notes

- Causality, or causation is:
  - The influence by which one event or process (i.e., cause) contributes to another (i.e. effect),
  - The cause is partly responsible for the effect, and the effect is partly dependent on the cause

- Causality relates to an extremely very wide domain of subjects: philosophy, science, management, humanity.

- Causality research is extremely complex
  - Researcher can never be completely certain that there are **no other factors** influencing the causal relationship,
  - In most cases, we can only say "probably" causal.

- Partially or partly, cannot be completeyl sure that there is anyhting else that something ti affecting the relationship

- (Probable) causal relations or non-causal?
  - New web design implemented ? Web page traffic increased Yes
  - Your height and weight ? Gets A in EE2211 No
  - Uploaded new app store images ? Downloads increased by 2X Yes
  - One works hard and attends lectures/tutorials ? Gets A in EE2211 Yes
  - Your favorite color ? Your GPA in NUS No

- One popular way to causal data analysis is **Randomized Controlled Trial (RCT)**
  - A study design that randomly assigns participants into an *experimental* group or a *control* group.
  - As the study is conducted, the only expected difference between two groups is the outcome variable being studied.

- Example:
  - To decide whether smoking and lung cancer has a causal relation, we put participants into *experimental* group (people who smoke) and *control* group (people who don't smoke), and check whether they develop lung cancer eventually.

- RCT is sometimes infeasible to conduct, and also has moral issues.

Causality is very difficult to study, but it's possible

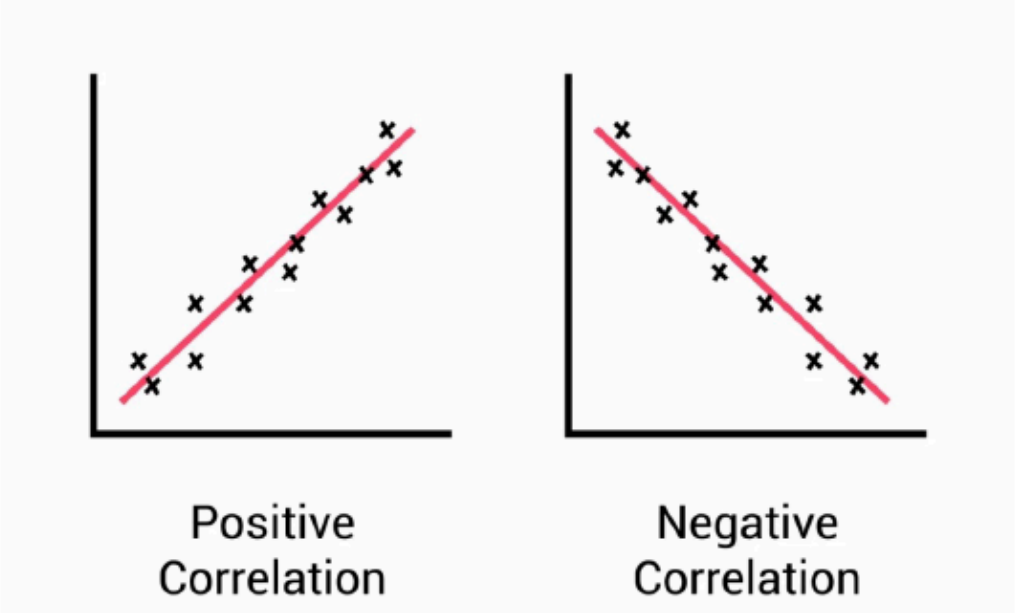
	Experiments	Observational Studies
Assignment	By researchers	Participants assign themselves
Randomisation	Preferable	Not possible
Ethical issues	Possible *(if intervention may be harmful)*	Unlikely
Confounders	Unlikely *(if randomisation is done on enough participants)*	Usually exist many
Possible to show causation	Yes (in the ideal case)	Very difficult
Able to show association	Yes	

Causality is a Statistical Relationship

- Decades of data show a clear causal relationship between smoking and cancer.
- If one smokes, it is a sure thing that his/her risk of cancer will increase.
- But it is not a sure thing that one will get cancer.
- The relationship is not deterministic.

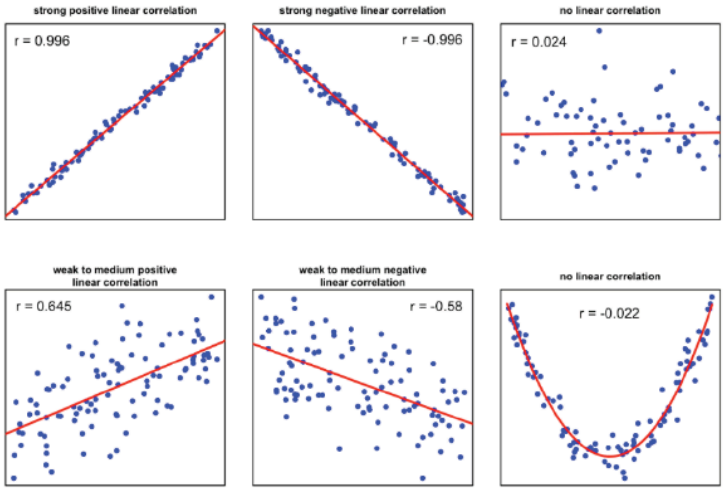
Correlation vs Causality

- In statistics, **correlation** is any statistical relationship, whether causal or not, between two random variables.
- Correlations are useful because they can indicate a predictive relationship that can be exploited in practice.



- Linear correlation coefficient, r, which is also known as the Pearson Coefficient.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} = \frac{s_{xy}}{s_x s_y},$$



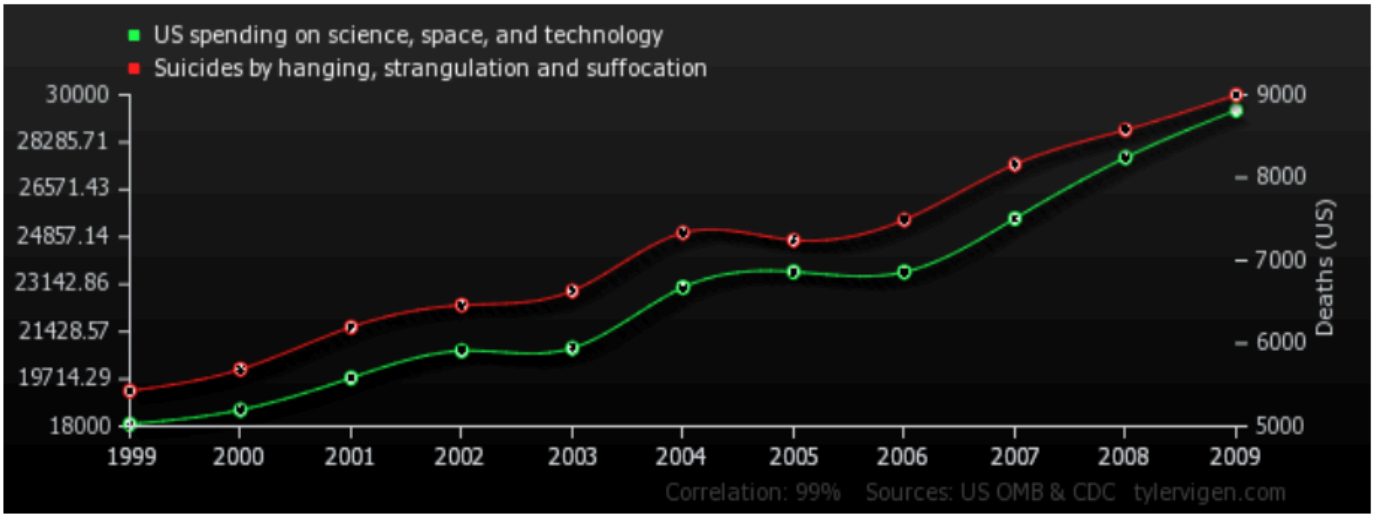
Strong linear relationship	$r > 0.9$
Medium linear relationship	$0.7 < r \leq 0.9$
Weak linear relationship	$0.5 < r \leq 0.7$
No or doubtful linear relationship	$0 < r \leq 0.5$

The same holds for negative values.

https://www.geo.fu-berlin.de/en/v/soga/

Correlation does not imply causation!

- Some great examples of correlations that can be calculated but are clearly not causally related appear at http://tylervigen.com/

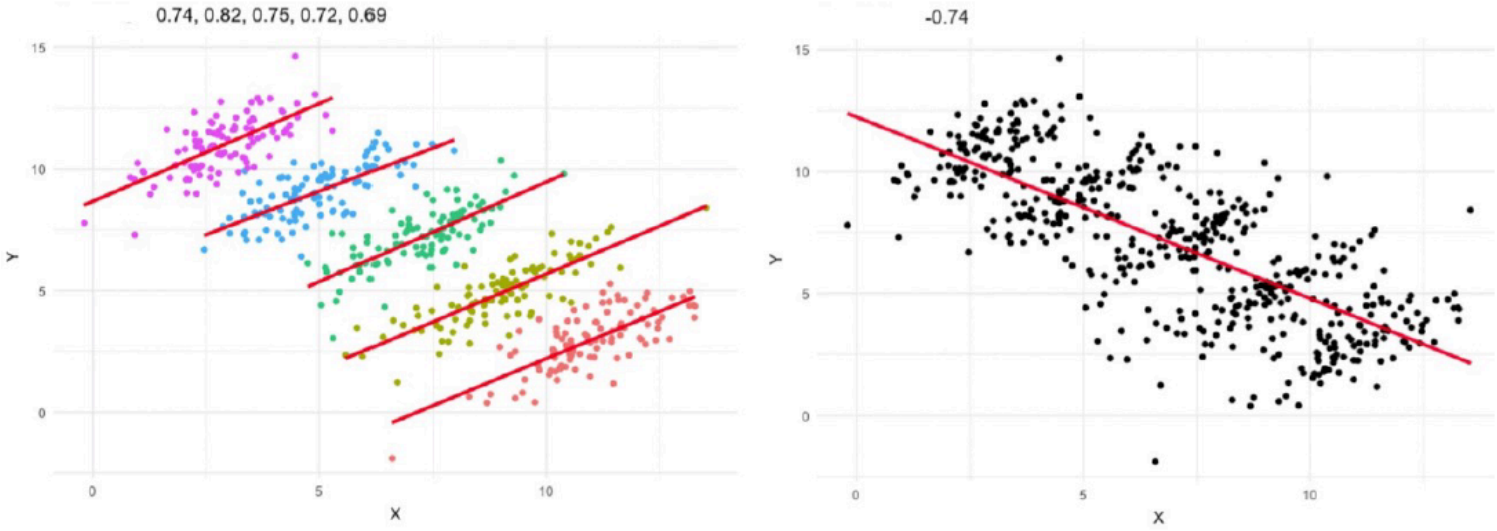


https://tylervigen.com

Simpson's paradox

Simpson's paradox: GEA1000 Notes

- **Simpson's paradox** is a phenomenon in probability and statistics, in which a trend appears in several different groups of data but disappears or reverses when these groups are combined.



The same set of samples!

Example

- Batting Average in professional baseball game
- Two well-known players, Derek Jeter and David Justice

Batter \ Year	1995		1996		Combined	
Derek Jeter	12/48	.250	183/582	.314	195/630	**.310**
David Justice	104/411	**.253**	45/140	**.321**	149/551	.270

#of wins #of games

Random Variable

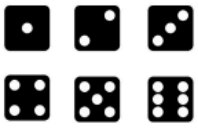
GEA1000 Notes

EE2012 Notes

- We describe a *random experiment* by describing its procedure and observations of its *outcomes*.
- *Outcomes* are mutual exclusive in the sense that only one outcome occurs in a specific trial of the random experiment.


outcome

  - This also means an outcome is not decomposable.
  - All unique outcomes form a *sample space*.


sample space

- A subset of sample space S, denoted as A, is an *event* in a random experiment $A \subset S$, that is meaningful to an application.
  - Example of an event: faces with numbers no greater than 3


event

- Some books used $P(\cdot)$ and $p(\cdot)$ to distinguish between the probability of discrete random variable and the probability of continuous random variables respectively.

- We shall use $Pr(\cdot)$ for both the above cases

- A **random variable**, usually written as an *italic* capital letter, like *X*, is a variable whose possible values are numerical outcomes of a random event.

- There are two types of random variables: discrete and continuous.

Axioms of Probability

Assuming events $A \subseteq S$ and $B \subseteq S$, the probabilities of events related with and must satisfy,

1. $Pr(A) \geq 0$
2. $Pr(S) = 1$
3. If $A \cap B = \emptyset$, then $Pr(A \cup B) = Pr(A) + Pr(B)$

*otherwise, $Pr(A \cup B) = Pr(A) + Pr(B) - Pr(A \cap B)$

Two Basic Rules

- Sum Rule

$$\Pr(X = x) = \sum_{Y} \Pr(X = x, Y = y_i)$$

- Product Rule

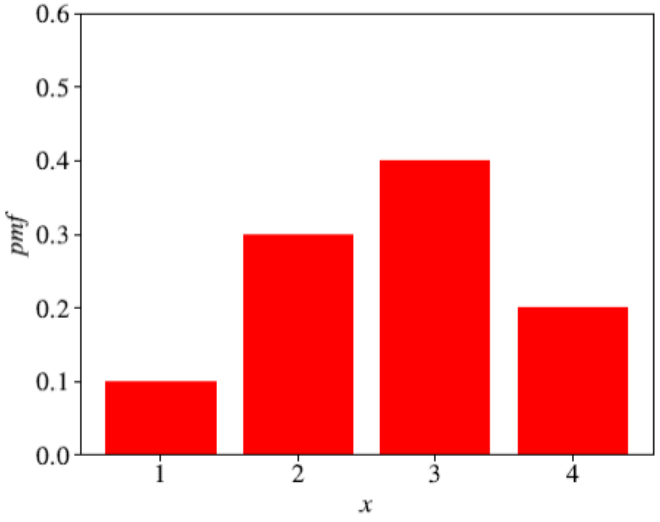
$$\Pr(X = x, Y = y) = \Pr(Y = y | X = x)\, P(X = x)$$

Discrete Random Variable

- A discrete random variable (DRV) takes on only a countable number of distinct values such as red, orange, blue or 1, 2, 3.

- The probability distribution of a discrete random variable is described by a list of probabilities associated with each of its possible values.

- This list of probabilities is called a probability mass function (pmf).
  - Like a histogram, except that here the probabilities sum to 1



A probability mass function

- Let a **discrete** random variable X have k possible values $\{x_i\}_{i=1}^{k}$.

- The **expectation** of X denoted as $E(x)$ is given by,

$$E(x) \stackrel{\text{def}}{=} \sum_{i=1}^{k} [x_i \cdot \Pr(X = x_i)]$$
$$= x_1 \cdot \Pr(X = x_1) + x_2 \cdot \Pr(X = x_2) + \cdots + x_k \cdot \Pr(X = x_k)$$

  where $\Pr(X = x_i)$ is the probability that X has the value x_i according to the pmf.

- The expectation of a random variable is also called the **mean**, **average** or **expected value** and is frequently denoted with the letter μ.

- Another important statistic is the **standard deviation**, defined as,

$$\sigma \stackrel{\text{def}}{=} \sqrt{E[(X - \mu)^2]}\,.$$

- **Variance**, denoted as σ^2 or $var(X)$, is defined as,

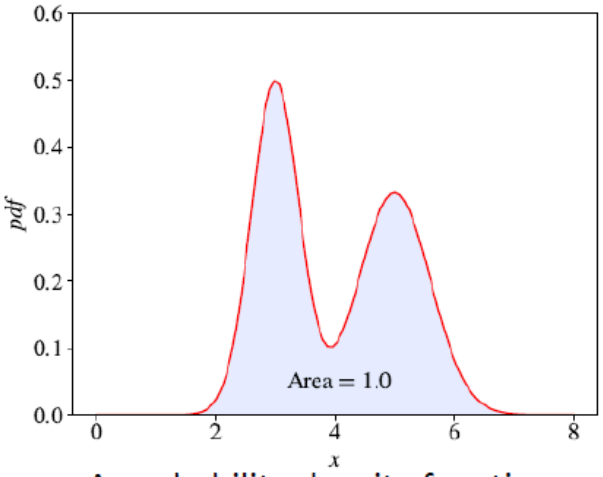
$$\sigma^2 = E[(X - \mu)^2]$$

- For a **discrete random variable**, the standard deviation is given by

$$\sigma = \sqrt{\Pr(X = x_1)(x_1 - \mu)^2 + \cdots + \Pr(X = x_k)(x_k - \mu)^2}$$

  where $\mu = E(X)$.

- A continuous random variable (CRV) takes an infinite number of possible values in some interval.
  - Examples include height, weight, and time.
  - The number of values of a continuous random variable X is infinite, the probability $\Pr(X = c)$ for any c is *0*
  - Therefore, instead of the list of probabilities, the probability distribution of a CRV (a continuous probability distribution) is described by a **probability density function** (pdf).
  - The pdf is a function whose range is nonnegative and the area under the curve is equal to 1.



A probability density function

- The **expectation** of a continuous random variable X is given by $E[x] \stackrel{\text{def}}{=} \int_R x\, f_X(x)\, \mathrm{d}x$
  where f_X is the pdf of the variable X and $\int_R$ is the integral of function $x\, f_X$.

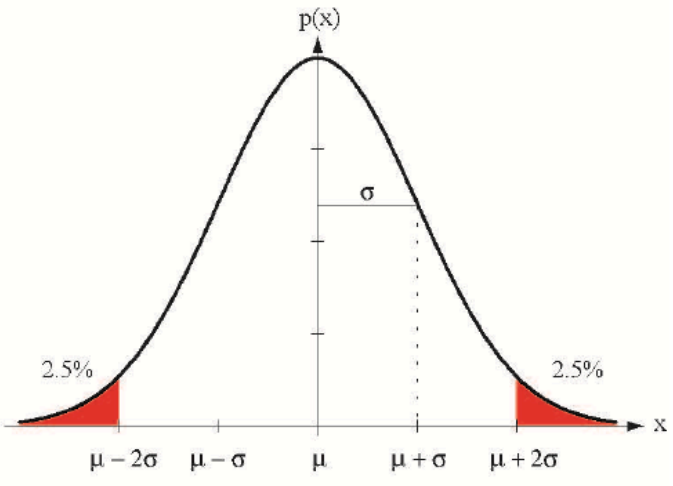
- The **variance** of a continuous random variable X is given by $\sigma^2 \stackrel{\text{def}}{=} \int_R (X-\mu)^2 f_X(x)\, \mathrm{d}x$

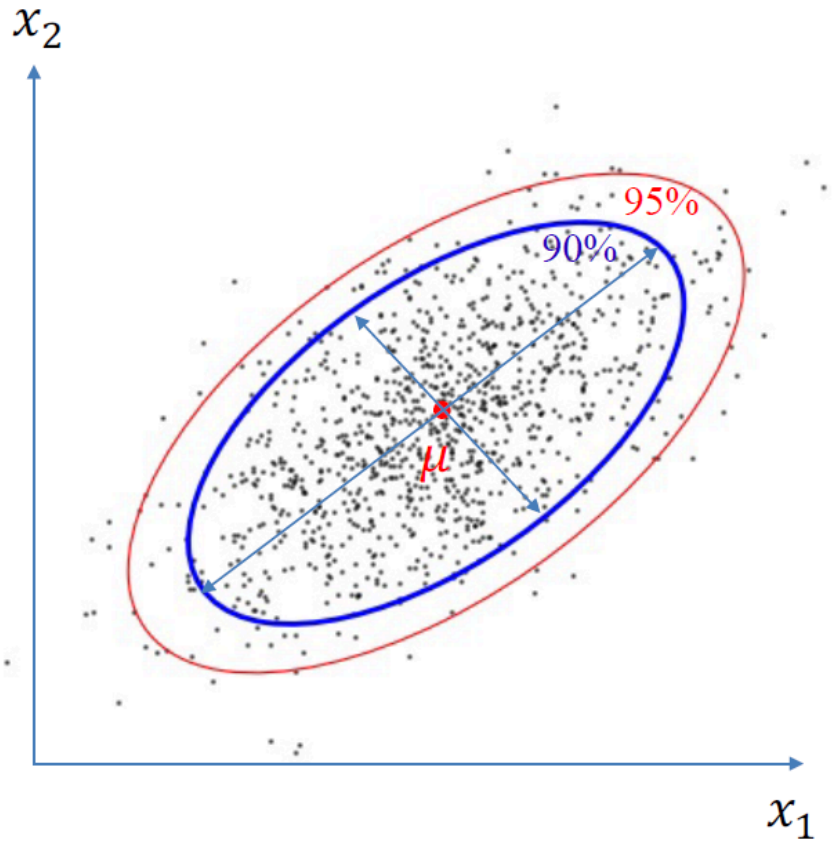
- Integral is an equivalent of the summation over all values of the function when the function has a continuous domain.

- It equals the area under the curve of the function.

- The property of the pdf that the **area under its curve is 1** mathematically means that $\int_R f_X(x)\, \mathrm{d}x = 1$

Mean and Standard Deviation of a Gaussian Distribution





Example 1

- **Independent random variables**
- Consider tossing a fair coin twice, what is the probability of having (H,H)? Assuming a coin has two sides, H=head and T=Tail
  - Pr*(x=H, y=H)* = Pr*(x=H)*Pr*(y=H)* = (1/2)(1/2) = 1/4

Example 2

- **Dependent random variables**
- Given 2 balls with different colors (Red and Black), what is the probability of first drawing B and then R? Assuming we are drawing the balls without replacement.

- The space of outcomes of taking two balls sequentially without replacement:

  B–R, R–B

  – Thus having B-R is 1⁄2 .

- Mathematically:

  – $\Pr(x=B, y=R) = \Pr(y=R \mid x=B) \Pr(x=B) = 1 \times (1/2) = 1/2$

Conditional Probability

Example 3

- **Dependent random variables**
- Given 3 balls with different colors (R,G,B), and we draw 2 balls. What is the probability of first having B and then G, if we draw without replacement?
- The space of outcomes of taking two balls sequentially without replacement:

  R–G | G–B | B–R

  R–B | G–R | B–G Thus, Pr(*y*=G, *x*=B) = 1/6
- Mathematically:

  Pr(*y*=G, *x*=B) = Pr(*y*=G | *x*=B) Pr(*x*=B)
  = (1/2) × (1/3)
  = 1/6

Bayes' Rule

- The conditional probability $\Pr(Y = y|X = x)$ is the probability of the random variable Y to have a specific value y, given that another random variable X has a specific value of x.

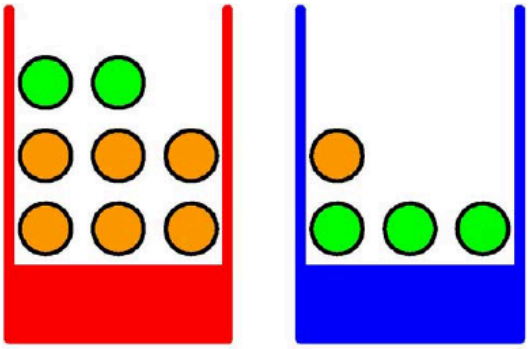
- The **Bayes' Rule** (also known as the **Bayes' Theorem**):

$$\underset{\text{posterior}}{\Pr(Y = y|X = x)} = \frac{\overset{\text{likelihood}}{\Pr(X = x|Y = y)} \overset{\text{prior}}{\Pr(Y = y)}}{\underset{\text{evidence}}{\Pr(X = x)}}$$

Example

- Drawing a sample of fruit from a box
  - First pick a box, and then draw a sample of fruit from it
  - B: variable for Box, can be *r* (red) or *b* (blue)
  - F: variable for Fruit, can be *o* (orange) or *a* (apple)

- $Pr(B=r)=0.4$ prior
- $Pr(F=o \mid B=r)= 0.75$ likelihood
- $Pr(F=o)= 0.45$ evidence



- $Pr(B=r \mid F=o) = Pr(F=o \mid B=r) * Pr(B=r) / Pr(F=o)$

$= 0.75*0.4/0.45 = 2/3$ posterior

Summary and Practice Question

Suppose the random variable X has the following probability mass function (pmf) listed in the table below. *k* is unknown.

X	1	2	3	4	5
Pr[X]	0.1	0.05	0.05	0.6	*k*

What is the probability that X takes a value of odd numbers?

k = 0.2

Pr(X = odd) = 0.35

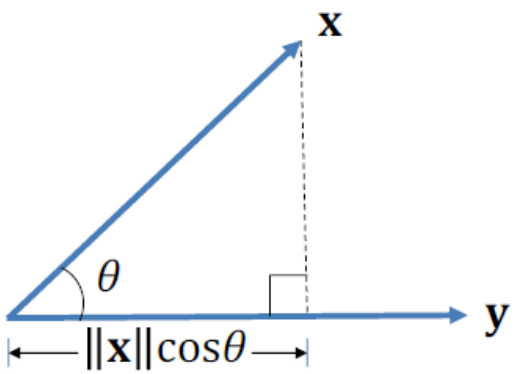
Lec4: Systems of linear equations

Vector and Matrix Operation

Dot Product = Inner Product

Dot Product or Inner Product of Vectors:

$$\mathbf{x} \cdot \mathbf{y} = \mathbf{x}^T \mathbf{y}$$
$$= [x_1 \quad x_2] \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$
$$= x_1\, y_1 + x_2 y_2$$



Geometric definition:

$$\mathbf{x} \cdot \mathbf{y} = \|\mathbf{x}\|\, \|\mathbf{y}\| \cos\theta$$

where θ is the angle between $\mathbf{x}$ and $\mathbf{y}$,
and $\|\mathbf{x}\| = \sqrt{\mathbf{x} \cdot \mathbf{x}}$ is the Euclidean length of vector $\mathbf{x}$

E. g. $\mathbf{a} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \mathbf{c} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ → $\mathbf{a} \cdot \mathbf{c}$ = 2*1 + 3 *0 = 2

requires the orthonormal basis of V to find the projection
1) get by using Gram-Schmidt
2) least square approximation

Orthogonal Projection of a Vector to a Subspace

$S = \{u_1, u_2, \ldots, u_k\}$ is a orthonormal basis for a subspace $V \subseteq \mathbb{R}^n$, if $w \in \mathbb{R}^n$, $w \notin V$

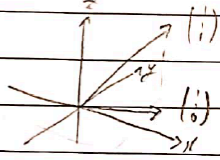
$w_p = (w \cdot u_1)u_1 + (w \cdot u_2)u_2 + \ldots + (w \cdot u_k)u_k$ = projection of w to V * if not orthonormal use the one we learn

· for any orthonormal basis for V, projection of w to V is unique.

Example: $\{\begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}\}$ & $\{\frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\0\end{pmatrix}, \frac{1}{\sqrt{2}}\begin{pmatrix}-1\\0\\0\end{pmatrix}\}$ are orthonormal bases for the x-y plane in $\mathbb{R}^3$

Let $w = \begin{pmatrix}1\\1\\1\end{pmatrix}$ then $w_p = \left(\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\0\\0\end{pmatrix}\right)\begin{pmatrix}1\\0\\0\end{pmatrix} + \left(\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}0\\1\\0\end{pmatrix}\right)\begin{pmatrix}0\\1\\0\end{pmatrix}$ $w_p = \left(\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\0\end{pmatrix}\right)\frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\0\end{pmatrix} + \left(\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\frac{1}{\sqrt{2}}\begin{pmatrix}-1\\0\\0\end{pmatrix}\right)\frac{1}{\sqrt{2}}\begin{pmatrix}-1\\0\\0\end{pmatrix}$

$= \begin{pmatrix}1\\0\\0\end{pmatrix} + \begin{pmatrix}0\\1\\0\end{pmatrix}$ ✓ $= \begin{pmatrix}1\\0\\0\end{pmatrix} + \begin{pmatrix}0\\1\\0\end{pmatrix} = \begin{pmatrix}1\\1\\0\end{pmatrix}$ ✓



- The dot product is the to find the projection of y to x or x to y

Inverse of Matrix

$$AB = \begin{pmatrix} a_1b_1 & a_1b_2 & \cdots & a_1b_n \\ \vdots & \vdots & \ddots & \vdots \\ a_mb_1 & a_mb_2 & \cdots & a_mb_n \end{pmatrix}$$

if A square, $(A|I) \to (R|O)$ R≠I, R has zero row/ non-pivot col

∴ need parameter ∴ non-trivial sol

Inverse of Matrix

inverse: invertible / singular } only for square matrices

for non-singular →

- A square matrix A of order n is invertible if there exists a square matrix B of order n such that $AB = I_n = BA$. ★ Thm. A square matrix is singular if it is not invertible.

if A is invertible

$A\underset{\sim}{x} = \underset{\sim}{b}$

- Uniqueness of inverse - if B & C are both inverses of a square matrix A, then B=C

$A^{-1}A\underset{\sim}{x} = A^{-1}\underset{\sim}{b}$

↳ $BA = I = AC$ so $B = BI = B(AC) = (BA)C = IC = C$ ∴ $B = C$ #

$\underset{\sim}{x} = A^{-1}\underset{\sim}{b}$

↳ since inverse is unique, it is given a notation A^{-1} for inverse of A. $AA^{-1} = I = A^{-1}A$ NOT $\frac{1}{A}$

∴ a unique solution exist & system is ~~correct~~ consistent

- for non-square matrix, if left inverse exists then there is no right inverse and vice versa. Right & Left inverse need not be unique.

↳ if both right & left inverse exist then A must be square & right = left inverse and are unique.

Proof

Cancellation Law for matrices - if A is invertible and $\begin{matrix} AB = AC \\ BA = CA \end{matrix}$ then $\begin{matrix} B = C \\ B = C \end{matrix}$ $AB = AC$, $A^{-1}AB = A^{-1}AC \Rightarrow B = C$ ∴ this only true if A^{-1} exist

i) $AA^{-1} = I = A^{-1}A$ ∴ inverse of $A^{-1} = A$

i) $(A^{-1})^{-1} = A$

v) inverse of symmetric matrix is symmetric: $A = A^T$, $(A^{-1})^T = (A^T)^{-1} = (A)^{-1} = (A^{-1})$

ii) $I = I^T = (AA^{-1})^T$, $I = (A^{-1})^T(A)^T$, $(A^{-1})^T = I(A^T)^{-1}$, $(A^{-1})^T = (A^T)^{-1}$ #

ii) for any non-zero $a \in \mathbb{R}$, (aA) is invertible with inverse $(aA)^{-1} = \frac{1}{a}A^{-1}$

iii) A^T is invertible with inverse $(A^T)^{-1} = (A^{-1})^T$, inverse of the transpose is the transpose of the inverse.

iv) $(AB)^{-1} = B^{-1}A^{-1}$, for $(AB)^{-1}$ to exist A, B must be invertible. $\Rightarrow (A_1A_2 \cdots A_k)^{-1} = A_k^{-1} \cdots A_2^{-1}A_1^{-1}$

$A^{-1} = (A^{-1})^T$ ∴ A^{-1} is symmetric #

$(AB)C = I = C(AB)$

Finding Inverse For 2×2

↳ $C = AB$ C is invertible if AB is invertible

↳ if A, B, C all invertible ABC must be invertible

↳ can explain using elementary matrix

$ABC = I$, $A^{-1}ABC = A^{-1}I$, $BC = A^{-1}$, $B^{-1}BC = B^{-1}A^{-1}$, $C = B^{-1}A^{-1}$ #

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

$$A^{-1} = \frac{1}{\det(A)}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

$\det A = ad - bc$

A^{-1} exist if $ad - bc \neq 0$

Finding Inverse General

inverse X of A satisfies $AX = I$

$\Rightarrow (A|I) \xrightarrow{RREF} (R|B)$

- unique solution: if $R = I$ then $B = A^{-1}$
- no solution: else R has a zero row & determinant of A = 0, A^{-1} doesn't exist.

Determinant computation $\det(\mathbf{A}) = \sum_{j=1}^{k} (-1)^{i+j} a_{ij} M_{ij}$

Example: 3x3 matrix, use the first row (i = 1)

$$|A| = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a \begin{vmatrix} \square & \square & \square \\ \square & e & f \\ \square & h & i \end{vmatrix} - b \begin{vmatrix} \square & \square & \square \\ d & \square & f \\ g & \square & i \end{vmatrix} + c \begin{vmatrix} \square & \square & \square \\ d & e & \square \\ g & h & \square \end{vmatrix}$$

$$= a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix}$$

$$= a(ei - fh) - b(di - fg) + c(dh - eg)$$

Consider a 3×3 matrix

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}.$$

The minor of $a_{22} = \begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix}$

Its cofactor matrix is

$$\mathbf{C} = \begin{pmatrix} +\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \\ -\begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} \\ +\begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{pmatrix}.$$

$$\text{adj}(\mathbf{A}) = \mathbf{C}^{\mathrm{T}}$$

$$\det(\mathbf{A}) = \sum_{j=1}^{k} = a_{ij} C_{ij} = (-1)^{i+j} a_{ij} M_{ij}$$

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})} \text{adj}(\mathbf{A})$$

Systems of Linear Equations

Consider a system of m linear equations with d variables or unknowns $w_1, \ldots, w_d$:

$$\begin{aligned} x_{1,1}w_1 + x_{1,2}w_2 + \cdots + x_{1,d}\, w_d &= y_1 \\ x_{2,1}w_1 + x_{2,2}w_2 + \cdots + x_{2,d}\, w_d &= y_2 \\ &\vdots \\ x_{m,1}w_1 + x_{m,2}w_2 + \cdots + x_{m,d}\, w_d &= y_m . \end{aligned}$$

- m is number of sample
- x is number of feature

These equations can be written compactly in matrix-vector notation:

$$\mathbf{X}\mathbf{w} = \mathbf{y}$$

Where

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,d} \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}.$$

Note:

- The data matrix $\mathbf{X} \in \mathcal{R}^{m \times d}$ and the target vector $\mathbf{y} \in \mathcal{R}^{m}$ are given
- The unknown vector of parameters $\mathbf{w} \in \mathcal{R}^{d}$ is to be learnt

- For number of feature d, is X, and number of sample m
- w is the classifier to map Rd → R, which is y

Types of System

A set of linear equations can have no solution, one solution, or multiple solutions:

$$\mathbf{X}\mathbf{w} = \mathbf{y}$$

Where

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,d} \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}.$$

$\mathbf{X}$ is Square	Even-determined	m = d	Equal number of equations and unknowns
$\mathbf{X}$ is Tall	Over-determined	m > d	More number of equations than unknowns
$\mathbf{X}$ is Wide	Under-determined	m < d	Fewer number of equations than unknowns

$\mathbf{X}$ is Square	Even-determined	m = d	One unique solution in general	$\hat{\mathbf{w}} = \mathbf{X}^{-1}\mathbf{y}$
$\mathbf{X}$ is Tall	Over-determined	m > d	No exact solution in general; An approximated solution	$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ Left-inverse
$\mathbf{X}$ is Wide	Under-determined	m < d	Infinite number of solutions in general; Unique constrained solution	$\hat{\mathbf{w}} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{y}$ Right-inverse

```
28  X = np.array([[1, 2], [0, 6], [1, 0]])
29  leftinverseX = helper.leftInverse(X)
30  print("leftinverseX =\n ", leftinverseX)
31  rightinverseX = helper.rightInverse(X)
32  print("rightinverseX =\n", rightinverseX)
33
34  Xpadded = helper.paddingOfOnes(X)
35  leftinverseX = helper.leftInverse(Xpadded)
36  print("leftinverseX =\n ", leftinverseX)
37  rightinverseX = helper.rightInverse(Xpadded)
38  print("rightinverseX =\n", rightinverseX)
39
40  print("inv(Xpadded) =\n", inv(Xpadded))
```

```
leftinverseX =
  [[ 0.47368421 -0.15789474  0.52631579]
 [ 0.02631579  0.15789474 -0.02631579]]
rightinverseX =
 [[ 0.      0.      1.    ]
 [-0.25    0.1875  0.25  ]]
leftinverseX =
  [[-3.   1.   3. ]
 [ 3.  -1.  -2. ]
 [ 0.5  0.  -0.5]]
rightinverseX =
 [[-3.0000000e+00  1.0000000e+00  3.0000000e+00]
 [ 3.0000000e+00 -1.0000000e+00 -2.0000000e+00]
 [ 5.0000000e-01  4.4408921e-16 -5.0000000e-01]]
inv(Xpadded) =
 [[-3.   1.   3. ]
 [ 3.  -1.  -2. ]
 [ 0.5 -0.  -0.5]]
```

-
  - If square and invertible, left and right inverse give the same thing
- Left inverse for over determined is actually the least square approximation
  - There is no solution in column space of X, hence find the closest one in Col(X)

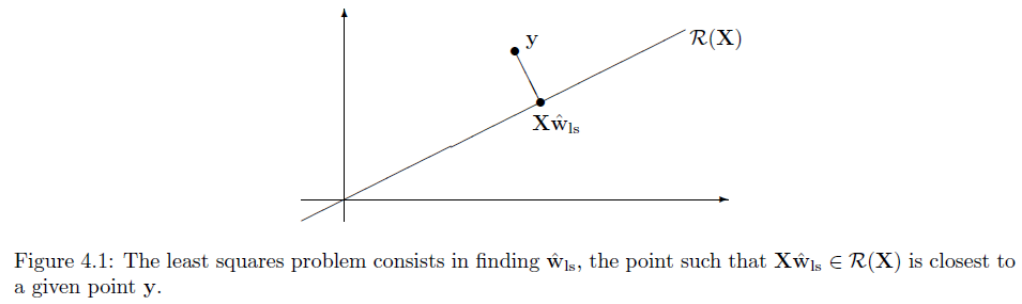


Figure 4.1: The least squares problem consists in finding $\hat{\mathbf{w}}_{\text{ls}}$, the point such that $\mathbf{X}\hat{\mathbf{w}}_{\text{ls}} \in \mathcal{R}(\mathbf{X})$ is closest to a given point $\mathbf{y}$.

- This means X^TX has to have non zero determine OR X is full rank

- Right inverse for under determined is actually the least norm solution
  - There is infinitely many solution, hence find the one closest to origin, having the least norm

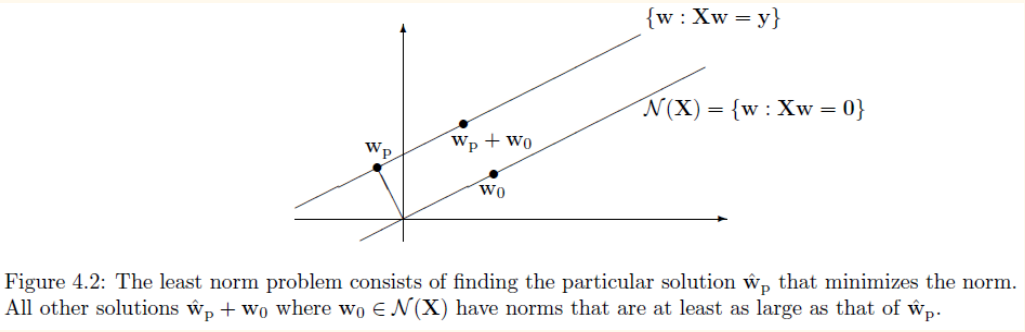


Figure 4.2: The least norm problem consists of finding the particular solution $\hat{\mathbf{w}}_{\text{p}}$ that minimizes the norm. All other solutions $\hat{\mathbf{w}}_{\text{p}} + \mathbf{w}_0$ where $\mathbf{w}_0 \in \mathcal{N}(\mathbf{X})$ have norms that are at least as large as that of $\hat{\mathbf{w}}_{\text{p}}$.

- This means XX^T has to have non zero determine OR X is full rank

The matrix $\mathbf{X} \in \mathbb{R}^{m\times d}$ and vector $\mathbf{y} \in \mathbb{R}^m$ are given and $\mathbf{w} \in \mathbb{R}^d$ is to be found. As mentioned, if the matrix $\mathbf{X}$ is square and full rank, $\mathbf{X}^{-1}$ exists and so we can solve for $\mathbf{w}$ by simple matrix inversion and multiplication $\mathbf{w} = \mathbf{X}^{-1}\mathbf{y}$. However, most of the time in engineering, $m \neq d$ and more care is needed to discuss the existence and uniqueness of solutions to the linear system in (4.6). For this, we appeal to the Rouché-Capelli Theorem. We need the notion of the *augmented matrix*

$$\tilde{\mathbf{X}} = \begin{bmatrix}\mathbf{X} & \mathbf{y}\end{bmatrix} \in \mathbb{R}^{m\times(d+1)}. \tag{4.8}$$

Note that the augmented matrix $\tilde{\mathbf{X}}$ has rank at least as large as that of $\mathbf{X}$, i.e., $\text{rank}(\mathbf{X}) \leq \text{rank}(\tilde{\mathbf{X}})$. This is because $\tilde{\mathbf{X}}$ has more columns than $\mathbf{X}$ so the dimension of its column space must be as large as that of $\mathbf{X}$.

Theorem 4.1 (Rouché-Capelli Theorem). *For the linear system in (4.6), the following hold:*

(i) The system in (4.6) admits a unique solution if and only if

$$\text{rank}(\mathbf{X}) = \text{rank}(\tilde{\mathbf{X}}) = d; \tag{4.9}$$

(ii) The system in (4.6) has no solution if and only if

$$\text{rank}(\mathbf{X}) < \text{rank}(\tilde{\mathbf{X}}); \tag{4.10}$$

(iii) The system in (4.6) has infinitely many solutions if and only if

$$\text{rank}(\mathbf{X}) = \text{rank}(\tilde{\mathbf{X}}) < d. \tag{4.11}$$

Proof sketch (Only the $\Longleftarrow$ directions). For part (i), we note that the condition that $\text{rank}(\mathbf{X}) = \text{rank}(\tilde{\mathbf{X}})$ means that $\mathbf{y}$ is in the column space of $\mathbf{X}$. This means that there exists $\{w_i\}_{i=1}^d \subset \mathbb{R}$ such that $\sum_{i=1}^d w_i \underline{x}_i = \mathbf{y}$ where the $\underline{x}_i$'s are the d columns of $\mathbf{X}$. Since $\text{rank}(\mathbf{X}) = d$, $\{\underline{x}_i\}_{i=1}^d$ span $\mathbb{R}^d$ and the representation $\sum_{i=1}^d w_i \underline{x}_i = \mathbf{y}$ is unique (see discussion after Definition 4.3), which means there is a unique solution.

For part (ii), the condition that $\text{rank}(\mathbf{X}) < \text{rank}(\tilde{\mathbf{X}})$ means that $\mathbf{y}$ is not in the column space of $\mathbf{X}$ so there is no solution.

For part (iii), since $\text{rank}(\mathbf{X}) < d$, $\{\underline{x}_i\}_{i=1}^d$ do not span $\mathbb{R}^d$ and the dimension of the nullspace of $\mathbf{X}$ is non-zero. This means that if $\hat{\mathbf{w}}_\text{p}$ is a particular solution so is $\hat{\mathbf{w}}_\text{p} + \mathbf{w}_0$ where $\mathbf{w}_0 \in \mathcal{N}(\mathbf{X})$. Hence, are infinitely many solutions. □

Square or even-determined system

1. Square or even-determined system: $m = d$

- Equal number of equations and unknowns, i.e., $\mathbf{X} \in \mathcal{R}^{d \times d}$
- **One unique solution** if $\mathbf{X}$ is invertible or all rows/columns of $\mathbf{X}$ are linearly independent
- If all rows or columns of $\mathbf{X}$ are linearly independent, then $\mathbf{X}$ is invertible.

Solution:

If $\mathbf{X}$ is invertible (or $\mathbf{X}^{-1}\mathbf{X} = \mathbf{I}$), then pre-multiply both sides by $\mathbf{X}^{-1}$

$$\mathbf{X}^{-1}\mathbf{X}\,\mathbf{w} = \mathbf{X}^{-1}\mathbf{y}$$

$$\Rightarrow \quad \hat{\mathbf{w}} = \mathbf{X}^{-1}\mathbf{y}$$

(Note: we use a *hat* on top of $\mathbf{w}$ to indicate that it is a specific point in the space of $\mathbf{w}$)

- Number of sample, m = number of features, d

Example 1

$$w_1 + w_2 = 4 \quad (1)$$
$$w_1 - 2w_2 = 1 \quad (2)$$

Two unknowns
Two equations

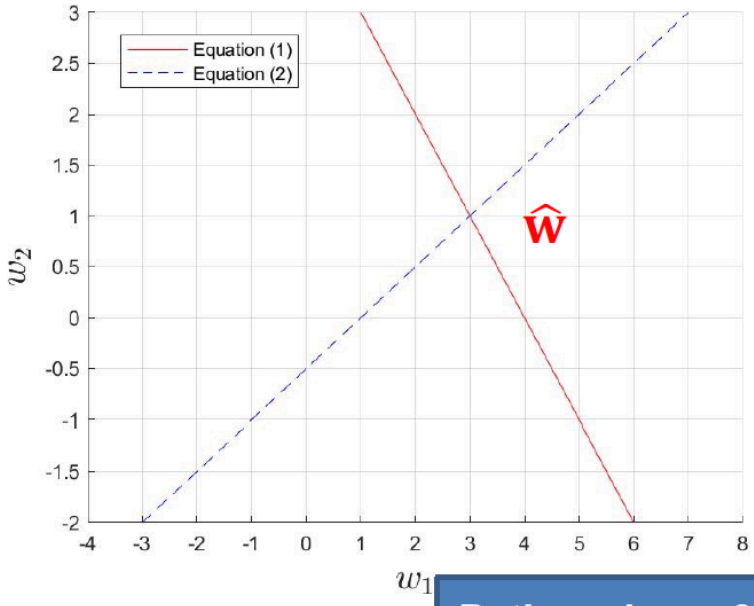
$$\mathbf{X} \quad \mathbf{w} \quad \mathbf{y}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \end{bmatrix}$$

$$\hat{\mathbf{w}} = \mathbf{X}^{-1}\mathbf{y}$$

$$= \begin{bmatrix} 1 & 1 \\ 1 & -2 \end{bmatrix}^{-1} \begin{bmatrix} 4 \\ 1 \end{bmatrix}$$

$$= \frac{-1}{3} \begin{bmatrix} -2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$



Python demo 3

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})}\text{adj}(\mathbf{A}) \qquad \text{adj}(\mathbf{A}) = \mathbf{C}^T = \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \qquad \det(\mathbf{A}) = ad - bc$$

2. Over-determined system: $m > d$

- More equations than unknowns
- $\mathbf{X}$ is non-square (tall) and hence not invertible
- **Has no exact solution in general** *
- An **approximated solution** is available using the left inverse

If the **left-inverse** of $\mathbf{X}$ exists such that $\mathbf{X}^{\dagger}\mathbf{X} = \mathbf{I}$, then pre-multiply both sides by $\mathbf{X}^{\dagger}$ results in

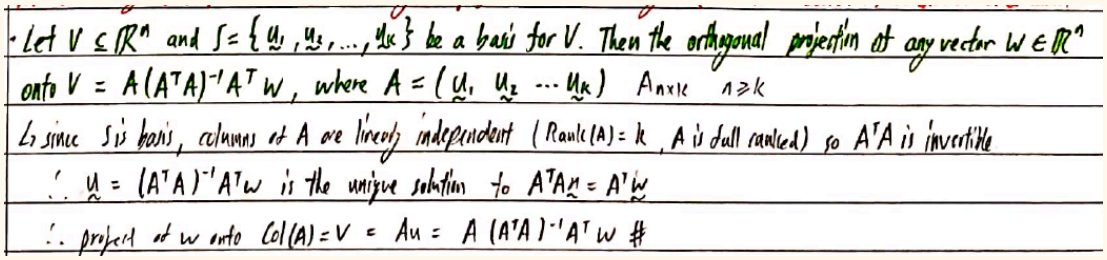
$$\mathbf{X}^{\dagger}\mathbf{X}\,\mathbf{w} = \mathbf{X}^{\dagger}\mathbf{y}$$
$$\Rightarrow \hat{\mathbf{w}} = \mathbf{X}^{\dagger}\mathbf{y}$$

Definition:

A matrix **B** that satisfies $\boldsymbol{B}_{d\,x\,m}\boldsymbol{A}_{m\,x\,d} = \mathbf{I}$ is called a **left-inverse** of $\mathbf{A}$.

The **left-inverse** of $\mathbf{X}$: $\mathbf{X}^{\dagger} = (\mathbf{X}^{T}\mathbf{X})^{-1}\mathbf{X}^{T}$ given $\mathbf{X}^{T}\mathbf{X}$ is invertible.

Note: * exception: when rank($\mathbf{X}$) = rank([$\mathbf{X}$,$\mathbf{y}$]), there is a solution.

- Number of sample, m > number of features, d
- No exact solution in general, only approximation (least square solution)
  - 

  - Except for rank(X) = rank([X, y]) which just means y is in the column space of X, hence there is an exact solution
- Lots of noisy measurement/sample to find something

Case 1 — Full Rank Matrices

Rank = no of col

Let A be $m \times n$, $m \geq n$. Then A has full rank, rank(A) = column = n, iff RREF(A) has the form:

$$R = \begin{pmatrix} I_n \\ 0_{(m-n)\times n} \end{pmatrix}$$

← n pivot columns, n non-zero rows, $m-n$ zero rows

1) rank$(A) = n$.

2) The rows of A spans $\mathbb{R}^n$, Row$(A) = \mathbb{R}^n$.

3) The columns of A are linearly independent.

4) Homogeneous system $A\underset{\sim}{x} = \underset{\sim}{0}$ has only the trivial solution, Null$(A) = \{\underset{\sim}{0}\}$.

5) A^TA is an invertible matrix of order n.

6) A has a left inverse. $= (A^TA)^{-1}A^T$

⑤→⑥: $I = (A^TA)^{-1}A^TA = ((A^TA)^{-1}A^T)A$ ⇒ left inverse of $A = (A^TA)^{-1}A^T$ #

⑥→⑤: Suppose C is a left inverse of A, then if $\underset{\sim}{u} \in \mathbb{R}^n$ such that $A\underset{\sim}{u} = \underset{\sim}{0}$,

$\underset{\sim}{u} = I\underset{\sim}{u} = CA\underset{\sim}{u} = C\underset{\sim}{0} = \underset{\sim}{0}$ ⇒ nullspace of A is trivial

since null(A^TA) = null(A) ⇒ nullspace of A^TA is trivial ⇒ A^TA is invertible #

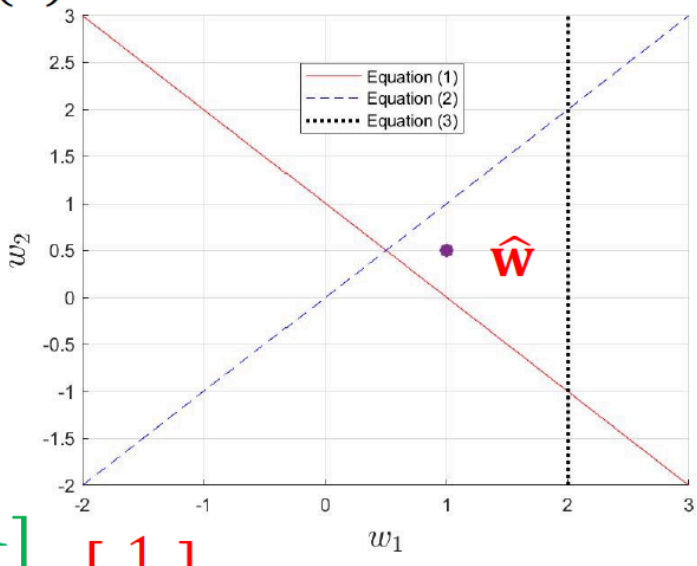
- to many samples for little features, cannot find inverse
  - w cannot fit the constraint needed
  - so need to be an approximation

Example 2

$$w_1 + w_2 = 1 \quad (1)$$
$$w_1 - w_2 = 0 \quad (2)$$
$$w_1 = 2 \quad (3)$$

Two unknowns
Three equations

$$\underset{\mathbf{X}}{\begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 0 \end{bmatrix}} \underset{\mathbf{w}}{\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}} = \underset{\mathbf{y}}{\begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}}$$



No exact solution

Approximated solution

$$\hat{\mathbf{w}} = \mathbf{X}^{\dagger}\mathbf{y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$= \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}$$

$\mathbf{X}^T\mathbf{X}$ is invertible

Python demo 4

- Consider the following over-determined system in which $m = 3$ and $d = 2$:

$$\mathbf{X} = \begin{bmatrix} 2 & 1 \\ 4 & 3 \\ 5 & 6 \end{bmatrix}, \qquad \text{and} \qquad \mathbf{y} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}. \tag{4.12}$$

The augmented matrix is

$$\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X} & \mathbf{y} \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 \\ 4 & 3 & 2 \\ 5 & 6 & 3 \end{bmatrix}. \tag{4.13}$$

In this case $\text{rank}(\mathbf{X}) = 2$ and $\text{rank}(\tilde{\mathbf{X}}) = 3$. This is case (ii) of the Rouché-Capelli Theorem and there is no solution. This is the usual case for over-determined systems. Note that in Python, you can find the rank of a matrix (2D array) $\mathbf{A}$ using `np.linalg.matrix_rank(A)`.

Consider the following over-determined system in which $m = 3$ and $d = 2$:

$$\mathbf{X} = \begin{bmatrix} 2 & 1 \\ 4 & 3 \\ 5 & 6 \end{bmatrix}, \qquad \text{and} \qquad \mathbf{y} = \begin{bmatrix} 4 \\ 10 \\ 17 \end{bmatrix}. \tag{4.14}$$

In this case $\text{rank}(\mathbf{X}) = 2$ and $\text{rank}(\tilde{\mathbf{X}}) = 2$. This is case (i) of the Rouché-Capelli Theorem and there is a unique solution even though the system is over-determined. Note that $\mathbf{y}$ is one times the first column of $\mathbf{X}$ plus two times the second column of $\mathbf{X}$, so it is in the linear span of the columns of $\mathbf{X}$.

Consider the following over-determined system in which $m = 3$ and $d = 2$:

$$\mathbf{X} = \begin{bmatrix} 2 & 1 \\ 4 & 2 \\ 6 & 3 \end{bmatrix}, \qquad \text{and} \qquad \mathbf{y} = \begin{bmatrix} 8 \\ 16 \\ 24 \end{bmatrix}. \tag{4.15}$$

In this case $\text{rank}(\mathbf{X}) = 1$ and $\text{rank}(\tilde{\mathbf{X}}) = 1$ and both these ranks are less than $d = 2$. This is case (iii) of the Rouché-Capelli Theorem and there are infinitely many solutions even though the system is over-determined. Note that the three columns of $\tilde{\mathbf{X}}$ are collinear.

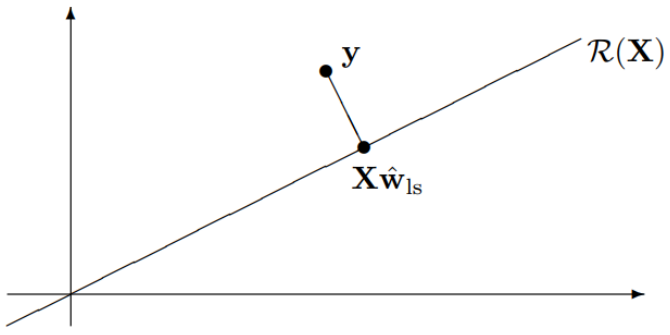


Figure 4.1: The least squares problem consists in finding $\hat{\mathbf{w}}_{\text{ls}}$, the point such that $\mathbf{X}\hat{\mathbf{w}}_{\text{ls}} \in \mathcal{R}(\mathbf{X})$ is closest to a given point $\mathbf{y}$.

Under-determined system

3. Under-determined system: $m < d$

– More unknowns than equations

– Infinite number of solutions in general *

If the **right-inverse** of $\mathbf{X}$ exists such that $\mathbf{X}\mathbf{X}^{\dagger} = \mathbf{I}$, then the d-vector $\mathbf{w} = \mathbf{X}^{\dagger}\mathbf{y}$ (one of the infinite cases) satisfies the equation $\mathbf{X}\mathbf{w} = \mathbf{y}$, i.e.,

$$\mathbf{X}\mathbf{w} = \mathbf{y} \quad \Rightarrow \quad \mathbf{X}\mathbf{X}^{\dagger}\mathbf{y} = \mathbf{y}$$
$$\Rightarrow \quad \mathbf{I}\mathbf{y} = \mathbf{y}$$

Definition:

A matrix **B** that satisfies $\mathbf{A}_{m\,x\,d}\mathbf{B}_{d\,x\,m} = \mathbf{I}$ is called a **right-inverse** of $\mathbf{A}$.
The **right-inverse** of $\mathbf{X}$: $\mathbf{X}^{\dagger} = \mathbf{X}^{T}(\mathbf{X}\mathbf{X}^{T})^{-1}$ given $\mathbf{X}\mathbf{X}^{T}$ is invertible.

If $\mathbf{X}$ is right−invertible, we can find a unique constrained solution.

Note: * exception: no solution if the system is inconsistent rank($\mathbf{X}$) < rank([$\mathbf{X}$,$\mathbf{y}$]

- Number of sample, m < number of features, d
  - Few patients but uses high resolution MRI
- Infinite number of solution in general
  - But can find unique constraint solution

A unique solution is yet possible by constraining the search using $\mathbf{w} = \mathbf{X}^{T}\mathbf{a}$

If $\mathbf{X}\mathbf{X}^{T}$ is invertible, let $\mathbf{w} = \mathbf{X}^{T}\mathbf{a}$, then

$$\mathbf{X}\mathbf{X}^{T}\mathbf{a} = \mathbf{y}$$
$$\Rightarrow \quad \hat{\mathbf{a}} = (\mathbf{X}\mathbf{X}^{T})^{-1}\mathbf{y}$$
$$\Rightarrow \hat{\mathbf{w}} = \mathbf{X}^{T}\hat{\mathbf{a}} = \underbrace{\mathbf{X}^{T}(\mathbf{X}\mathbf{X}^{T})^{-1}}_{\mathbf{X}^{\dagger}}\mathbf{y}$$

right-inverse

- By constraint w to be the form of $X^{T}a$, then from infinite solution to unique solution
- This is finding the least norm solution

  - Or exception when rank(X) < rank([X, y]), meaning y is linearly independent of the column space of X, so confirm no solution

Case 2

Rank = no of rows

Let A be $m \times n$, $n \geq m$. Then A has a full rank, rank(A) = no of rows = m, iff R = RREF has the form:

$R = \begin{pmatrix} 0 \cdots 1 \cdots 0 \cdots 0 \cdots \\ 0 \cdots 0 \cdots 1 \cdots 0 \cdots \\ 0 \cdots 0 \cdots 0 \cdots 1 \cdots \\ \vdots \quad \vdots \quad \vdots \quad \vdots \\ 0 \cdots 0 \cdots 0 \cdots 0 \cdots \end{pmatrix}$ ← m non-zero rows, m pivot column, n-m non-pivot column

* columns before the 1st pivot column can be zero column also ⇒ # col in RREF ≥ 1

full rank = 3: $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

1) rank(A) = m.

2) The columns of A spans $\mathbb{R}^m$, $Col(A) = \mathbb{R}^m$. ← R has m pivot column with m entries ⇒ spans $\mathbb{R}^m$

3) The rows of A are linearly independent. ← All non-zero

4) The linear system $A\underset{\sim}{x} = \underset{\sim}{b}$ is consistent for every $\underset{\sim}{b} \in \mathbb{R}^m$.

5) AA^T is an invertible matrix of order m.

6) A has a right inverse. = $A^T(AA^T)^{-1}$

⑤→④: $AA^T(AA^T)^{-1} = I \Rightarrow A^T(AA^T)^{-1}$ is the right inverse of A. For any $\underset{\sim}{b}$, $A\left(A^T(AA^T)^{-1}\underset{\sim}{b}\right) = \underset{\sim}{b}$, $\underset{\sim}{x} = A^T(AA^T)^{-1}\underset{\sim}{b}$ ∴ consistent ✓

④→⑤: Suppose $A\underset{\sim}{x} = \underset{\sim}{b}$ is consistent for every $\underset{\sim}{b} \in \mathbb{R}^m$. Let $\underset{\sim}{b_i}$ be a solution to $A\underset{\sim}{x} = \underset{\sim}{e_i}$, $\underset{\sim}{e_i}$ = i-th vector in the standard basis, for i = 1, ..., m.

Let $B = (\underset{\sim}{b_1}\ \underset{\sim}{b_2} \cdots \underset{\sim}{b_m})$, then $AB = A(\underset{\sim}{b_1}\ \underset{\sim}{b_2} \cdots \underset{\sim}{b_m}) = \underset{\sim}{e_1}\ \underset{\sim}{e_2} \cdots \underset{\sim}{e_m} = I_m \Rightarrow$ ∴ B is a right inverse of A ✓

- To satisfy both case 1 & 2, A must be a square matrix with R = I. → RREF (full rank square matrix) = I
- Case 1 & 2 are mutually exclusive for non-square matrix ⇒ can have at most a left or a right inverse not both.

4: if r > c, then can't have right inverse: if A has right inverse, then rank A = r but r > c ∴ rank A ≤ r

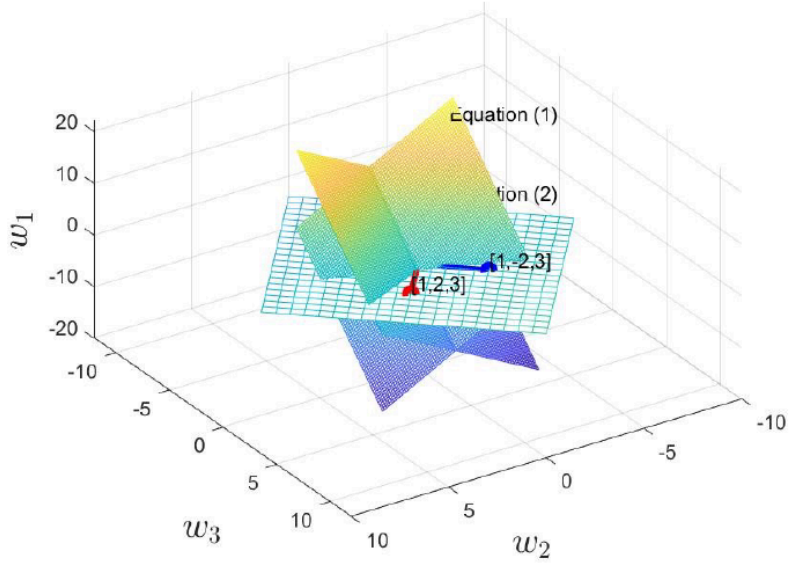
- Too many features, too little samples
  - too many possible solution that is possible and most are not good

Example 3

$$w_1 + 2w_2 + 3w_3 = 2 \quad (1)$$
$$w_1 - 2w_2 + 3w_3 = 1 \quad (2)$$

Three unknowns
Two equations

$$\underset{\mathbf{X}}{\begin{bmatrix} 1 & 2 & 3 \\ 1 & -2 & 3 \end{bmatrix}} \underset{\mathbf{w}}{\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}} = \underset{\mathbf{y}}{\begin{bmatrix} 2 \\ 1 \end{bmatrix}}$$



Infinitely many solutions along the intersection line

Here $\mathbf{X}\mathbf{X}^T$ is invertible

$$\hat{\mathbf{w}} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{y}$$

$$= \begin{bmatrix} 1 & 1 \\ 2 & -2 \\ 3 & 3 \end{bmatrix} \begin{bmatrix} 14 & 6 \\ 6 & 14 \end{bmatrix}^{-1} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.15 \\ 0.25 \\ 0.45 \end{bmatrix}$$

Constrained solution

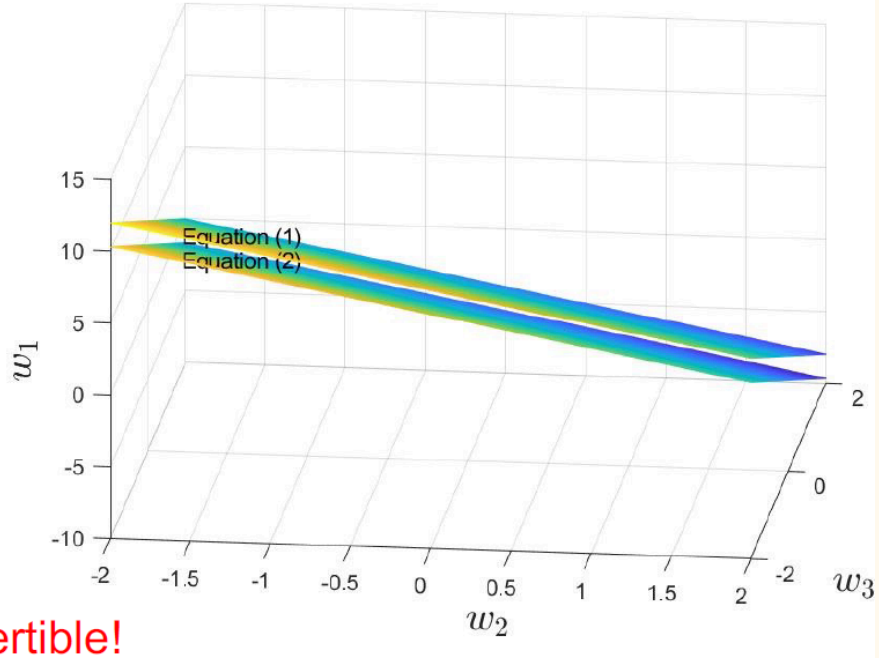
- The two original planes, intersect in a line, infinite solution
- Putting in the constraint, introduce one more plane and hence only one point of intersection

Example 4

$$w_1 + 2w_2 + 3w_3 = 2 \quad (1)$$
$$3w_1 + 6w_2 + 9w_3 = 1 \quad (2)$$

Three unknowns
Two equations

$$\begin{matrix} \mathbf{X} & & \mathbf{w} & \mathbf{y} \end{matrix}$$
$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 6 & 9 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$



Both $\mathbf{X}\mathbf{X}^T$ and $\mathbf{X}^T\mathbf{X}$ are not invertible!

There is no solution for the system

- Let A be m×n, n ≥ m. Then A has a full rank, rank(A) = no of rows = m, iff R = RREF has the form:

$$R = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 & \cdots \\ 0 & \cdots & 1 & \cdots & 0 & \cdots & 0 & \cdots \\ 0 & \cdots & 0 & \cdots & 1 & \cdots & 0 & \cdots \\ \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 & \cdots \end{pmatrix}$$

← m non-zero rows, m pivot columns, n-m non-pivot column

* columns before the 1st pivot column can be zero column also

- X is not full rank so cannot find right inverse

Consider the following under-determined system in which $m = 2$ and $d = 3$:

$$\mathbf{X} = \begin{bmatrix} 2 & 1 & 3 \\ 4 & 2 & 5 \end{bmatrix}, \qquad \text{and} \qquad \mathbf{y} = \begin{bmatrix} 10 \\ 7 \end{bmatrix}. \tag{4.16}$$

In this case, $\text{rank}(\mathbf{X}) = 2$ and $\text{rank}(\tilde{\mathbf{X}}) = 2$ but $d = 3$. This is case (iii) of the Rouché-Capelli Theorem and there are infinitely many solutions. This is the usual case for under-determined systems.

Consider the following under-determined system in which $m = 2$ and $d = 3$:

$$\mathbf{X} = \begin{bmatrix} 2 & 1 & 3 \\ 4 & 2 & 6 \end{bmatrix}, \qquad \text{and} \qquad \mathbf{y} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}. \tag{4.17}$$

In this case, $\text{rank}(\mathbf{X}) = 1$ and $\text{rank}(\tilde{\mathbf{X}}) = 2$ because $\mathbf{y} \notin \mathcal{R}(\mathbf{X})$. This is case (ii) of the Rouché-Capelli Theorem and there is no solution. Note that $\mathbf{y}$ boosts the rank of $\mathbf{X}$ by 1 in the augmented matrix $\tilde{\mathbf{X}}$, i.e., $\mathbf{y}$ is not in the column space of $\mathbf{X}$, which is the ray $\{[t, 2t]^\top : t \in \mathbb{R}\}$.

Least Norm Solution

4.4 Least Norm Solution for $m < d$

Now we consider the case in which $\mathbf{X}$ is wide ($m < d$) and full rank. This means that $\text{rank}(\mathbf{X}) = m$; equivalently, all rows are linearly independent (full row rank). This *under-determined* situation also occurs a lot in engineering.

Example 4.3. *For example, in control engineering (a field of study within mechanical and electrical engineering), one often considers the following discrete-time state-space system (e.g., describing the dynamics of a robot operating over a quantized time interval):*

$$v_{i+1} = av_i + bw_i, \quad i = 0, 1, \ldots, d-1, \tag{4.21}$$

where v_i is the state *of the system at time i and w_i is our* control. *We assume the system starts at the origin $v_0 = 0$. We desire to design the w_i's such that the terminal state $v_d = y$ (for some given y) while minimizing the* cost *of the control $\sum_{i=0}^{d-1} w_i^2$. After some algebra, this can be rewritten as*

$$\textit{minimize} \quad \|\mathbf{w}\|^2 \quad \textit{subject to} \quad y = \begin{bmatrix} b & ab & a^2b & \cdots & a^{d-1}b \end{bmatrix} \begin{bmatrix} w_{d-1} \\ w_{d-2} \\ \vdots \\ w_0 \end{bmatrix}. \tag{4.22}$$

*This is exactly an under-determined problem if we make the identifications $\mathbf{X} = \begin{bmatrix} b & ab & a^2b & \cdots & a^{d-1}b \end{bmatrix}$ (for obvious reasons, this is called the d-*step reachability matrix*) and $\mathbf{w} = \begin{bmatrix} w_{d-1} & w_{d-2} & \cdots & w_0 \end{bmatrix}^\top$ and y is scalar (i.e., $m = 1$).*

We return to the equation $\mathbf{X}\mathbf{w} = \mathbf{y}$ in which $m < d$ and the matrix $\mathbf{X}$ has full row rank. It is clear that $(\mathbf{X}\mathbf{X}^\top)^{-1}$ exists and

$$\hat{\mathbf{w}}_\mathrm{p} = \mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{y} \tag{4.23}$$

is a solution to the equation (check!). The subscript p indicates that this is a *particular* solution to the linear system. From the usual case of the Rouché-Capelli Theorem (case (iii)), we know that there are infinitely many solutions. Where are these infinitely many solutions? Let $\mathbf{w}_0 \in \mathcal{N}(\mathbf{X})$ be any vector in the nullspace of $\mathbf{X}$. Note that the nullspace has positive dimension because $\mathrm{rank}(\mathbf{X}) = m < d$ so $\mathbf{w}_0$ can be chosen to be a non-zero vector. Then $\hat{\mathbf{w}}_\mathrm{p} + \mathbf{w}_0$ is also a solution to (4.6) (check!). In the following, we argue that among all the solutions, $\hat{\mathbf{w}}_\mathrm{p}$ has a special place in our hearts because is the *least norm solution* to (4.6).

Suppose that $\mathbf{w}$ is any solution to $\mathbf{X}\mathbf{w} = \mathbf{y}$. Then since $\hat{\mathbf{w}}_\mathrm{p}$ is also a solution, we have $\mathbf{X}(\mathbf{w} - \hat{\mathbf{w}}_\mathrm{p}) = \mathbf{0}$ and

$$(\mathbf{w} - \hat{\mathbf{w}}_\mathrm{p})^\top \hat{\mathbf{w}}_\mathrm{p} \overset{(4.23)}{=} (\mathbf{w} - \hat{\mathbf{w}}_\mathrm{p})^\top \mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{y} = \big(\mathbf{X}(\mathbf{w} - \hat{\mathbf{w}}_\mathrm{p})\big)^\top(\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{y} = \mathbf{0}. \tag{4.24}$$

This means that $\mathbf{w} - \hat{\mathbf{w}}_\mathrm{p}$ is orthogonal to $\hat{\mathbf{w}}_\mathrm{p}$. By the Pythagorean theorem,

$$\|\mathbf{w}\|^2 = \|(\mathbf{w} - \hat{\mathbf{w}}_\mathrm{p}) + \hat{\mathbf{w}}_\mathrm{p}\|^2 = \|\mathbf{w} - \hat{\mathbf{w}}_\mathrm{p}\|^2 + \|\hat{\mathbf{w}}_\mathrm{p}\|^2 \geq \|\hat{\mathbf{w}}_\mathrm{p}\|^2, \tag{4.25}$$

which shows that $\hat{\mathbf{w}}_\mathrm{p}$ is the least norm solution to $\mathbf{X}\mathbf{w} = \mathbf{y}$, so we also denote this as

$$\hat{\mathbf{w}}_\mathrm{ln} = \mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{y} \tag{4.26}$$

See the geometry of the least norm problem in Fig. 4.2.

Finally, we say a few words about the matrix $\mathbf{X}^\dagger = \mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top)^{-1}$. This matrix is called the *pseudo-inverse* of the full-rank wide matrix $\mathbf{X}$. It is also know as the *right-inverse* of $\mathbf{X}$ (why?). The right-inverse of a matrix with full row rank can be implemented in Python as `np.linalg.pinv(X)`.

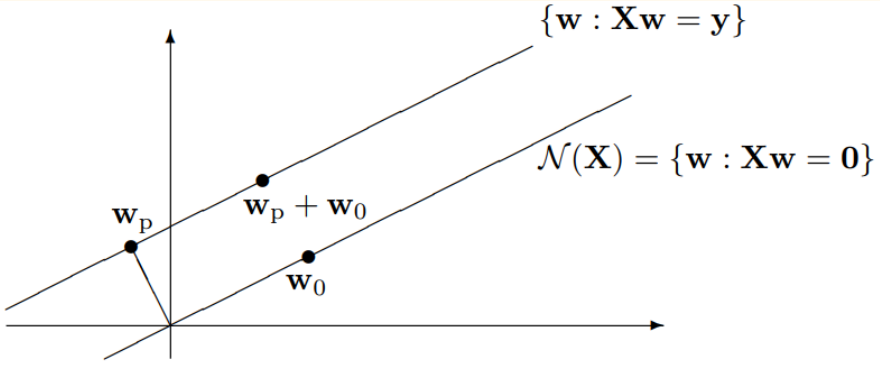


Figure 4.2: The least norm problem consists of finding the particular solution $\hat{\mathbf{w}}_\mathrm{p}$ that minimizes the norm. All other solutions $\hat{\mathbf{w}}_\mathrm{p} + \mathbf{w}_0$ where $\mathbf{w}_0 \in \mathcal{N}(\mathbf{X})$ have norms that are at least as large as that of $\hat{\mathbf{w}}_\mathrm{p}$.

Set

- A **set** is an unordered collection of unique elements
  - Denoted as a calligraphic capital character e.g., $\mathcal{S}, \mathcal{R}, \mathcal{N}$ etc
  - When an element x belongs to a set $\mathcal{S}$, we write $x \in \mathcal{S}$
- A set of numbers can be finite - include a fixed amount of values
  - Denoted using accolades, e.g. {1, 3, 18, 23, 235} or $\{x_1, x_2, x_3, x_4, \ldots, x_d\}$
- A set can be infinite and include all values in some interval
  - If a set of real numbers includes all values between *a* and *b*, including *a* and *b*, it is denoted using square brackets as [*a*, *b*]
  - If the set does not include the values *a* and *b*, it is denoted using parentheses as (*a*, *b*)
- Examples:
  - The special set denoted by $\mathcal{R}$ includes all real numbers from minus infinity to plus infinity
  - The set [0, 1] includes values like 0, 0.0001, 0.25, 0.9995, and 1.0

- **Intersection** of two sets:

$$\mathcal{S}_3 \leftarrow \mathcal{S}_1 \cap \mathcal{S}_2$$

Example: {1,3,5,8} ∩ {1,8,4} = {1,8}

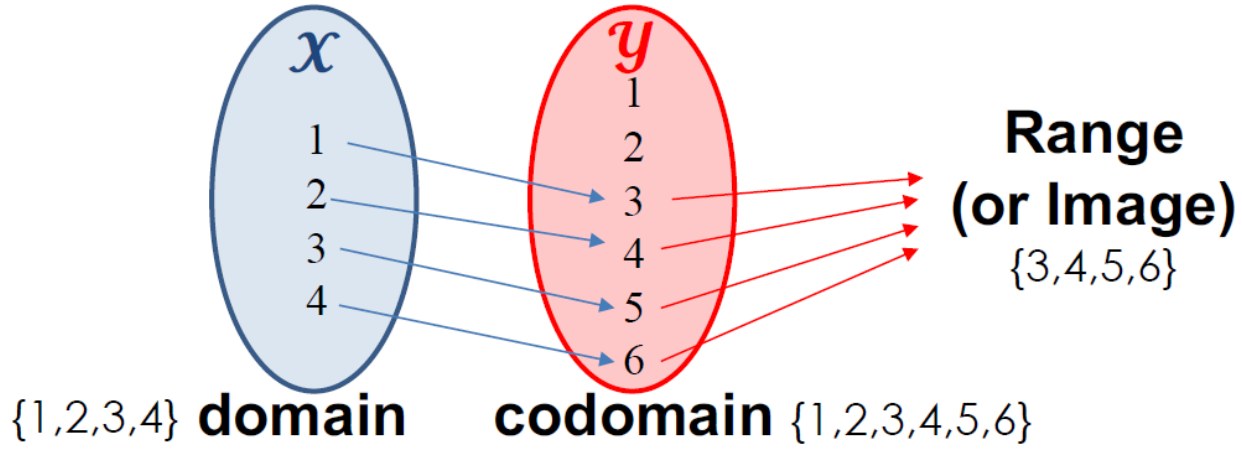
- **Union** of two sets:

$$\mathcal{S}_3 \leftarrow \mathcal{S}_1 \cup \mathcal{S}_2$$

Example: {1,3,5,8} ∪ {1,8,4} = {1,3,4,5,8}

Function

- A **function** is a relation that associates each element x of a set $\mathcal{X}$, the **domain** of the function, to a single element y of another set $\mathcal{Y}$, the **codomain** of the function
- If the function is called f, this relation is denoted $y = f(x)$
  - The element x is the argument or input of the function
  - y is the value of the function or the output
- The symbol used for representing the input is the variable of the function
  - $f(x)$ f is a function of the variable x; $f(x, w)$ f is a function of the variable x and w



- A scalar function can have vector argument
  - E.g. $y = f(\mathbf{x}) = x_1 + x_2 + 2x_3$
- A vector function, denoted as $\mathbf{y} = \mathbf{f}(\mathbf{x})$ is a function that returns a vector $\mathbf{y}$
  - Input argument can be a **vector** $\mathbf{y} = \mathbf{f}(\mathbf{x})$ or a **scalar** $\mathbf{y} = \mathbf{f}(x)$
  - E.g. $\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} -x_1 \\ x_2 \end{bmatrix}$
  - E.g. $\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} -2x_1 \\ 3x_1 \end{bmatrix}$

- The notation $f: \mathcal{R}^d \rightarrow \mathcal{R}$ means that f is a function that maps real d-vectors to real numbers
  - i.e., f is a scalar-valued function of d-vectors
- If $\mathbf{x}$ is a d-vector argument, then $f(\mathbf{x})$ denotes the value of the function f at $\mathbf{x}$
  - i.e., $f(\mathbf{x}) = f(x_1, x_2, \ldots, x_d)$, $\mathbf{x} \in \mathcal{R}^d$, $f(\mathbf{x}) \in \mathcal{R}$

- Example: we can define a function $f: \mathcal{R}^4 \rightarrow \mathcal{R}$ by

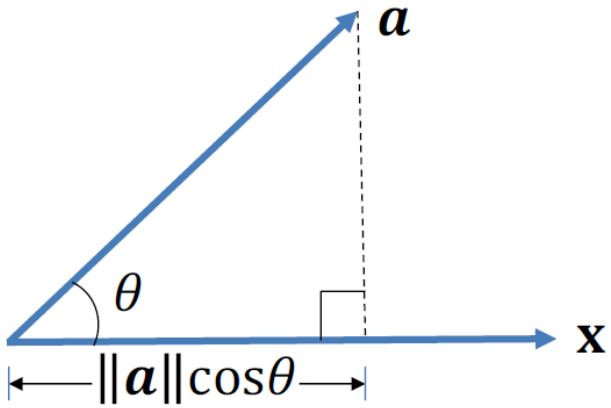
$$f(\mathbf{x}) = x_1 + x_2 - x_4^2$$

The inner product function

- Suppose $\boldsymbol{a}$ is a d-vector. We can define a scalar valued function f of d-vectors, given by

$$f(\mathbf{x}) = \boldsymbol{a}^T\mathbf{x} = a_1x_1 + a_2x_2 + \cdots a_dx_d \qquad (1)$$

  for any d-vector $\mathbf{x}$
- The inner product of its d-vector argument $\mathbf{x}$ with some (fixed) d-vector $\boldsymbol{a}$
- We can also think of f as forming a **weighted sum** of the elements of $\mathbf{x}$; the elements of $\boldsymbol{a}$ give the weights



- Weighted sum, a can be called weight or coefficient
- a is to be found to map x to a scalar

A function $f: \mathcal{R}^d \rightarrow \mathcal{R}$ is **linear** if it satisfies the following two properties:

- **Homogeneity**
  - For any d-vector $\mathbf{x}$ and any scalar α, $f(\alpha\mathbf{x}) = \alpha f(\mathbf{x})$
  - Scaling the (vector) argument is the same as scaling the function value
- **Additivity**
  - For any d-vectors $\mathbf{x}$ and $\mathbf{y}$, $f(\mathbf{x}+\mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$
  - Adding (vector) arguments is the same as adding the function values

Superposition and linearity

- The inner product function $f(\mathbf{x}) = \boldsymbol{a}^T\mathbf{x}$ defined in equation (1) (slide 42) satisfies the property

$$\begin{aligned} f(\alpha\mathbf{x}+\beta\mathbf{y}) &= \boldsymbol{a}^T(\alpha\mathbf{x}+\beta\mathbf{y}) \\ &= \boldsymbol{a}^T(\alpha\mathbf{x}) + \boldsymbol{a}^T(\beta\mathbf{y}) \\ &= \alpha(\boldsymbol{a}^T\mathbf{x}) + \beta(\boldsymbol{a}^T\mathbf{y}) \\ &= \alpha f(\mathbf{x}) + \beta f(\mathbf{y}) \end{aligned}$$

  for all d-vectors $\mathbf{x}$, $\mathbf{y}$, and all scalars α, β.

- This property is called superposition, which consists of homogeneity and additivity
- A function that satisfies the superposition property is called linear

- If a function f is linear, superposition extends to linear combinations of any number of vectors:

$$f(\alpha_1\mathbf{x}_1 + \cdots + \alpha_k\mathbf{x}_k) = \alpha_1 f(\mathbf{x}_1) + \cdots + \alpha_k f(\mathbf{x}_k)$$

  for any d vectors $\mathbf{x}_1 + \cdots + \mathbf{x}_k$, and any scalars $\alpha_1 + \cdots + \alpha_k$.

Affine Function

A linear function plus a constant is called an affine function

A linear function $f: \mathcal{R}^d \rightarrow \mathcal{R}$ is **affine** if and only if it can be expressed as $f(\mathbf{x}) = \boldsymbol{a}^T\mathbf{x} + b$ for some d-vector $\boldsymbol{a}$ and scalar b, which is called the offset (or bias)

Example:

$$f(\mathbf{x}) = 2.3 - 2x_1 + 1.3x_2 - x_3$$

This function is affine, with $b = 2.3$, $\boldsymbol{a}^T = [-2,\ 1.3,\ -1]$.

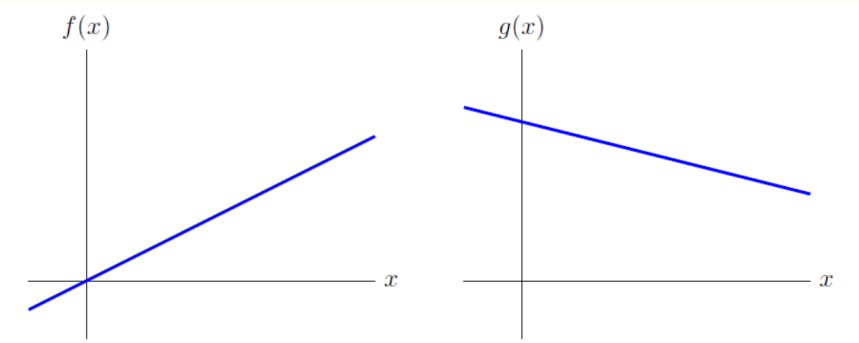
- Linear function plus DC offset constant get non linear function, get affine function
- 

Figure 2.1 *Left.* The function f is linear. *Right.* The function g is affine, but not linear.

Class Quiz

100 elderly participated in a research study on brain imaging and cognitive decline in ageing. Each subject has 10 brain measures. If we would like to use these measures to predict cognition, this is an over-determined system.

Yes, overdetermined as no of sample > no of feature

If each of the 100 older adults has 100 brain and body measures, this is an even-determined system. We can usually find a unique solution.

Yes, even determine as no of sample = no of feature

If only a subset of 40 older adults have completed 100 brain and body measures, this is an under-determined system, i.e. more equations than unknowns. To predict cognition, we need to use right inverse.

No, under determined as no of sample < no of feature, find right inverse, but it's less equations than unknown

Summary

Linear Functions

A function $f: \mathcal{R}^d \rightarrow \mathcal{R}$ is **linear** if it satisfies the following two properties:

- **Homogeneity** $f(\alpha \mathbf{x}) = \alpha f(\mathbf{x})$ Scaling
- **Additivity** $f(\mathbf{x} + \mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$ Adding

Inner product function

$$f(\mathbf{x}) = \boldsymbol{a}^T \mathbf{x} = a_1 x_1 + a_2 x_2 + \cdots a_d x_d$$

Affine function

$f(\mathbf{x}) = \boldsymbol{a}^T \mathbf{x} + b$ scalar b is called the offset (or bias)

- Operations on Vectors and Matrices
  - Dot-product, matrix inverse
- Systems of Linear Equations $\mathbf{X}\mathbf{w} = \mathbf{y}$
  - Matrix-vector notation, linear dependency, invertible
  - Even-, over-, under-determined linear systems
- Set and Functions

Assignment 1 (week 7 Wed)
Tutorial 4

$\mathbf{X}$ is Square	Even-determined	m = d	One unique solution in general	$\hat{\mathbf{w}} = \mathbf{X}^{-1}\mathbf{y}$
$\mathbf{X}$ is Tall	Over-determined	m > d	No exact solution in general; An approximated solution	$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ Left-inverse
$\mathbf{X}$ is Wide	Under-determined	m < d	Infinite number of solutions in general; Unique constrained solution	$\hat{\mathbf{w}} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{y}$ Right-inverse

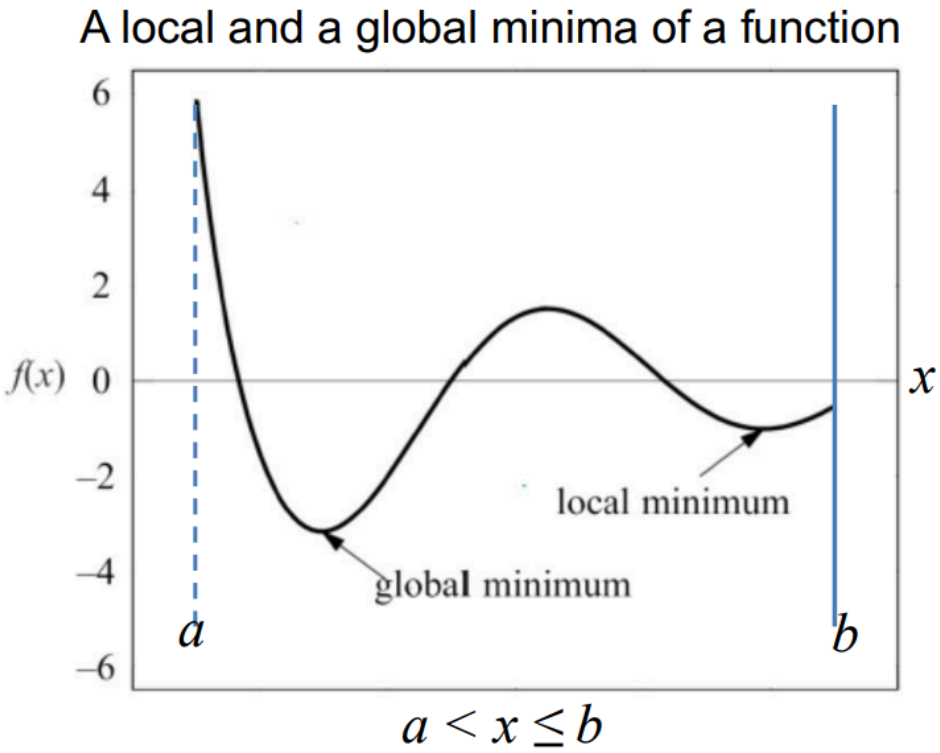
- Scalar and vector functions
- Inner product function
- Linear and affine functions

python package *numpy*
Inverse: *numpy.linalg.inv(X)*
Transpose: *X.T*

Lec5: Least Squares and Linear Regression

Functions: Maximum and Minimum

- $f(x)$ has a **local minimum** at $x = c$ if $f(x) \geq f(c)$ for every x in some open interval around $x = c$
- $f(x)$ has a **global minimum** at $x = c$ if $f(x) \geq f(c)$ for all x in the domain of f



Note: An **interval** is a set of real numbers with the property that any number that lies between two numbers in the set is also included in the set.
An **open interval** does not include its endpoints and is denoted using parentheses. E.g. (0, 1) means “all numbers greater than 0 and less than 1”.

- optimisation problem

Max and Arg Max

- Given a set of values $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$,
- The operator $\max_{a \epsilon \mathcal{A}} f(a)$ returns the highest value $f(a)$ for all elements in the set $\mathcal{A}$
- The operator $\arg\max_{a \epsilon \mathcal{A}} f(a)$ returns the element of the set $\mathcal{A}$ that maximizes $f(a)$
- When the set is **implicit** or **infinite**, we can write

$$\max_a f(a) \quad \text{or} \quad \arg\max_a f(a)$$

E.g. $f(a) = 3a$, $a \epsilon [0,1]$ → $\max_a f(a) = 3$ and $\arg\max_a f(a) = 1$

Min and **Arg Min** operate in a similar manner

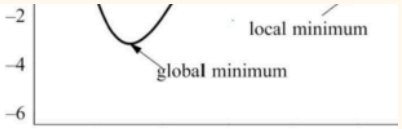
Note: **arg max** returns a value from the **domain** of the function and **max** returns from the **range (codomain)** of the function.

- f(a) is what we want to optimise

Derivative and Gradient

Optimise in terms of cost functions, so need to find the maximum or minimum of the cost functions

- The **derivative f'** of a function f is a function that describes how fast f grows (or decreases)
  - If the derivative is a constant value, e.g. 5 or −3
    - The function f grows (or decreases) constantly at any point x of its domain
  - When the derivative f' is a function
    - If f' is positive at some x, then the function f grows at this point
    - If f' is negative at some x, then the function f decreases at this point
    - The derivative of zero at x means that the function's slope at x is horizontal (e.g. maximum or minimum points)



- The process of finding a derivative is called **differentiation.**

- **Gradient** is the generalization of derivative for functions that take several inputs (or one input in the form of a vector or some other complex structure).

The gradient of a function is a vector of **partial derivatives**

Differentiation of a scalar function w.r.t. a vector

If $f(\mathbf{x})$ is a scalar function of d variables, $\mathbf{x}$ is a d x1 vector. Then differentiation of $f(\mathbf{x})$ w.r.t. $\mathbf{x}$ results in a d x1 vector

$$\frac{df(\mathbf{x})}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_d} \end{bmatrix}$$

$I : R^3 \rightarrow R^1$

(x, y, z)

3×1

This is referred to as the **gradient** of $f(\mathbf{x})$ and often written as $\nabla_{\mathbf{x}} f$.

E.g. $f(\mathbf{x}) = ax_1 + bx_2 \quad \nabla_{\mathbf{x}} f = \begin{bmatrix} a \\ b \end{bmatrix}$

Ref: Duda, Hart, and Stork, "Pattern Classification", 2001 (Appendix)

$\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z})$

$\nabla f = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z})$

- f takes in a vector and returns a scalar
- differentiating f with respect to the vector it takes in is nothing but the gradient of f = ∇f
- Imagine V(x, y, z), then ∇V = - **E**
- Partial derivative of f with respect to all the variable in **x**

Differentiation of a vector function w.r.t. a vector

Differentiation of a vector function w.r.t. a vector

If $\mathbf{f}(\mathbf{x})$ is a vector function of *size* h x1 and $\mathbf{x}$ is a d x1 vector. Then differentiation of $\mathbf{f}(\mathbf{x})$ results in a h x d matrix

$$\frac{d\mathbf{f}(\mathbf{x})}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_h}{\partial x_1} & \cdots & \frac{\partial f_h}{\partial x_d} \end{bmatrix}$$

The matrix is referred to as the **Jacobian** of $\mathbf{f}(\mathbf{x})$

- f is vector valued function of vector
- equivalent to the matrix multiplication of f as a column vector (h x 1) and ∇ as a row vector (1 x d)
    - first row is the partial derivative of the first scalar function in f

Properties

Derivative and Gradient

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$

Some Vector-Matrix Differentiation Formulae

$$\frac{d\mathbf{Ax}}{d\mathbf{x}} = \mathbf{A} \quad (2\times 1 \text{ result}, \; 3\times 1 \text{ input}, \; 2\times 3 \text{ matrix}) \qquad \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 \end{bmatrix}$$

$$\frac{d(\boldsymbol{b}^T\mathbf{x})}{d\mathbf{x}} = \boldsymbol{b} \qquad \frac{d(\mathbf{y}^T\mathbf{Ax})}{d\mathbf{x}} = \mathbf{A}^T\mathbf{y} \quad (\mathbf{y}^T \text{ is } b^T)$$

$$\frac{d(\mathbf{x}^T\mathbf{Ax})}{d\mathbf{x}} = (\mathbf{A} + \mathbf{A}^T)\mathbf{x}$$

$$f(\mathbf{x}) = \boldsymbol{a}^T\mathbf{x} = a_1x_1 + a_2x_2 + \cdots a_dx_d$$

https://math.stackexchange.com/questions/312077/differentiate-fx-xtax

- last one is hard lmao

Linear Regression

$$\sum_{i=1}^{m} (f_{\mathbf{w},b}(\mathbf{x}_i) - \mathrm{y}_i)^2$$

- All model-based learning algorithms have a **loss function**
- What we do to find the best model is **to minimize the objective** known as the **cost function**
- **Cost function** is a sum of **loss functions** over training set plus possibly some model complexity penalty (regularization)

- In linear regression, the cost function is given by the *average loss*, also called the **empirical risk** because we do not have all the data (e.g. testing data)
  - The average of all penalties is obtained by applying the model to the training data

- Loss function is the penalty of being inaccurate
- Empirical risk, because, sample may not be representative the real world distribution, the best possible thing doable is to minimise the empirical risk

Linear regression is a popular regression learning algorithm that learns a model which is a linear combination of features of the input example.

$$\mathbf{Xw} = \mathbf{y}, \quad \mathbf{X} \in \mathcal{R}^{m \times d}, \mathbf{w} \in \mathcal{R}^{d \times 1}, \mathbf{y} \in \mathcal{R}^{m \times 1}$$

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,d} \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

Problem Statement: To predict the unknown y for a given $\mathbf{x}$ **(testing)**

- We have a collection of labeled examples (**training**) $\{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$
  - m is the size of the collection
  - $\mathbf{x}_i$ is the d-dimensional feature vector of example $i = 1, \ldots, m$ (input)
  - y_i is a real-valued target (1-D)
  - Note:
    - when y_i is **continuous** valued, it is a **regression problem**
    - when y_i is **discrete** valued, it is a **classification problem**

- We want to build a model $f_{\mathbf{w},b}(\mathbf{x})$ as a linear combination of features of example $\mathbf{x}$: $f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + b$

  where $\mathbf{w}$ is a d-dimensional vector of parameters and b is a real number.
- The notation $f_{\mathbf{w},b}$ means that the model f is parametrized by two values: $\mathbf{w}$ and b

Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "Introduction to Applied Linear Algebra", Cambridge University Press, 2018 (chp.14)

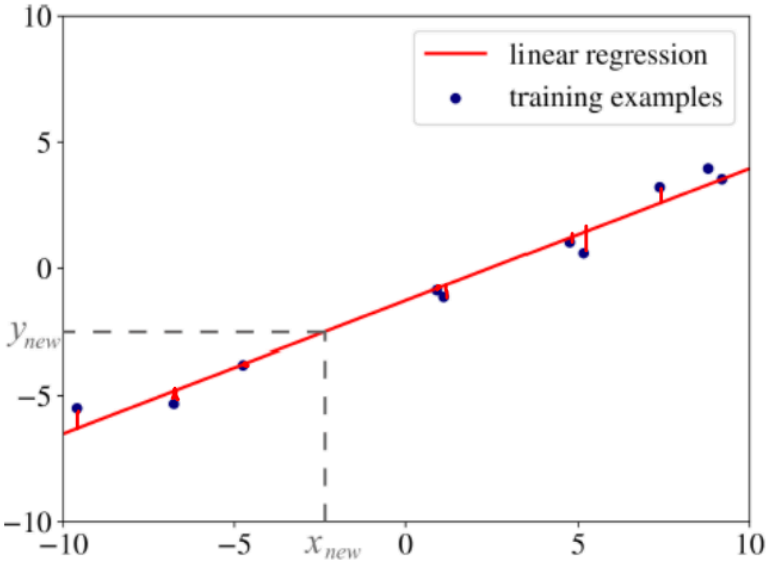
- during training cannot use test data, only training data, the use of test data is called data leakage
- a model is a function that turns x into y
  - parametrized by w and b, so these are the values to be found

Learning objective function

- To find the optimal values for $\mathbf{w}^*$ and b^* which **minimizes** the following expression:

$$\frac{1}{m}\sum_{i=1}^{m}(f_{\mathbf{w},b}(\mathbf{x}_i) - \mathrm{y}_i)^2$$

- In mathematics, the expression we minimize or maximize is called an **objective function**, or, simply, an **objective**



$(f_{\mathbf{w}}(\mathbf{x}_i) - \mathrm{y}_i)^2$ is called the **loss function**: a measure of the difference between $f_{\mathbf{w}}(\mathbf{x}_i)$ and y_i or a penalty for misclassification of example *i*.

- Objective function = cost function, minimise or maximise this to find the best regression, ie, the best value of w and b
  - sum of squared error or mean squared error
  - sum of the loss functions is the objective function
- Compare the value predicted by the model and the ground truth value
- Bias term is needed to shift the line in

Learning objective function (using simplified notation hereon)

- To find the optimal values for $\mathbf{w}^*$ which **minimizes** the following expression:

$$\sum_{i=1}^{m} (f_{\mathbf{w}}(\mathbf{x}_i) - \mathrm{y}_i)^2$$

with $f_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{x}^T\mathbf{w}$,

where we define $\mathbf{w} = [b, w_1, \ldots w_d]^T = [w_0, w_1, \ldots w_d]^T$,

and $\mathbf{x}_i = [1, x_{i,1}, \ldots x_{i,d}]^T = [x_{i,0}, x_{i,1}, \ldots x_{i,d}]^T$, $i = 1, \ldots, m$

- This particular choice of the loss function is called **squared error loss**

Note: The normalization factor $\frac{1}{m}$ can be omitted as it does not affect the optimization.

- putting b inside w to simplify the expression, adjust the x accordingly put a bunch of 1

Learning (Training)

- Consider the set of feature vector $\mathbf{x}_i$ and target output y_i indexed by $i = 1, \ldots, m$, a linear model $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$ can be stacked as

$$\boldsymbol{f}_{\mathbf{w}}(\mathbf{X}) = \mathbf{X}\mathbf{w} \quad \Longleftrightarrow \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

Learning Model

$$= \begin{bmatrix} \mathbf{x}_1^T\mathbf{w} \\ \vdots \\ \mathbf{x}_m^T\mathbf{w} \end{bmatrix}$$

Learning target vector

where $\mathbf{x}_i^T\mathbf{w} = [1, x_{i,1}, \ldots, x_{i,d}] \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$

Note: The **bias/offset term** is responsible for **translating** the line/plane/hyperplane away from the origin.

- shove all the sample together and form matrix X

Least Squares Regression

In vector-matrix notation, the minimization of the objective function can be written compactly using $\mathbf{e} = \mathbf{X}\mathbf{w} - \mathbf{y}$:

$$\begin{aligned} \mathrm{J}(\mathbf{w}) &= \mathbf{e}^T\mathbf{e} \\ &= (\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y}) \\ &= (\mathbf{w}^T\mathbf{X}^T - \mathbf{y}^T)(\mathbf{X}\mathbf{w} - \mathbf{y}) \\ &= \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{w}^T\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\mathbf{w} + \mathbf{y}^T\mathbf{y} \\ &= \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{y}^T\mathbf{X}\mathbf{w} + \mathbf{y}^T\mathbf{y}. \end{aligned}$$

Note: when $\boldsymbol{f}_{\mathbf{w}}(\mathbf{X}) = \mathbf{X}\mathbf{w}$, then

$$\sum_{i=1}^{m} (f_{\mathbf{w}}(\mathbf{x}_i) - \mathrm{y}_i)^2 = (\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y}).$$

- minimise sum of square of error ⇒ least squares regression
- J(w) is a scalar

Differentiating $\mathbf{J(w)}$ with respect to $\mathbf{w}$ and setting the result to $\mathbf{0}$: $\frac{\partial}{\partial \mathbf{w}} \mathbf{J(w)} = \mathbf{0}$

$$\frac{\partial}{\partial \mathbf{w}} (\mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{y}^T\mathbf{X}\mathbf{w} + \mathbf{y}^T\mathbf{y}) = \mathbf{0}$$

$$\Rightarrow 2\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{X}^T\mathbf{y} = \mathbf{0}$$

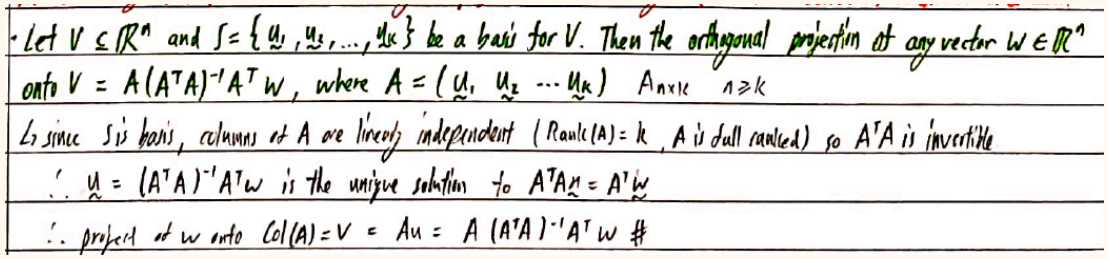
$$\Rightarrow \mathbf{X}^T\mathbf{X}\mathbf{w} = \mathbf{X}^T\mathbf{y}$$

$\Rightarrow$ Any minimizer $\hat{\mathbf{w}}$ of $\mathbf{J(w)}$ must satisfy $\mathbf{X}^T(\mathbf{Xw} - \mathbf{y}) = \mathbf{0}$

If $\mathbf{X}^T\mathbf{X}$ is invertible, then

Learning/training: $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

Prediction/testing: $\hat{f}_{\mathbf{w}}(\mathbf{X}_{new}) = \mathbf{X}_{new}\hat{\mathbf{w}}$

- differentiate with respect to w to minimise the cost, to find the best w
- 

- the least square is the left inverse
- XTX must be invertible
- note that w include the b

Example 1: one dimension input, 1 feature

Example 1 Training set $\{(x_i, y_i)\}_{i=1}^{m}$

$$\{x = -9\} \rightarrow \{y = -6\}$$
$$\{x = -7\} \rightarrow \{y = -6\}$$
$$\{x = -5\} \rightarrow \{y = -4\}$$
$$\{x = 1\} \rightarrow \{y = -1\}$$
$$\{x = 5\} \rightarrow \{y = 1\}$$
$$\{x = 9\} \rightarrow \{y = 4\}$$

$$\mathbf{X} \quad \mathbf{w} \quad \mathbf{y}$$

$$\begin{bmatrix} 1 & -9 \\ 1 & -7 \\ 1 & -5 \\ 1 & 1 \\ 1 & 5 \\ 1 & 9 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} -6 \\ -6 \\ -4 \\ -1 \\ 1 \\ 4 \end{bmatrix}$$

This set of linear equations has no exact solution

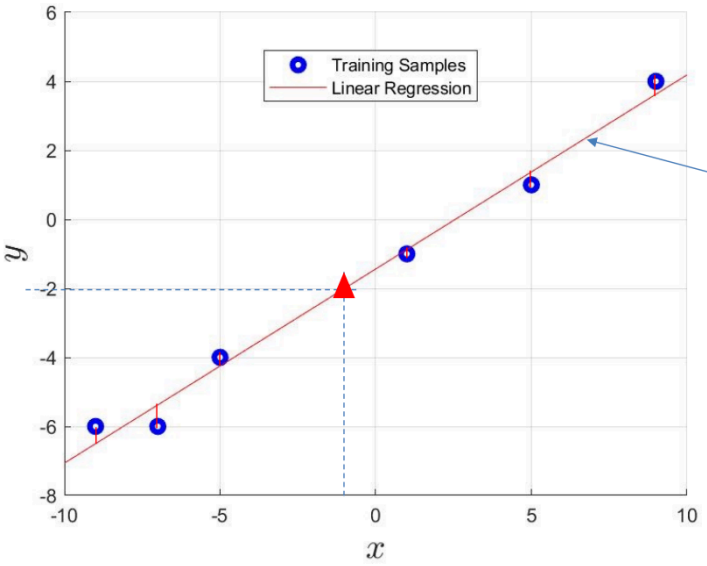
However, $\mathbf{X}^T\mathbf{X}$ is invertible

Least square approximation

$$\hat{\mathbf{w}} = \mathbf{X}^{\dagger}\mathbf{y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$= \begin{bmatrix} 6 & -6 \\ -6 & 262 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ -9 & -7 & -5 & 1 & 5 & 9 \end{bmatrix} \begin{bmatrix} -6 \\ -6 \\ -4 \\ -1 \\ 1 \\ 4 \end{bmatrix} = \begin{bmatrix} -1.4375 \\ 0.5625 \end{bmatrix}$$

- remember to add 1 to ge the bias in (augmentation)
- then shove the x into X along with the padded 1
- Overdetermined system, likely to have left inverse
    - no exact solution, has least square solution



Linear Regression on one-dimensional samples

$$\hat{y} = \mathbf{X}\hat{\mathbf{w}} = \mathbf{X}\begin{bmatrix}-1.4375\\0.5625\end{bmatrix}$$

$$y = -1.4375+0.5625x$$

Prediction:
Test set

$$\{x = -1\} \rightarrow \{y = ?\}$$

$$\hat{y} = [1 \quad -1]\begin{bmatrix}-1.4375\\0.5625\end{bmatrix} = -2$$

Python demo 1

```
1
2   # EE2211 Lecture 5 Demo 1 linear regression
3   import pandas as pd
4   #import matplotlib.pyplot as plt
5   import numpy as np
6   from numpy.linalg import inv
7   from sklearn.metrics import mean_squared_error
8   ##
9   X = np.array([[1, -9], [1, -7], [1, -5], [1, 1], [1, 5], [1, 9]])
10  Y = np.array([[-6], [-6], [-4], [-1], [1], [4]])
11  w = inv(X.T @ X) @ X.T @ Y
12  print(w)
13
14  Xnew = np.array([1, -1])
15  Ynew = Xnew@w
16  print(Ynew)
17
18  ## difference
19  print("Mean squared error between Y and XW")
20  Ytest=X@w
21  MSE = np.square(np.subtract(Ytest,Y)).mean()
22  print(MSE)
23  MSE = mean_squared_error(Ytest,Y)
24  print(MSE)
```

- manually do MSE or use built in function
- remember to pad 1 for X

```
[[-1.4375]
 [ 0.5625]]
[-2.]
Mean squared error between Y and XW
0.16666666666666666
0.16666666666666666
```

Example 2: 3 dimension input, 3 features

Example 2

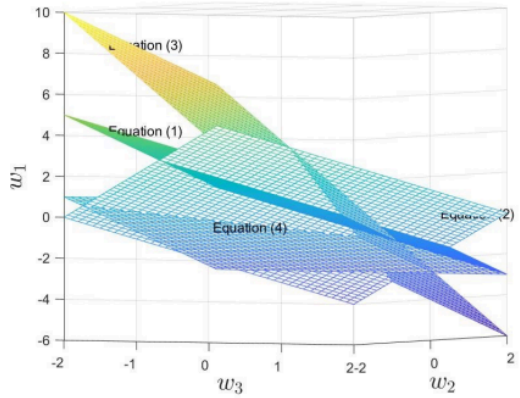
$\{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$

$\{x_1 = 1, x_2 = 1, \quad x_3 = 1\} \rightarrow \{y = 1\}$
$\{x_1 = 1, x_2 = -1, x_3 = 1\} \rightarrow \{y = 0\}$
$\{x_1 = 1, x_2 = 1, \quad x_3 = 3\} \rightarrow \{y = 2\}$
$\{x_1 = 1, x_2 = 1, \quad x_3 = 0\} \rightarrow \{y = -1\}$

Training set

$$\mathbf{X} \quad \mathbf{w} \quad \mathbf{y}$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 3 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 2 \\ -1 \end{bmatrix}$$



This set of linear equations has no exact solution

However, $\mathbf{X}^T\mathbf{X}$ is invertible

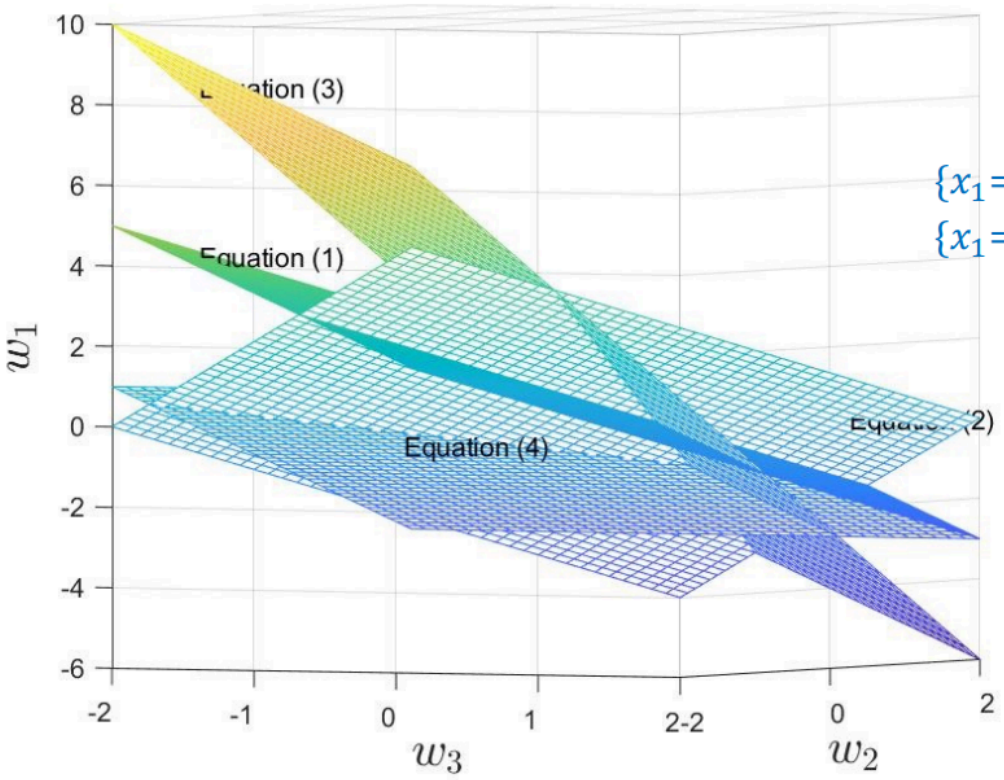
$$\hat{\mathbf{w}} = \mathbf{X}^{\dagger}\mathbf{y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Least square approximation

$$= \begin{bmatrix} 4 & 2 & 5 \\ 2 & 4 & 3 \\ 5 & 3 & 11 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & 3 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 2 \\ -1 \end{bmatrix} = \begin{bmatrix} -0.7500 \\ 0.1786 \\ 0.9286 \end{bmatrix}$$

- in this case the first feature is all one (or a constant), so no need to pad ones
  - so w1 used as the offset

The four linear equations



Prediction:

Test set

$\{x_1 = 1, x_2 = 6, \quad x_3 = 8\} \rightarrow \{y = ?\}$

$\{x_1 = 1, x_2 = 0, x_3 = -1\} \rightarrow \{y = ?\}$

$$\hat{y} = \hat{f}_{\mathbf{w}}(\mathbf{X}_{new}) = \mathbf{X}_{new}\hat{\mathbf{w}}$$

$$\hat{y} = \begin{bmatrix} 1 & 6 & 8 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} -0.7500 \\ 0.1786 \\ 0.9286 \end{bmatrix} = \begin{bmatrix} 7.7500 \\ -1.6786 \end{bmatrix}$$

Learning of Vectored Function (Multiple Outputs)

For one sample: a linear model $\mathbf{f}_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^T\mathbf{W}$ Vector function

For m samples: $\mathbf{F}_{\mathbf{w}}(\mathbf{X}) = \mathbf{XW} = \mathbf{Y}$

Sample 1 ⋮ Sample m

$$\begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_m^T \end{bmatrix} \mathbf{W} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m,1} & \cdots & x_{m,d} \end{bmatrix} \begin{bmatrix} w_{0,1} & \cdots & w_{0,h} \\ w_{1,1} & \cdots & w_{1,h} \\ \vdots & \ddots & \vdots \\ w_{d,1} & \cdots & w_{d,h} \end{bmatrix}$$

(h columns)

Sample 1's output ⋮ Sample m's output

$$= \begin{bmatrix} y_{1,1} & \cdots & y_{1,h} \\ \vdots & & \\ y_{m,1} & \cdots & y_{m,h} \end{bmatrix}$$

(m rows; h columns)

$$\mathbf{X} \in \mathcal{R}^{m\times(d+1)}, \ \mathbf{W} \in \mathcal{R}^{(d+1)\times h}, \ \mathbf{Y} \in \mathcal{R}^{m\times h}$$

- w is no longer a vector, its a matrix W
- its solving the same system of equation but with different y, the features are the same, to predict the different output, the weights must necessarily be different
- But now must be least squares for all the output predictions

Objective: $\sum_{i=1}^{m}(\mathbf{f_w}(\mathbf{x}_i) - \mathbf{y}_i)^2 = \boldsymbol{E^T E}$

Least Squares Regression of Multiple Outputs

In matrix notation, the sum of squared errors cost function can be written compactly using $\mathbf{E = XW - Y}$:

$$\begin{aligned} \mathrm{J}(\mathbf{W}) &= \mathrm{trace}(\mathbf{E}^T\mathbf{E}) \\ &= \mathrm{trace}[(\mathbf{XW}-\mathbf{Y})^T(\mathbf{XW}-\mathbf{Y})] \end{aligned}$$

- With error matrix E = XW - Y, E is m x h
- Minimising J is hard, trace of J is easier
- E^TE is h x h with the trace of E^TE being the sum of error squared

Least Squares Regression of Multiple Outputs

$$\mathrm{J}(\mathbf{W}) = \mathrm{trace}(\mathbf{E}^T\mathbf{E})$$

$$= \mathrm{trace}\left(\begin{bmatrix}\mathbf{e}_1^T \\ \vdots \\ \mathbf{e}_h^T\end{bmatrix}[\mathbf{e}_1 \quad \mathbf{e}_2 \quad \dots \quad \mathbf{e}_h]\right)$$

$$= \mathrm{trace}\left(\begin{bmatrix}\mathbf{e}_1^T\mathbf{e}_1 & \mathbf{e}_1^T\mathbf{e}_2 & \dots & \mathbf{e}_1^T\mathbf{e}_h \\ \mathbf{e}_2^T\mathbf{e}_1 & \mathbf{e}_2^T\mathbf{e}_2 & \dots & \mathbf{e}_2^T\mathbf{e}_h \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{e}_h^T\mathbf{e}_1 & \mathbf{e}_h^T\mathbf{e}_2 & \dots & \mathbf{e}_h^T\mathbf{e}_h\end{bmatrix}\right) = \sum_{k=1}^{h}\mathbf{e}_k^T\mathbf{e}_k$$

- If $\mathbf{X}^T\mathbf{X}$ is invertible, then

  Learning/training: $\widehat{\mathbf{W}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$

  Prediction/testing: $\hat{\mathbf{F}}_{\mathbf{w}}(\mathbf{X}_{new}) = \mathbf{X}_{new}\widehat{\mathbf{W}}$

  - least square estimate for multiple outputs is the similar as the for single output

- The regression is the same as the outputs are independent, and can just do the same regression for the output separately. But doing together can save computation resource

Example 3:

Training set $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{m}$

$\{x_1 = 1, x_2 = 1, \quad x_3 = 1\} \rightarrow \{y_1 = 1, \quad y_2 = 0\}$

$\{x_1 = 1, x_2 = -1, x_3 = 1\} \rightarrow \{y_1 = 0, \quad y_2 = 1\}$

$\{x_1 = 1, x_2 = 1, \quad x_3 = 3\} \rightarrow \{y_1 = 2, y_2 = -1\}$

$\{x_1 = 1, x_2 = 1, \quad x_3 = 0\} \rightarrow \{y_1 = -1, y_2 = 3\}$

$\mathbf{X}$ $\mathbf{W}$ $\mathbf{Y}$

Bias →
$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 3 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2 & -1 \\ -1 & 3 \end{bmatrix}$$

This set of linear equations has NO exact solution

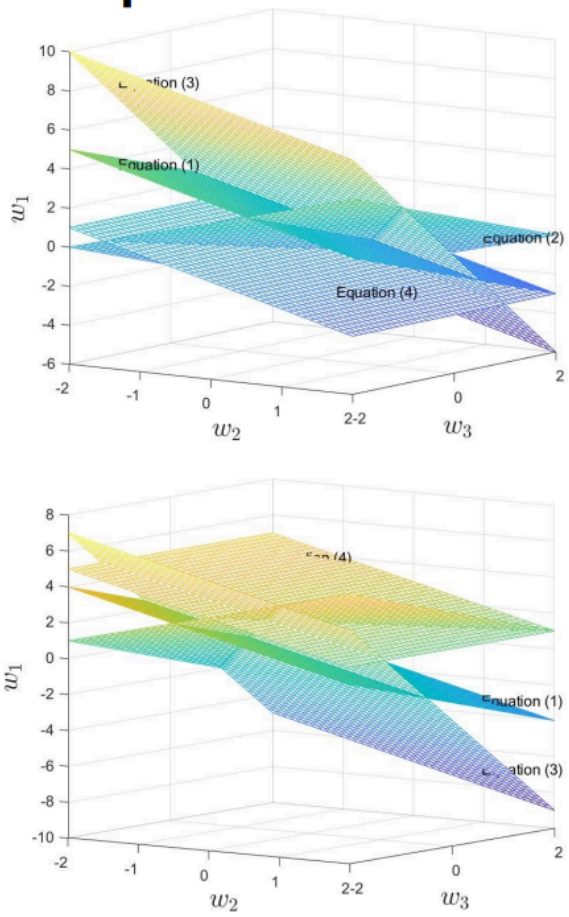
Least square approximation

$$\widehat{\mathbf{W}} = \mathbf{X}^{\dagger}\mathbf{Y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

$\mathbf{X}^T\mathbf{X}$ is invertible

$$= \begin{bmatrix} 4 & 2 & 5 \\ 2 & 4 & 3 \\ 5 & 3 & 11 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & 3 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2 & -1 \\ -1 & 3 \end{bmatrix} = \begin{bmatrix} -0.75 & 2.25 \\ 0.1786 & 0.0357 \\ 0.9286 & -1.2143 \end{bmatrix}$$

Example 3



Prediction:

Test set: two new samples

$\{x_1 = 1, x_2 = 6, \quad x_3 = 8\} \rightarrow \{y_1 = ?, y_2 = ?\}$

$\{x_1 = 1, x_2 = 0, x_3 = -1\} \rightarrow \{y_1 = ?, y_2 = ?\}$

$$\hat{\mathbf{Y}} = \mathbf{X}_{new}\,\widehat{\mathbf{W}}$$

Bias →

$$= \begin{bmatrix} 1 & 6 & 8 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} -0.75 & 2.25 \\ 0.1786 & 0.0357 \\ 0.9286 & -1.2143 \end{bmatrix}$$

$$= \begin{bmatrix} 7.75 & -7.25 \\ -1.6786 & 3.4643 \end{bmatrix}$$

Python demo 2

```python
# EE2211 Lecture 5 Demo 2
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from numpy.linalg import inv
from sklearn.metrics import mean_squared_error
##
X = np.array([[1, 1, 1], [1, -1, 1], [1, 1, 3], [1, 1, 0]])
Y = np.array([[1, 0], [0, 1], [2, -1], [-1, 3]])
w = inv(X.T @ X) @ X.T @ Y
print("the estimated w")
print(w)

Xnew = np.array([[1, 6, 8], [1, 0, -1]])
Ynew = Xnew@w
print("testing Ynew")
print(Ynew)

## difference
print("Mean squared error between Y and XW")
Ytest=X@w
MSE = np.square(np.subtract(Ytest,Y)).mean()
print(MSE)
MSE = mean_squared_error(Ytest,Y)
print(MSE)
```

```
the estimated w
[[-0.75        2.25      ]
 [ 0.17857143  0.03571429]
 [ 0.92857143 -1.21428571]]
testing Ynew
[[ 7.75       -7.25      ]
 [-1.67857143  3.46428571]]
Mean squared error between Y and XW
0.3035714285714286
0.30357142857142855
```

Example 4

The values of feature x and their corresponding values of multiple outputs target **y** are shown in the table below.

Based on the least square regression, what are the values of **w**?
Based on the current mapping, when x = 2, what is the value of **y**?

x	[3]	[4]	[10]	[6]	[7]
y	[0, 5]	[1.5, 4]	[-3, 8]	[-4, 10]	[1, 6]

$$\widehat{\mathbf{W}} = \mathbf{X}^{\dagger}\mathbf{Y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} = \begin{bmatrix} 1.9 & 3.6 \\ -0.4667 & 0.5 \end{bmatrix}$$

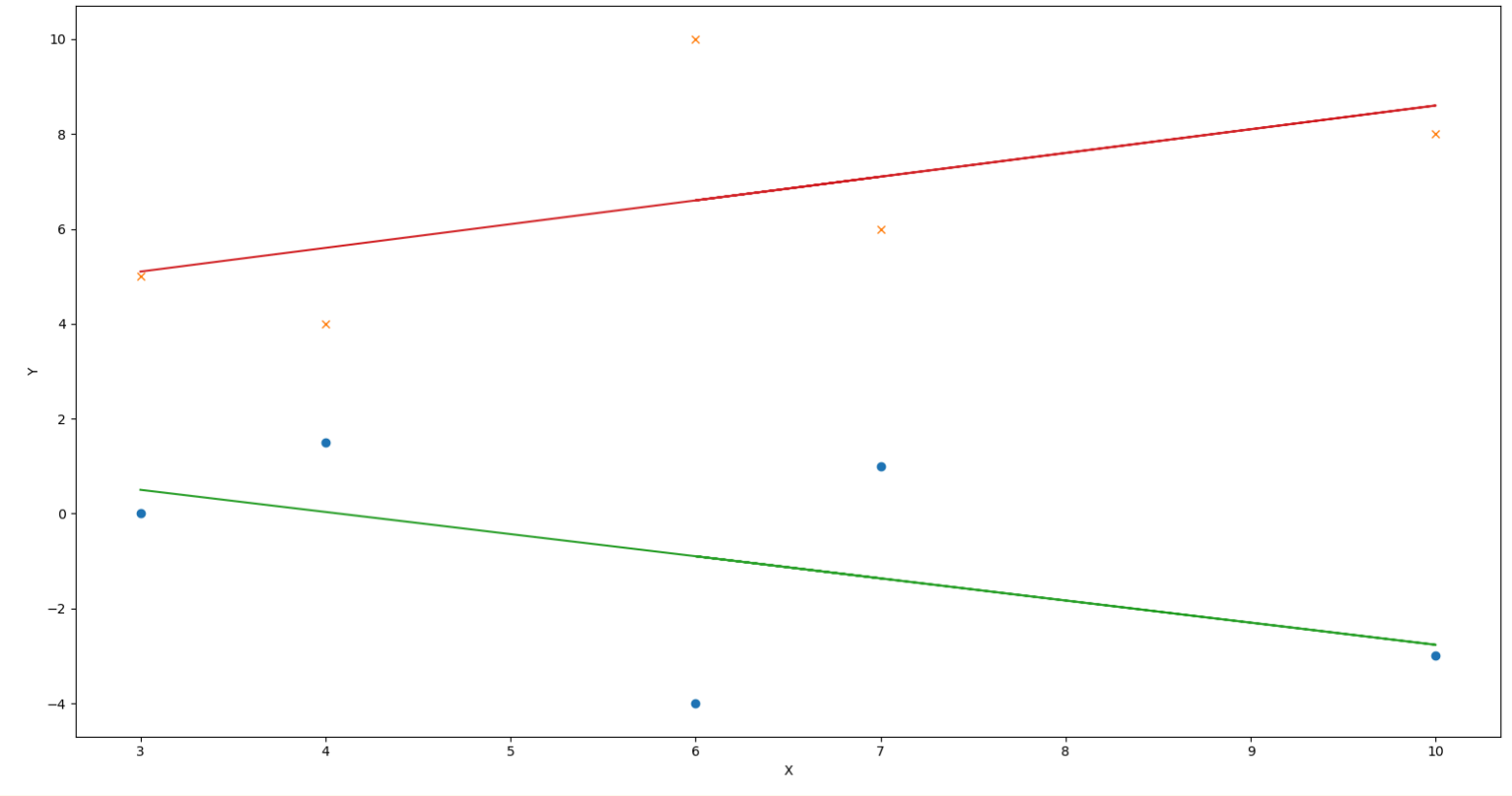
Python demo 3

$$\widehat{\mathbf{Y}_{new}} = \mathbf{X}_{new}\,\widehat{\mathbf{W}} = [1 \quad 2]\,\widehat{\mathbf{W}} = [\,0.9667 \quad 4.6]$$

Prediction

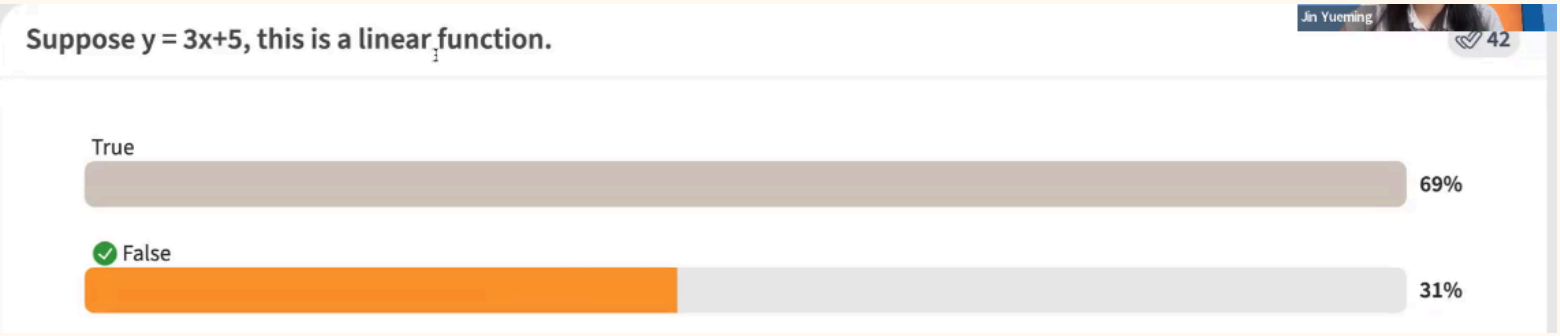
```python
# EE2211 Lecture 5 Demo 3
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from numpy.linalg import inv
from sklearn.metrics import mean_squared_error
##
X = np.array([[1, 3], [1, 4], [1, 10], [1, 6], [1, 7]])
Y = np.array([[0, 5], [1.5, 4], [-3, 8], [-4, 10], [1, 6]])
w = inv(X.T @ X) @ X.T @ Y
print('W')
print(w)

Xnew = np.array([[1, 2]])

Ynew = Xnew@w

print('Ynew')
print(Xnew)
print(Xnew.shape)
print(Ynew)

Ytest= X@w
print('Ytest')
print(Ytest)
plt.plot(X[:,1], Y[:,0], 'o', label = 'Y1')
plt.plot(X[:,1], Y[:,1], 'x', label = 'Y2')
plt.plot(X[:,1], Ytest[:,0])
plt.plot(X[:,1], Ytest[:,1])
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```

```
W
[[ 1.9         3.6       ]
 [-0.46666667  0.5       ]]
Ynew
[[1 2]]
(1, 2)
[[0.96666667 4.6       ]]
Ytest
[[ 0.5         5.1       ]
 [ 0.03333333  5.6       ]
 [-2.76666667  8.6       ]
 [-0.9         6.6       ]
 [-1.36666667  7.1       ]]
```

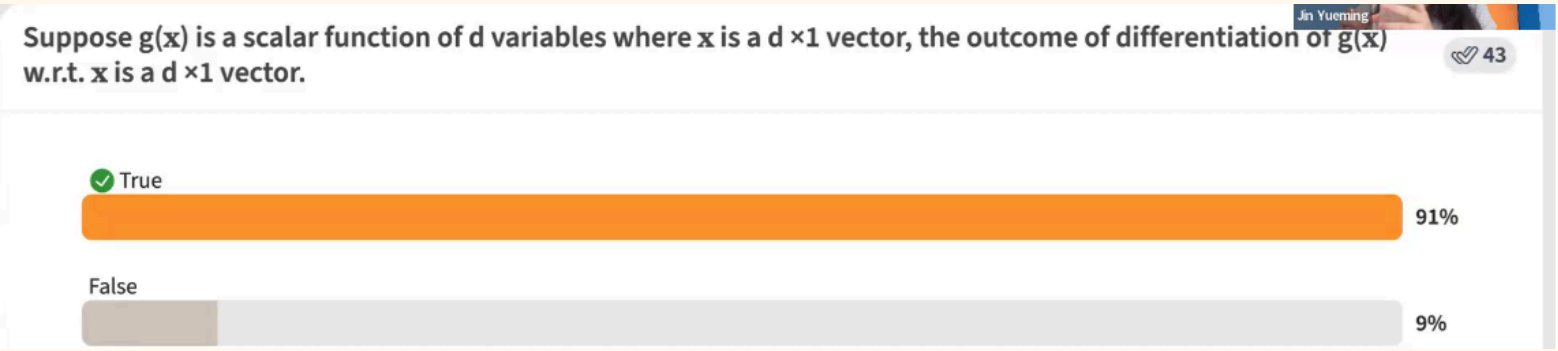


- green is Y1 and red is Y2

Class Quiz



- its affine



Summary

Midterm (L1 to L5)
Trial quiz

- Notations, Vectors, Matrices
- Operations on Vectors and Matrices
  - Dot-product, matrix inverse
- Systems of Linear Equations $f_{\mathbf{w}}(\mathbf{X}) = \mathbf{X}\mathbf{w} = \mathbf{y}$
  - Matrix-vector notation, linear dependency, invertible
  - Even-, over-, under-determined linear systems
- Functions, Derivative and Gradient
  - Inner product, linear/affine functions
  - Maximum and minimum, partial derivatives, gradient
- Least Squares, Linear Regression
  - Objective function, loss function
  - Least square solution, training/learning and testing/prediction
  - Linear regression with multiple outputs

Learning/training $\hat{\mathbf{w}} = (\mathbf{X}_{train}^T \mathbf{X}_{train})^{-1} \mathbf{X}_{train}^T \mathbf{y}_{train}$

Prediction/testing $\mathbf{y}_{test} = \mathbf{X}_{test}\, \hat{\mathbf{w}}$

- Classification
- Ridge Regression
- Polynomial Regression

Python packages: numpy, pandas, matplotlib.pyplot, numpy.linalg, and sklearn.metrics (for mean_squared_error), numpy.linalg.pinv

Lec6: Ridge regression & Polynomial regression

Linear Regression for Classification

- We have a collection of labeled examples
  - m is the size of the collection
  - $\mathbf{x}_i$ is the d-dimensional feature vector of example $i = 1, \ldots, m$
  - y_i is discrete target label (e.g., $y_i \in \{-1, +1\}$ or $\{0, 1\}$ for binary classification problems)
  - Note:
    - when y_i is **continuous** valued → a **regression problem**
    - when y_i is **discrete** valued →a **classification problem**
- Linear model: $f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + b$ or in compact form $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$ (having the offset term absorbed into the inner product)

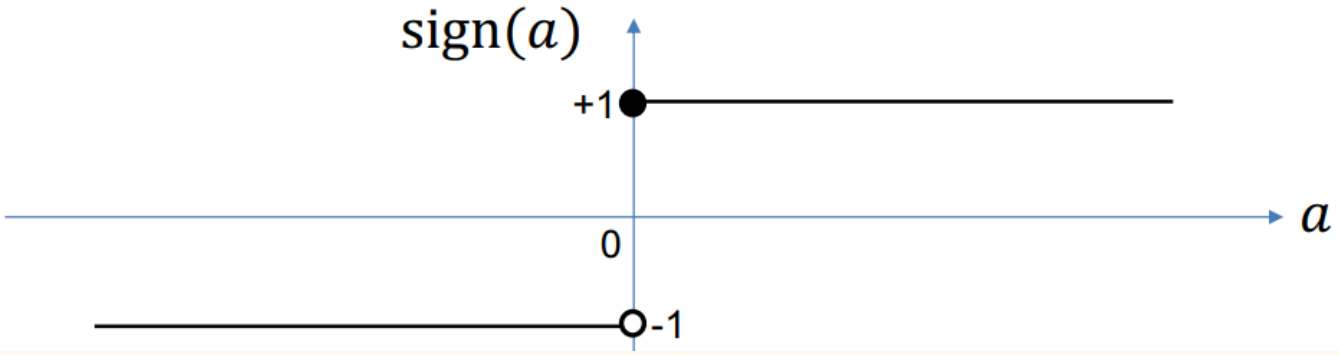
Binary Classification

If $\mathbf{X}^T\mathbf{X}$ is invertible, then

Learning: $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$, $y_i \in \{-1, +1\}, i = 1, \ldots, m$

Prediction: $\hat{f}^c_{\mathbf{w}}(\mathbf{x}_{new}) = \text{sign}(\mathbf{x}^T_{new}\hat{\mathbf{w}})$ for each row $\mathbf{x}^T_{new}$ of $\mathbf{X}_{new}$

$\mathbf{sign}(a) = +1$ for $a \geq 0$ and -1 for $a < 0$



- If classified by 0, 1, take threshold as 0.5
  - or use -1 +1
  - don't use larger number incase large number overshadows smaller number
- Use sign function for binary

Example 1 **Training set** $\{x_i, y_i\}_{i=1}^{m}$

$\{x = -9\} \rightarrow \{y = -1\}$
$\{x = -7\} \rightarrow \{y = -1\}$
$\{x = -5\} \rightarrow \{y = -1\}$
$\{x = 1\} \rightarrow \{y = +1\}$
$\{x = 5\} \rightarrow \{y = +1\}$
$\{x = 9\} \rightarrow \{y = +1\}$

$$\mathbf{X} \quad \mathbf{w} \quad \mathbf{y}$$

Bias $\rightarrow$
$$\begin{bmatrix} 1 & -9 \\ 1 & -7 \\ 1 & -5 \\ 1 & 1 \\ 1 & 5 \\ 1 & 9 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

This set of linear equations has NO exact solution

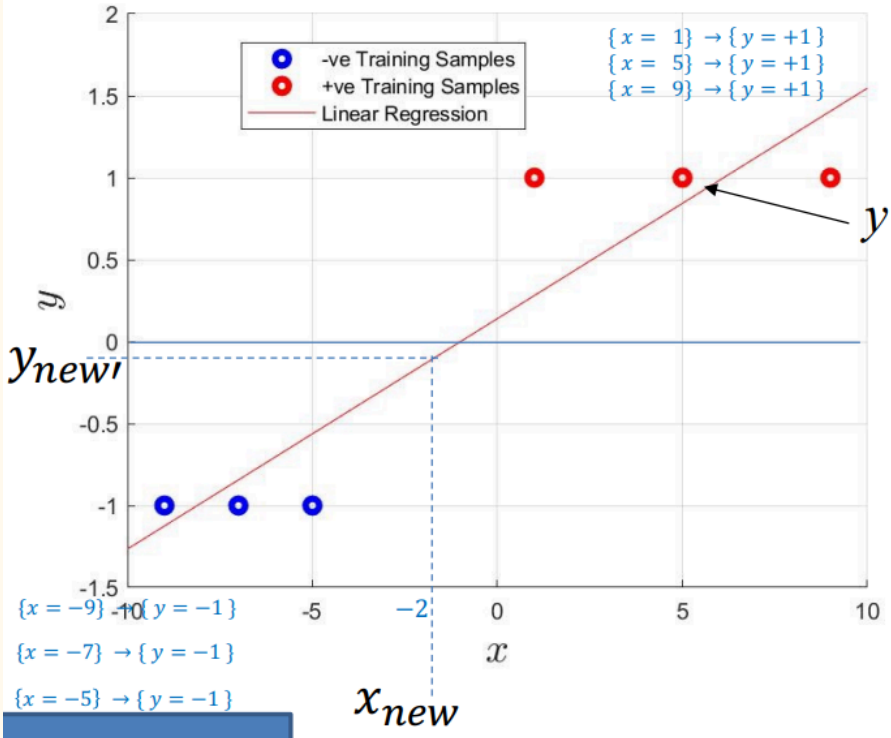
$$\hat{\mathbf{w}} = \mathbf{X}^{\dagger}\mathbf{y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$\mathbf{X}^T\mathbf{X}$ is invertible

$$= \begin{bmatrix} 6 & -6 \\ -6 & 262 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ -9 & -7 & -5 & 1 & 5 & 9 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.1406 \\ 0.1406 \end{bmatrix}$$

Least square approximation

Example 1



$$\hat{y} = \text{sign}(\mathbf{X}\hat{\mathbf{w}}) = \text{sign}(\mathbf{X}\begin{bmatrix}0.1406\\0.1406\end{bmatrix})$$

$$y' = \mathbf{X}\hat{\mathbf{w}} = 0.1406 + 0.1406x$$

Prediction:

Test set $\{x = -2\} \rightarrow \{y = ?\}$

$$y_{new} = \hat{f}^{c}_{\mathbf{w}}(\mathbf{x}_{new}) = \text{sign}(\mathbf{x}_{new}\hat{\mathbf{w}})$$

$$= \text{sign}(\underset{\text{Bias}}{\begin{bmatrix}1 & -2\end{bmatrix}}\begin{bmatrix}0.1406\\0.1406\end{bmatrix})$$

$$= \text{sign}(-0.1406) = -1$$

Python demo 1

Linear Regression for one-dimensional classification

```
# EE2211 Lecture 6 Demo 1 Binary classification
import numpy as np
from numpy.linalg import inv

X = np.array([[1,-9], [1,-7], [1,-5], [1,1], [1,5], [1, 9]])
y = np.array([[-1], [-1], [-1], [1], [1], [1]])
## Linear regression for classification
w = inv(X.T @ X) @ X.T @ y
print("Estimated w")
print(w)
Xt = np.array([[1,-2]])
y_predict = Xt @ w
print("Predicte y")
print(y_predict)
y_class_predict = np.sign(y_predict)
print("Predicted y class")
print(y_class_predict)
```

```
Estimated w
[[0.140625]
 [0.140625]]
Predicte y
[[-0.140625]]
Predicted y class
[[-1.]]
```

If $\mathbf{X}^T\mathbf{X}$ is invertible, then

Learning: $\widehat{\mathbf{W}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}, \quad \mathbf{Y} \in \mathbf{R}^{m\times C}$

Prediction: $\hat{f}^c_{\mathbf{w}}(\mathbf{x}_{new}) = \arg\max_{k=1,\dots,C}\left(\mathbf{x}^T_{new}\widehat{\mathbf{W}}(:,k)\right)$ for each $\mathbf{x}^T_{new}$ of $\mathbf{X}_{new}$

Each row (of $i = 1, \dots, m$) in $\mathbf{Y}$ has an **one-hot** encoding/assignment:

e.g., target for class-1 is labelled as $\mathbf{y}_i^T = [1, 0, 0, \dots, 0]$ for the ith sample,
target for class-2 is labelled as $\mathbf{y}_j^T = [0, 1, 0, \dots, 0]$ for the jth sample,
target for class-C is labelled as $\mathbf{y}_m^T = [0, 0, \dots, 0, 1]$ for the mth sample.

(C)

- Use one hot encoding and multiple output linear regression
- One column for each class
- Then for prediction, the column with the biggest value to determine which class it is

Training set $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^m$

$\{x_1 = 1, \quad x_2 = 1\} \rightarrow \{y_1 = 1, y_2 = 0, y_3 = 0\}$ Class 1
$\{x_1 = -1, \quad x_2 = 1\} \rightarrow \{y_1 = 0, y_2 = 1, y_3 = 0\}$ Class 2
$\{x_1 = 1, \quad x_2 = 3\} \rightarrow \{y_1 = 1, y_2 = 0, y_3 = 0\}$ Class 1
$\{x_1 = 1, \quad x_2 = 0\} \rightarrow \{y_1 = 0, y_2 = 0, y_3 = 1\}$ Class 3

$$\underset{\text{Bias}\rightarrow}{\mathbf{X}}\begin{bmatrix}1 & 1 & 1\\ 1 & -1 & 1\\ 1 & 1 & 3\\ 1 & 1 & 0\end{bmatrix}\underset{\mathbf{W}}{\begin{bmatrix}w_{1,1} & w_{1,2} & w_{1,3}\\ w_{2,1} & w_{2,2} & w_{2,3}\\ w_{3,1} & w_{3,2} & w_{3,3}\end{bmatrix}} = \underset{\mathbf{Y}}{\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}}$$

This set of linear equations has NO exact solution.

Least square approximation

$\widehat{\mathbf{W}} = \mathbf{X}^{\dagger}\mathbf{Y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$ $\mathbf{X}^T\mathbf{X}$ is invertible

$$= \begin{bmatrix}4 & 2 & 5\\ 2 & 4 & 3\\ 5 & 3 & 11\end{bmatrix}^{-1}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & 1\\ 1 & 1 & 3 & 0\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}0 & 0.5 & 0.5\\ 0.2857 & -0.5 & 0.2143\\ 0.2857 & 0 & -0.2857\end{bmatrix}$$

Test set $\mathbf{X}_{new}$ $\{x_1 = 6,\ x_2 = 8\} \rightarrow \{class\ 1, 2, or\ 3?\}$
$\{x_1 = 0,\ x_2 = -1\} \rightarrow \{class\ 1, 2, or\ 3?\}$

$$\hat{\mathbf{Y}} = \mathbf{X}_{new}\,\hat{\mathbf{W}} = \begin{bmatrix} 1 & 6 & 8 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0.2857 & -0.5 & 0.2143 \\ 0.2857 & 0 & -0.2857 \end{bmatrix}$$

Category prediction:

$$\hat{f}^{c}_{\mathbf{w}}(\mathbf{X}_{new}) = \arg\max_{k=1,\ldots,C}(\hat{\mathbf{Y}}(:,k))$$
$$= \arg\max_{k=1,\ldots,C}\left(\begin{bmatrix} 4 & -2.50 & -0.50 \\ -0.2587 & 0.50 & 0.7857 \end{bmatrix}\right)$$
$$= \begin{bmatrix} 1 \\ 3 \end{bmatrix} \begin{matrix} \longrightarrow \text{Class 1} \\ \longrightarrow \text{Class 3} \end{matrix}$$

Python demo 2

For each row of **Y**, the **column position** of the **largest** number (across all columns for that row) determines the **class label**.

E.g. in the first row, the maximum number is 4 which is in column 1. Therefore, the resulting predicted class is 1.

```
# EE2211 Lecture 6 Demo 2 Multi-class classification
import numpy as np
from numpy.linalg import inv
from sklearn.preprocessing import OneHotEncoder
X = np.array([[1, 1, 1], [1, -1, 1], [1, 1, 3], [1, 1, 0]])
y_class = np.array([[1], [2], [1], [3]])
y_onehot = np.array([[1, 0, 0], [0, 1, 0], [1, 0, 0], [0, 0, 1]])
print("One-hot encoding manual")
print(y_class)
print(y_onehot)

print("One-hot encoding function")
onehot_encoder=OneHotEncoder(sparse=False)
print(onehot_encoder)
Ytr_onehot = onehot_encoder.fit_transform(y_class)
print(Ytr_onehot)

#reshaped = y_class.reshape(len(y_class), 1)
#print(reshaped)
#Ytr_onehot = onehot_encoder.fit_transform(reshaped)
```

```
## Linear Classification
print("Estimated W")
W = inv(X.T @ X) @ X.T @ Ytr_onehot
print(W)
X_test = np.array([[1, 6, 8], [1, 0, -1]])
yt_est = X_test@W;
print("Test")
print(yt_est)
yt_class = [[1 if y == max(x) else 0 for y in x] for x in yt_est ]
print("class label test")
print(yt_class)
```

```
One-hot encoding manual
[[1]
 [2]
 [1]
 [3]]
[[1 0 0]
 [0 1 0]
 [1 0 0]
 [0 0 1]]
One-hot encoding function
OneHotEncoder(sparse=False)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [1. 0. 0.]
 [0. 0. 1.]]
Estimated W
[[ 1.11022302e-16  5.00000000e-01  5.00000000e-01]
 [ 2.85714286e-01 -5.00000000e-01  2.14285714e-01]
 [ 2.85714286e-01  2.77555756e-17 -2.85714286e-01]]
Test
[[ 4.         -2.5        -0.5       ]
 [-0.28571429  0.5         0.78571429]]
class label test
[[1, 0, 0], [0, 0, 1]]
```

Ridge Regression

Recall Linear regression

Objective: $\widehat{\mathbf{w}} = \text{argmin} \sum_{i=1}^{m} (f_{\mathbf{w}}(\mathbf{x}_i) - \mathrm{y}_i)^2 = (\mathbf{X}\mathbf{w} - \mathbf{y})^T (\mathbf{X}\mathbf{w} - \mathbf{y})$

The learning computation: $\widehat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

We cannot guarantee that the matrix $\mathbf{X}^T\mathbf{X}$ is invertible

Ridge regression: shrinks the regression coefficients *w* by imposing a penalty on their size

Objective: $\widehat{\mathbf{w}} = \text{argmin} \sum_{i=1}^{m} (f_{\mathbf{w}}(\mathbf{x}_i) - \mathrm{y}_i)^2 + \lambda \sum_{j=1}^{d} w_j^2$

$= \text{argmin} (\mathbf{X}\mathbf{w} - \mathbf{y})^T (\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^T \mathbf{w}$

Here $\lambda \geq 0$ is a complexity parameter that controls the amount of shrinkage: the larger the value of λ, the greater the amount of shrinkage.

Note: *m* samples & *d* parameters

- Least square solution for linear regression requires X^TX must be invertible, but this is not guarantee for any given sample
- Ridge regression introduce additional objective term to shrink the w
  - first part is minimise square error, second part is to minimise size of w
  - reduce size of weights
- Ridge regression to avoid overfitting issue
- Multiple output ridge regression is out of scope

Derivation

Using a linear model:

$$\min_{\mathbf{w}} (\mathbf{Xw} - \mathbf{y})^T (\mathbf{Xw} - \mathbf{y}) + \lambda \mathbf{w}^T \mathbf{w}$$

Solution:

$$\frac{\partial}{\partial \mathbf{w}} \left((\mathbf{Xw} - \mathbf{y})^T (\mathbf{Xw} - \mathbf{y}) + \lambda \mathbf{w}^T \mathbf{w} \right) = \mathbf{0}$$

$$\Rightarrow 2\mathbf{X}^T\mathbf{Xw} - 2\mathbf{X}^T\mathbf{y} + 2\lambda\mathbf{w} = \mathbf{0}$$

$$\Rightarrow \mathbf{X}^T\mathbf{Xw} + \lambda\mathbf{w} = \mathbf{X}^T\mathbf{y}$$

$$\Rightarrow (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\mathbf{w} = \mathbf{X}^T\mathbf{y}$$

where **I** is the *dxd* identity matrix

Here on, we shall focus on single column of output **y** in derivations in the sequel

Learning: $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$

- Adding a small value along the diagonal means even if X^TX is not invertible, $X^TX + \lambda I$ is.

Primal and Dual Form

- dual and prime give the same answer but one is computationally less expensive
- Always use the one that requires to find the inverse of the smaller square matrix
-

```
## dual solution m < d (without ridge)
w_dual = P.T @ inv(P @ P.T) @ y
print("Under-determined system")
print("Unique constrained solution, no ridge")
print(w_dual)

print("************************************")
print("Approximation with dual ridge regression")
print(P.shape)
reg_L2 = 0.0001*np.identity(P.shape[0]) #number of rows of P = Dual I
print(reg_L2)
w_dual_ridge = P.T @ (inv(P @ P.T + reg_L2)) @ y
print(w_dual_ridge)

print("************************************")
## primal ridge
print("Approximation with primal ridge regression")
print(P.shape)
reg_L = 0.0001*np.identity(P.shape[1]) #number of columns of P = Primal I
print(reg_L)
w_primal_ridge = inv(P.T @ P + reg_L) @ P.T @ y
print(w_primal_ridge)
```

```
Under-determined system
Unique constrained solution, no ridge
[[-1.]
 [ 1.]
 [ 1.]
 [ 1.]
 [-4.]
 [ 1.]]
************************************
Approximation with dual ridge regression
(4, 6)
[[0.0001 0.     0.     0.    ]
 [0.     0.0001 0.     0.    ]
 [0.     0.     0.0001 0.    ]
 [0.     0.     0.     0.0001]]
[[-0.9993004 ]
 [ 0.99940033]
 [ 0.99940033]
 [ 0.99940033]
 [-3.99790115]
 [ 0.99940033]]
************************************
Approximation with primal ridge regression
(4, 6)
[[0.0001 0.     0.     0.     0.     0.    ]
 [0.     0.0001 0.     0.     0.     0.    ]
 [0.     0.     0.0001 0.     0.     0.    ]
 [0.     0.     0.     0.0001 0.     0.    ]
 [0.     0.     0.     0.     0.0001 0.    ]
 [0.     0.     0.     0.     0.     0.0001]]
[[-0.9993004 ]
 [ 0.99940033]
 [ 0.99940033]
 [ 0.99940033]
 [-3.99790115]
 [ 0.99940033]]
```

Ridge Regression in Primal Form (when m > d)

$(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})$ is invertible for $\lambda > 0$,

Learning: $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$

Prediction: $\hat{f}_{\mathbf{w}}(\mathbf{X}_{new}) = \mathbf{X}_{new}\hat{\mathbf{w}}$

- Primal form used for m > d, over determined
  - always find inverse in smaller in dimension
- X is m x d
- XTX is d x d
- I is d x d

Ridge Regression in Dual Form (when m < d)

$(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})$ is invertible for $\lambda > 0$,

Learning: $\hat{\mathbf{w}} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\mathbf{y}$

Prediction: $\hat{f}_{\mathbf{w}}(\mathbf{X}_{new}) = \mathbf{X}_{new}\hat{\mathbf{w}}$

- Dual form used for m < d, under determined
  - always find inverse in smaller in dimension
- X is m x d
- XXT is m x m
- I is m x m

Derivation as homework (see tutorial 6).

Hint: start off with $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\mathbf{w} = \mathbf{X}^T\mathbf{y}$ and make use of $\mathbf{w} = \mathbf{X}^T\mathbf{a}$ and $\boldsymbol{a} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w}), \lambda > 0$

Given $(X^T X + \lambda I) w = X^T y$

derive $\hat{w} = X^T (XX^T + \lambda I)^{-1} y$

Hint: $w = X^T a$ where

$a = \lambda^{-1} (y - Xw)$, $\lambda > 0$

: $X^T X w + \lambda I w = X^T y$

$\lambda w = X^T y - X^T X w$

$= X^T (y - Xw)$

$w = \lambda^{-1} X^T (y - Xw)$

$= X^T a$ — (1)

: $a = \lambda^{-1} (y - Xw)$

$\lambda a = y - Xw$

$= y - X X^T a$

$XX^T a + \lambda a = y$

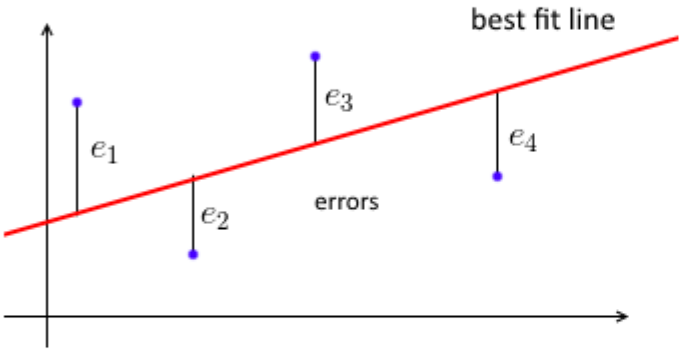
$(XX^T + \lambda) a = y$

$a = (XX^T + \lambda I)^{-1} y$ — (2)

Short answer: no difference between Primal and Dual - it's only about the way of arriving to the solution. Kernel ridge regression is essentially the same as usual ridge regression, but uses the kernel trick to go non-linear.

Linear Regression

First of all, a usual Least Squares Linear Regression tries to fit a straight line to the set of data points in such a way that the sum of squared errors is minimal.



We parametrize the best fit line with $\mathbf{w}$ and for each data point $(\mathbf{x}_i, y_i)$ we want $\mathbf{w}^T\mathbf{x}_i \approx y_i$. Let $e_i = y_i - \mathbf{w}^T\mathbf{x}_i$ be the error - the distance between predicted and true values. So our goal is to minimize the sum of squared errors $\sum e_i^2 = \|\mathbf{e}\|^2 = \|X\mathbf{w} - \mathbf{y}\|^2$ where $X = \begin{bmatrix} — \mathbf{x}_1 — \\ — \mathbf{x}_2 — \\ \vdots \\ — \mathbf{x}_n — \end{bmatrix}$ - a data matrix with each $\mathbf{x}_i$ being a row, and $\mathbf{y} = (y_1, \ \dots \ , y_n)$ a vector with all y_i's.

Thus, the objective is $\min_{\mathbf{w}} \|X\mathbf{w} - \mathbf{y}\|^2$, and the solution is $\mathbf{w} = (X^TX)^{-1}X^T\mathbf{y}$ (known as "Normal Equation").

For a new unseen data point $\mathbf{x}$ we predict its target value $\hat{y}$ as $\hat{y} = \mathbf{w}^T\mathbf{x}$.

Ridge Regression

When there are many correlated variables in linear regression models, the coefficients $\mathbf{w}$ can become poorly determined and have lots of variance. One of the solutions to this problem is to restrict weights $\mathbf{w}$ so they don't exceed some budget C. This is equivalent to using L_2-regularization, also known as "weight decay": it will decrease the variance at the cost of sometimes missing the correct results (i.e. by introducing some bias).

The objective now becomes $\min_{\mathbf{w}} \|X\mathbf{w} - y\|^2 + \lambda \|\mathbf{w}\|^2$, with λ being the regularization parameter. By going through the math, we obtain the following solution: $\mathbf{w} = (X^T X + \lambda I)^{-1} X^T \mathbf{y}$. It's very similar to the usual linear regression, but here we add λ to each diagonal element of $X^T X$.

Note that we can re-write $\mathbf{w}$ as $\mathbf{w} = X^T (XX^T + \lambda I)^{-1} \mathbf{y}$ (see here for details). For a new unseen data point $\mathbf{x}$ we predict its target value $\hat{y}$ as $\hat{y} = \mathbf{x}^T \mathbf{w} = \mathbf{x}^T X^T (XX^T + \lambda I)^{-1} \mathbf{y}$. Let $\boldsymbol{\alpha} = (XX^T + \lambda I)^{-1} \mathbf{y}$. Then $\hat{y} = \mathbf{x}^T X^T \boldsymbol{\alpha} = \sum_{i=1}^{n} \alpha_i \cdot \mathbf{x}^T \mathbf{x}_i$.

Ridge Regression Dual Form

We can have a different look at our objective - and define the following quadratic program problem:

$$\min_{\mathbf{e},\mathbf{w}} \sum_{i=1}^{n} e_i^2 \text{ s.t. } e_i = y_i - \mathbf{w}^T\mathbf{x}_i \text{ for } i = 1\,..\,n \text{ and } \|\mathbf{w}\|^2 \leqslant C.$$

It's the same objective, but expressed somewhat differently, and here the constraint on the size of $\mathbf{w}$ is explicit. To solve it, we define the Lagrangian $\mathcal{L}_p(\mathbf{w}, \mathbf{e}; C)$ - this is the primal form that contains primal variables $\mathbf{w}$ and $\mathbf{e}$. Then we optimize it w.r.t. $\mathbf{e}$ and $\mathbf{w}$. To get the dual formulation, we put found $\mathbf{e}$ and $\mathbf{w}$ back to $\mathcal{L}_p(\mathbf{w}, \mathbf{e}; C)$.

So, $\mathcal{L}_p(\mathbf{w}, \mathbf{e}; C) = \|\mathbf{e}\|^2 + \boldsymbol{\beta}^T(\mathbf{y} - X\mathbf{w} - \mathbf{e}) - \lambda\left(\|\mathbf{w}\|^2 - C\right)$. By taking derivatives w.r.t. $\mathbf{w}$ and $\mathbf{e}$, we obtain $\mathbf{e} = \frac{1}{2}\boldsymbol{\beta}$ and $\mathbf{w} = \frac{1}{2\lambda}X^T\boldsymbol{\beta}$. By letting $\boldsymbol{\alpha} = \frac{1}{2\lambda}\boldsymbol{\beta}$, and putting $\mathbf{e}$ and $\mathbf{w}$ back to $\mathcal{L}_p(\mathbf{w}, \mathbf{e}; C)$, we get dual Lagrangian $\mathcal{L}_d(\boldsymbol{\alpha}, \lambda; C) = -\lambda^2\|\boldsymbol{\alpha}\|^2 + 2\lambda\,\boldsymbol{\alpha}^T y - \lambda\|X^T\boldsymbol{\alpha}\| - \lambda C$. If we take a derivative w.r.t. $\boldsymbol{\alpha}$, we get $\boldsymbol{\alpha} = (XX^T - \lambda I)^{-1}\mathbf{y}$ - the same answer as for usual Kernel Ridge regression. There's no need to take a derivative w.r.t λ - it depends on C, which is a regularization parameter - and it makes λ regularization parameter as well.

Next, put $\boldsymbol{\alpha}$ to the primal form solution for $\mathbf{w}$, and get $\mathbf{w} = \frac{1}{2\lambda}X^T\boldsymbol{\beta} = X^T\boldsymbol{\alpha}$. Thus, the dual form gives the same solution as usual Ridge Regression, and it's just a different way to come to the same solution.

https://stats.stackexchange.com/questions/92672/difference-between-primal-dual-and-kernel-ridge-regression

Polynomial Regression

Motivation: nonlinear decision surface

- Based on the sum of products of the variables
- E.g. when the input dimension is $d=2$,

a polynomial function of degree = 2 is:

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + w_{12}\,x_1x_2 + w_{11}\,x_1^2 + w_{22}\,x_2^2.$$

Example

XOR problem

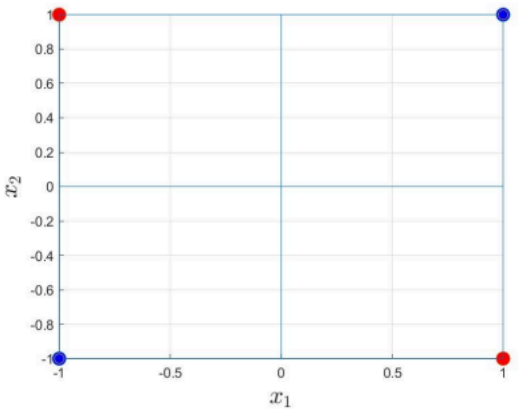
$\mathbf{x}_1 = [+1 \quad +1]^\top \qquad y_1 = +1$

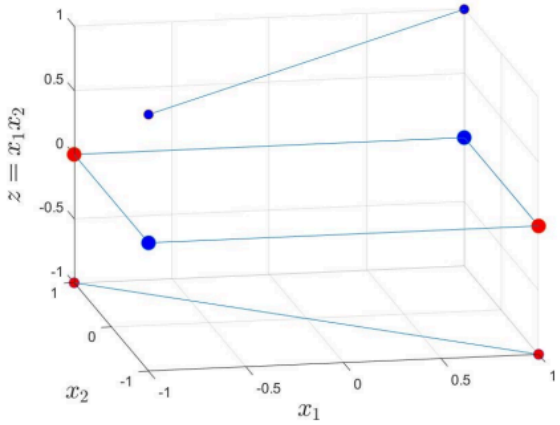
$\mathbf{x}_2 = [-1 \quad +1]^\top \qquad y_2 = -1$

$\mathbf{x}_3 = [+1 \quad -1]^\top \qquad y_3 = -1$

$\mathbf{x}_4 = [-1 \quad -1]^\top \qquad y_4 = +1$

$f_{\mathbf{w}}(\mathbf{x}) = x_1 x_2$





- Must need non linear regression to fit the situation, not possible to split into non linear, not linearly separable
- Pic on left shows how the axis of x1x2 separate the blue and red dots

Polynomial Expansion

- The linear model $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$ can be written as

$$f_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$$
$$= \sum_{i=0}^{d} x_i w_i , \qquad x_0 = 1$$
$$= w_0 + \sum_{i=1}^{d} x_i w_i .$$

- By including additional terms involving the products of pairs of components of $\mathbf{x}$, we obtain a quadratic model:

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij} x_i x_j .$$

2nd order: $f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + w_{12} x_1 x_2 + w_{11} x_1^2 + w_{22} x_2^2$

3rd order: $f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + w_{12} x_1 x_2 + w_{11} x_1^2 + w_{22} x_2^2 + \sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{k=1}^{d} w_{ijk} x_i x_j x_k, \; d = 2$

$C(n,r) = \frac{(n+r-1)!}{r!\,(n-1)!}$

Ref: Duda, Hart, and Stork, "Pattern Classification", 2001 (Chp.5)

- d is number of feature
- 2nd order has 6 parameter/weight
  - x1x2 is the same as x2x1 need to combine
- 3rd order has 10 parameter/weight
  - same as 2nd order, + 4
- nCr, ????

Polynomial Regression

1
C(3,1)
C(3,2)
C(3,3)

Generalized Linear Discriminant Function

- In general:

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij} x_i x_j + \sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{k=1}^{d} w_{ijk} x_i x_j x_k + \cdots$$

Weierstrass Approximation Theorem: Every continuous function defined on a closed interval [*a*, *b*] can be uniformly approximated as closely as desired by a polynomial function.

- Suppose f is a continuous real-valued function defined on the real interval $[a, b]$.
- For every $\varepsilon > 0$, there exists a polynomial p such that for all x in $[a, b]$, we have $|f(x) - p(x)| < \varepsilon$.

(Ref: https://en.wikipedia.org/wiki/Stone%E2%80%93Weierstrass_theorem)

Notes:

- For high dimensional input features (large d *value*) and high *polynomial order*, the number of polynomial terms becomes explosive! (i.e., grows exponentially)
- For high dimensional problems, polynomials of order larger than 3 is seldom used.

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij} x_i x_j + \sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{k=1}^{d} w_{ijk} x_i x_j x_k + \cdots$$

$$\boldsymbol{f}_{\mathbf{w}}(\mathbf{x}) = \mathbf{P}\mathbf{w} \qquad (\text{Note: } \mathbf{P} \triangleq \mathbf{P}(\mathbf{X}) \text{ for symbol simplicity})$$

$$= \begin{bmatrix} \boldsymbol{p}_1^T\mathbf{w} \\ \vdots \\ \boldsymbol{p}_m^T\mathbf{w} \end{bmatrix}$$

where $\boldsymbol{p}_l^T\mathbf{w} = [1, x_{l,1}, \ldots, x_{l,d}, \ldots, x_{l,i}x_{l,j}, \ldots, x_{l,i}x_{l,j}x_{l,k}, \ldots]$

$$\begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \\ \vdots \\ w_{ij} \\ \vdots \\ w_{ijk} \\ \vdots \end{bmatrix}$$

$l = 1, \ldots, m$; d denotes the dimension of input features; m denotes the number of samples

Example 3

$\{x_1 = 0, x_2 = 0\} \rightarrow \{y = -1\}$
$\{x_1 = 1, x_2 = 1\} \rightarrow \{y = -1\}$
$\{x_1 = 1, x_2 = 0\} \rightarrow \{y = +1\}$
$\{x_1 = 0, x_2 = 1\} \rightarrow \{y = +1\}$

Training set $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{m}$

2nd order polynomial model

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + w_{12} x_1 x_2 + w_{11} x_1^2 + w_{22} x_2^2$$

$$= \begin{bmatrix} 1 & x_1 & x_2 & x_1 x_2 & x_1^2 & x_2^2 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_{12} \\ w_{11} \\ w_{22} \end{bmatrix}$$

Python demo 3

Stack the 4 training samples as a matrix

$$\mathbf{P} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & x_{1,1}x_{1,2} & x_{1,1}^2 & x_{1,2}^2 \\ 1 & x_{2,1} & x_{2,2} & x_{2,1}x_{2,2} & x_{2,1}^2 & x_{2,2}^2 \\ 1 & x_{3,1} & x_{3,2} & x_{3,1}x_{3,2} & x_{3,1}^2 & x_{3,2}^2 \\ 1 & x_{4,1} & x_{4,2} & x_{4,1}x_{4,2} & x_{4,1}^2 & x_{4,2}^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

- Number of rows in P = number of samples
- Number of columns in P = number of terms in the polynomial expansion of with d feature up till order O
- Sub in the value of x1 and x2, and solve the linear system y = Pw

Example 3 (cont'd)

$\{x_1 = 0, x_2 = 0\} \rightarrow \{y = -1\}$
$\{x_1 = 1, x_2 = 1\} \rightarrow \{y = -1\}$
$\{x_1 = 1, x_2 = 0\} \rightarrow \{y = +1\}$
$\{x_1 = 0, x_2 = 1\} \rightarrow \{y = +1\}$

Training set

2nd order polynomial model

$$\mathbf{P} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & x_{1,1}x_{1,2} & x_{1,1}^2 & x_{1,2}^2 \\ 1 & x_{2,1} & x_{2,2} & x_{2,1}x_{2,2} & x_{2,1}^2 & x_{2,2}^2 \\ 1 & x_{3,1} & x_{3,2} & x_{3,1}x_{3,2} & x_{3,1}^2 & x_{3,2}^2 \\ 1 & x_{4,1} & x_{4,2} & x_{4,1}x_{4,2} & x_{4,1}^2 & x_{4,2}^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$\hat{\mathbf{w}} = \mathbf{P}^T(\mathbf{P}\mathbf{P}^T)^{-1}\mathbf{y}$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 6 & 3 & 3 \\ 1 & 3 & 3 & 1 \\ 1 & 3 & 1 & 3 \end{bmatrix}^{-1} \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 1 \\ -4 \\ 1 \\ 1 \end{bmatrix}$$

Python demo 3

Example 3 (cont'd)

Prediction

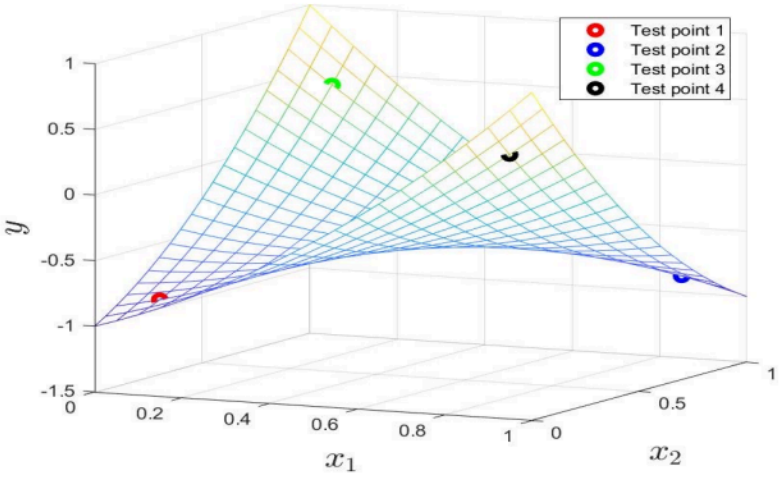
Test set

Test point 1: $\{x_1 = 0.1, x_2 = 0.1\} \rightarrow \{y = \text{class } -1 \text{ or } +1?\}$
Test point 2: $\{x_1 = 0.9, x_2 = 0.9\} \rightarrow \{y = \text{class } -1 \text{ or } +1?\}$
Test point 3: $\{x_1 = 0.1, x_2 = 0.9\} \rightarrow \{y = \text{class } -1 \text{ or } +1?\}$
Test point 4: $\{x_1 = 0.9, x_2 = 0.1\} \rightarrow \{y = \text{class } -1 \text{ or } +1?\}$

$$\hat{\mathbf{y}} = \mathbf{P}_{new}\,\hat{\mathbf{w}} = \begin{bmatrix} 1 & 0.1 & 0.1 & 0.01 & 0.01 & 0.01 \\ 1 & 0.9 & 0.9 & 0.81 & 0.81 & 0.81 \\ 1 & 0.1 & 0.9 & 0.09 & 0.01 & 0.81 \\ 1 & 0.9 & 0.1 & 0.09 & 0.81 & 0.01 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ 1 \\ -4 \\ 1 \\ 1 \end{bmatrix}$$

$$[1 \quad x_1 \quad x_2 \quad x_1x_2 \quad x_1^2 \quad x_2^2]$$

$$\hat{f}^c_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \text{sign}(\hat{\mathbf{y}}) = \text{sign}\left(\begin{bmatrix} -0.82 \\ -0.82 \\ 0.46 \\ 0.46 \end{bmatrix}\right) = \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix} \begin{array}{l} \dashrightarrow \text{Class } -1 \\ \dashrightarrow \text{Class } -1 \\ \dashrightarrow \text{Class } +1 \\ \dashrightarrow \text{Class } +1 \end{array}$$



- non linear decision surface, each value of x1 and x2 to mapped onto the surface, then compare to threshold value 0 to get which class it is

```
#EE2211 Lecture 6 Demo 3 Polynomial regression
import numpy as np
from numpy.linalg import inv
from numpy.linalg import matrix_rank
from sklearn.preprocessing import PolynomialFeatures
X = np.array([ [0, 0], [1, 1], [1, 0], [0, 1]])
y = np.array([[-1], [-1], [1], [1]])
## Generate polynomial features
order = 2
poly = PolynomialFeatures(order)
print(poly)
P = poly.fit_transform(X)
print("matrix P")
print(P)

print("************************************")
#print(matrix_rank(P))
#PY = np.vstack((P.T, y.T))
#print(matrix_rank(PY.T))
```

```
PolynomialFeatures()
matrix P
[[1. 0. 0. 0. 0. 0.]
 [1. 1. 1. 1. 1. 1.]
 [1. 1. 0. 1. 0. 0.]
 [1. 0. 1. 0. 0. 1.]]
************************************
```

-
  - input to polynomial fit transform doesn't need to pad 1, it automatically does it

```
## dual solution m < d (without ridge)
w_dual = P.T @ inv(P @ P.T) @ y
print("Under-determined system")
print("Unique constrained solution, no ridge")
print(w_dual)

print("************************************")
print("Approximation with dual ridge regression")
print(P.shape)
reg_L2 = 0.0001*np.identity(P.shape[0]) #number of rows of P = Dual I
print(reg_L2)
w_dual_ridge = P.T @ (inv(P @ P.T + reg_L2)) @ y
print(w_dual_ridge)

print("************************************")
## primal ridge
print("Approximation with primal ridge regression")
print(P.shape)
reg_L = 0.0001*np.identity(P.shape[1]) #number of columns of P = Primal I
print(reg_L)
w_primal_ridge = inv(P.T @ P + reg_L) @ P.T @ y
print(w_primal_ridge)
```

```
Under-determined system
Unique constrained solution, no ridge
[[-1.]
 [ 1.]
 [ 1.]
 [ 1.]
 [-4.]
 [ 1.]]
************************************
Approximation with dual ridge regression
(4, 6)
[[0.0001 0.     0.     0.    ]
 [0.     0.0001 0.     0.    ]
 [0.     0.     0.0001 0.    ]
 [0.     0.     0.     0.0001]]
[[-0.9993004 ]
 [ 0.99940033]
 [ 0.99940033]
 [ 0.99940033]
 [-3.99790115]
 [ 0.99940033]]
************************************
Approximation with primal ridge regression
(4, 6)
[[0.0001 0.     0.     0.     0.     0.    ]
 [0.     0.0001 0.     0.     0.     0.    ]
 [0.     0.     0.0001 0.     0.     0.    ]
 [0.     0.     0.     0.0001 0.     0.    ]
 [0.     0.     0.     0.     0.0001 0.    ]
 [0.     0.     0.     0.     0.     0.0001]]
[[-0.9993004 ]
 [ 0.99940033]
 [ 0.99940033]
 [ 0.99940033]
 [-3.99790115]
 [ 0.99940033]]
```

```
#EE2211 Lecture 6 Demo 3 Testing/prediction
import numpy as np
from numpy.linalg import inv
from numpy.linalg import matrix_rank
from sklearn.preprocessing import PolynomialFeatures
X = np.array([ [0, 0], [1, 1], [1, 0], [0, 1]])
y = np.array([[-1], [-1], [1], [1]])
## Generate polynomial features
order = 2
poly = PolynomialFeatures(order)
print(poly)
P = poly.fit_transform(X)
print("matrix P")
print(P)
print("Under-determined system")
#print(matrix_rank(P))
#PY = np.vstack((P.T, y.T))
#print(matrix_rank(PY.T))

## dual solution m < d (without ridge)
w_dual = P.T @ inv(P @ P.T) @ y
print("Unique constrained solution, no ridge")
print(w_dual)

#testing
print("Prediction")
Xnew= np.array([ [0.1, 0.1], [0.9, 0.9], [0.1, 0.9], [0.9, 0.1]])
Pnew = poly.fit_transform(Xnew)
Ynew=Pnew@w_dual
print(Ynew)
print(np.sign(Ynew))
```

```
PolynomialFeatures()
matrix P
[[1. 0. 0. 0. 0. 0.]
 [1. 1. 1. 1. 1. 1.]
 [1. 1. 0. 1. 0. 0.]
 [1. 0. 1. 0. 0. 1.]]
Under-determined system
Unique constrained solution, no ridge
[[-1.]
 [ 1.]
 [ 1.]
 [ 1.]
 [-4.]
 [ 1.]]
Prediction
[[-0.82]
 [-0.82]
 [ 0.46]
 [ 0.46]]
[[-1.]
 [-1.]
 [ 1.]
 [ 1.]]
```

Ridge Regression in Primal Form (m > d)

For $\lambda > 0$,

Learning: $\hat{\mathbf{w}} = (\mathbf{P}^T\mathbf{P} + \lambda\mathbf{I})^{-1}\,\mathbf{P}^T\mathbf{y}$

Prediction: $\hat{f}_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \mathbf{P}_{new}\hat{\mathbf{w}}$

Ridge Regression in Dual Form (m < d)

For $\lambda > 0$,

Learning: $\hat{\mathbf{w}} = \mathbf{P}^T(\mathbf{P}\mathbf{P}^T + \lambda\mathbf{I})^{-1}\,\mathbf{y}$

Prediction: $\hat{f}_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \mathbf{P}_{new}\hat{\mathbf{w}}$

Note: Change **X** to **P** with reference to slides 15/16; m & d refers to the size of **P** (not **X**)

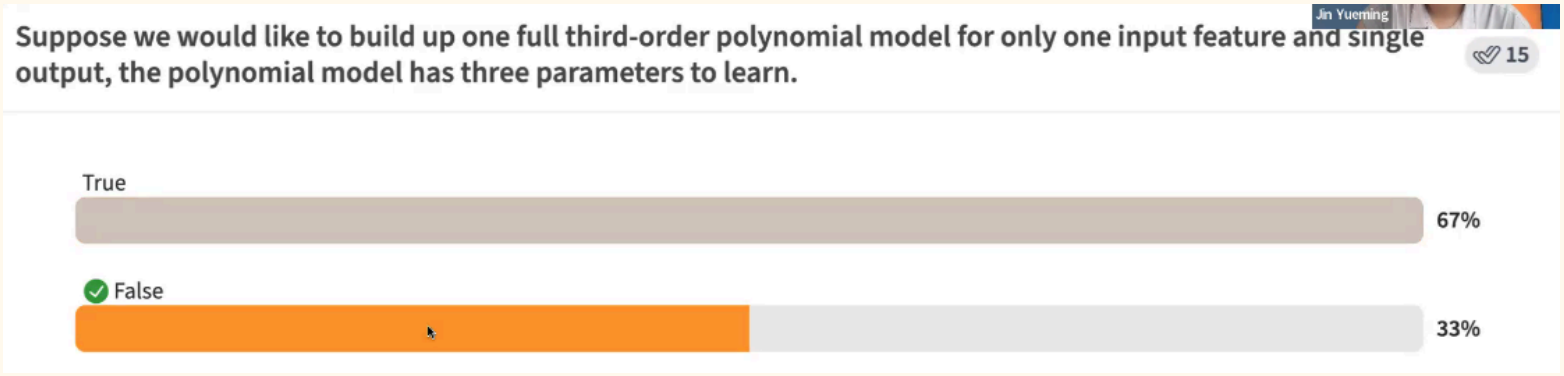
For Regression Applications

- Learn **continuous** valued y using either primal form or dual form
- Prediction: $\hat{f}_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \mathbf{P}_{new}\hat{\mathbf{w}}$

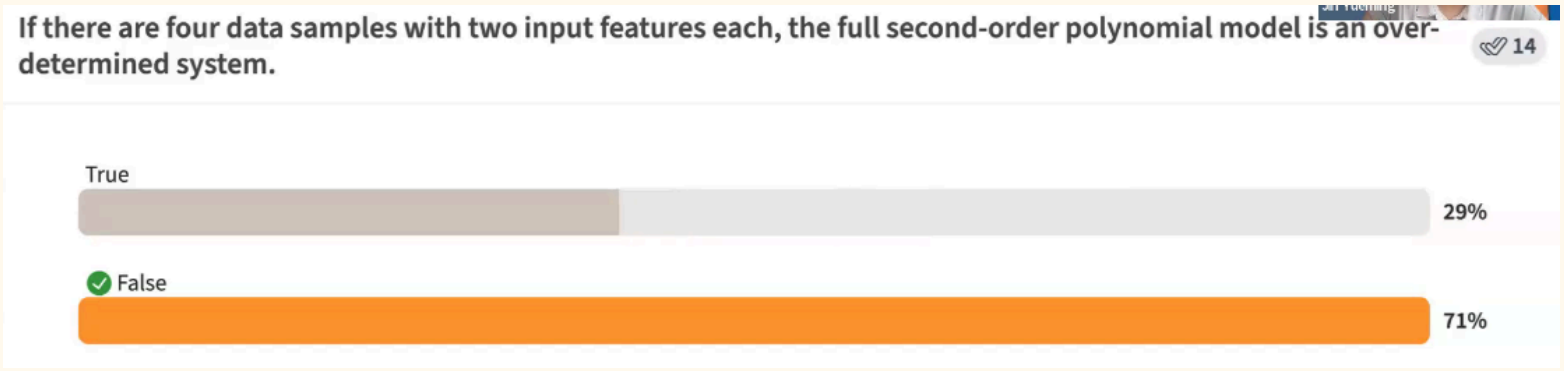
For Classification Applications

- Learn **discrete** valued y ($y \in \{-1, +1\}$) or $\mathbf{Y}$ (one-hot) using either primal form or dual form
- Binary Prediction: $\hat{f}^c_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \text{sign}(\mathbf{P}_{new}\hat{\mathbf{w}})$
- Multi-Category Prediction: $\hat{f}^c_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \arg\max_{k=1,\ldots,C}(\mathbf{P}_{new}\hat{\mathbf{W}}(:,k))$

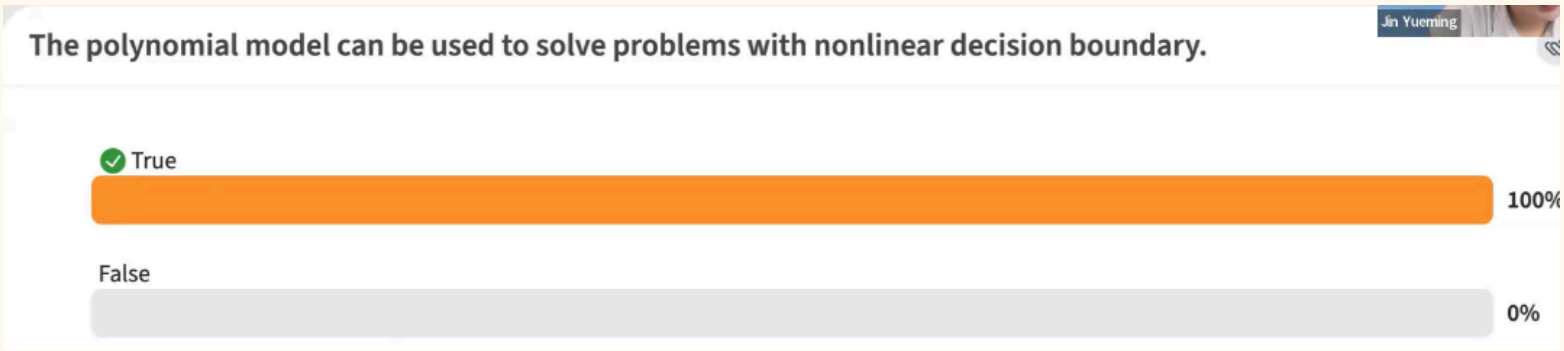
In Class Quiz



- One variable (feature) 3rd order means $y = c + w_1x + w_2x^2 + w_3x^3$
- Four parameters/weights



- $f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + w_{12}x_1x_2 + w_{11}x_1^2 + w_{22}x_2^2$
- 6 columns, 4 rows, under determined



Lec7: Over-fitting, bias/variance trade-off

- Linear is special case of polynomial => use "**P**" instead of "**X**" from now on
- Training/Learning (primal) on training set

$$\hat{\mathbf{w}} = (\mathbf{P}_{train}^T \mathbf{P}_{train})^{-1} \mathbf{P}_{train}^T \mathbf{y}_{train}$$

- Prediction/Testing/Evaluation on test set

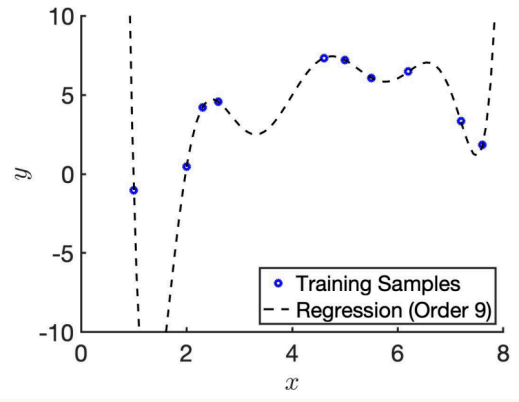
$$\hat{\mathbf{y}}_{test} = \mathbf{P}_{test} \hat{\mathbf{w}}$$

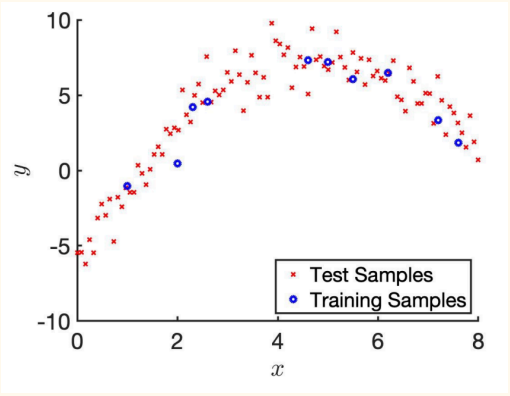
- There should be **zero** overlap between training & test sets
- Important goal of regression: prediction on **new unseen data**, i.e., test set
- Why is test set important for evaluation?

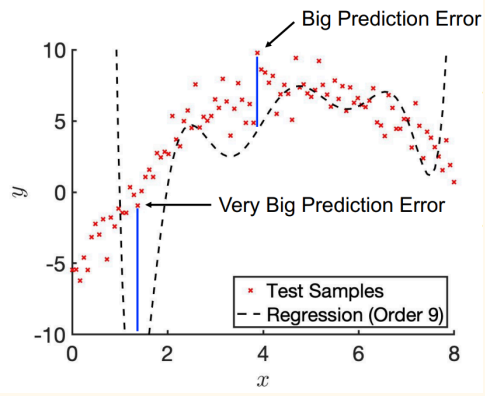
- Ideally training set >> test set

Overfitting, Underfitting & Model Complexity

Overfitting





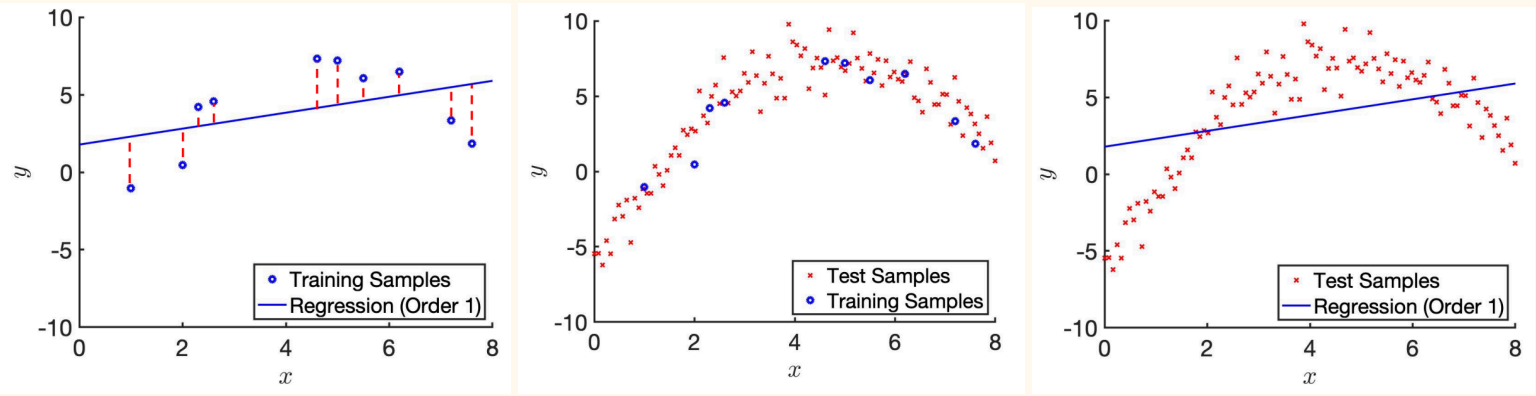


- Regression fits the training data well but not the test data
- has very big prediction error when using test data (MSE is big)
- Because training data set is small while trying to fit a very complicated model of order 9, alot parameter
- Training good, test bad ⇒ over fitting

- **Overfitting** occurs when model predicts the training data well, but predicts new data (e.g., from test set) poorly
- **Reason 1**
  - Model is too complex for the data
  - Previous example: Fit order 9 polynomial to 10 data points
- **Reason 2**
  - Too many features but number of training samples too small
  - Even linear model can overfit, e.g., linear model with 9 input features (i.e., w is 10-D) and 10 data points in training set => data might not be enough to estimate 10 unknowns well
- **Solutions**
  - Use **simpler models** (e.g., lower order polynomial)
  - Use **regularization** (see next part of lecture)

- Can use less features also

Underfitting

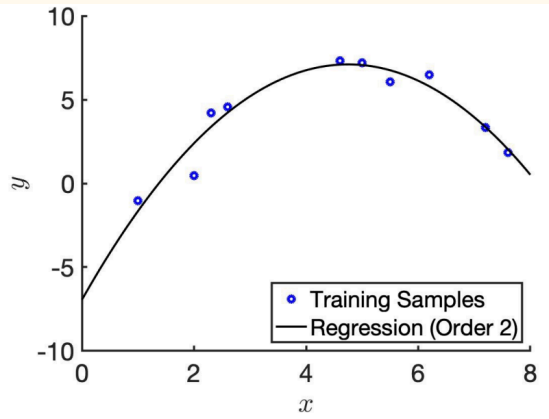


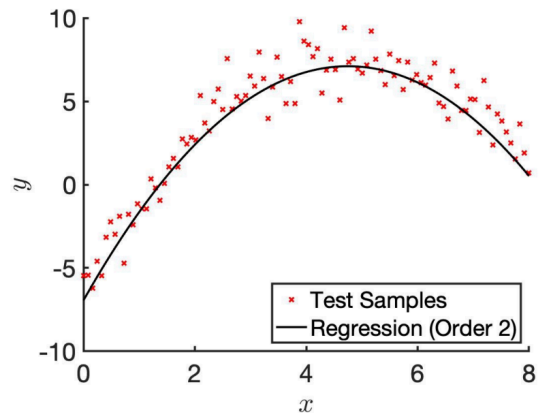
- Regression is bad for the training data and the test data ⇒ under fitting
- Means model is too simple

- **Underfitting** is the inability of trained model to predict the targets in the training set
- **Reason 1**
  - Model is too simple for the data
  - Previous example: Fit order 1 polynomial to 10 data points that came from an order 2 polynomial
  - **Solution**: Try more complex model
- **Reason 2**
  - Features are not informative enough
  - **Solution**: Try to develop more informative features

- Need more features, or have features that doesn't tell much
- Be better at feature selection

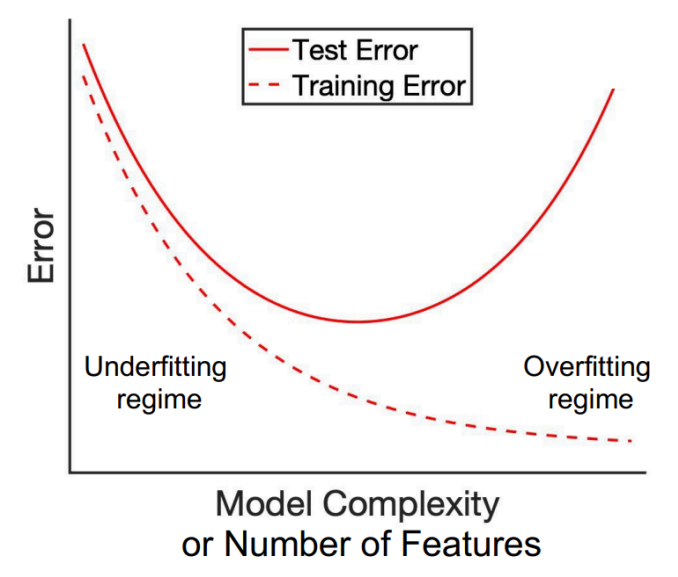
Good Fit





- Regression does not pass through all the training data, the MSE is actually larger than the 9 order overfitting case
- But this is due to quadratic only having 3 parameters, so the regression is also still good
- The MSE to the test data is also very good

Overfitting / Underfitting Schematic



- When both training and test error are high, is when model complexity is too low or number of features is not enough ⇒ underfitting
- When training error is low but test error is high, is when model complexity is too high or number of features is too much ⇒ over fitting
    - if the model is very complicated and has a lot of parameter, the parameter can be chosen to fit the training data very well
- note the test error has a u-shape indicating there is a sweet spot between under and over fitting

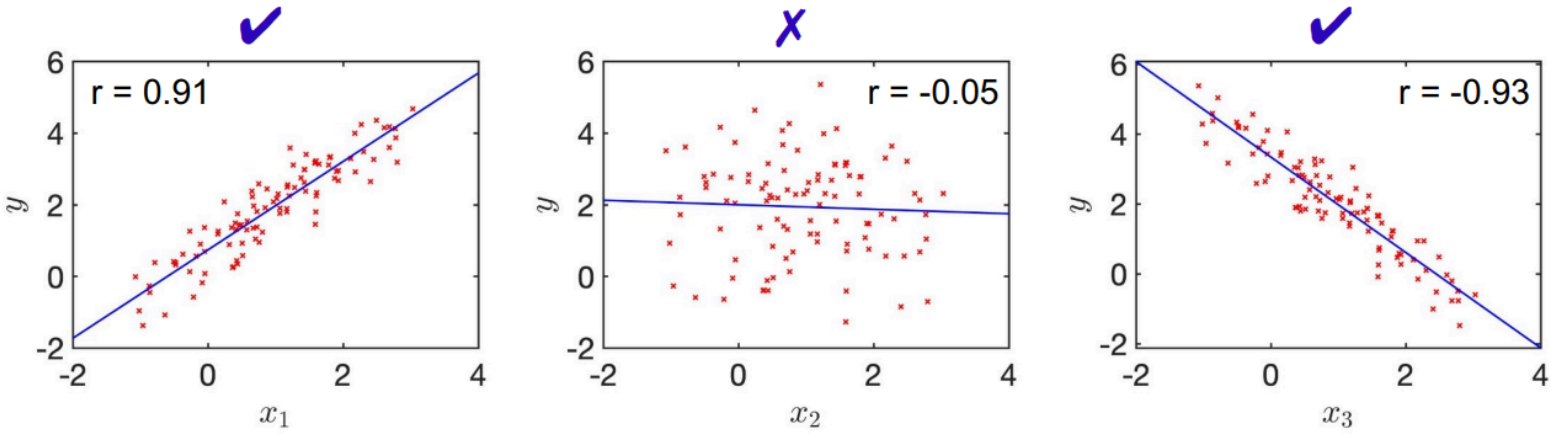
Feature Selection

- Less features might reduce overfitting
  - Want to discard useless features & keep good features, so perform feature selection
- Feature selection procedure
  - Step 1: feature selection in training set
  - Step 2: fit model using selected features in training set
  - Step 3: evaluate trained model using test set
- Very common mistake
  - Feature selection with test set (or full dataset) leads to inflated performance
  - Do not perform feature selection with test data

- Do not train with test set
- Do not feature select from test set
- Both will lead to information leakage
- If need to compare two models, give both the same training set and test with the same test set

Selecting Features With Pearson's r

- Given features x, we want to predict target y
- Assume x & y both continuous
- Compute Pearson's correlation coefficient between each feature & target y in the training set
  - Pearson's correlation r measures linear relationship between two variables



- Two options
  - Option 1: Pick K features with largest absolute correlations
  - Option 2: Pick all features with absolute correlations > C
  - K & C are "magic" numbers set by the ML practitioner
- Other metrics besides Pearson's correlation are possible

Regularisation

- Regularization is an umbrella term that includes methods forcing learning algorithm to build less complex models.
- Motivation 1: Solve an ill-posed problem
  - For example, estimate 10th order polynomial with just 5 datapoints
- Motivation 2: Reduce overfitting
- For example, in previous lecture, we added $\lambda \mathbf{w}^T \mathbf{w}$:

$$\underset{\mathbf{w}}{\operatorname{argmin}} (\mathbf{P}\mathbf{w} - \mathbf{y})^T (\mathbf{P}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^T \mathbf{w}$$

- Minimizing with respect to $\mathbf{w}$, primal solution is

$$\hat{\mathbf{w}} = (\mathbf{P}^T \mathbf{P} + \lambda \mathbf{I})^{-1} \mathbf{P}^T \mathbf{y}$$

- For $\lambda > 0$, matrix becomes invertible (Motivation 1)

- $\hat{\mathbf{w}}$ might also perform better in test set, i.e., reduces overfitting (Motivation 2) – will show example later

- ill posed problem means trying to solve under determine system
- with λ, w reduces fitting by making sure features that does not contribute much have low weights, so can fix over fitting

- Consider minimization from previous slide

$$\underset{\mathbf{w}}{\operatorname{argmin}} (\mathbf{P}\mathbf{w} - \mathbf{y})^T (\mathbf{P}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^T \mathbf{w}$$

Cost function quantifying data fitting error in training set

Regularization

- Consider minimization from previous slide

$$\underset{\mathbf{w}}{\operatorname{argmin}} (\mathbf{Pw} - \mathbf{y})^T (\mathbf{Pw} - \mathbf{y}) + \lambda \mathbf{w}^T \mathbf{w}$$

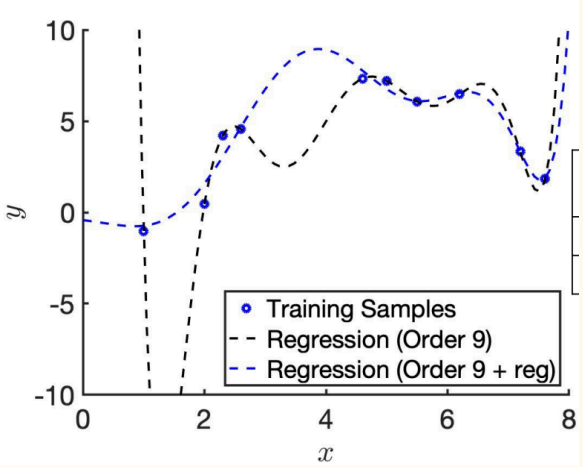
- $\mathbf{w}^T \mathbf{w} = w_0^2 + w_1^2 + \cdots + w_d^2$ — L2 - Regularization
- Encourage $w_0, \cdots, w_d$ to be small (also called shrinkage or weight-decay) => constrain model complexity
- More generally, most machine learning algorithms can be formulated as the following optimization problem

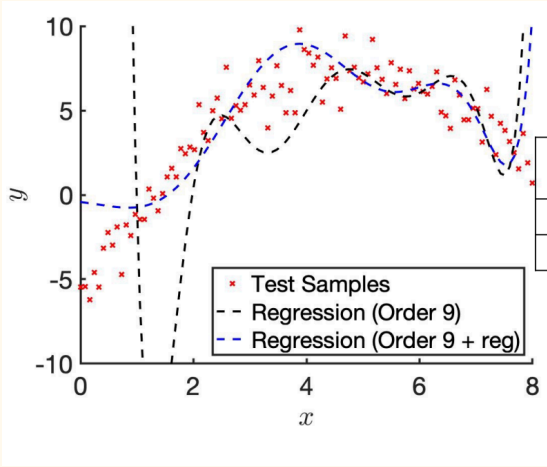
$$\underset{\mathbf{w}}{\operatorname{argmin}}\ \mathbf{Data\text{-}Loss}(\mathbf{w}) + \lambda \mathbf{Regularization}(\mathbf{w})$$

- **Data-Loss(w)** quantifies fitting error to training set given parameters **w:** smaller error => better fit to training data
- **Regularization(w)** penalizes more complex models

  - Because its squared, its called L2 - Regularization
  - Bigger λ means regularisation dominates, forces model to be less complex at the cost of increasing MSE to training data

Regularisation Example





	Training Set Fit	Test Set Fit
Order 9	Good	Bad
Order 9, λ = 1	Good	Good

Python demo

  - Can see that with Ridge regression, the blue curve does not fit the training data as well anymore. But it fit the overall data much better

Bias-Variance Trade-off

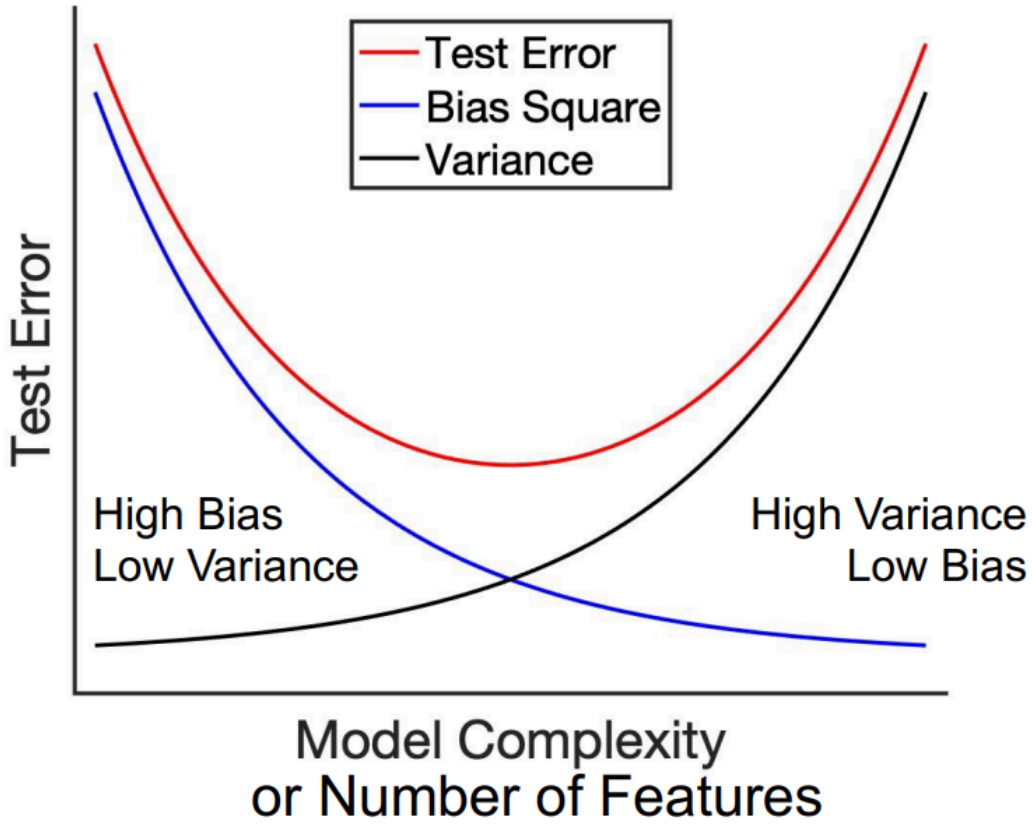
- Suppose we are trying to predict red target below:

- Both bias and variance contribute to prediction error

Bias-Variance Decomposition Theorem

- **Test error** = Bias Squared + Variance + Irreducible Noise
  - Mathematical details in optional uploaded material (won't be tested)
- "**Variance**" refers to variability of prediction models across different training sets
  - In previous example, every time the training set of 10 samples changes, the trained model changes
  - "Variance" quantifies variability across trained models
- "**Bias**" refers to how well an average prediction model will perform
  - In previous example, every time the training set of 10 samples changes, the trained model changes
  - If we average the trained models, how well will this average trained model perform?
- "**Irreducible Noise**" reflects the fact that even if we are perfect modelers, it might not be possible to predict target y with 100% accuracy from feature(s) x

- Test error = Bias Squared + Variance + Irreducible Noise



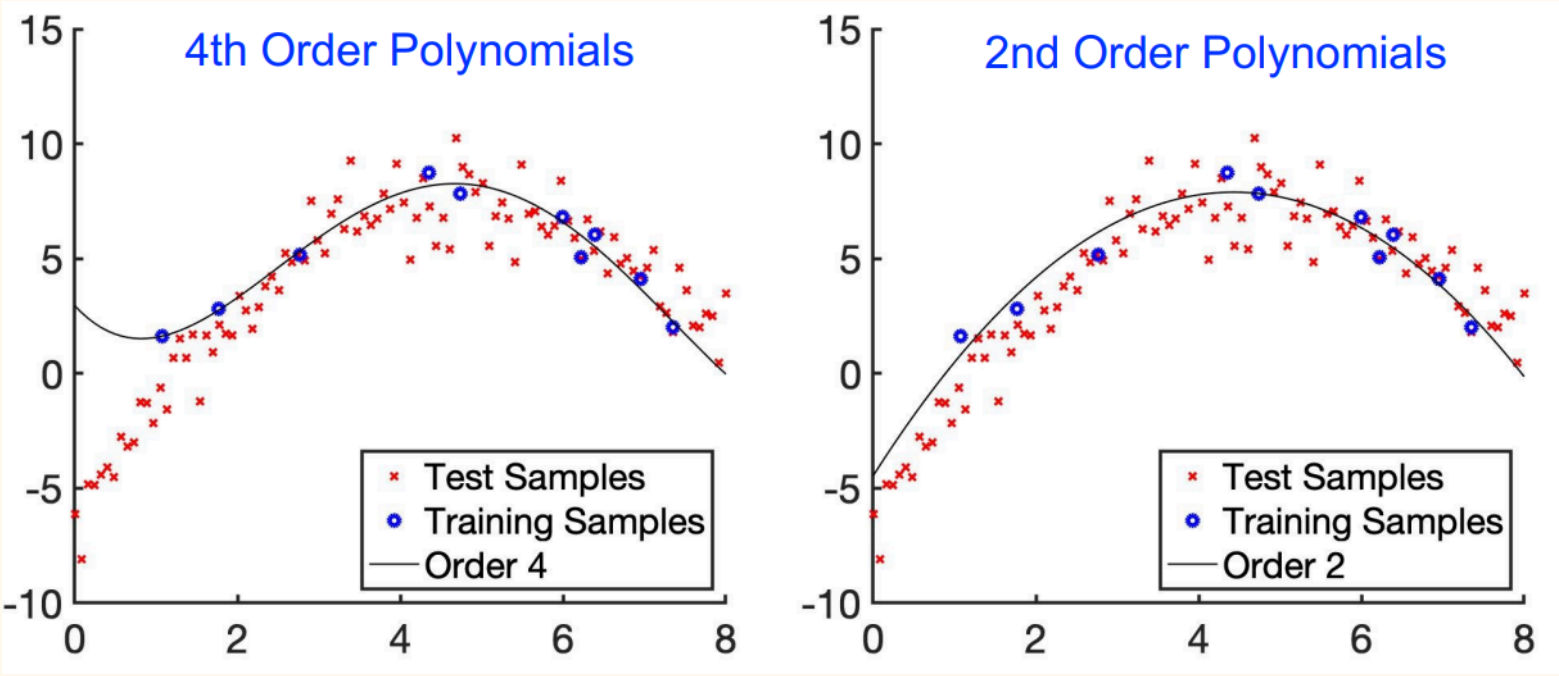
- test error with u shape can be explained here
- With simple models, predictions has high bias but low variance
- With complex models, predictions has low bias but high variance
- This is the bias variance trade off

Example

- Simulate data from order 2 polynomial (+ noise)
- Randomly sample 10 training samples each time
- Fit with order 2 polynomial
- Fit with order 4 polynomial

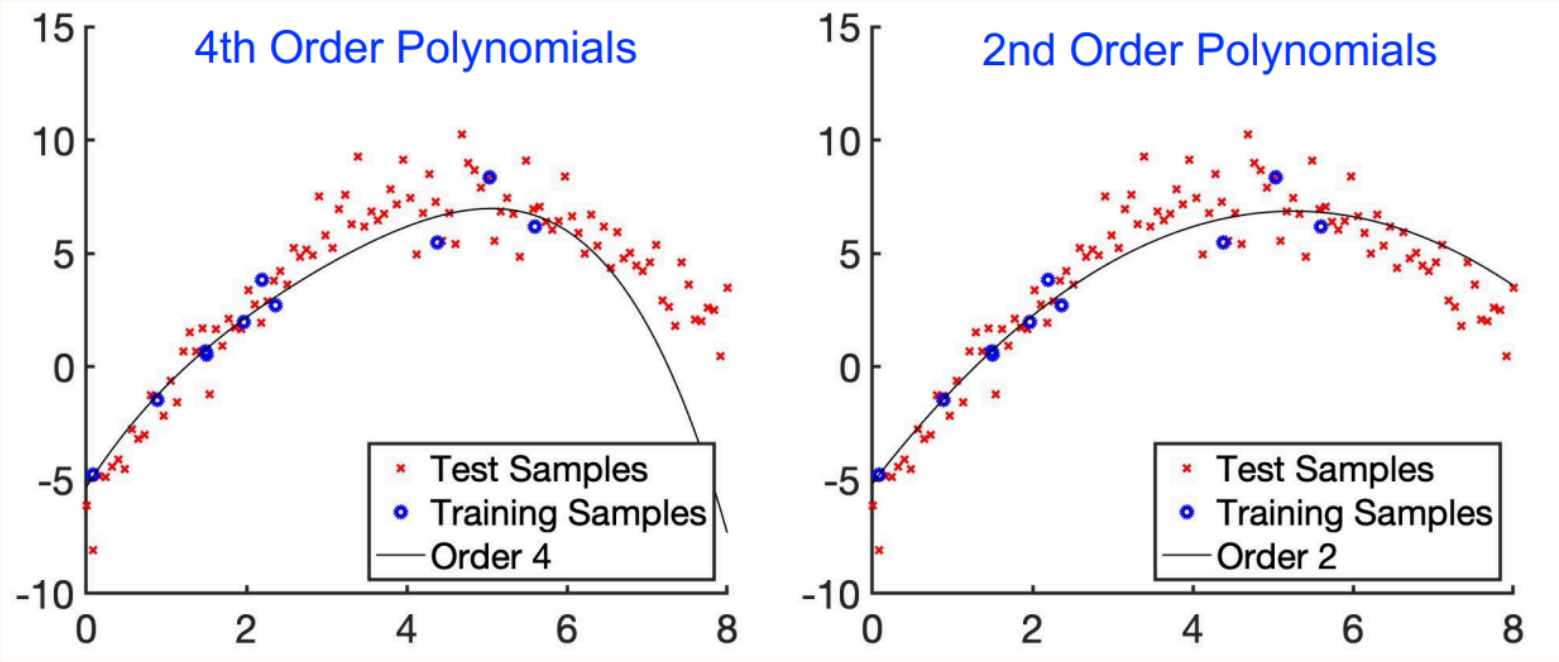
- The data points are the same for all
- The same two models is trained on the same data set each time the experiment is repeated
    - each time the sample is randomly chosen
- Note the variance the 4th order one has

Test 1



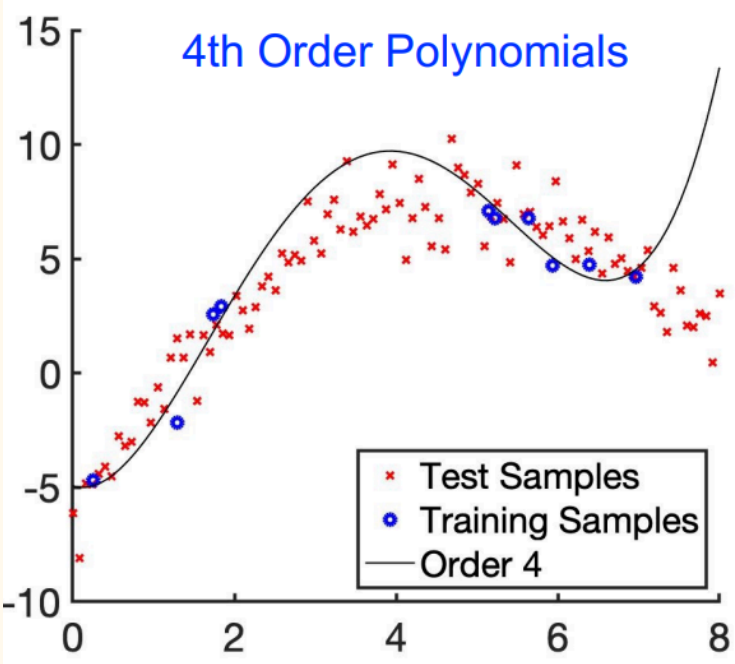
- 4th order has big error on LHS
- 2nd order fit training and test well
- Both fit training quite well

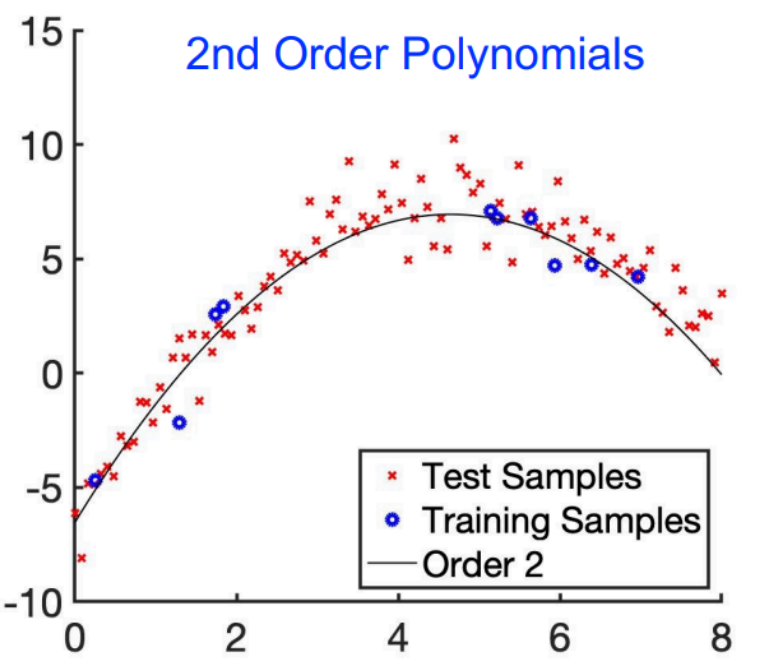
Test 2



- 4th order has big error on RHS
- 2nd order fit training and test well
- Both fit training quite well

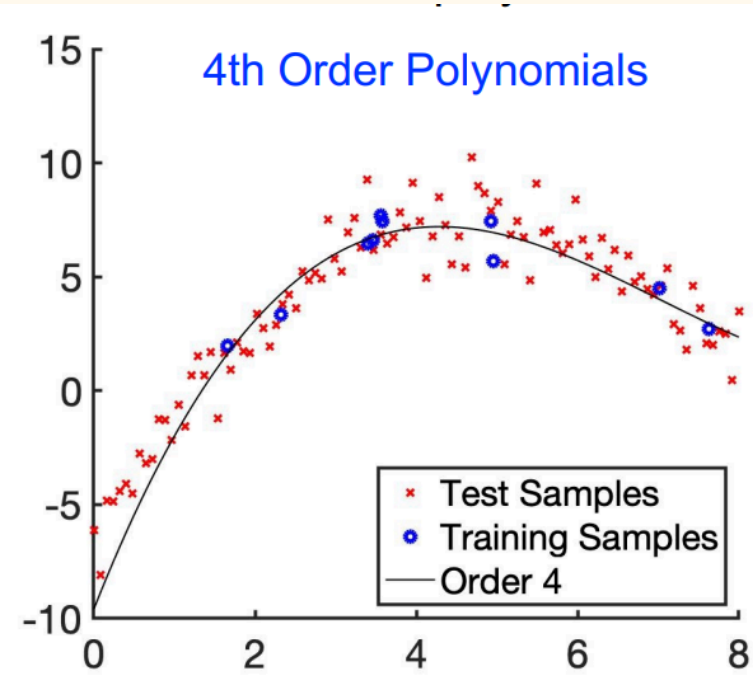
Test 3

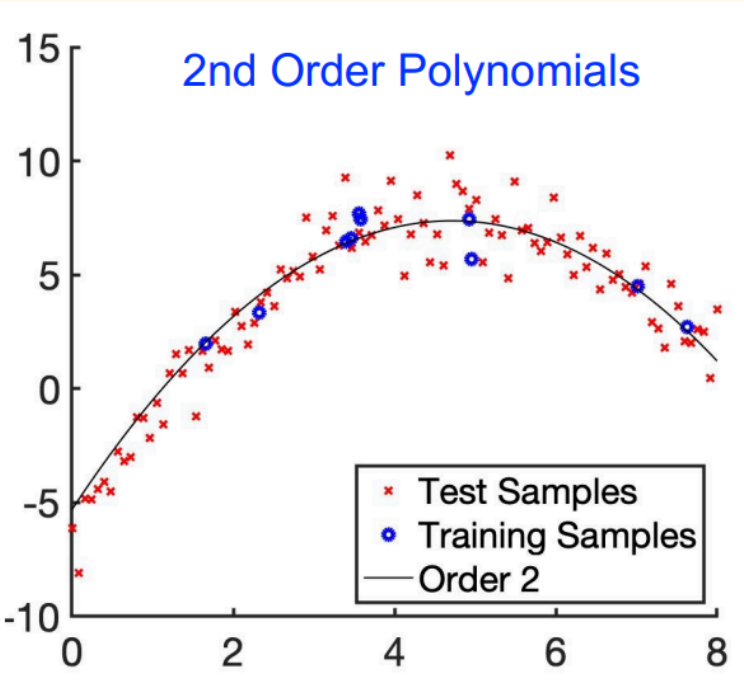




- 4th order has big error on RHS
- 2nd order fit training and test well
- Both fit training quite well

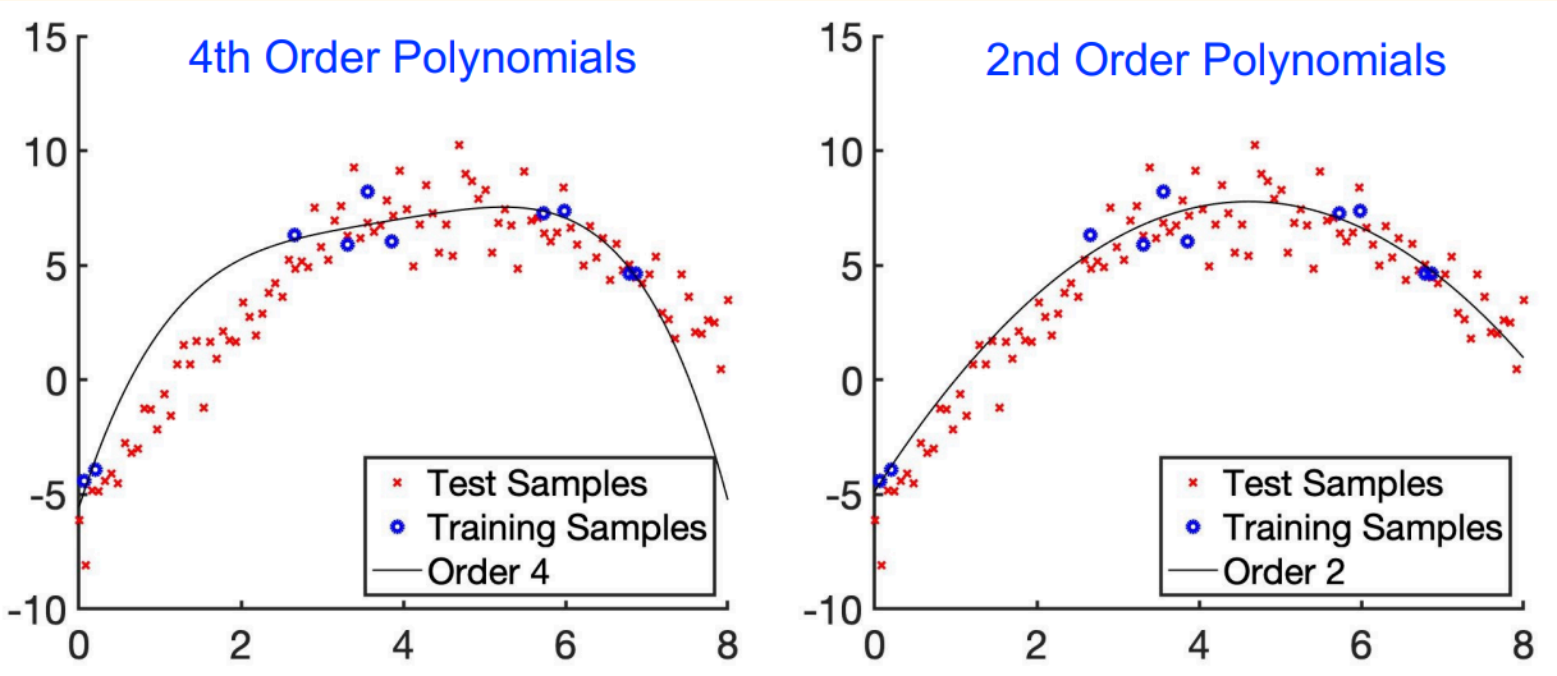
Test 4





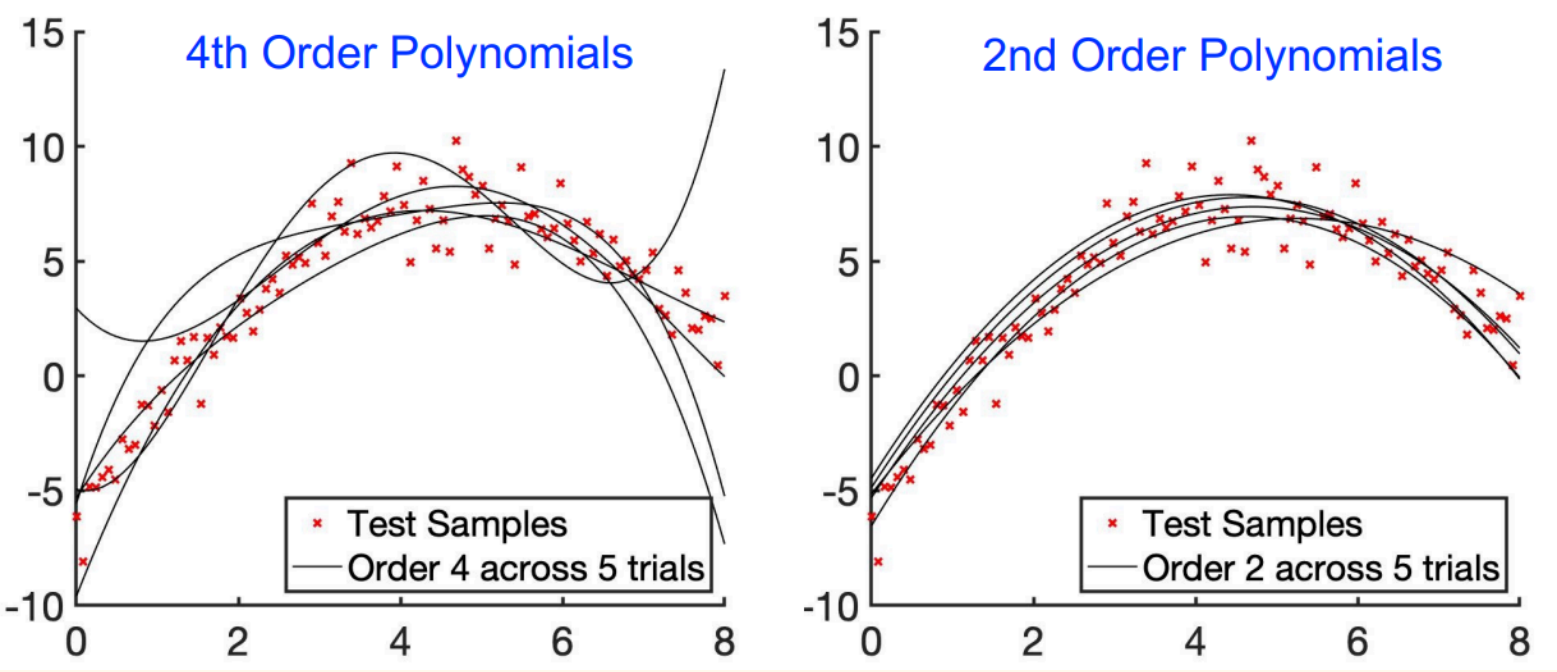
- Both fit training and test quite well

Test 5



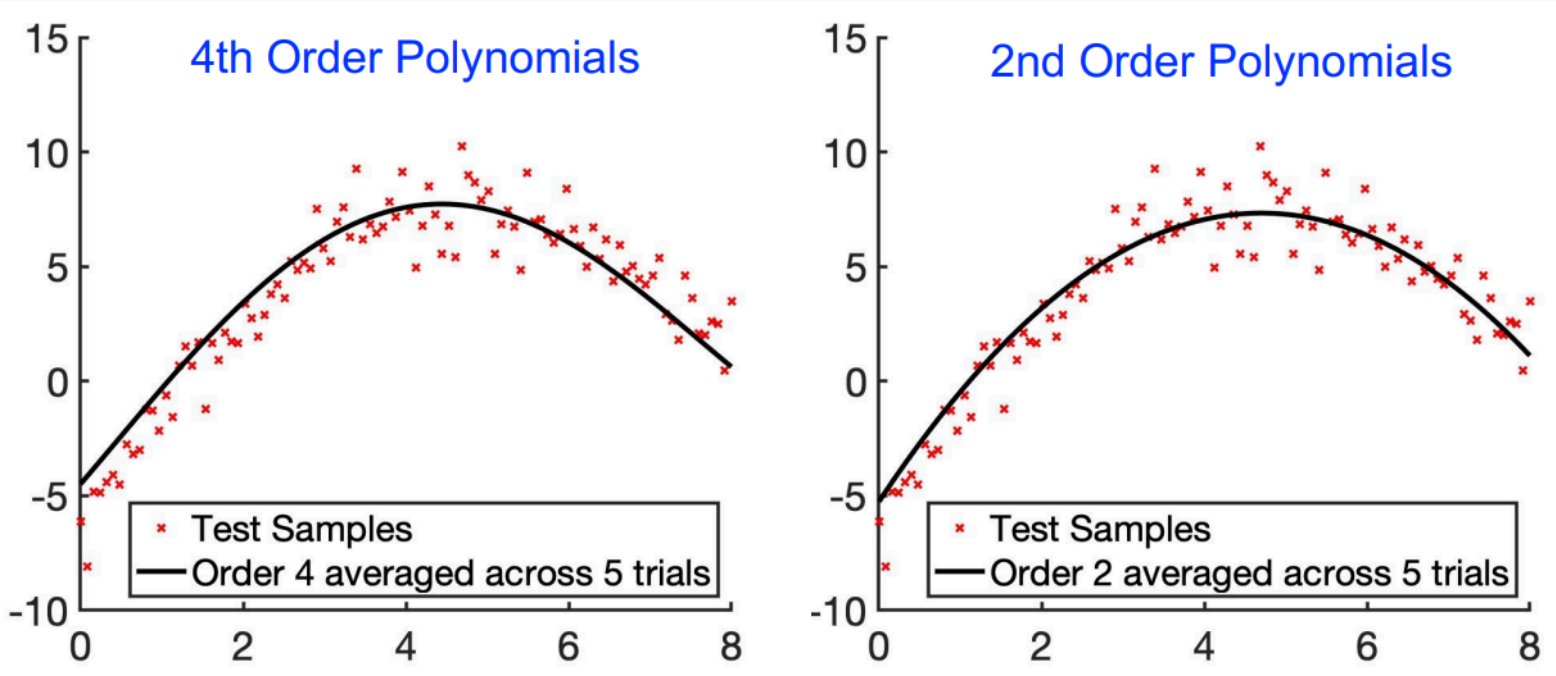
- 4th order has big error on LHS
- 2nd order fit training and test well
- Both fit training quite well

All together



- 4th varies a lot, high variance
- 2nd all quite similar, low variance

Average



- Both averages follows the test data well
- Both has low bias

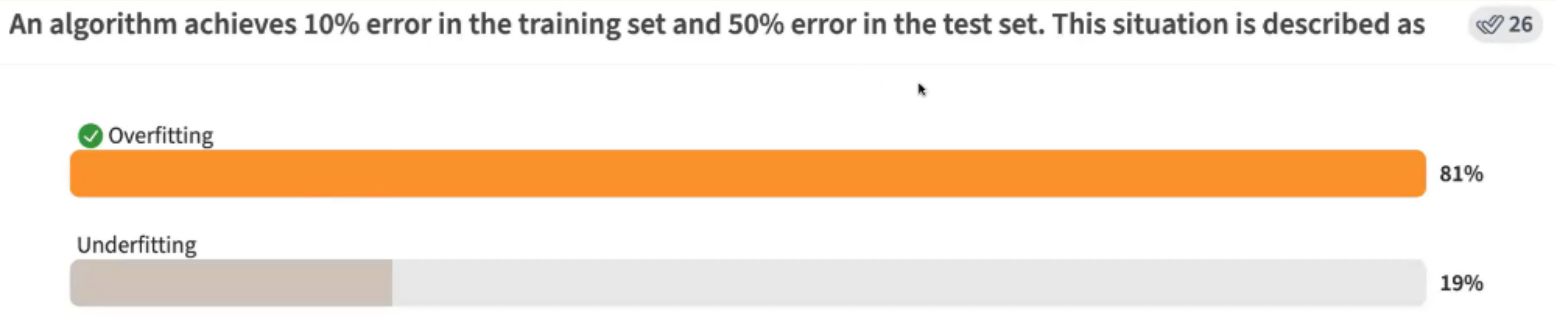
Conclusion

- Fit with order 2 polynomial: low variance, low bias
- Fit with order 4 polynomial: high variance, low bias

} Order 2 Achieves Lower Test Error

- 2nd order performs better and has lower test error

In Class Quiz



Python and Conda Stuff

need to run the conda commands in the conda prompt

2.1 Installation

After installing Anaconda [3], you will probably find `python=3.8.8` in your base(root) environment as shown in the Anaconda Navigator under the environment tab. In this module we will use

```
python=3.8 numpy=1.24
```

So let's create a new environment for EE2211 and we will call it `ee2211`. To do this we enter the following command in the terminal (for Windows user search `Anaconda Powershell Prompt`, for MacOS user just search for `terminal`)

```
conda create --name ee2211 python=3.8 numpy=1.24 matplotlib scikit-learn pandas
```

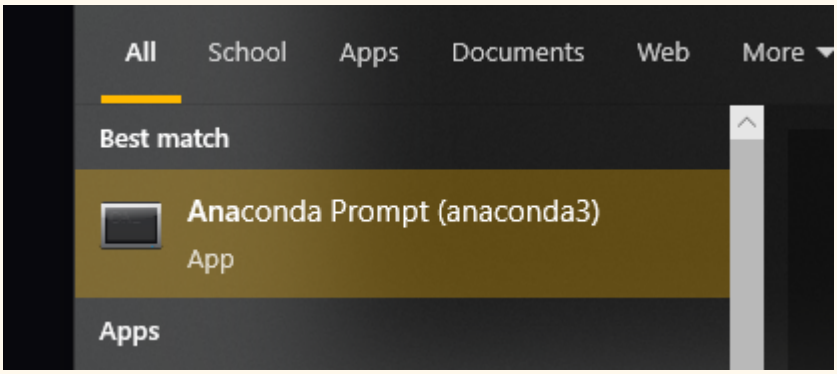
Now in the conda navigator you should find a new environment we just created `ee2211`, with the version we want :-)

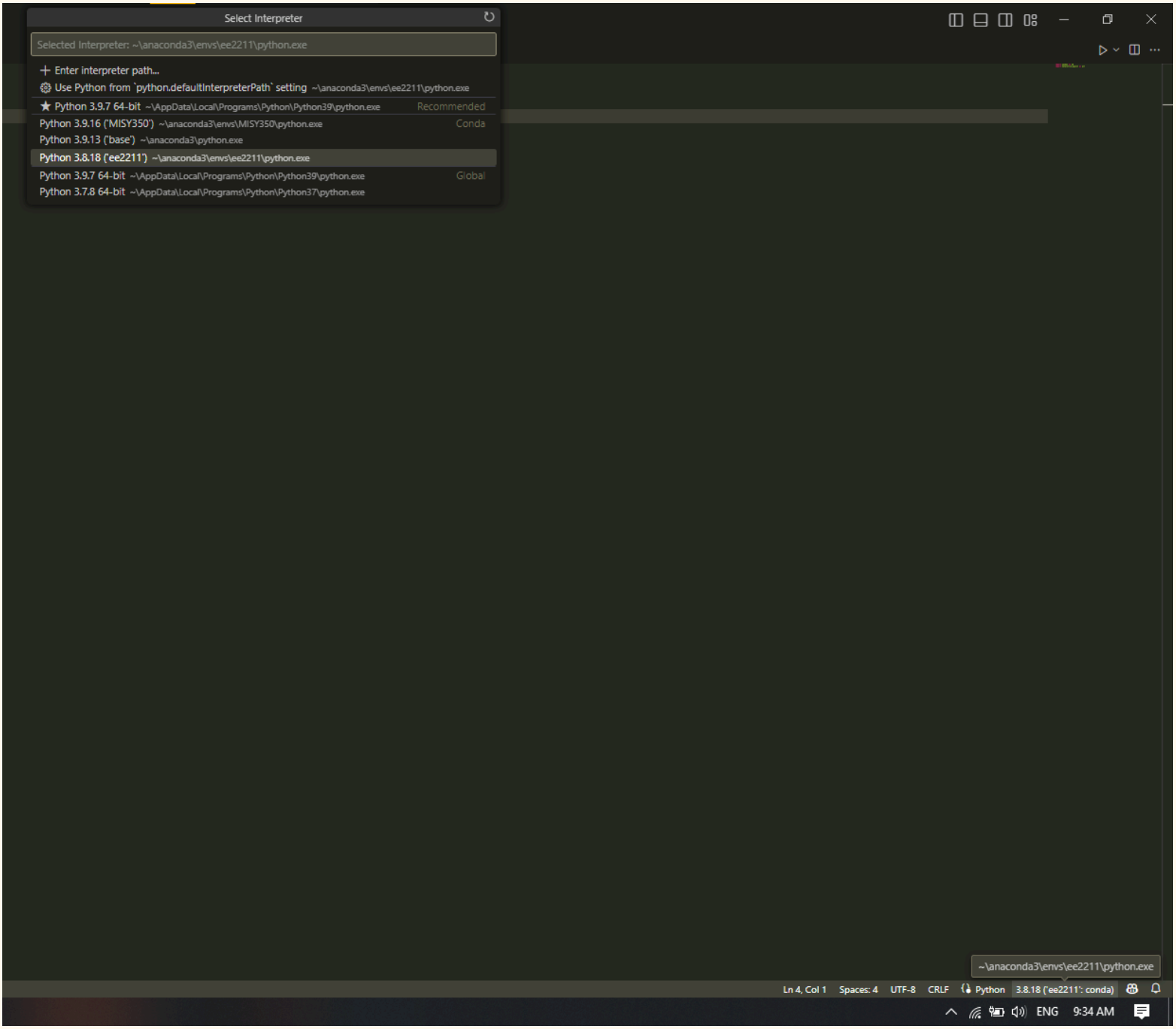
Go ahead and activate the `ee2211` environment, by the following command

```
conda activate ee2211
```

In order to view the Jupyter notebook, let's install it in ee2211

```
conda install -c conda-forge jupyterlab
```





Tutorial 1

Tutorial 2

pandas mean python data analysis
matplotlib to plot
sklearn for machine learning

csv easy to read and create, used for data processing and manipulation

pandas reading csv will add index to the data

```
csv_file = pd.read_csv('GovernmentExpenditureonEducation.csv')
print(csv_file)
```

	year	total_expenditure_on_education
0	1981	942517
1	1982	1358430
2	1983	1611647
3	1984	1769728
4	1985	1812376
5	1986	1641893
6	1987	1654115
7	1988	1604473
8	1989	1765250
9	1990	2056374
10	1991	2816371
11	1992	2597894
12	1993	2902886
13	1994	3318956
14	1995	3443857
15	1996	3771955

https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html

Q1

(Data Reading and Visualization, simple data structure)

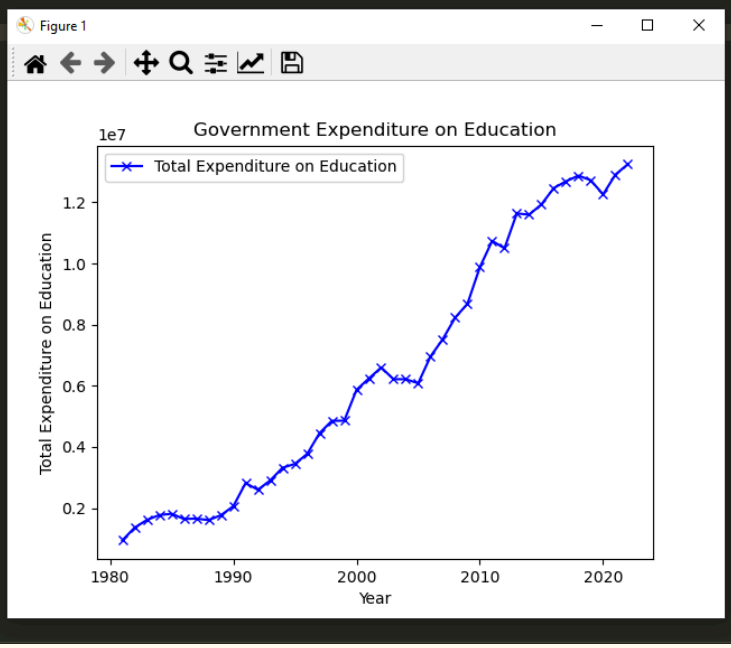
Question 1:

A Comma Separated Values (CSV) file is a plain text file that contains a list of data. These files are often used for exchanging data between different applications. Download the file "government-expenditure-on-education.csv" from https://data.gov.sg/dataset/government-expenditure-on-education. Plot the educational expenditure over the years. (Hint: you might need "`import pandas as pd`" and "`import matplotlib.pyplot as plt`".)

tut2q1.py | tut2q2.py | tut2q3.py

C: > University Stuff > Y2S2 > EE2211 > Tutut > tut2 > tut2q1.py > ...

```
import pandas as pd
import matplotlib.pyplot as plt

dataframe = pd.read_csv('GovernmentExpenditureonEducation.csv')

expenditureList = dataframe['total_expenditure_on_education'].tolist()
yearList = dataframe['year'].tolist()

plt.plot(yearList, expenditureList, color='blue', marker='x')
plt.xlabel('Year')
plt.ylabel('Total Expenditure on Education')
plt.title('Government Expenditure on Education')
plt.legend({'Total Expenditure on Education'})
plt.show()

print("done")
```



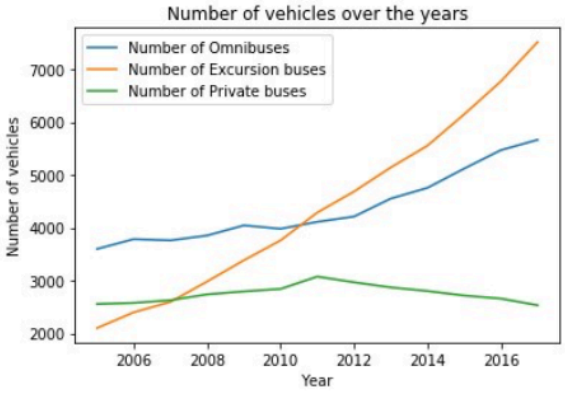
- if absolute file path not used, need to run in the same directory
- use .tolist() to make column into list in python

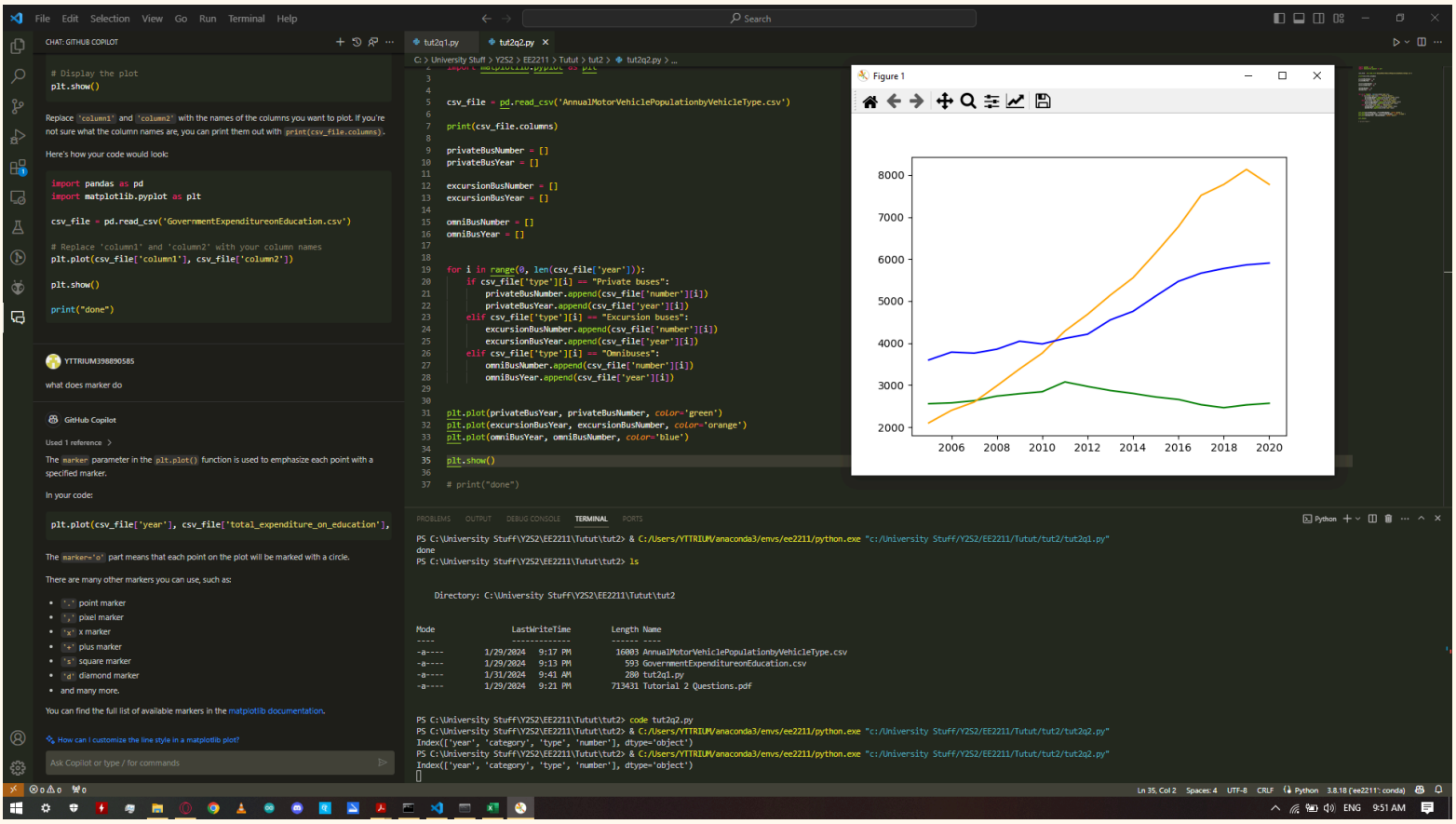
Q2

(Data Reading and Visualization, slightly more complicated data structure)

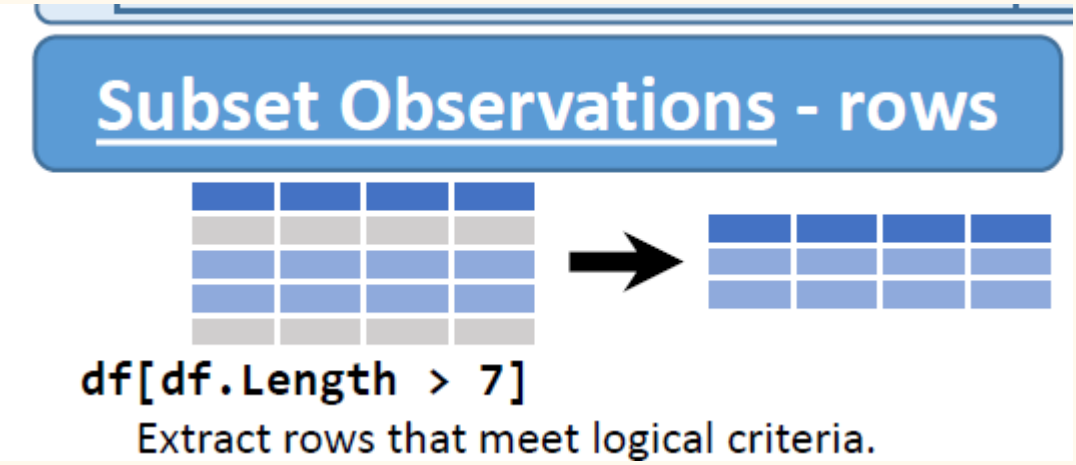
Question 2:

Download the CSV file from https://data.gov.sg/dataset/annual-motor-vehicle-population-by-vehicle-type. Extract and plot the number of Omnibuses, Excursion buses and Private buses over the years as shown below. (Hint: you might need "`import pandas as pd`" and "`import matplotlib.pyplot as plt`".)





- instead of using for loop to extract rows



- extract the index out instead of getting the whole list

```
import pandas as pd
import matplotlib.pyplot as plt


dataframe = pd.read_csv('AnnualMotorVehiclePopulationbyVehicleType.csv')

privateBusIndexList = dataframe[dataframe['type'] == "Private buses"].index.tolist()

print(privateBusIndexList)

privateBusIndexList = dataframe.loc[dataframe['type'] == "Private buses"].index

print(privateBusIndexList)
```

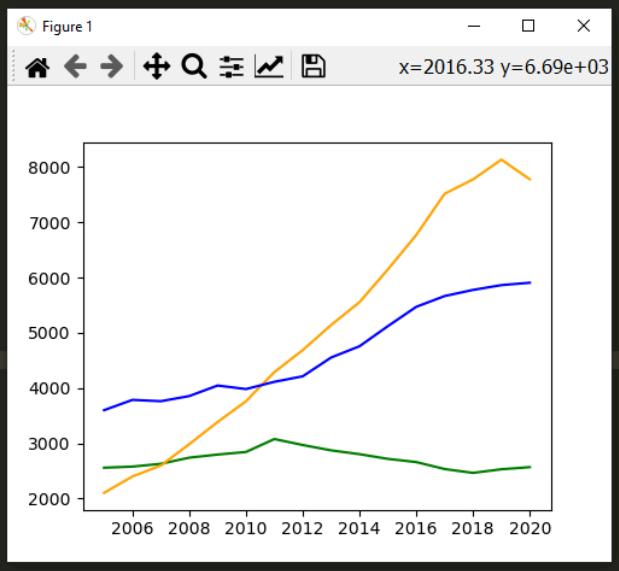
```
[132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 216, 237, 258, 279, 300, 321]
Index([132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 216, 237, 258, 279,
       300, 321],
      dtype='int64')
PS C:\University Stuff\Y2S2\EE2211\Tutut\tut2>
```

  - with .index, the data type is in a diff form

tut2q1.py | tut2q2.py | tut2q3.py

C: > University Stuff > Y2S2 > EE2211 > Tutut > tut2 > tut2q2.py > ...

```python
import pandas as pd
import matplotlib.pyplot as plt


dataframe = pd.read_csv('AnnualMotorVehiclePopulationbyVehicleType.csv')

# privateBusIndexList = dataframe[dataframe['type'] == "Private buses"].index.tolist()
# print(privateBusIndexList)

# privateBusIndexList = dataframe.loc[dataframe['type'] == "Private buses"].index
# excursionBusIndexList = dataframe.loc[dataframe['type'] == "Excursion buses"].index
# omniBusIndexList = dataframe.loc[dataframe['type'] == "Omnibuses"].index

# privateBusDataframe = dataframe.loc[privateBusIndexList]
# excursionBusDataframe = dataframe.loc[excursionBusIndexList]
# omniBusDataframe = dataframe.loc[omniBusIndexList]

# or without extracting the index first
privateBusDataframe = dataframe.loc[dataframe['type'] == "Private buses"]
excursionBusDataframe = dataframe.loc[dataframe['type'] == "Excursion buses"]
omniBusDataframe = dataframe.loc[dataframe['type'] == "Omnibuses"]

plt.plot(privateBusDataframe['year'], privateBusDataframe['number'], color='green')
plt.plot(excursionBusDataframe['year'], excursionBusDataframe['number'], color='orange')
plt.plot(omniBusDataframe['year'], omniBusDataframe['number'], color='blue')

plt.show()



# privateBusNumber = []
# privateBusYear = []

# excursionBusNumber = []
# excursionBusYear = []

# omniBusNumber = []
```

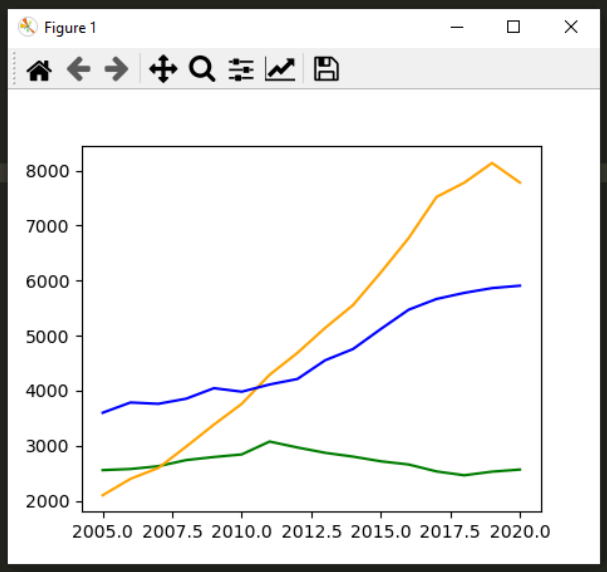


- without extracting index first

tut2q1.py | tut2q2.py | tut2q3.py

C: > University Stuff > Y2S2 > EE2211 > Tutut > tut2 > tut2q2.py > ...

```python
import pandas as pd
import matplotlib.pyplot as plt


dataframe = pd.read_csv('AnnualMotorVehiclePopulationbyVehicleType.csv')

# privateBusIndexList = dataframe[dataframe['type'] == "Private buses"].index.tolist()
# print(privateBusIndexList)

privateBusIndexList = dataframe.loc[dataframe['type'] == "Private buses"].index
excursionBusIndexList = dataframe.loc[dataframe['type'] == "Excursion buses"].index
omniBusIndexList = dataframe.loc[dataframe['type'] == "Omnibuses"].index

privateBusDataframe = dataframe.loc[privateBusIndexList]
excursionBusDataframe = dataframe.loc[excursionBusIndexList]
omniBusDataframe = dataframe.loc[omniBusIndexList]

# or without extracting the index first
# privateBusDataframe = dataframe.loc[dataframe['type'] == "Private buses"]
# excursionBusDataframe = dataframe.loc[dataframe['type'] == "Excursion buses"]
# omniBusDataframe = dataframe.loc[dataframe['type'] == "Omnibuses"]

plt.plot(privateBusDataframe['year'], privateBusDataframe['number'], color='green')
plt.plot(excursionBusDataframe['year'], excursionBusDataframe['number'], color='orange')
plt.plot(omniBusDataframe['year'], omniBusDataframe['number'], color='blue')

plt.show()



# privateBusNumber = []
# privateBusYear = []
```



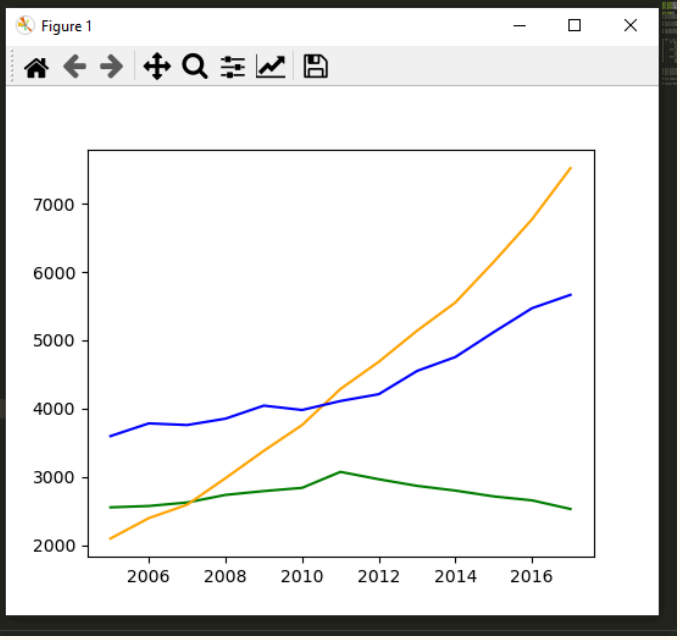
- extracting index first

Cut away the year after 2017

tut2q1.py | tut2q2.py | tut2q3.py

C: > University Stuff > Y2S2 > EE2211 > Tutut > tut2 > tut2q2.py > ...

```python
privateBusIndexList = dataframe.loc[dataframe['type'] == "Private buses"].index
excursionBusIndexList = dataframe.loc[dataframe['type'] == "Excursion buses"].index
omniBusIndexList = dataframe.loc[dataframe['type'] == "Omnibuses"].index

privateBusDataframe = dataframe.loc[privateBusIndexList]
excursionBusDataframe = dataframe.loc[excursionBusIndexList]
omniBusDataframe = dataframe.loc[omniBusIndexList]

# or without extracting the index first
# privateBusDataframe = dataframe.loc[dataframe['type'] == "Private buses"]
# excursionBusDataframe = dataframe.loc[dataframe['type'] == "Excursion buses"]
# omniBusDataframe = dataframe.loc[dataframe['type'] == "Omnibuses"]

plt.plot(privateBusDataframe['year'], privateBusDataframe['number'], color='green')
plt.plot(excursionBusDataframe['year'], excursionBusDataframe['number'], color='orange')
plt.plot(omniBusDataframe['year'], omniBusDataframe['number'], color='blue')

plt.show()

yearCut = 2017

print("cutting away years after {}".format(yearCut))
privateBusDataframe = privateBusDataframe.loc[privateBusDataframe['year'] <= yearCut]
excursionBusDataframe = excursionBusDataframe.loc[excursionBusDataframe['year'] <= yearCut]
omniBusDataframe = omniBusDataframe.loc[omniBusDataframe['year'] <= yearCut]

plt.plot(privateBusDataframe['year'], privateBusDataframe['number'], color='green')
plt.plot(excursionBusDataframe['year'], excursionBusDataframe['number'], color='orange')
plt.plot(omniBusDataframe['year'], omniBusDataframe['number'], color='blue')

plt.show()
# privateBusNumber = []
# privateBusYear = []

# excursionBusNumber = []
# excursionBusYear = []
```

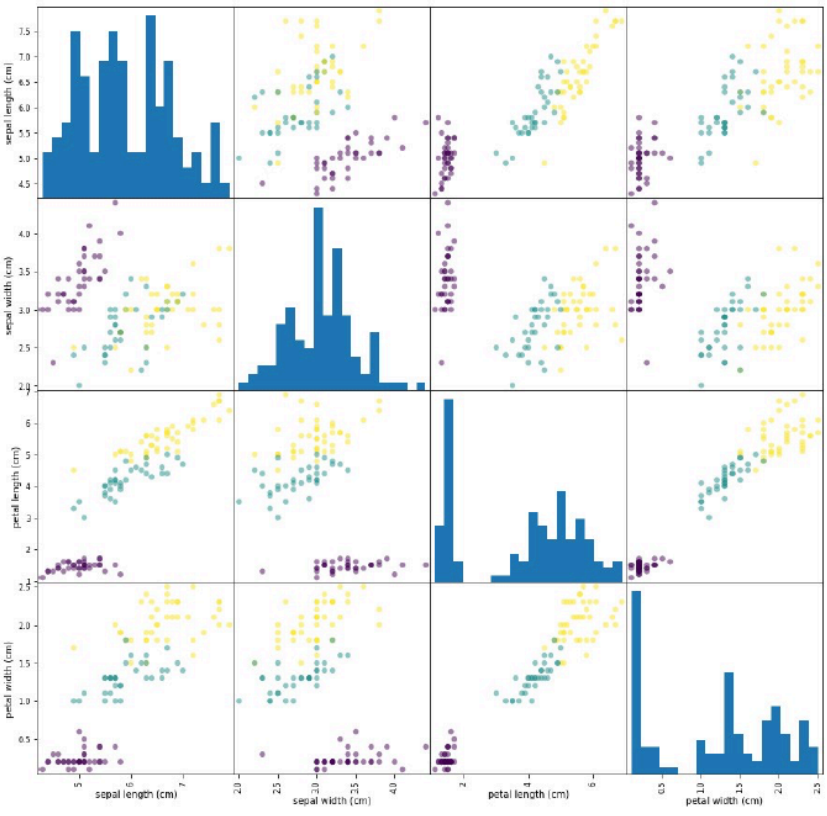


Q3

(Data Reading and Visualization, distribution)

Question 3:

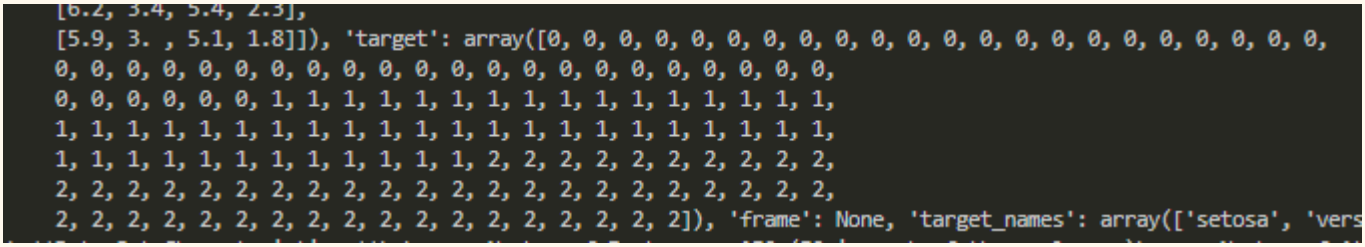
The "iris" flower data set consists of measurements such as the length, width of the petals, and the length, width of the sepals, all measured in centimeters, associated with each iris flower. Get the data set "from sklearn.datasets import load_iris" and do a scatter plot as shown below. (Hint: you might need "`from pandas.plotting import scatter_matrix`")

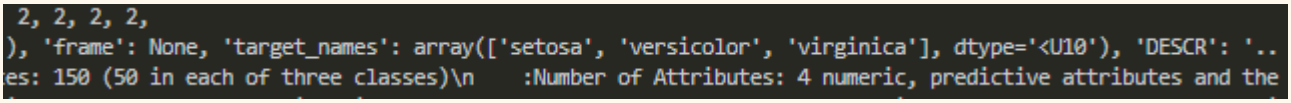


C: > University Stuff > Y2S2 > EE2211 > Tutut > tut2 > tut2q3.py > ...

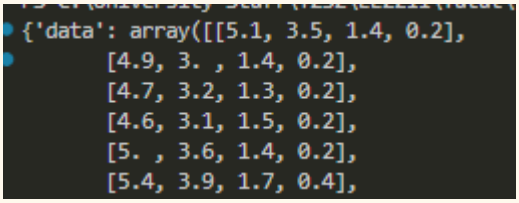
```python
from sklearn.datasets import load_iris

import pandas as pd

from pandas.plotting import scatter_matrix

import matplotlib.pyplot as plt

# Load the iris dataset
iris_dataset = load_iris()

print(iris_dataset)
```

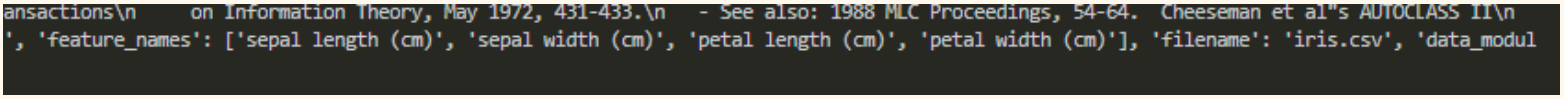
sklearn build in dataset for trying shit out





- target name is specified here,
- target name/species name should be the output of learning algo





- feature is here
- feature should be the input to learning algo

sklearn.model_selection.train_test_split

`sklearn.model_selection.train_test_split(*arrays, test_size=None, train_size=None, random_state=None, shuffle=True, stratify=None)` [source]

Split arrays or matrices into random train and test subsets.

Quick utility that wraps input validation, `next(ShuffleSplit().split(X, y))`, and application to input data into a single call for splitting (and optionally subsampling) data into a one-liner.

Read more in the User Guide.

Parameters:

arrays : sequence of indexables with same length / shape[0]
Allowed inputs are lists, numpy arrays, scipy-sparse matrices or pandas dataframes.

test_size : *float or int, default=None*
If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If int, represents the absolute number of test samples. If None, the value is set to the complement of the train size. If `train_size` is also None, it will be set to 0.25.

train_size : *float or int, default=None*
If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the train split. If int, represents the absolute number of train samples. If None, the value is automatically set to the complement of the test size.

random_state : *int, RandomState instance or None, default=None*
Controls the shuffling applied to the data before applying the split. Pass an int for reproducible output across multiple function calls. See Glossary.

shuffle : *bool, default=True*
Whether or not to shuffle the data before splitting. If shuffle=False then stratify must be None.

stratify : *array-like, default=None*
If not None, data is split in a stratified fashion, using this as the class labels. Read more in the User Guide.

Returns:

splitting : *list, length=2 * len(arrays)*
List containing train-test split of inputs.

New in version 0.16: If the input is sparse, the output will be a `scipy.sparse.csr_matrix`. Else, output type is the same as the input type.

- random state is the seed
- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

https://pandas.pydata.org/docs/reference/api/pandas.plotting.scatter_matrix.html

Q5

(Missing Data)

Question 5:

The Pima Indians Diabetes Dataset involves predicting the onset of diabetes within 5 years in Pima Indians given medical details. Download the Pima-Indians-Diabetes data from https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv.

It is a binary (2-class) classification problem. The number of observations for each class is not balanced. There are 768 observations with 8 input variables and 1 output variable. The variable names are as follows:

0. Number of times pregnant.
1. Plasma glucose concentration a 2 hours in an oral glucose tolerance test.
2. Diastolic blood pressure (mm Hg).
3. Triceps skinfold thickness (mm).
4. 2-Hour serum insulin (mu U/ml).
5. Body mass index (weight in kg/(height in m)^2).
6. Diabetes pedigree function.
7. Age (years).
8. Class variable (0 or 1).

(a) Print the summary statistics of this data set.
(b) Count the number of "0" entries in columns [1,2,3,4,5].
(c) Replace these "0" values by "NaN".

(Hint: you might need the "`.describe()`" and "`.replace(0, numpy.NaN)`" functions "`from pandas import read_csv`".)

- {dataset}.describe() gives summary states, print out to read before processing data

Tutorial 3

Q5

pdf(x, loc=0, scale=1)	Probability density function.
logpdf(x, loc=0, scale=1)	Log of the probability density function.
cdf(x, loc=0, scale=1)	Cumulative distribution function.

The probability density function for `norm` is:

$$f(x) = \frac{\exp(-x^2/2)}{\sqrt{2\pi}}$$

for a real number x.

The probability density above is defined in the "standardized" form. To shift and/or scale the distribution use the `loc` and `scale` parameters. Specifically, `norm.pdf(x, loc, scale)` is identically equivalent to `norm.pdf(y) / scale` with `y = (x - loc) / scale`. Note that shifting the location of a distribution does not make it a "noncentral" distribution; noncentral generalizations of some distributions are available in separate classes.

- loc and scale is mean and standard deviation

Q10-Q11

Question 10: (Fill-in-blank)

The rank of the matrix $\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$ is _BLANK_.

2

Question 11: (Fill-in-blank)

The rank of the matrix $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ is _BLANK_.

3

2, must do ero to ref

- find rank must do ref, for > 2x2, cannot eye ball

Tutorial 4

- X^TX is always symmetrical if $\det(X^TX) \neq 0$
- XX^T is always symmetrical if $\det(XX^T) \neq 0$

Q1

(Systems of Linear Equations)

Question 1:

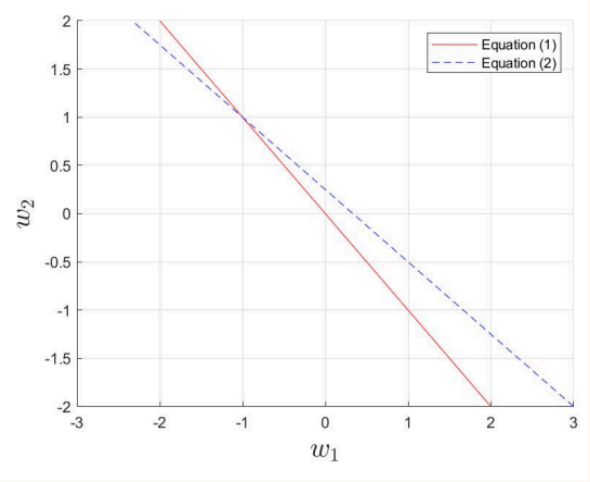
Given $\mathbf{Xw} = \mathbf{y}$ where $\mathbf{X} = \begin{bmatrix} 1 & 1 \\ 3 & 4 \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

even

det X = 1 != 0, invertible

(a) What kind of system is this? (even-, over- or under-determined?)
(b) Is $\mathbf{X}$ invertible? Why?
(c) Solve for $\mathbf{w}$ if it is solvable.

w = (-1, 1)



Q2

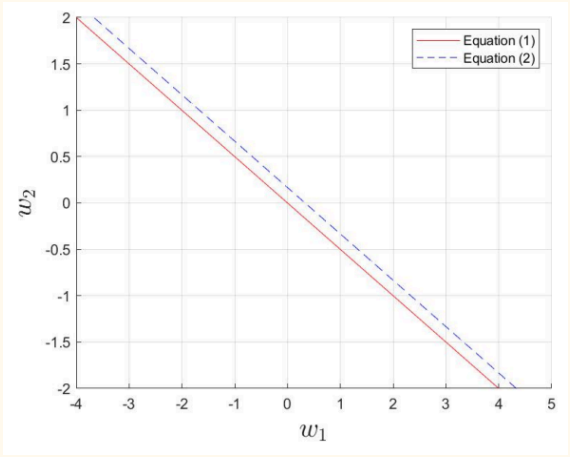
(Systems of Linear Equations)
Question 2:

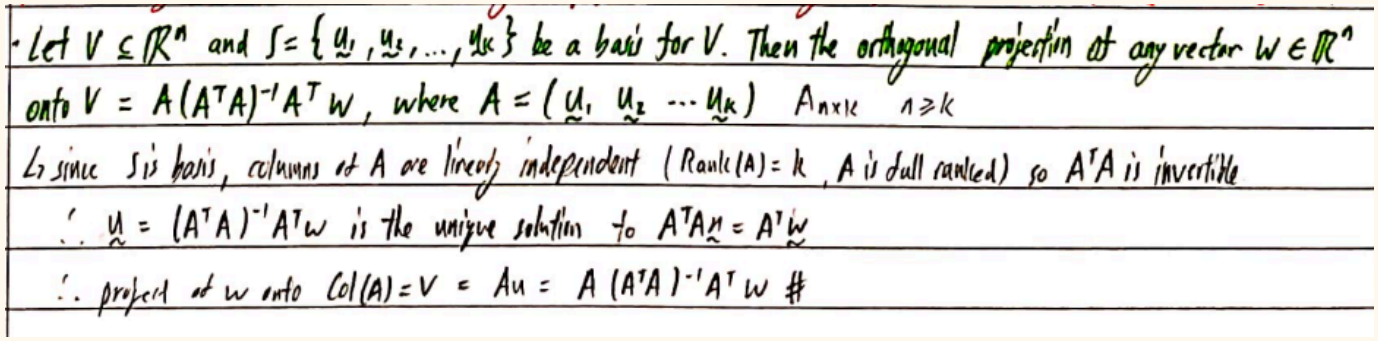
Given $\mathbf{Xw} = \mathbf{y}$ where $\mathbf{X} = \begin{bmatrix} 1 & 2 \\ 3 & 6 \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

even
det X = 0, non invertible

(a) What kind of system is this? (even-, over- or under-determined?)
(b) Is $\mathbf{X}$ invertible? Why?
(c) Solve for $\mathbf{w}$ if it is solvable.

not solvable? can use least square?

- Two parallel lines confirm no solution, there is no least square solution also
- 

- 
- Let $V \subseteq \mathbb{R}^n$ and $S = \{u_1, u_2, \dots, u_k\}$ be a basis for V. Then the orthogonal projection of any vector $w \in \mathbb{R}^n$ onto $V = A(A^TA)^{-1}A^Tw$, where $A = (u_1\ u_2\ \cdots\ u_k)$ $A_{n\times k}$ $n \geq k$
  - since S is basis, columns of A are linearly independent (Rank(A) = k, A is full ranked) so A^TA is invertible
  - $\therefore u = (A^TA)^{-1}A^Tw$ is the unique solution to $A^TAu = A^Tw$
  - $\therefore$ project of w onto $Col(A) = V = Au = A(A^TA)^{-1}A^Tw$ #
  - column must be independent so cannot use this
  - if take out one column then can apply

(Systems of Linear Equations)
Question 3:

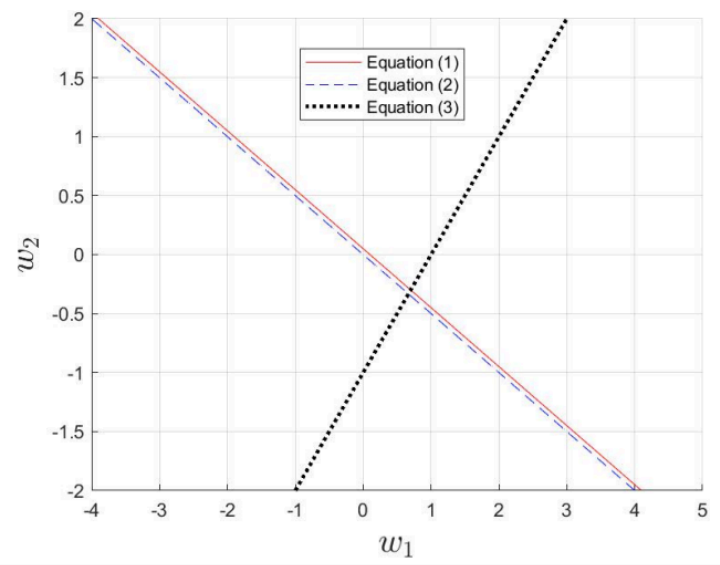
Given $\mathbf{Xw} = \mathbf{y}$ where $\mathbf{X} = \begin{bmatrix} 1 & 2 \\ 2 & 4 \\ 1 & -1 \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} 0 \\ 0.1 \\ 1 \end{bmatrix}$.

(a) What kind of system is this? (even-, over- or under-determined?)
(b) Is $\mathbf{X}$ invertible? Why?
(c) Solve for $\mathbf{w}$ if it is solvable.

over

X is full rank, rank = 2, left inverse exist

w = [0.68 -0.32]

- Two parallel line and one non parallel line intersecting at two points, the w which is least square, gives the between ish between the two intersection points
- 

Q6

(Systems of Linear Equations)
Question 6:

Given $\mathbf{w}^T\mathbf{X} = \mathbf{y}^T$ where

$\mathbf{X} = \begin{bmatrix} 1 & 2 \\ 2 & 4 \\ 1 & -1 \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

(a) What kind of system is this? (even-, over- or under-determined?)
(b) Is $\mathbf{X}$ invertible? Why?
(c) Solve for $\mathbf{w}$ if it is solvable.

XT w = y

after transpose its under determined

over

rank 2 =ful rnak, left inverse exist
w = [0.06666667 0.13333333 -0.33333333]

= [1/15, 2/15, -1/3]

(a) This is an under-determined system (there are 3 unknowns with 2 equations).
(b) $\mathbf{X}$ is NOT invertible but $\mathbf{X}^T\mathbf{X}$ is. The determinant of $\mathbf{X}^T\mathbf{X} = 6\times21 - 9\times9 = 45$.
(c) A constrained solution (exact) is given by

$\hat{\mathbf{w}}^T = (\mathbf{X}\mathbf{a})^T$ (The 3-dimensional vector $\mathbf{w}$ can be constrained by projecting $\mathbf{X}$ onto a 2-dimensional vector $\mathbf{a}$)

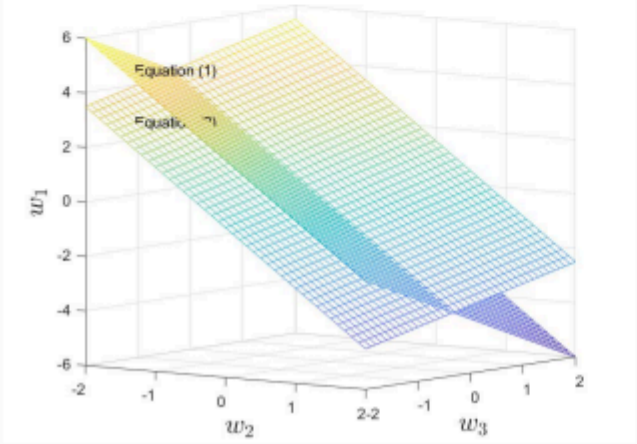
$$= \mathbf{a}^T\mathbf{X}^T$$

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

$$= [0 \quad 1]\begin{bmatrix} 0.4667 & -0.2 \\ -0.2 & 0.1333 \end{bmatrix}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & -1 \end{bmatrix}$$

$$= [0.0667 \quad 0.1333 \quad -0.3333\ ].$$

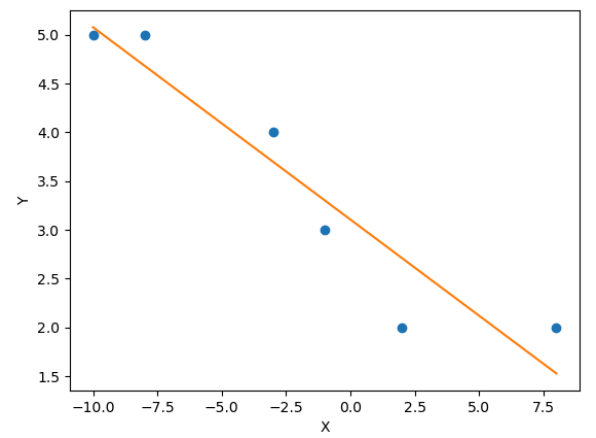
Note: dim($\mathbf{X}$) is 3×2, dim($\mathbf{a}$) is 2×1, estimation is done/constrained on/to the lower dimension of (3×2) and then projected back to the higher dimension 3.

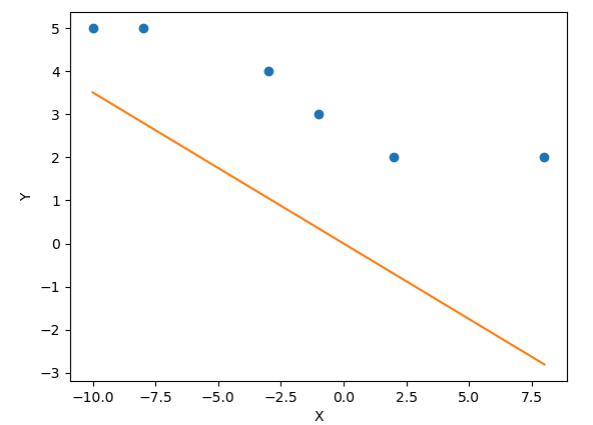


Tutorial 5

Q1

```
# Question 1
print("Question 1")

X = np.array([[-10], [-8], [-3], [-1], [2], [8]])
Y = np.array([[5], [5], [4], [3], [2], [2]])
w = helper.linearRegressionWithBias(X, Y, printResult=False, printFeature=1)
print(w)

X = np.array([[-10], [-8], [-3], [-1], [2], [8]])
Y = np.array([[5], [5], [4], [3], [2], [2]])
w = helper.linearRegressionWithoutBias(X, Y, printResult=False, printFeature=0)
print(w)
```



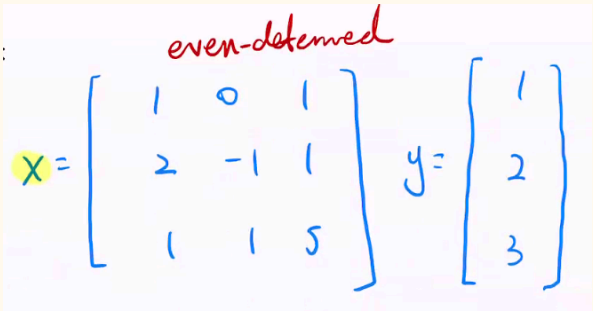



```
W =
 [[ 3.10550459]
 [-0.19724771]]
```

```
W =
 [[-0.35123967]]
```

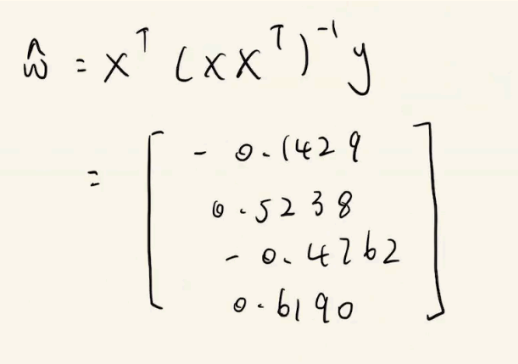
bias term allows the regression to move away from the origin and reduce error

Q2

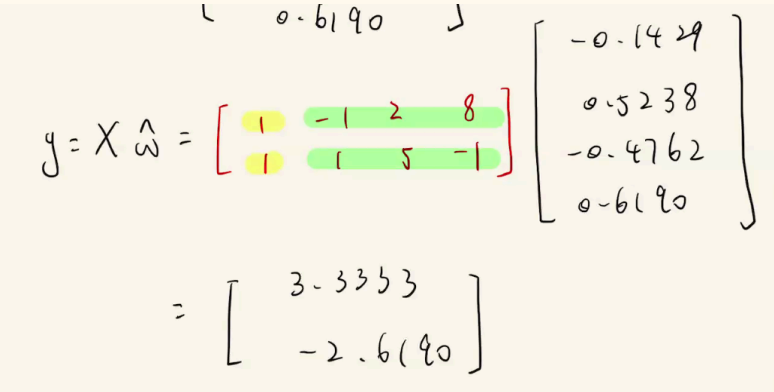


- even determined with full rank can get answer by finding the inverse

```
Question 2
rank(X) =  3
W =
 [[ 0.33333333]
 [-0.66666667]
 [ 0.66666667]]
MSE =  1.4725403564125555e-29
Ytest1_nobias =
 [[3.66666667]]
Ytest2_nobias =
 [[-3.66666667]]
rank(X) =  3
W =
 [[-1.5 ]
 [-1.  ]
 [ 3.  ]
 [-0.25]]
MSE =  34.895833333333336
Ytest1_bias =
 [[3.5]]
Ytest2_bias =
 [[12.75]]
```



- should use right inverse



-

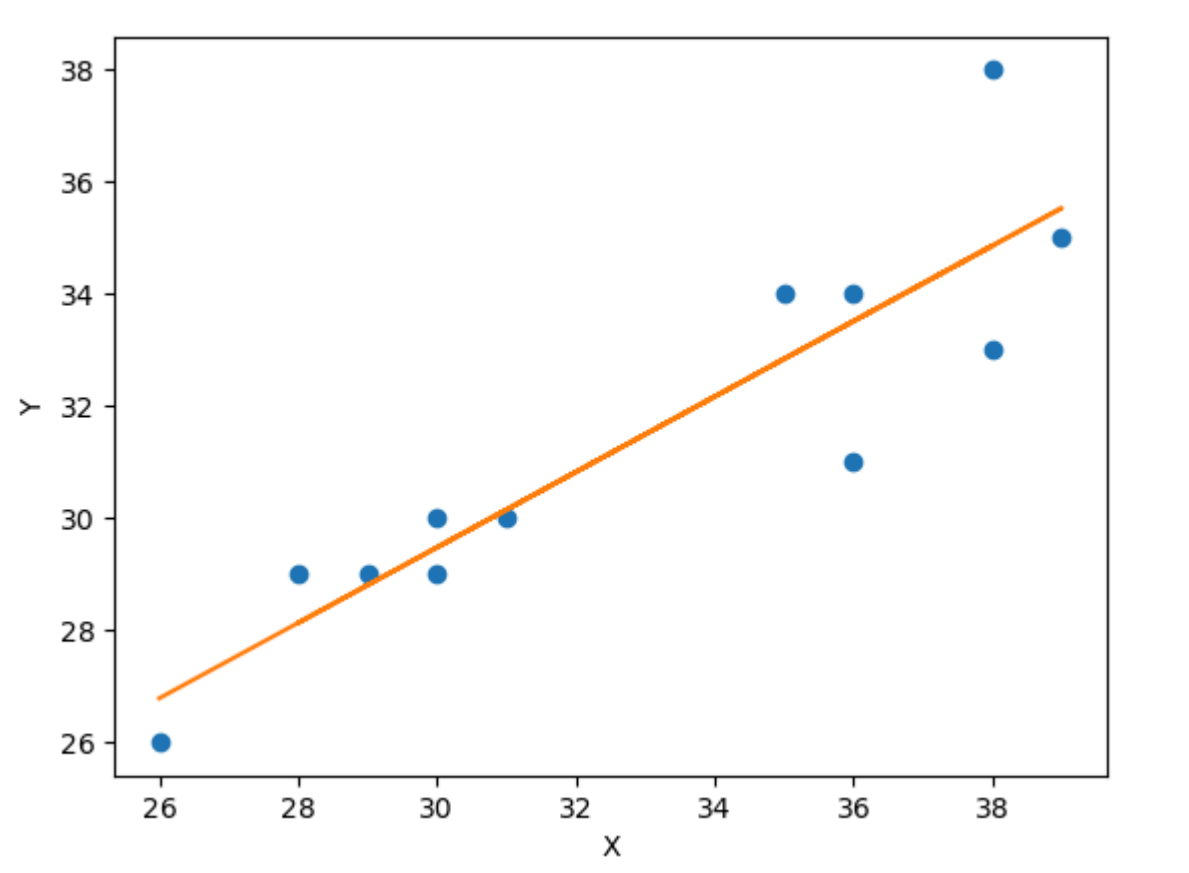
```python
# this is wrong, should use right inverse
W_bias = helper.linearRegressionWithBias(
    X, Y, printResult=True, printFeature=None)

# add in bias
Xtest1 = np.array([[1, -1, 2, 8]])
Xtest2 = np.array([[1, 1, 5, -1]])

Ytest1_bias = helper.testData(Xtest1, W_bias, printResult=False)
print("Ytest1_bias =\n", Ytest1_bias)
Ytest2_bias = helper.testData(Xtest2, W_bias, printResult=False)
print("Ytest2_bias =\n", Ytest2_bias)

# after adding bias, its a wider matrix so should solve with right inverse
W_bias_right = helper.rightInverse(helper.paddingOfOnes(X)) @ Y
print("W_bias_right =\n", W_bias_right)
      You, 1 second ago • Uncommitted changes
Ytest1_bias = helper.testData(Xtest1, W_bias_right, printResult=False)
print("Ytest1_bias =\n", Ytest1_bias)
Ytest2_bias = helper.testData(Xtest2, W_bias_right, printResult=False)
print("Ytest2_bias =\n", Ytest2_bias)
```

```
W_bias_right =
 [[-0.14285714]
 [ 0.52380952]
 [-0.47619048]
 [ 0.61904762]]
Ytest1_bias =
 [[3.33333333]]
Ytest2_bias =
 [[-2.61904762]]
```

The wide matrix has a left inverse as the 3x3 is already full rank, so the dim of row is 3, after padding ones, dimension remains

- hence can find left inverse, but it's not the most accurate regression as there is no interpretation of least squares error in this case

Q3



```
rank(X) =  2
W =
 [[9.3       ]
 [0.67272727]]
MSE =  1.9530303030303038
Ytest1 =
 [[29.48181818]]
Ytest2 =
 [[12.66363636]]
```

always include bias if not specified

- predicted that 30 student buy around 30 books, one per student
- predicted that 5 student buy around 12 books, 2 per student
- 5 student is out of the range of regression, not appropriate to use to same regression
    - not sure whether the regression still fits, cannot extrapolate
    - like gea1000

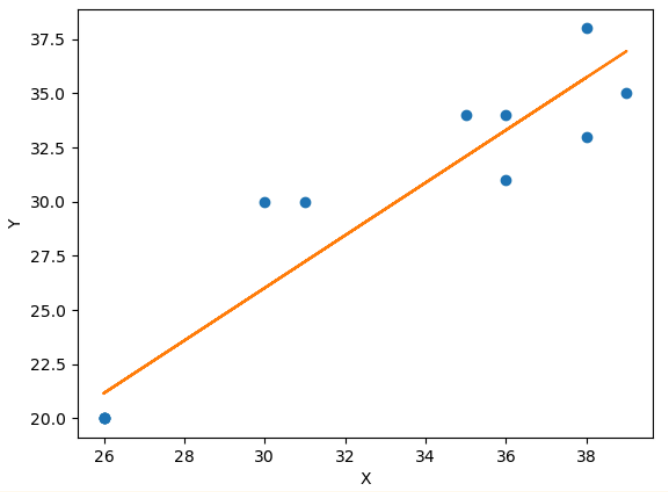
Q4

Semester	Students	Books
1	36	31
2	26	20
3	35	34
4	39	35
5	26	20
6	30	30
7	31	30
8	38	38
9	36	34
10	38	33
11	26	20
12	26	20

Without removing duplicates VS removing the duplicates

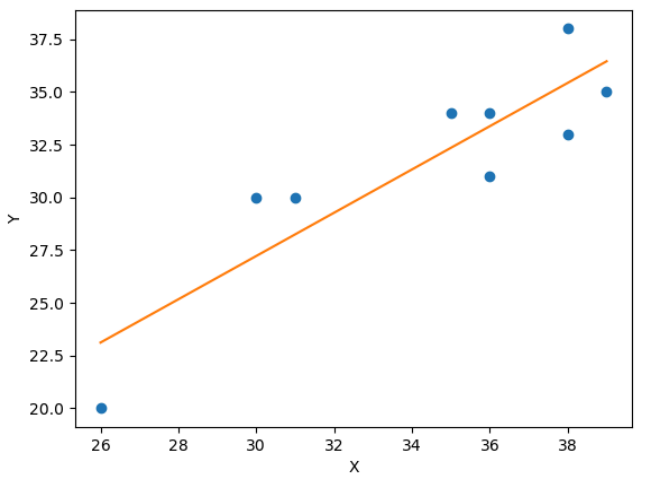
```
X = np.array([[36], [26], [35], [39], [26], [30],
              [31], [38], [36], [38], [26], [26]])
Y = np.array([[31], [20], [34], [35], [20], [30],
              [30], [38], [34], [33], [20], [20]])

# xw = Y regression
w = helper.linearRegressionWithBias(X, Y, printResult=True, printFeature=1)

# predict
Xtest1 = np.array([[1, 30]])
Ytest1 = helper.testData(Xtest1, w, printResult=False)
print("Ytest1 =\n", Ytest1)
```

```
rank(X) =  2
W =
 [[-10.41257051]
 [  1.21434327]]
MSE =  4.562181036798282
Ytest1 =
 [[26.01772764]]
```



```
# remove duplicates
print("X =\n", X)
print("Y =\n", Y)
combined = np.hstack((X, Y))
print("combined =\n", combined)
new_array = [tuple(row) for row in combined]
print("new_array =\n", new_array)
uniques = np.unique(new_array, axis=0)
print("uniques =\n", uniques)
X_unique = uniques[:, 0].reshape(-1, 1)
Y_unique = uniques[:, 1].reshape(-1, 1)
# In this context, -1 is used as a placeholder for "figure out what the dimension should be". It means that the length in that dimension is inferred from the length of the array and remaining dimensions.
# So, .reshape(-1, 1) means to reshape the array to have one column and as many rows as necessary to accommodate the original data.
print("X_unique =\n", X_unique)
print("Y_unique =\n", Y_unique)

# xw = Y regression
w_unique = helper.linearRegressionWithBias(X_unique, Y_unique, printResult=True, printFeature=1)

# predict
Ytest1_unique = helper.testData(Xtest1, w_unique, printResult=False)
print("Ytest1_unique =\n", Ytest1_unique)
```

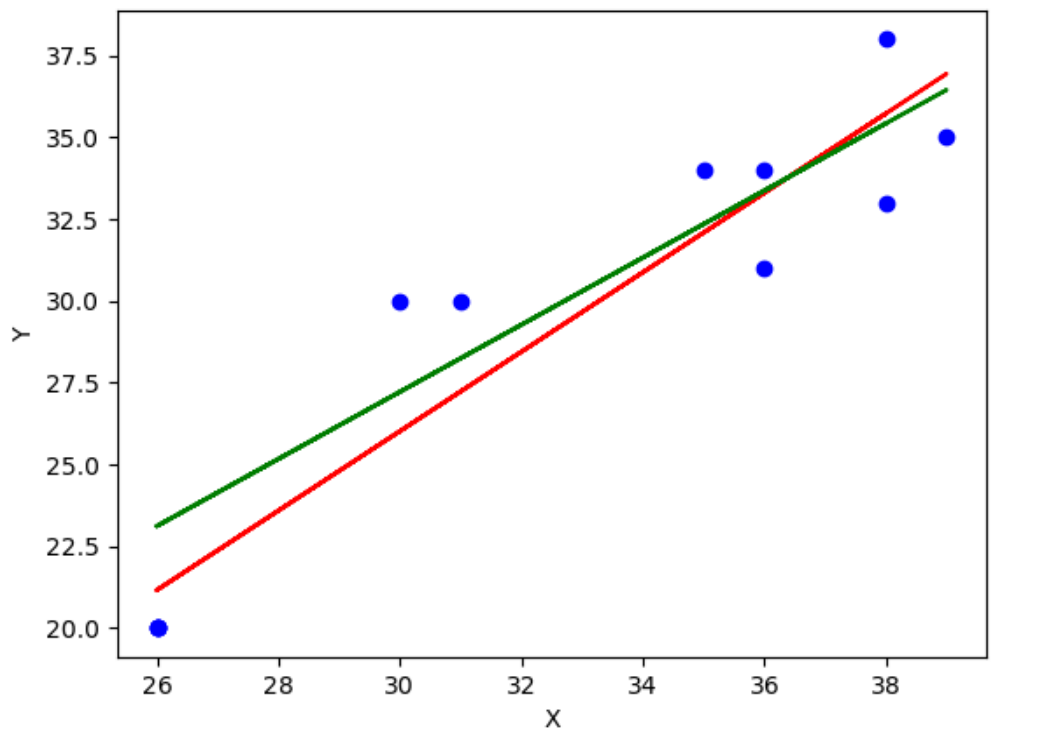
```
rank(X) =  2
W =
 [[-3.55844156]
 [ 1.02597403]]
MSE =  4.8773448773448775
Ytest1_unique =
 [[27.22077922]]
```



- can see that the effect (26, 20) has on the regression is less

Combine so it's easier to compare

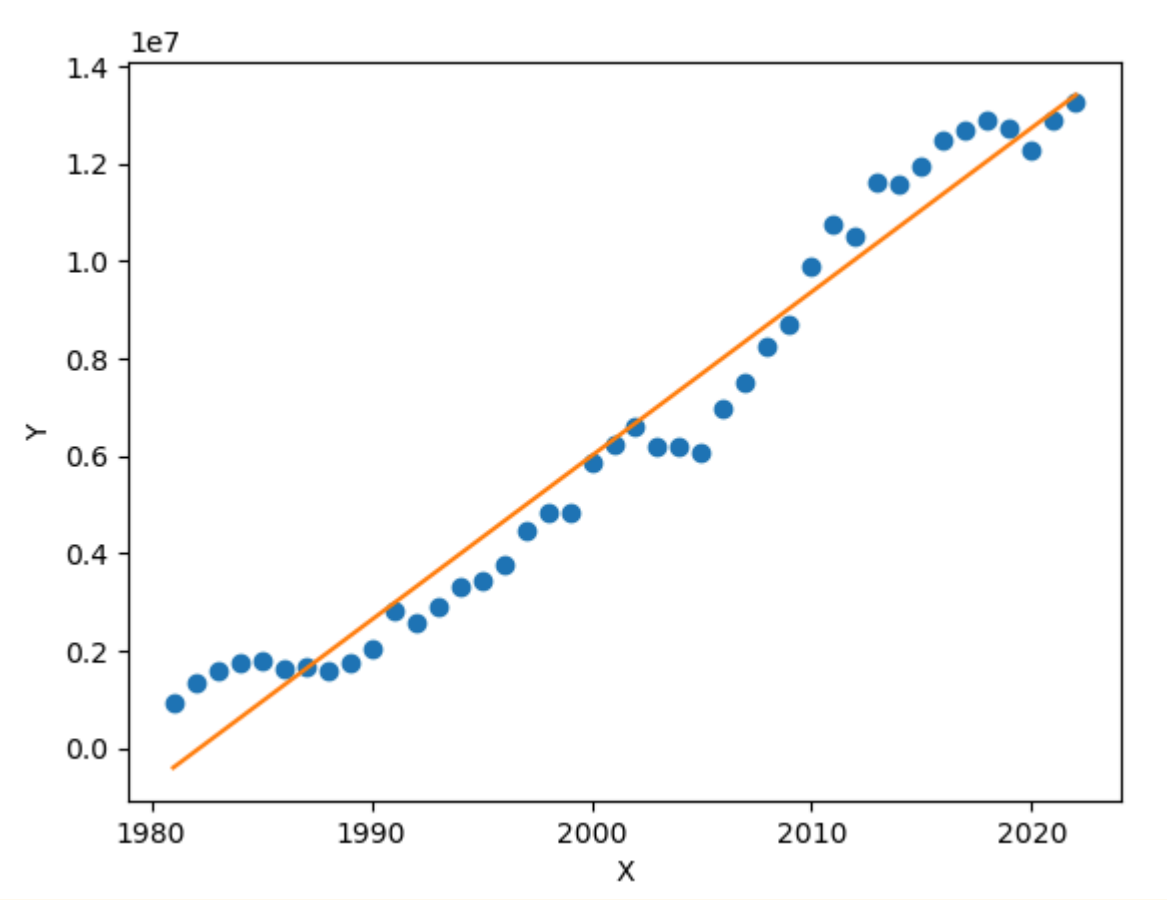
```
# Combining the plots for easier comparison
plt.plot(X, Y, 'o', color='blue')

# plot the graph for the unique data, X needs to be padded for the offset
plt.plot(X, np.hstack((np.ones((X.shape[0], 1)), X)) @ w, '-', color='red')

# plot the graph for the unique data
plt.plot(X, np.hstack((np.ones((X.shape[0], 1)), X)) @ w_unique, '-', color='green')

plt.xlabel('X')
plt.ylabel('Y')

plt.show()
```



- more optimistic after purging before 37

Q5

```
# Question 5 ##################################################################################
print("Question 5 ##################################################################################")
exp_df = pd.read_csv("GovernmentExpenditureonEducation.csv")
exp_df.info()

X = np.array(exp_df['year']).reshape(-1, 1)
Y = np.array(exp_df['total_expenditure_on_education']).reshape(-1, 1)
print("X =\n", X)
print("Y =\n", Y)

# xw = Y regression
W = helper.linearRegressionWithBias(X, Y, printResult=True, printFeature=1)

x_test = helper.paddingOfOnes(np.array([[2021]]))

y_predict = helper.testData(x_test, W, printResult=False)

print("y_predict =\n", y_predict)       You, 1 second ago • Uncommitted changes
```



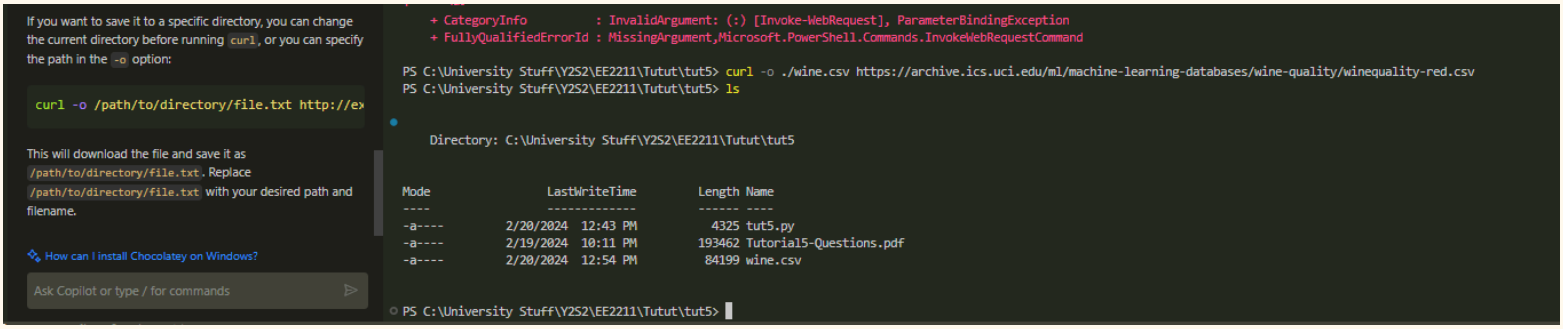
y_predict =
[[13075210.44420362]]

Answer:

The predicted educational expenditure in year 2021 is 12102904.270643068.

- diff from answer probably cuz diff dataset

Q6



- downloading files using curl

```
# Question 6 ###############################################################################
print("Question 6 ###############################################################################")

# wine_df = pd.read_csv("wine.csv", sep=';')
# print(wine_df)

wine_df = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv", sep=";")
wine_df.info()
```

- get data without downloading

```
wine_df = pd.read_csv("wine.csv", sep=';')
print(wine_df)

X = np.array(wine_df.drop('quality', axis=1)) # drop the quality column, the other columns are the features
Y = np.array(wine_df['quality'].tolist()).reshape(-1, 1) # the quality column is the output

print("X\n", X)
print("Y\n", Y)

Xtrain = X[:1500]
Ytrain = Y[:1500]
print("Xtrain\n", Xtrain)
print("Ytrain\n", Ytrain)

Xtest = X[1500:]
Ytest = Y[1500:]
print("Xtest\n", Xtest)
print("Ytest\n", Ytest)

# Training
w = helper.linearRegressionWithBias(Xtrain, Ytrain, printResult=True, printFeature=None)

# Prediction Test
Ytest_pred = helper.testData(helper.paddingOfOnes(Xtest), w, printResult=False)
print("Ytest_pred\n", Ytest_pred)

MSE = mean_squared_error(Ytest_pred, Ytest)
print("MSE = ", MSE)
```

```
W =
 [[ 2.22330327e+01]
 [ 2.68702621e-02]
 [-1.12838019e+00]
 [-2.06141685e-01]
 [ 1.22000584e-02]
 [-1.77718503e+00]
 [ 4.29357454e-03]
 [-3.18953315e-03]
 [-1.81795124e+01]
 [-3.98142390e-01]
 [ 8.92474793e-01]
 [ 2.77147239e-01]]
MSE =  0.4216923253120808
```

MSE on training data

```
MSE =  0.3435263811803818
```

MSE on testing data, lower than 0.2 better regression

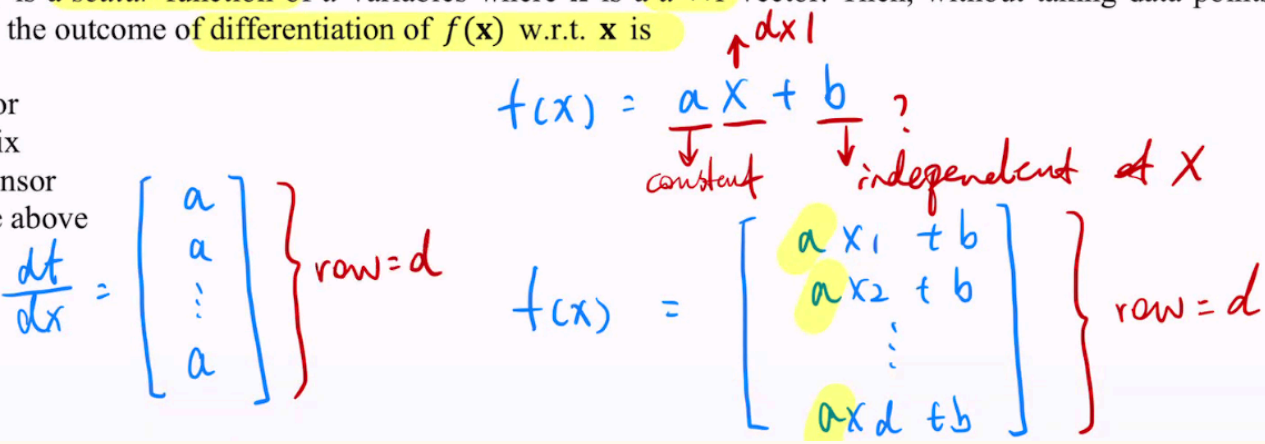
Q8

Question 8:

MCQ: There could be more than one answer.
Suppose $f(\mathbf{x})$ is a *scalar* function of d variables where $\mathbf{x}$ is a $d \times 1$ vector. Then, without taking data points into consideration, the outcome of differentiation of $f(\mathbf{x})$ w.r.t. $\mathbf{x}$ is

a) a scalar
b) a $d \times 1$ vector
c) a $d \times d$ matrix
d) a $d \times d \times d$ tensor
e) None of the above

Answer: b)



Q9

```
X = np.array([[3, -1, 0], [5, 1, 2], [9, -1, 3], [-6, 7, 2], [3, -2, 0]])
Y = np.array([[1, -1], [-1, 0], [1, 2], [0, 3], [1, -2]])

# Training
w = helper.linearRegressionWithBias(X, Y, printResult=True, printFeature=None)

Xtest = helper.paddingOfOnes(np.array([[8, 0, 2]]))
Ytest = helper.testData(Xtest, w, printResult=False)
print("Ytest\n", Ytest)
```

```
rank(X) =  4
W =
 [[ 1.14668974 -0.95997404]
 [-0.630463   -0.33427088]
 [-1.10601471 -0.24426655]
 [ 1.3595846   1.77953267]]
MSE =  0.254110774556469
Ytest
 [[-1.17784509 -0.07507572]]
```

Tutorial 6

Q7

Question 7

MCQ: there could be more than one answer. Given three samples of two-dimensional data points $\mathbf{X} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 3 & 3 \end{bmatrix}$ with corresponding target vector $\mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$. Suppose you want to use a full third-order polynomial model to fit these data. Which of the following is/are true?

a) The polynomials model has 10 parameters to learn
b) The polynomial learning systemis an under-determined one
c) The learning of the polynomial model has infinite number of solutions
d) The input matrix $\mathbf{X}$ has linearly dependent samples
e) None of the above

Answer: a, b, c, d

Q8

Question 8

MCQ: there could be more than one answer. Which of the following is/are true?

a) The polynomial model can be used to solve problems with nonlinear decision boundary.

b) The ridge regression cannot be applied to multi-target regression.

c) The solution for learning feature $\mathbf{X}$ with target $\mathbf{y}$ based on linear ridge regression can be written as $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$ for $\lambda > 0$. As λ increases, $\hat{\mathbf{w}}^T\hat{\mathbf{w}}$ decreases.

d) If there are four data samples with two input features each, the full second-order polynomial model is an over-determined system.

Answer: a, c

PY

Three balls are drawn from three urns sequentially, one ball from each urn. The first urn contains 1 blue and 7 red balls, the second urn contains 2 blue and 6 red balls, and the third urn contains 3 red and 5 green balls. Find the probability that 2 red balls are chosen. (3 marks)

Currently Selected: B

- **A** 270/1024
- **B** 270/512 ✓
- **C** 226/64
- **D** 226/512
- **E** None of the rest.

This question is related to the understanding of linear systems and partial derivatives. Which of the following statements below is correct?

Currently Selected: A

- **A** The system $\begin{bmatrix} 1 & 4 \\ 2 & 7 \\ -3 & 11 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -2.5 \\ 4 \end{bmatrix}$ has no *exact* solution but an approximated solution is available using the left inverse. ✓
- **B** ~~If $\mathbf{f}(\mathbf{x})$ is a vector-valued function of *size* p x1 and $\mathbf{x}$ is an m x1 vector, then differentiation of $\mathbf{f}(\mathbf{x})$ with respect to $\mathbf{x}$ is an m x p matrix.~~
- **C** ~~A linear function needs to satisfy the properties of homogeneity only.~~
- **D** ~~In over-determined linear systems, the number of parameters is greater than the number of unknown equations.~~
- **E** None of the other options.

Which of the following task is likely to be achieved via supervised learning?

Currently Selected: A

- **A** Using historical data for weather forecast. ✓
- **B** Grouping together users with similar viewing patterns in order to recommend similar content.
- **C** Grouping a number of oranges by their size.
- **D** None of the rest.

Causality is a deterministic relationship; suppose we know A and B have causal relation, if A occurs, B is for sure to take place.

Currently Selected: A

- **A** True ✓
- **B** False

- False
- [Causality is a Statistical Relationship]

A set of linear equations is written as $\mathbf{w}^T\mathbf{X} = \mathbf{y}^T$ where $\mathbf{X} \in \mathcal{R}^{3\times 2}$ and $\mathbf{y} \in \mathcal{R}^{2\times 1}$. How many simultaneous equations are there in this set of equations?

Currently Selected: B

- **A** 1
- **B** 2 ✓
- **C** 3
- **D** 5
- **E** 4

A discrete random variable takes a finite number of values, while a continuous random variable can only take infinite number of values. (2 marks)

Currently Selected: A

- **A** True ✓
- **B** False

- False: Discrete can be infinite also

This question is related to the understanding of modelling assumptions. f(x) = 5x - 3 is a linear function. (2 marks)

Currently Selected: B

- **A** True
- **B** False ✓

A machine learning algorithm takes the temperature as one of its input features. The temperature is measured in Celsius. Please select the correct option.

Currently Selected: E

- **A** The temperature in Celsius is considered as interval data.
- **B** We can calculate the mean and standard deviation of temperature.
- **C** The temperature in Celsius is considered as ratio data.
- **D** None of the rest.
- **E** (a), and (b) ✓
- **F** (a), and (c)

A machine learning algorithm takes the letter grade of students as one of its input features. The letter grade can take any element from {A+, A, A−, B+, B, B−, C+, C, D+, D, F}, subject to some distribution curving. For example, A+ corresponds to 95%, A corresponds to 85%, and A- corresponds to 80%. Which of the following statements is/are true?

Currently Selected: F

A The letter grade is an example of nominal variable.

B The letter grade is an example of ordinal variable.

C The letter grade is an example of interval variable.

D The letter grade is an example of discrete variable.

E (a), and (c)

F (b), and (d) ✓

G (b), and (c)

A person draws 2 cards from a deck of 52 cards, one after another without replacing the previous card back. What is the probability of drawing two Queens in a row?

Currently Selected: B

A 4/52

B 1/221 ✓

C 3/51

D 2/52

One key step in Data Cleaning is to check the missing features of data samples. When we have insufficient number of training samples in our dataset, we may consider removing the examples with missing features.

Currently Selected: B

A True

B False ✓

Suppose the random variable X has a probability mass function (pmf) given in the table below.

X	1	2	3	4	5
Pr[X]	0.1	(BLANK1)	0.2	0.4	(BLANK2)

We also know that the expected value of X is 3.5.

1) What is the probability of Pr[X=5]? 1 0.2 (2 Marks)

2) What is the probability of Pr[X<=2]? 2 0.2 (2 Marks)

Answers 1 - 2

1. 0.2

2. 0.2

1. Range - Min:0.19999 Max:0.20001
2. Range - Min:0.19999 Max:0.20001

We have a collection of 1,000 images from three classes: cat, bird, and dog. With these images, we would like to train an image classifier that categorizes an input image into one of the three classes.

To ensure the 1,000 images are of good quality for training the classifier, we ask a well-trained human inspector to go through all the images, to label the images and remove noisy ones. Eventually, we revmoved 200 images of low quality suggested by the inspector, and use the remaining 800 images to train the classifier; the 800 images comprise 200 cat images, 300 bird images, and 300 dog images.

Please select the correct option.

Currently Selected: A

A The human inspection process can be considered as a data cleaning step. ✓

B ~~If we are to use one-hot encoding for the labels of the three classes, we can set~~
~~Cat = [1 1 1]~~
~~Dog = [0 1 0]~~
~~Bird = [0 0 0].~~

C ~~The image classification conducted here is an unsuperivsed-learning task.~~

D If we keep the 200 noisy images (suggested by the human inspector), we will end up having more training images and hence a better-performed image classifier.

E (a) and (b)

F (a), (b), and (d)

G None of others is correct.

You are given a collection of 5 training data points of two features (x_1, x_2) and their target output (y) which are packed as follows:

Feature matrix: $\mathbf{X} = \begin{bmatrix} 1 & 2 \\ 0 & 6 \\ 1 & 0 \\ 0 & 5 \\ 1 & 7 \end{bmatrix}$, Target output: $\mathbf{y} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$.

Predict the output (up to 4 decimal places) of $(x_1, x_2) = (1, 3)$ using the linear regression model. (4 marks)

1) What is the mean of squared error of the estimated model? (1) 1.3887 (up to 4 decimal places, 2 mark)

2) The prediction for y is (2) 2.6603 (up to 4 decimal places, 2 marks).

Answers 1 - 2

1. 1.3887

2. 2.6603

1. Range - Min:1.3886 Max:1.3888
2. Range - Min:2.9999 Max:3.0001

- type in wrongly lmao, get 3

```
X = np.array([[1, 2], [0, 6], [1, 0], [0, 5], [1, 7]])
Y = np.array([[1], [2], [3], [4], [5]])

w = helper.linearRegressionWithBias(X, Y, printResult=True, printFeature=None)

xtest = np.array([[1, 3]])
xtest = helper.paddingOfOnes(xtest)
ytest = helper.testData(xtest, w, True)
```

```
rank(X) =  3
W =
 [[1.13207547]
 [0.8490566 ]
 [0.33962264]]
MSE =  1.388679245283019
X @ W = Y =
 [[3.]]
```

The values of feature x and their corresponding values of target y are shown in the table below.

x	3	4	5	6	7
y	5	4	3	2	1

Find the least square regression line y = a *x* + b and then estimate the value of y when *x* = 8.

Currently Selected: A

A y = 0 ✓

B y = −1

C y = +1

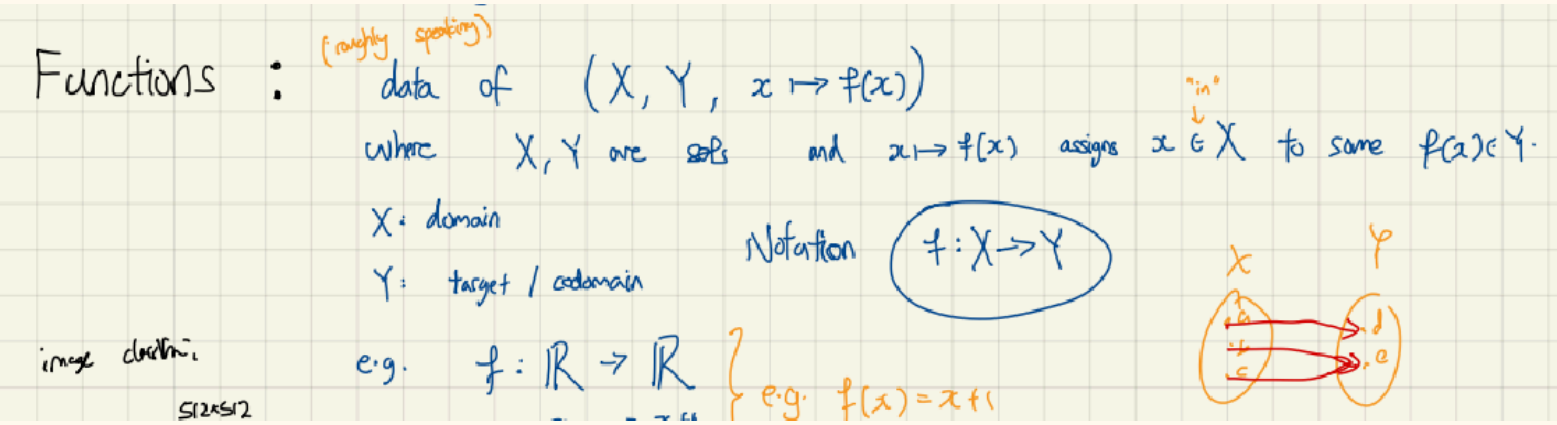
D y = 8

E None of the above

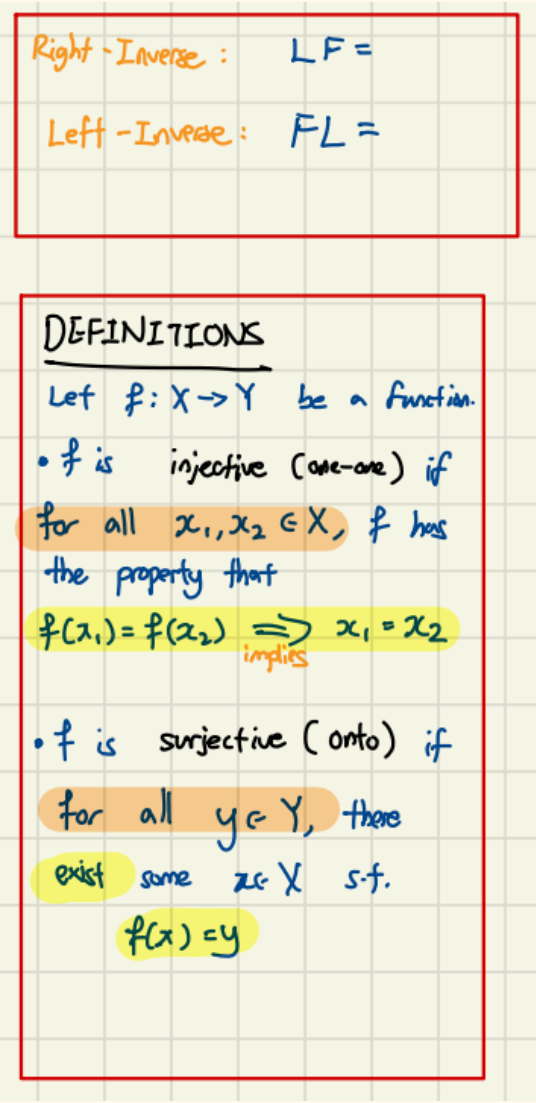
```
X = np.array([[3], [4], [5], [6], [7]])
Y = np.array([[5], [4], [3], [2], [1]])

w = helper.linearRegressionWithBias(X, Y, printResult=True, printFeature=None)

xtest = np.array([[8]])
xtest = helper.paddingOfOnes(xtest)
ytest = helper.testData(xtest, w, True)
```

```
rank(X) =  2
W =
 [[ 8.]
 [-1.]]
MSE =  4.733165431326071e-30
X @ W = Y =
 [[4.4408921e-15]]
```

Functions



- function is defined with three stuff
  - input space X
  - output space Y
  - and the transformation



- Injective means the function
  - $x_1 != x_2 \Rightarrow f(x_1) != f(x_2)$, if input different, output must be different
  - $f(x_1) = f(x_2) \Rightarrow x_1 = x_2$, if output same, input must be same

- Contrapositive example:
  - if rain ⇒ cloud
  - not contrapositive, converse: no rain ⇒ no cloud
    - this is not the same statement, no rain could be cloud or no cloud
    - if apply contrapositive to (no rain ⇒ no cloud): cloud ⇒ rain which is confirm not the same statement
  - contrapositive: no cloud ⇒ no rain
    - this is the same statement, if rain must have cloud, if no cloud must be no rain
  - contrapositive is done by negation and flipping ⇒
- for linear function:
  - injective ⇔ (f(x) = 0 ⇒ x = 0)
    - showing ⇐
      - assume (f(x) = 0 ⇒ x = 0) is true
      - want to show: f(x1) = f(x2) ⇒ x1 = x2
        - assume f(x1) = f(x2)
        - f(x1)- f(x2) = 0
        - by linearity of f, f(x1-x2) = 0
        - by hypothesis, x1-x2 = 0
        - x1 = x2
      - have shown f(x1) = f(x2) ⇒ x1 = x2
    - showing ⇒
      - assume injective: $f(x_1) = f(x_2) \Rightarrow x_1 = x_2$
      - want to show: f(x) = 0 ⇒ x = 0
        - assume f(x) = 0
        -
      - have shown f(x) = 0 ⇒ x = 0
  - Example: f(x) = mx for m != 0
    - $f(x_1) = f(x_2)$
    - $mx_1 = mx_2$
    - $x_1 = x_2$
    - ⇒ injective

- Surjective means the function can output all y in Y as define in the set
  - surjectivity of a function depends on the Y that is defined
- Example: $f(x) = x^2$, X = R, Y = R
  - f is not injective as x^2 maps many to one
  - f is not surjective as f cannot map to negative numbers
  - f does not have inverse, nor right nor left inverse
- Example: $f(x) = x^2$, X = R, Y = non negative R
  - f is not injective as x^2 maps many to one
  - f is is surjective as f can map to all values in Y
  - f has a right inverse (means apply right inverse to input first)
    - sqrt() then square
  -

A2 Dump

https://stackoverflow.com/questions/50132322/how-does-multiple-target-ridge-regression-work-in-scikit-learn

Admin

- Introduction and Preliminaries (Xinchao)
  - Introduction
  - Data Engineering
  - Introduction to Probability and Statistics
- Fundamental Machine Learning Algorithms I (Yueming)
  - Systems of linear equations
  - Least squares, Linear regression
  - Ridge regression, Polynomial regression
- Fundamental Machine Learning Algorithms II (Yueming)
  - Over-fitting, bias/variance trade-off
  - Optimization, Gradient descent
  - Decision Trees, Random Forest
- Performance and More Algorithms (Xinchao)
  - Performance Issues
  - K-means Clustering
  - Neural Networks

- Schedule
  - **12 Weeks Lectures**, starting from Week 1
  - **12 Weeks Tutorials**, starting from Week 2
  - **2 Programming Tutorials** (optional and highly recommended)
    - Week 1 – 2, Friday
    - Right after the lecture (i.e., 2 to 3 PM)
  - **1 Mid-term Quiz** (using ExamSoft)
    - Tentatively held offline on 9 March 2024 (Recess Week)
    - Content up to Week 5 (inclusive)
  - **1 Briefing Session on ExamSoft**
    - Tentative on Week 3, exact time to be confirmed with CIT Staff
  - **1 Final Exam** (using ExamSoft)
    - Held on 3-May-2024
  - **3 Assignments**
    - Assignment 1: released on Week 4, due on Week 6 (tentatively)
    - Assignment 2: released on Week 6, due on Week 9 (tentatively)
    - Assignment 3: released on Week 9, due on Week 13 (tentatively)

- midterm is 16 march

- 3 Assignments (36%) + Tutorial Attendance (4%)
- 1 Mid-term (30%)
- 1 Final Exam (30%)

- Held online:
  - Lectures

- Held offline (in classrooms):
  - Tutorials

- Videos of lectures are made available after lectures.

- midterm is open book