EE2213 Introduction to AI

Lecture 7: Introduction to Machine Learning

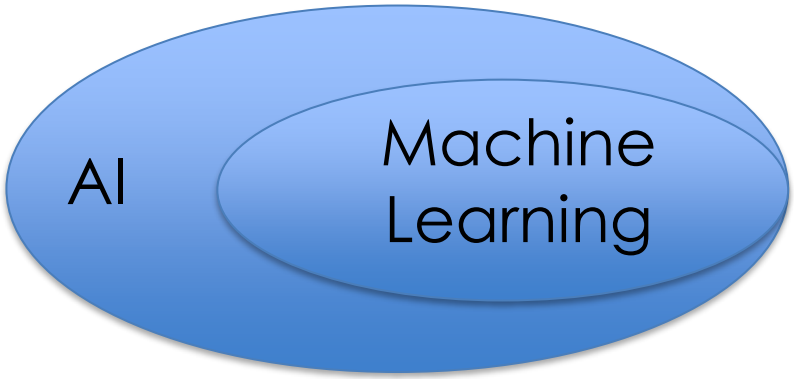
Dr. WANG Si
si.wang@nus.edu.sg
Electrical and Computer Engineering Department
National University of Singapore

OVERVIEW OF COURSE CONTENTS

- **Introduction (Shaojing)**
  - What is AI
  - Applications of AI
  - AI agent
- **Search (Shaojing)**
  - Uninformed search algorithms: breadth-first, depth-first, uniform-cost(Dijkstra's algorithm)
  - Informed search algorithms: greedy best-first, A*
  - Applications
- **Optimisation (Shaojing)**
  - Linear programming
  - Convex problems
  - Applications
- **Machine learning (Wang Si) (Weeks 6-9)**
  - Supervised and unsupervised learning: regression, classification, clustering
  - Neural networks and deep learning
  - Applications
- **Knowledge representation (Wang Si) (Weeks 10-11)**
  - Knowledge Representation and Reasoning
  - Propositional Logic
  - Applications
- **Ethical considerations (Shaojing)**
  - Bias in AI
  - Privacy concerns
  - Societal impact

(Week 10, Monday)

No Class on 20 Oct.! A recording will be uploaded on Canvas.



What is Machine Learning?

AI Techniques that give computers **the ability to learn** without being explicitly programmed to do so

- Arthur Samuel

What is Machine Learning?

A computer program is said to learn if its performance at task **T**, as measured by **P**, improves with experience **E**.

- Tom Mitchell

e.g., Google Maps finding the fastest route based on current traffic?

Google Maps predicting traffic jams based on historical traffic patterns?

millions of past trips from other users

Experience (**E**)

traffic jam prediction

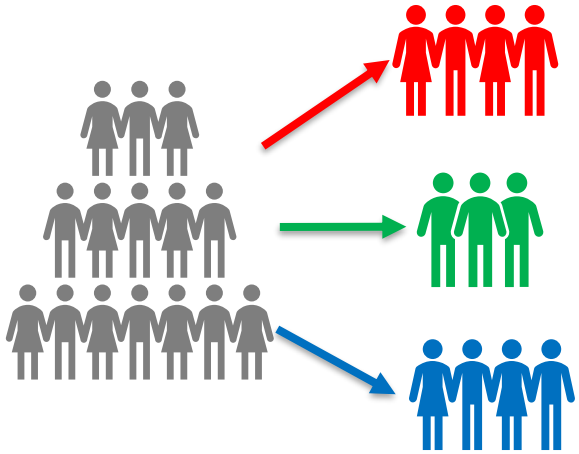
Task (**T**)

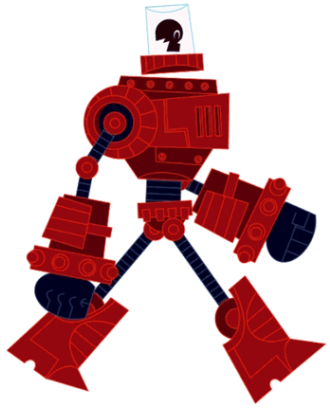
Prediction accuracy

Performance Measure (**P**)

What is Machine Learning?

Examples: Task (**T**) Experience (**E**) Performance (**P**)





T: Spam Email Detection

E: Email Data

P: Detection Accuracy

T: Customer Segmentation

E: Purchase data

P: Silhouette Scores

T: Robot Walking

E: Past records of Walking

P: Walking time

performance measure

Types of Machine Learning

-Supervised Learning

-Unsupervised Learning

-Reinforcement Learning

regression and classification tasks
(training samples with desired output.
Train a model to input -> function -> label)

clustering tasks (use samples with no expected labels, and cluster to identify underlying patterns

seq of states, actions and delayed rewards

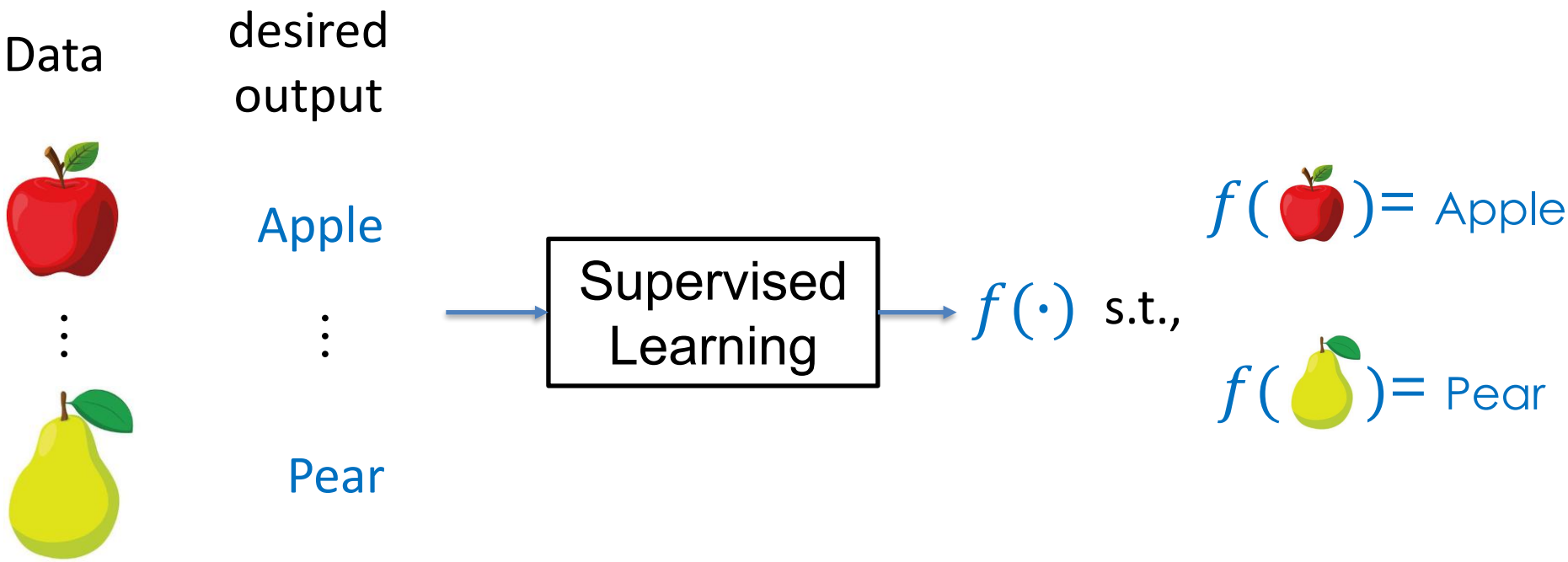
Types of Machine Learning

➢ Supervised Learning

by the output of the training data fed in.

Input: data with desired output

Output: A function(i.e., model) that maps data to the desired output

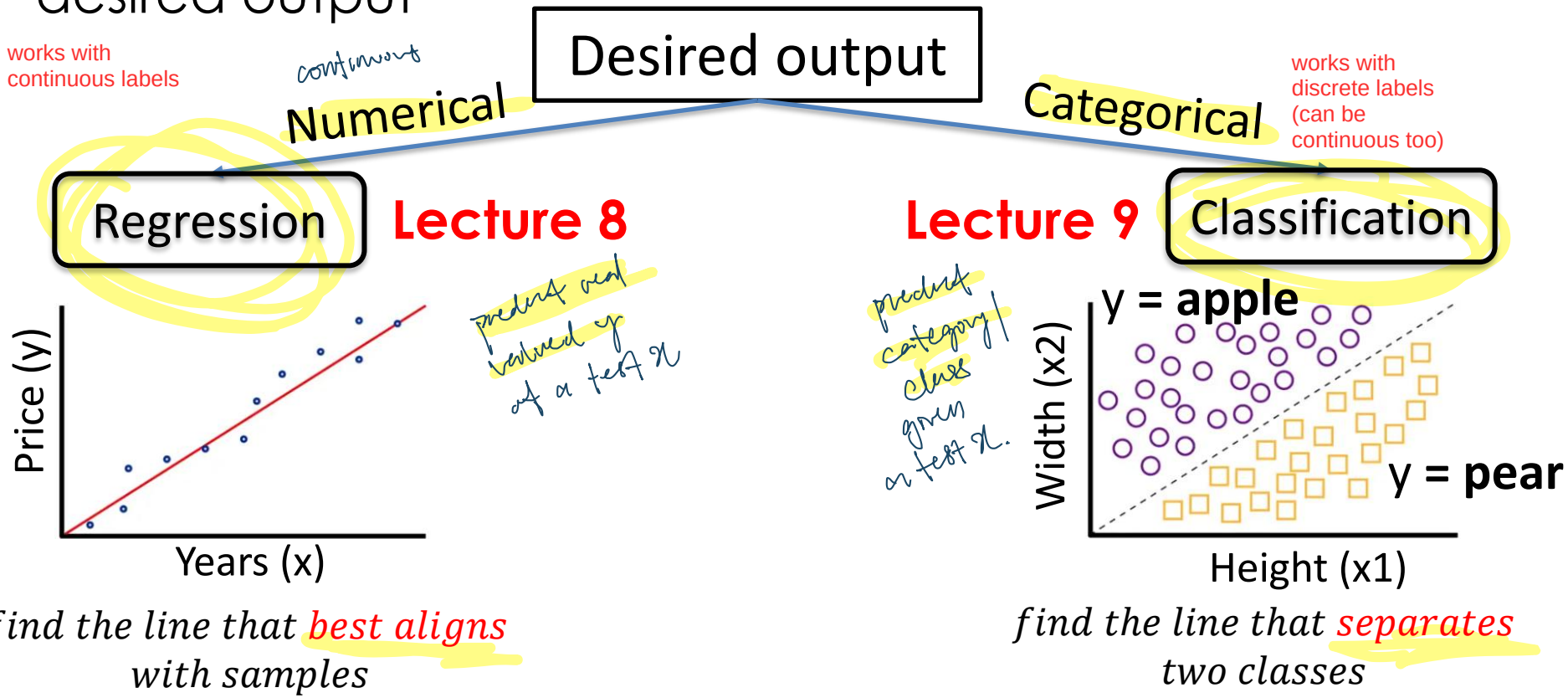


Types of Machine Learning

➢ Supervised Learning

Input: data with desired output

Output: A function(i.e., model) that maps data to the desired output



Types of Machine Learning

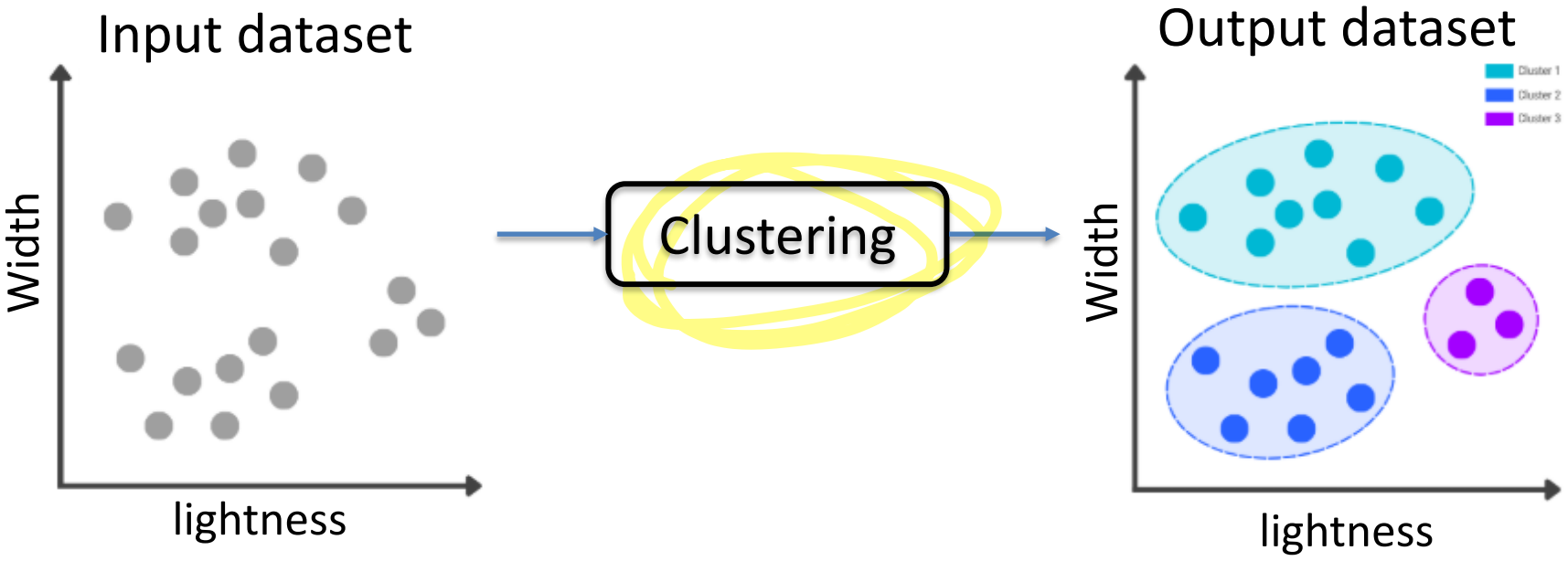
➢ Unsupervised Learning

Input: data (No desired output)

Output: Underlying patterns in data

No label given
No expected output given

Clustering **Lecture 10**



Types of Machine Learning

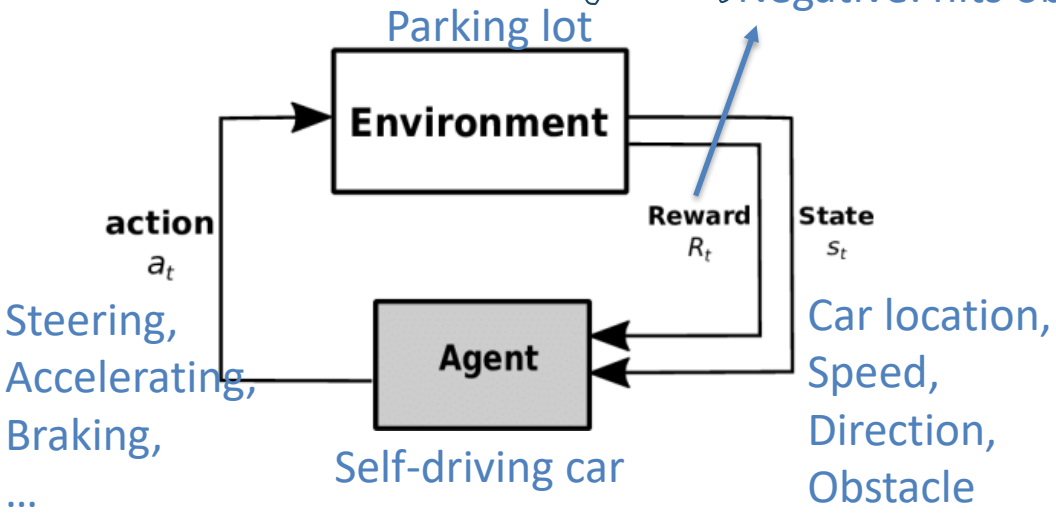
➢ Reinforcement Learning

Input: Sequence of States, Actions, and Delayed Rewards

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

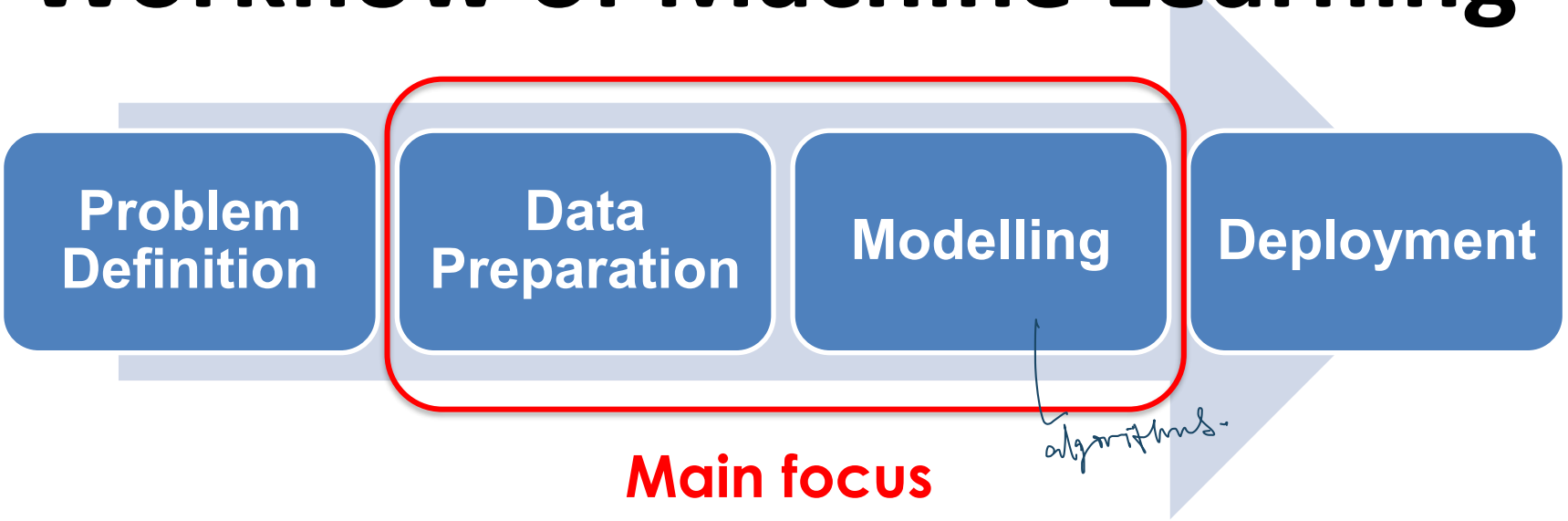
Positive: closer to the correct spot

Negative: hits obstacles or parks badly



Self-driving car learn to park

Workflow of Machine Learning



Workflow of Machine Learning

Problem Definition | Data Preparation | Modelling | Deployment

Example

1. Clarify the objective

What problem are you trying to solve? — Predict House Prices

2. Determine if ML is appropriate

Can this problem be learned from data? — Yes

3. Specify inputs and outputs

What input features will the model use? — Size, location, no. rooms, etc.

What is the model trying to predict? — Sale price in SGD

4. Choose the task type

Classification? Regression? Clustering?... — Regression

5. Decide on performance measure

Accuracy? Mean Squared Error?... — Mean Squared Error

Workflow of Machine Learning

NUS National University of Singapore

Problem Definition | **Data Preparation** | Modelling | Deployment

1. Data Collection

Gather raw data from sensors, databases, files, etc.

e.g., Predict House Prices

No.	Size	No. rooms	Location	Sold Price
1				
…				
100				

Features

Desired output

2. Exploratory Data Analysis

Flag any anomalies and inconsistencies.

No.	Size	No. rooms	Location	Sold Price
1	112	2	“Pioneer”	600000
2		4	“Clementi"	801256
3	100	3	4	“650K”
4	116	3	“City Hall”	2300000
…				
100	112	2	“Pioneer”	600000

Missing value

Inconsistent data values

Outlier

Duplicate data

Workflow of Machine Learning

Problem Definition | **Data Preparation** | Modelling | Deployment

3. Data Preprocessing

- Address the problems flagged in the previous step.

  e.g., handling missing value through **dropping** or **imputation**

If the number of samples is large.

Option 1: take the average value of the corresponding feature.

Option 2: take a value outside the normal range.

Missing value

No.	Size	No. rooms	Location	Sold Price
1	112	2	"Pioneer"	600000
2		4	"Clementi"	801256
3	100	3	4	"650K"
4	116	3	"City Hall"	1300000
…				
100	112	2	"Pioneer"	600000

Which of the following preprocessing steps would be the most suitable to address
missing values in this dataset before applying linear regression?
A. Remove all samples with missing values (CORRECT)
B. Fill the missing values with zeros
C. Remove all attributes (columns) with missing values
D. Ignore the missing values as they will not impact model performance

Workflow of Machine Learning

Problem Definition | **Data Preparation** | Modelling | Deployment

3. Data Preprocessing

- Format the data
  - ❖ Convert non-numerical data into numerical form.

e.g., Use postal code for location

Convert categories into a binary vector through **one-hot encoding.**

Categories: ['Cat', 'Dog', 'Bird'] → **one-hot encoding**

Cat: [1,0,0]
Dog: [0,1,0]
Bird: [0,0,1]

- ❖ **Normalization (Ensure Features Contribute Fairly)**

Across Samples for Each Feature

Use only training data Statistics!

Min-max normalization: $x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$

linear regression is not sensitive to scale of input features
but it is important for iterative algos like gradient descent/MLP

Z-score standardization: $x_{scaled} = \frac{x - \mu}{\sigma}$ → Mean, Standard deviation

No.	Feature 1	Feature 2
1	3	-1
2	2	0
3	4	1

Workflow of Machine Learning

Problem Definition | **Data Preparation** | Modelling | Deployment

4. Feature Selection or Feature Extraction

Manipulating existing features to obtain **more relevant** features that improves performance

Feature Selection	Feature Extraction
Chooses from **existing features**	Creates **new features** from original ones
e.g., drop "Favorite color"	e.g., combine "height" and "weight" into "BMI"

Example task: Predict health risk → categorical → use classification (supervised)

	Height (m)	Weight (kg)	Age	Gender	Favorite color	Risk
1	1.75	59	33	Male	Red	High
…	…	…	…	…	…	…
100	1.65	50	31	Female	Blue	Medium

Workflow of Machine Learning

Problem Definition | **Data Preparation** | Modelling | Deployment

4. Feature Selection and Feature Extraction

Manipulating existing features to obtain **more relevant** features that improves performance

-Feature Selection with **Pearson's correlation coefficient (r)**

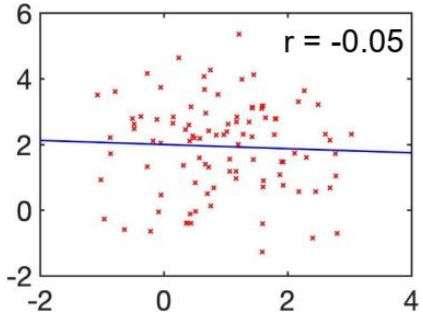
(Measures linear relationship between two numerical variables)

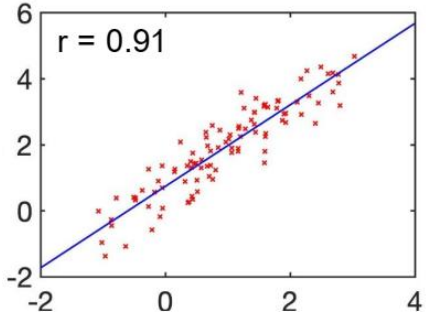
Given N pairs of data points $\{(a_1, b_1),..., (a_N, b_N)\}$

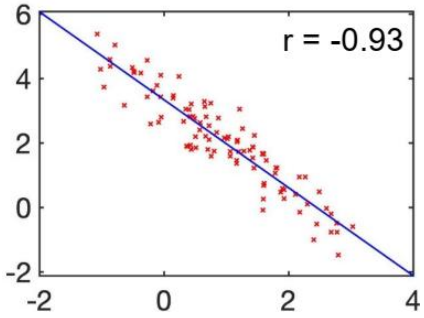
Use only training data!

$$r = \frac{\frac{1}{N}\sum_{i=1}^{N}(a_i - \overline{a})(b_i - \overline{b})}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(a_i - \overline{a})^2}\sqrt{\frac{1}{N}\sum_{i=1}^{N}(b_i - \overline{b})^2}}$$

where $\overline{a} = \frac{1}{N}\sum_{i=1}^{N} a_i$, $\overline{b} = \frac{1}{N}\sum_{i=1}^{N} b_i$







Workflow of Machine Learning

Problem Definition | **Data Preparation** | Modelling | Deployment

4. Feature Selection and Feature Extraction

Manipulating existing features to obtain **more relevant** features that improves performance

-Feature Selection with **Pearson's correlation coefficient (r)**

(Measures linear relationship between two numerical variables)

Way 1:
- Compute r between each input feature and the target output in the **training set**.
- Select features with high absolute r.

→ more predictive, for supervised learning as regression

Way 2:
- Compute r between every pair of input features in the **training set**
- Remove features with high absolute r.

→ can view as redundant feature instead to reduce feature space.

Workflow of Machine Learning

Problem Definition | **Data Preparation** | Modelling | Deployment

5. Split data into training, validation and test sets

Training	Validation	Test
For training the ML models (learn internal parameters)	For choosing the hyperparameter of the model	For testing the "real" performance and generalization

extend DA by rules

to see which algo / model is the best

to choose model before testing, look at validation accuracy, then training accuracy, then one with FEWER number of parameters

Seen/visible data

Unseen/hidden data

No.	Size	No. rooms	Location	Sold Price
1				
…		Training Set		
80				

No.	Size	No. rooms	Location	Sold Price
81				
…		Validation Set		
90				

No.	Size	No. rooms	Location	Sold Price
91				
…		Test Set		
100				

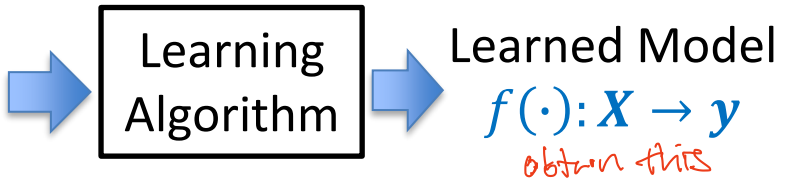
Workflow of Machine Learning

Problem Definition | Data Preparation | **Modelling** | Deployment

1. Train model on the training set

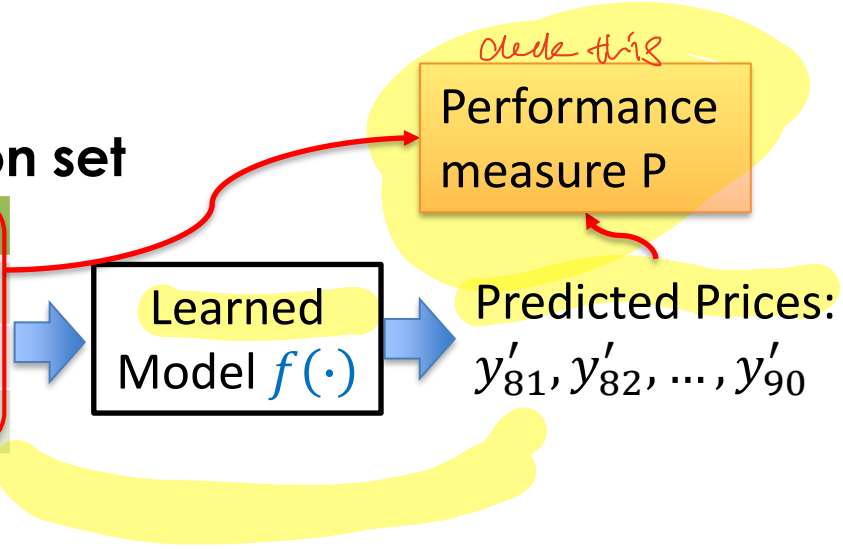
	No.	Size	No. rooms	Location	Sold Price
$\boldsymbol{x}_1$	1	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	y_1
	…				
$\boldsymbol{x}_{80}$	80	$x_{80,1}$	$x_{80,2}$	$x_{80,3}$	y_{80}

$$\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_{80}, y_{80})\}$$



2. Evaluate the model on the validation set

	No.	Size	No. rooms	Location	Sold Price
$\boldsymbol{x}_{81}$	81	$x_{81,1}$	$x_{81,2}$	$x_{81,3}$	y_{81}
	…				
$\boldsymbol{x}_{90}$	90	$x_{90,1}$	$x_{90,2}$	$x_{90,3}$	y_{90}



Workflow of Machine Learning

NUS National University of Singapore

Problem Definition | Data Preparation | **Modelling** | Deployment

3. Hyperparameter Tuning

Change the hyperparameters and repeat steps 1&2 to find the best hyperparameters that have the best validation performance.

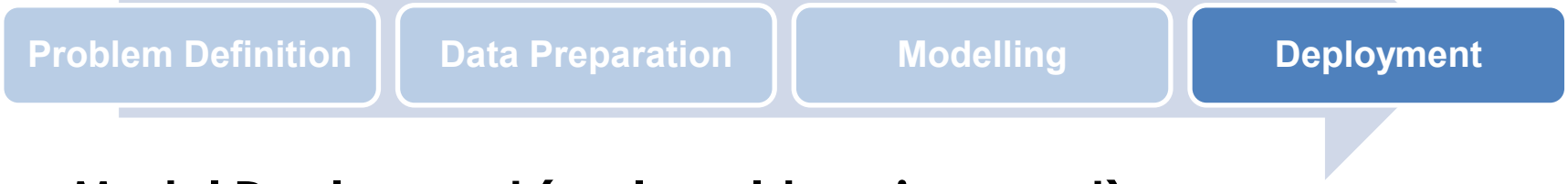
	Hyperparameters	Parameters
Definition	External settings chosen before training	Internal variables learned from the data
Learned during the training?	No	Yes
Set by user?	Yes	No
Examples	-Learning rate -Training steps	-Regression Coefficents in Linear Regression

e.g. in gradient descent

4. Test model on the test set

Unbiased evaluation of the final model

Workflow of Machine Learning



- **Model Deployment (real-world environment)**

  Web API : Model exposed as an endpoint that APPs can call

  Cloud: AWS, GCP, or Azure provide managed ML serving platforms.

  Edge: Model runs locally on devices like smartphones, IoT devices, etc.

Summary by Quick Quiz

NUS National University of Singapore

✓ Three Components in ML Definition

Task, experience, performance measure

✓ Three Types of ML

Supervised (regression for actual y/numerical, classification for label/class/categorical)
Unsurpervised (no expected output)
Reinforcement learning (from seq of states and actions and delayed rewards/positive/negative feedback, to determine best actions)

✓ Three ways of Handling Missing Value

Dropping, average value, out of normal range value

✓ Data Formatting

Categorical data? Different Scales for Features?

One hot encoding

Normalisation (min-max, z score)

✓ One Criterion for Feature Selection

pearson correlation coefficient r
(only high r or remove high r)

✓ Dataset Partition

Training set to train the model, validation set to choose best hyperparameter for model,, test set to evaluate model performance

Review of Fundamentals

Review of Fundamentals

❖ Scalars, Vectors, Matrices

- A **scalar** is a simple numerical value (e.g., 10, -4.2,).

  Denoted by an *italic* letter (e.g., x, y)

- A **vector** is an ordered list of scalar values.

  Denoted by a **bold** letter (e.g., $\mathbf{x}$, $\mathbf{y}$)

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

- A **matrix** is a rectangular array of numbers arranged in rows and columns

  Denoted by a **bold** CAPITAL letter (e.g., $\mathbf{X}$, $\mathbf{Y}$)

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.5 \\ 2 & 9 & -3.2 \end{bmatrix}$$

Review of Fundamentals

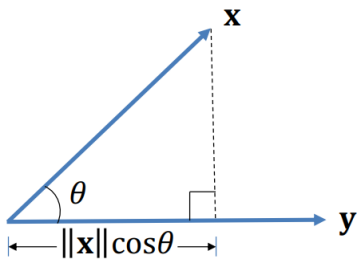
❖ Operations on Vectors and Matrices

$$\mathbf{x}+\mathbf{y}=\begin{bmatrix}x_1\\x_2\end{bmatrix}+\begin{bmatrix}y_1\\y_2\end{bmatrix}=\begin{bmatrix}x_1+y_1\\x_2+y_2\end{bmatrix}$$

$$\mathbf{x}-\mathbf{y}=\begin{bmatrix}x_1\\x_2\end{bmatrix}-\begin{bmatrix}y_1\\y_2\end{bmatrix}=\begin{bmatrix}x_1-y_1\\x_2-y_2\end{bmatrix}$$

$$a\,\mathbf{x}=a\begin{bmatrix}x_1\\x_2\end{bmatrix}=\begin{bmatrix}ax_1\\ax_2\end{bmatrix}$$

$$\mathbf{x}=\begin{bmatrix}x_1\\x_2\end{bmatrix},\ \mathbf{x}^T=[x_1\quad x_2]$$



$$\mathbf{x}\cdot\mathbf{y}=\mathbf{x}^T\mathbf{y}=\|\mathbf{x}\|\,\|\mathbf{y}\|\cos\theta$$
$$=[x_1\quad x_2]\begin{bmatrix}y_1\\y_2\end{bmatrix}=x_1\,y_1+x_2y_2$$

$Xw=y$

if $(w^TX)^T=(y^T)^T \Rightarrow X^Tw=y$

$$\mathbf{W}\mathbf{x}=\begin{bmatrix}w_{1,1}&w_{1,2}&w_{1,3}\\w_{2,1}&w_{2,2}&w_{2,3}\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}$$
$$=\begin{bmatrix}w_{1,1}x_1+w_{1,2}x_2+w_{1,3}x_3\\w_{2,1}x_1+w_{2,2}x_2+w_{2,3}x_3\end{bmatrix}$$

$$\mathbf{x}^T\mathbf{W}=[x_1\quad x_2]\begin{bmatrix}w_{1,1}&w_{1,2}&w_{1,3}\\w_{2,1}&w_{2,2}&w_{2,3}\end{bmatrix}$$
$$=[(x_1w_{1,1}+x_2w_{2,1})\quad(x_1w_{1,2}+x_2w_{2,2})\quad(x_1w_{1,3}+x_2w_{2,3})]$$

$\boldsymbol{A}_{3\times3}\boldsymbol{B}_{3\times3}$ (same) (output)

$$=\begin{bmatrix}a_{11}&a_{12}&a_{13}\\a_{21}&a_{22}&a_{23}\\a_{31}&a_{32}&a_{33}\end{bmatrix}\times\begin{bmatrix}b_{11}&b_{12}&b_{13}\\b_{21}&b_{22}&b_{23}\\b_{31}&b_{32}&b_{33}\end{bmatrix}=\begin{bmatrix}a_{11}\times b_{11}+a_{12}\times b_{21}+a_{13}\times b_{31}&a_{11}\times b_{12}+a_{12}\times b_{22}+a_{13}\times b_{32}&a_{11}\times b_{13}+a_{12}\times b_{23}+a_{13}\times b_{33}\\a_{21}\times b_{11}+a_{22}\times b_{21}+a_{23}\times b_{31}&a_{21}\times b_{12}+a_{22}\times b_{22}+a_{23}\times b_{32}&a_{21}\times b_{13}+a_{22}\times b_{23}+a_{23}\times b_{33}\\a_{31}\times b_{11}+a_{32}\times b_{21}+a_{33}\times b_{31}&a_{31}\times b_{12}+a_{32}\times b_{22}+a_{33}\times b_{32}&a_{31}\times b_{13}+a_{32}\times b_{23}+a_{33}\times b_{33}\end{bmatrix}$$

Review of Fundamentals

❖ Operations on Vectors and Matrices

or matlab

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

python

```
import numpy as np

# Calculate inverse
A_inv = np.linalg.inv(A)
```

- det(**A**): determinant of **A**

$\det(\mathbf{A}) = \sum_{j=1}^{k} a_{ij}\mathbf{C}_{ij}$, where cofactor $\mathbf{C}_{ij} = (-1)^{i+j}\mathbf{M}_{ij}$

$\mathbf{M}_{ij}$: minor of a_{ij} is the determinant obtained by deleting the row and column in which a_{ij} lies.

cofactor expansion along any row/col gives the same result. ⇒ most convenient: most zeros.

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

$\mathbf{M}_{11} = \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} = a_{22} \times a_{33} - a_{23} \times a_{32}$

$\mathbf{C}_{11} = (-1)^{1+1}\mathbf{M}_{11}$

$\det(\mathbf{A}) = \sum_{j=1}^{3} a_{ij}\mathbf{C}_{ij} = a_{i1}\mathbf{C}_{i1} + a_{i2}\mathbf{C}_{i2} + a_{i3}\mathbf{C}_{i3}$

Review of Fundamentals

❖ Operations on Vectors and Matrices

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

- adj(**A**): adjugate/adjoint of **A**

  adj(**A**)=$\mathbf{C}^T$, where cofactor $\mathbf{C}_{ij} = (-1)^{i+j}\mathbf{M}_{ij}$

  $\mathbf{M}_{ij}$: minor of a_{ij} is the determinant obtained by deleting the row and column in which a_{ij} lies.

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

$\mathbf{M}_{11} = \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} = a_{22} \times a_{33} - a_{23} \times a_{32}$

$\mathbf{C}_{11} = (-1)^{1+1}\mathbf{M}_{11}$

$$\text{adj}(\mathbf{A})=\mathbf{C}^T = \begin{bmatrix} \mathbf{M}_{11} & -\mathbf{M}_{21} & \mathbf{M}_{31} \\ -\mathbf{M}_{12} & \mathbf{M}_{22} & \mathbf{M}_{32} \\ \mathbf{M}_{13} & -\mathbf{M}_{23} & \mathbf{M}_{33} \end{bmatrix}$$

Review of Fundamentals

❖ Operations on Vectors and Matrices

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

- $\det(\mathbf{A}) = \sum_{j=1}^{k} a_{ij}\mathbf{C}_{ij}$, where cofactor $\mathbf{C}_{ij} = (-1)^{i+j}\mathbf{M}_{ij}$
- $\text{adj}(\mathbf{A}) = \mathbf{C}^T$

$\mathbf{M}_{ij}$: minor of a_{ij} is the determinant obtained by deleting the row and column in which a_{ij} lies.

e.g., What is the cofactor matrix of $\mathbf{A} = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix}$ $\mathbf{C} = \begin{bmatrix} -3 & 6 & -3 \\ 6 & -12 & 6 \\ -3 & 6 & -3 \end{bmatrix}$

$$\mathbf{C}_{11} = (-1)^{1+1}\begin{vmatrix} 4 & 5 \\ 7 & 8 \end{vmatrix} = -3,\ \mathbf{C}_{12} = (-1)^{1+2}\begin{vmatrix} 3 & 5 \\ 6 & 8 \end{vmatrix} = 6,\ \mathbf{C}_{13} = (-1)^{1+3}\begin{vmatrix} 3 & 4 \\ 6 & 7 \end{vmatrix} = -3$$

$$\mathbf{C}_{21} = (-1)^{2+1}\begin{vmatrix} 1 & 2 \\ 7 & 8 \end{vmatrix} = 6,\ \mathbf{C}_{22} = (-1)^{2+2}\begin{vmatrix} 0 & 2 \\ 6 & 8 \end{vmatrix} = -12,\ \mathbf{C}_{23} = (-1)^{2+3}\begin{vmatrix} 0 & 1 \\ 6 & 7 \end{vmatrix} = 6,$$

$$\mathbf{C}_{31} = (-1)^{3+1}\begin{vmatrix} 1 & 2 \\ 4 & 5 \end{vmatrix} = -3,\ \mathbf{C}_{32} = (-1)^{3+2}\begin{vmatrix} 0 & 2 \\ 3 & 5 \end{vmatrix} = 6,\ \mathbf{C}_{33} = (-1)^{3+3}\begin{vmatrix} 0 & 1 \\ 3 & 4 \end{vmatrix} = -3,$$

Review of Fundamentals

❖ Operations on Vectors and Matrices

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

Matrix inverse $\mathbf{A}^{-1}$ exists if and only if the matrix $\mathbf{A}$ is **square and nonsingular** (i.e., **invertible)**

inv(A) in matlab.

In other words, there exists a square matrix $\mathbf{B}$, such that **AB=BA=I** (Identity matrix)

$$\mathbf{I}_{d\times d} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & 0 & \ddots & 0 & \vdots \\ 0 & \vdots & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

Review of Fundamentals

❖ Operations on Vectors and Matrices

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

Matrix inverse $\mathbf{A}^{-1}$ exists if and only if the matrix $\mathbf{A}$ is **square and nonsingular** (i.e., **invertible)**

2 methods to check invertibility

det(**A**)=0?

rank($\mathbf{A}_{d\times d}$)=d?

rank(**A**): maximal number of **linearly independent** columns/rows of **A**

$\beta_1\mathbf{a}_1 + \beta_2\mathbf{a}_2 + \cdots + \beta_m\mathbf{a}_m = 0$ only holds for $\beta_1 = \cdots = \beta_m = 0$

Compute rank(**A**): Step 1. Transform into row echelon form using elementary row operations

Step 2. The number of non-zero rows in the row echelon form.

https://stattrek.com/matrix-algebra/echelon-transform#MatrixA

Review of Fundamentals

❖ Operations on Vectors and Matrices

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

just matlab.

Matrix inverse $\mathbf{A}^{-1}$ exists if and only if the matrix $\mathbf{A}$ is **square and nonsingular** (i.e., **invertible)**

e.g., $\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$ $\begin{bmatrix} 1 & 2 \end{bmatrix}$ $\begin{bmatrix} 3 & 1 \\ 9 & 3 \end{bmatrix}$

$\begin{bmatrix} 1 & 3 \\ 0 & 9 \\ 3 & 4 \end{bmatrix}$ $\begin{bmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \\ 3 & 4 & 5 \end{bmatrix}$ $-1 \times R_2 + R_1 = R_3$

Review of Fundamentals

❖ Some properties on Matrix Operations

$$\mathbf{A}+\mathbf{B}=\mathbf{B}+\mathbf{A}$$

$$(\mathbf{A}+\mathbf{B})+\mathbf{C}=\mathbf{A}+(\mathbf{B}+\mathbf{C})$$

$$(\mathbf{AB})\mathbf{C}=\mathbf{A}(\mathbf{BC})$$

$$(\mathbf{A}+\mathbf{B})\mathbf{C}=\mathbf{AC}+\mathbf{BC}$$

$$\mathbf{I}_m\mathbf{A}_{m\times n}=\mathbf{A}=\mathbf{A}_{m\times n}\mathbf{I}_n$$

$$(\mathbf{A}^T)^T=\mathbf{A}$$

$$(\mathbf{A}+\mathbf{B})^T=\mathbf{A}^T+\mathbf{B}^T$$

$$(r\mathbf{A})^T=r\mathbf{A}^T$$

$$(\mathbf{AB})^T=\mathbf{B}^T\mathbf{A}^T$$

very important for $X\omega = y$.

$$\mathbf{A}\mathbf{A}^{-1}=\mathbf{A}^{-1}\mathbf{A}=\mathbf{I}$$

$$(r\mathbf{A})^{-1}=r^{-1}\mathbf{A}^{-1}$$

$$(\mathbf{AB})^{-1}=\mathbf{B}^{-1}\mathbf{A}^{-1}$$

$$\mathbf{I}^{-1}=\mathbf{I}$$

$$(\mathbf{A}^T)^{-1}=(\mathbf{A}^{-1})^T$$

$$(\mathbf{A}^{-1})^{-1}=\mathbf{A}$$

Review of Fundamentals

❖ Sets, Functions

-A **Set** is an unordered collection of unique elements

Denoted by a calligraphic capital letter (e.g., $\mathcal{X}, \mathcal{Y}$)

$x \in \mathcal{X}$: the element x belongs to the set $\mathcal{X}$

- ✓ Finite set: a fixed number of elements

  Denoted by accolades (e.g., $\{1, 3.5, 7, -1\}$)

- ✓ Infinite set: include all values in some interval

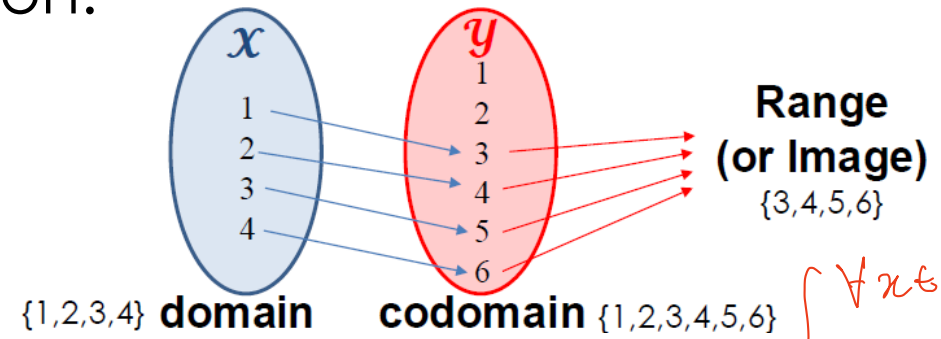
  Denoted by square brackets as [a, b] if including a and b (e.g., [1,2])

  Denoted by parentheses as (a, b) if does not include a and b (e.g., (1,2))

Review of Fundamentals

❖ Sets, Functions

-A **Function** is a relation that associates each element x of a set $\boldsymbol{x}$, the domain of the function, to a single element y of another set $\boldsymbol{y}$, the codomain of the function.



Value/output

$y = f(x)$

Argument/input

$\begin{cases} \forall x \in A \ \exists y \in B \ (x,y) \in f \\ \forall x \in A \ \forall y_1, y_2 \in B \ ((x,y_1) \in f \wedge (x,y_2) \in f \Rightarrow y_1 = y_2) \end{cases}$

every x is related at least once
at most once y.

✓ Scaler function: function returns a scalar (e.g., $f(x)$ *or* $f(\mathbf{x})$)

✓ Vector function: function returns a vector (e.g., $\mathbf{f}(x)$ *or* $\mathbf{f}(\mathbf{x})$)

Review of Fundamentals

❖ Operations on Functions

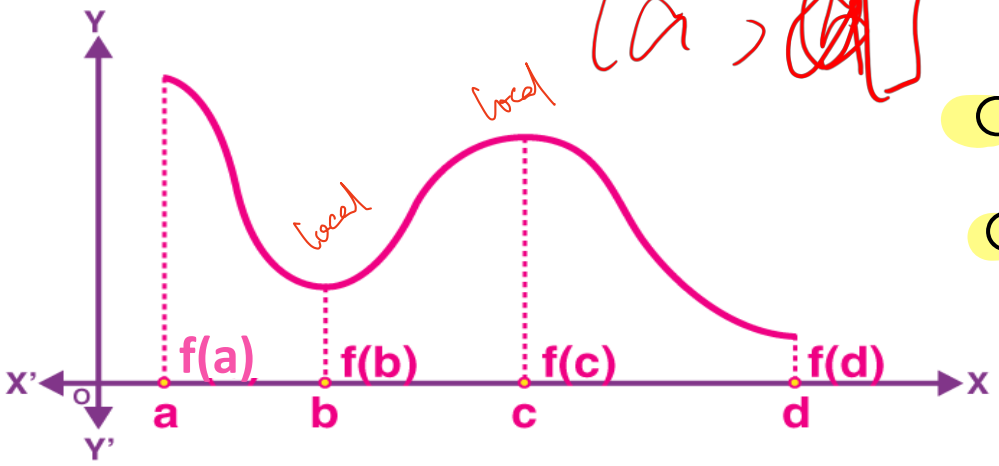
-Maximum and Minimum

$\max_x f(x)$: returns the highest value of $f(x)$ across its domain

$\operatorname{argmax}_x f(x)$: returns x that maximizes $f(x)$

$\min_x f(x)$: returns the lowest value of $f(x)$ across its domain

$\operatorname{argmin}_x f(x)$: returns x that minimizes $f(x)$



Global maximum = $\operatorname{argmax}_x f(x)$

Global minimum = $\operatorname{argmin}_x f(x)$

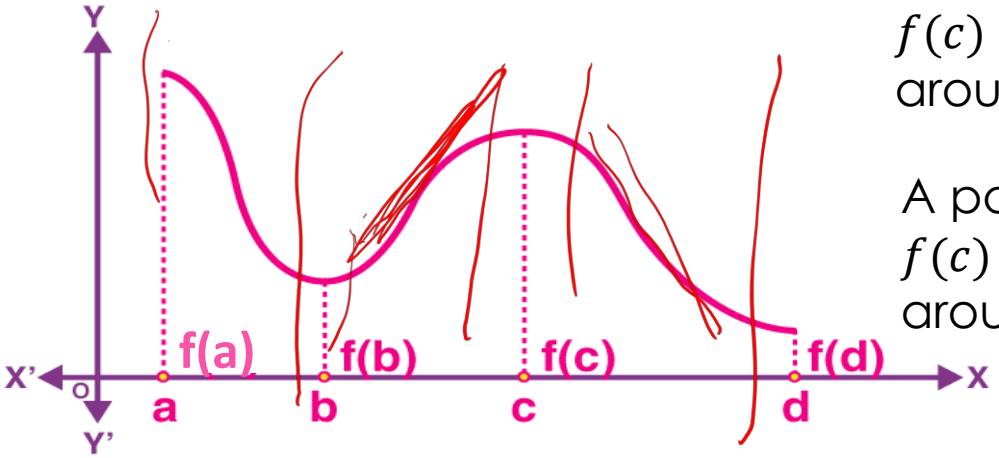
Review of Fundamentals

❖ Operations on Functions

-Derivative (single-input)

The derivative f' of a function f is a function that describes how fast f grows (or decreases)

- If f' is positive at some x, f grows at this x
- If f' is negative at some x, f decreases at this x
- If f' is zero at some x, f's slope at x is horizontal



A point x =c is a **local maximum** of $f(x)$ if $f(c) \geq f(x)$ for all x in some **open** interval around c

A point x =c is a **local minimum** of $f(x)$ if $f(c) \leq f(x)$ for all x in some **open** interval around c

Review of Fundamentals

❖ Operations on Functions

-Gradient (multiple-input scalar function)

Generalization of derivative for scalar functions with multiple inputs.

Denoted by $\nabla_{\mathbf{x}} f$

diff of a scalar function $f(\underline{x})$ w.r.t a vector $\underline{x}$

A vector of **partial derivatives**

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

A matrix of **partial derivatives**

$$\nabla_{\mathbf{X}} f = \frac{df(\mathbf{X})}{d\mathbf{X}} = \begin{bmatrix} \frac{\partial f}{\partial x_{1,1}} & \cdots & \frac{\partial f}{\partial x_{1,K}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial x_{d,1}} & \cdots & \frac{\partial f}{\partial x_{d,K}} \end{bmatrix}$$

same size as **X**

Differentiation of $f(\mathbf{x})$ w.r.t. $\mathbf{x}$

if $f(\underline{x})$ has d variables, $\underline{x}$ is a $d \times 1$ vector.
result $\nabla_{\underline{x}} f$ is a $d \times 1$ vector

e.g., $f(\mathbf{x}) = 2x_1 + 3x_2$, what is $\nabla_{\mathbf{x}} f$?

$$\begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

Review of Fundamentals

❖ Operations on Functions

-Jacobian (multiple-input vector function)

Generalization of derivative for vector functions with multiple inputs.

If $\underset{\sim}{f}(\underset{\sim}{X})$ is a vector function of size $h \times 1$ and $\underset{\sim}{X}$ is a $d \times 1$ vector, the Jacobian results in a $h \times d$ matrix

A matrix of **partial derivatives**

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

Differentiation of $\mathbf{f}(\mathbf{x})$ of size hx1 w.r.t. $\mathbf{x}$ of size dx1.

e.g., $\mathbf{f}(\mathbf{x}) = \begin{bmatrix} 2x_1 + 3x_2 \\ 5x_1x_2 \end{bmatrix}$, what is the Jacobian matrix $\frac{d\mathbf{f}(\mathbf{x})}{d\mathbf{x}}$?

$\underset{\sim}{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \Rightarrow 2 \times 1$

Size = 2x2

$\begin{bmatrix} 2 & 3 \\ 5x_2 & 5x_1 \end{bmatrix}$

Review of Fundamentals

❖ Common Differentiation Rules

- Constant Rule: the derivative of a constant is 0
- Sum & Difference Rule: $\big(f(x)+g(x)\big)' = f'(x)+g'(x)$

  $\big(f(x)-g(x)\big)' = f'(x)-g'(x)$
- Product Rule: $\big(f(x)g(x)\big)' = f(x)g'(x)+f'(x)g(x)$
- Quotient Rule: $\left(\frac{f(x)}{g(x)}\right)' = \frac{f'(x)g(x)-f(x)g'(x)}{(g(x))^2}$
- Chain Rule: $(f(g(x)))' = f'(g(x))g'(x)$ or $\frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}$

EE2213 Introduction to AI

Lecture 8: Regression

Dr. WANG Si
si.wang@nus.edu.sg

Electrical and Computer Engineering Department
National University of Singapore

OVERVIEW OF COURSE CONTENTS

- **Introduction (Shaojing)**
  - What is AI
  - Applications of AI
  - AI agent
- **Search (Shaojing)**
  - Uninformed search algorithms: breadth-first, depth-first, uniform-cost(Dijkstra's algorithm)
  - Informed search algorithms: greedy best-first, A*
  - Applications
- **Optimisation (Shaojing)**
  - Linear programming
  - Convex problems
  - Applications
- **Machine learning (Wang Si) (Weeks 6-9)**
  - Supervised and unsupervised learning: regression, classification, clustering
  - Neural networks and deep learning
  - Applications
- **Knowledge representation (Wang Si) (Weeks 10-11)**
  - Knowledge Representation and Reasoning
  - Propositional Logic
  - Applications
- **Ethical considerations (Shaojing)**
  - Bias in AI
  - Privacy concerns
  - Societal impact

(Week 10, Monday)

No Class on 20 Oct.! A recording will be uploaded on Canvas.

AGENDA

- We will discuss:
  - Linear Regression
  - Polynomial Regression
  - Ridge Regression (for regularization)
  - Evaluation Metrics

EE2211

At the end of this lecture, you should be able to:

- ✓ Explain the working principles
- ✓ Apply the techniques to solve regression problems.
- ✓ Evaluate the performance of a regression task

Recap: Regression

Input: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_3, y_N)$

Output: $f(\mathbf{x})$ that predict **numerical** *y* given **x**

map feat x to a y value.

Applications:

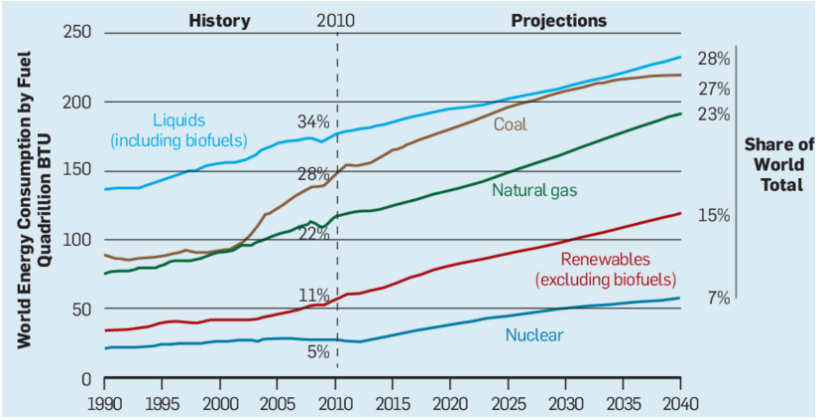
House Price Prediction

Student Performance Prediction

	Attendance (%)	Assignment	Quiz	Study Hours/Week	Final Exam
1	90	85	80	10	88
2	75	70	65	5	72
…	…	…	…	…	…

Sales Forecasting

Energy Consumption Estimation



Linear Regression

if over determined, need left-inverse
under determined, need right-inverse
} see EE2211

Linear Regression (Scalar Function)

- **Linear Regression** learns a model which is a **linear combination of features** of the input example.

Input: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$,

where $\mathbf{x}$ is a d-dimensional feature vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$$

Output: $f_{\mathbf{w}}(\mathbf{x}) = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d = \mathbf{x}^T \mathbf{w}$

where $\mathbf{w}$ is a d-dimensional vector of **parameters**

$$\mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}$$

e.g., House Price Prediction

No.	Size	No. rooms	Location	Sold Price
1	112	2	648310	600000
…				
90	100	4	129580	801256

Linear Regression (Scalar Function)

Learning (Training)

Find the optimal value $\mathbf{w}^*$ which **minimizes**:

$$J(\mathbf{w}) = \sum_{i=1}^{N} (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$$

Squared error loss

Least Squares Regression

- **Objective function**: the expression to be minimized or maximized

- **Loss function**: a measure of the difference between predicted output $f_{\mathbf{w}}(\mathbf{x}_i)$ and actual/desired/target output y_i.

$$\sum_{i=1}^{N} (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = (\mathbf{Xw} - \mathbf{y})^T (\mathbf{Xw} - \mathbf{y})$$

where $\mathbf{X} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,d} \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$

Linear Regression (Scalar Function)

Learning (Training)

$$\underset{\mathbf{w}}{\operatorname{argmin}} \; (\mathbf{Xw}-\mathbf{y})^T(\mathbf{Xw}-\mathbf{y})$$

$$J(\mathbf{w}) = (\mathbf{Xw}-\mathbf{y})^T(\mathbf{Xw}-\mathbf{y}) = (\mathbf{w}^T\mathbf{X}^T-\mathbf{y}^T)(\mathbf{Xw}-\mathbf{y})$$

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

The optimal value $\mathbf{w}^*$ should have **zero gradient:**

$$\frac{dJ(\mathbf{w})}{d\mathbf{w}} = \mathbf{0}$$

Why?

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

$$\frac{d(\mathbf{w}^T\mathbf{Aw})}{d\mathbf{w}} = (\mathbf{A}+\mathbf{A}^T)\mathbf{w}$$

$$\frac{d(\mathbf{y}^T\mathbf{Aw})}{d\mathbf{w}} = \mathbf{A}^T\mathbf{y}$$

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

$$\mathbf{X}^T\mathbf{Xw} = \mathbf{X}^T\mathbf{y}$$

If $\mathbf{X}^T\mathbf{X}$ is invertible, $\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

Linear Regression (Scalar Function)

➢ Learning (Training)

- Geometric Interpretation of Least Squares Solution

$$\mathbf{X}^T(\mathbf{Xw}-\mathbf{y}) = \mathbf{0} = X^T e$$

$\mathbf{e}$ (error term)

$\mathbf{e}$ derived from $\frac{dJ(w)}{dw}$ to find w^*, equated to zero to minimize error.

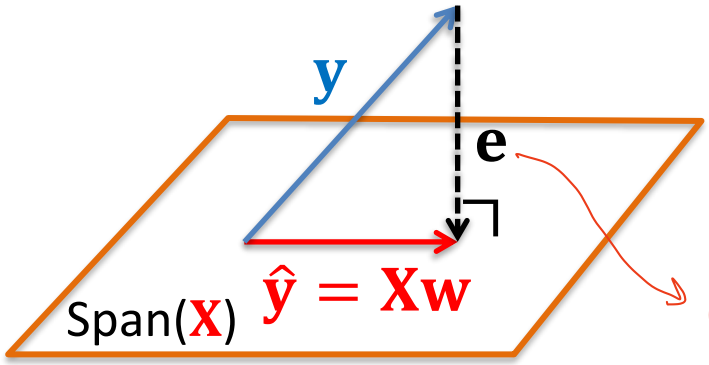
Xw^* is the projection of y on the column space $Col(X)$

↳ linear combination of col of X. $x_1w_1 + x_2w_2 \dots$

$$\begin{bmatrix} x_{1,1} & \cdots & x_{N,1} \\ \vdots & \ddots & \vdots \\ x_{1,d} & \cdots & x_{N,d} \end{bmatrix} \mathbf{e} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

$\mathbf{e}$ is perpendicular to each row of $\mathbf{X}^T$ (i.e., each column of $\mathbf{X}$)

$\mathbf{e}$ is perpendicular to the column space of $\mathbf{X}$ (i.e., Span($\mathbf{X}$))



orthogonal projection of vector y.

Least Square Approximation (lsq(A,b)) → from CS2040

y : actual correct outputs.

$\hat{y}$: estimated outputs using given features X and trained parameters/weights w.

e: error between predicted and actual ⇒ $\hat{y} - y \Rightarrow Xw - y$

we are minimizing the error e by finding w^* such that $\|Xw^* - y\| \leq \|Xv - y\|$ for any other $v \in \mathbb{R}^n$

⇒ can find w^* the lsq solution such that $\hat{y}$ is predicted best.

Linear Regression (Scalar Function)

➢ Prediction (Testing)

$$\mathbf{f}_{\mathbf{w}^*}(\mathbf{X}_{new}) = \mathbf{X}_{new}\mathbf{w}^* = \begin{bmatrix} x_{1,1}^{new} & \cdots & x_{1,d}^{new} \\ \vdots & \ddots & \vdots \\ x_{m,1}^{new} & \cdots & x_{m,d}^{new} \end{bmatrix} \begin{bmatrix} w_1^* \\ \vdots \\ w_d^* \end{bmatrix}$$

$$= \begin{bmatrix} (\mathbf{x}_1^{new})^T\mathbf{w}^* \\ \vdots \\ (\mathbf{x}_m^{new})^T\mathbf{w}^* \end{bmatrix} = \hat{y}$$ the predicted output vector.

If w^* is best, $\hat{y}$ should be close to y by minimum squared errors.

How to ensure solution w^* is the global minimum i.e. the true minimum/best weight to get best $\hat{y} = Xw^*$

Is the obtained $\mathbf{w}^*$ a global minimum?

Yes! Sum of squared errors is convex $\Rightarrow$ any local min is global min.

Linear Regression (Scalar Function)

Example: Given a training set $\{(x_i, y_i)\}_{i=1}^{6} = \{(-10,5),(-8,5),(-3,4),(-1,3),(2,2),(8,2)\}$, predict the test dataset $\{7,9\}$ using least squares regression.

$$\mathbf{X} = \begin{bmatrix} -10 \\ -8 \\ -3 \\ -1 \\ 2 \\ 8 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 5 \\ 5 \\ 4 \\ 3 \\ 2 \\ 2 \end{bmatrix}$$

Step 1: Learning (work out $\mathbf{w}^*$)

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

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

Since $\mathbf{X}^T\mathbf{X} = [242]$ is invertible

$$= [242]^{-1}[-10,-8,-3,-1,2,8] \begin{bmatrix} 5 \\ 5 \\ 4 \\ 3 \\ 2 \\ 2 \end{bmatrix}$$

$$= -0.3512$$

Step 2: Prediction

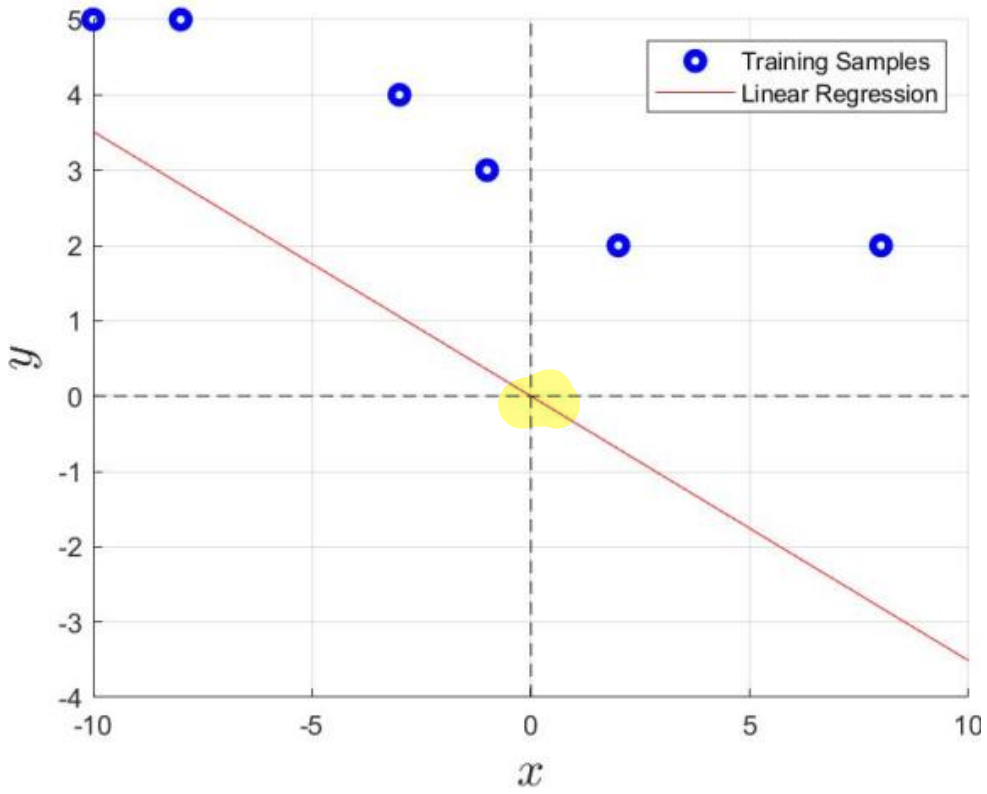
$$\mathbf{X}_t = \begin{bmatrix} 7 \\ 9 \end{bmatrix}$$

$$\mathbf{y}_t = \mathbf{f}_{\mathbf{w}^*}(\mathbf{X}_t) = \mathbf{X}_t\mathbf{w}^* = \begin{bmatrix} 7 \\ 9 \end{bmatrix}[-0.3512]$$

$$= \begin{bmatrix} -2.4587 \\ -3.1612 \end{bmatrix}$$

Linear Regression (Scalar Function)

Example: Given a training set $\{(x_i, y_i)\}_{i=1}^{6} = \{(-10, 5), (-8, 5), (-3, 4), (-1, 3), (2, 2), (8, 2)\}$, predict the test dataset $\{7, 9\}$ using least squares regression.



$$f_{\mathbf{w}}(\mathbf{x}) = w_1 x_1 = -0.3512 x_1$$

The line always passes through the origin.

because no offset!

Linear Regression (Scalar Function)

➢ Linear Regression with bias/offset

Input: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$,

Output: $f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d = \mathbf{x}^T \mathbf{w}$

where $\mathbf{x}$ is a $(d+1)$-dimensional feature vector

$\mathbf{w}$ is a $(d+1)$-dimensional vector of parameters

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_d \end{bmatrix}$$

extra constant/intercept to allow the model to make non-zero predictions when all inputs are zero.

$\hat{y} = Xw^* + b$ ($b \to w_0$)

The bias/offset term w_0 is responsible for translating the line/plane/hyperplane away from the origin

Linear Regression (Scalar Function)

➢ Linear Regression with bias/offset

Input: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N),$

Output: $f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d = \mathbf{x}^T \mathbf{w}$

see prev. slide on why transpose. written as vector of weights instead of matrix

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

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_d \end{bmatrix}$$

$$\text{where } \mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & \cdots & x_{N,d} \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

-Prediction: $\mathbf{f}_{\mathbf{w}^*}(\mathbf{X}_t) = \mathbf{X}_t\mathbf{w}^*$

$$\text{where } \mathbf{X}_t = \begin{bmatrix} 1 & x_{1,1}^t & \cdots & x_{1,d}^t \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m,1}^t & \cdots & x_{m,d}^t \end{bmatrix}, \mathbf{w}^* = \begin{bmatrix} w_0^* \\ \vdots \\ w_d^* \end{bmatrix}$$

Linear Regression (Scalar Function)

➢ Linear Regression with bias/offset

Example: Given a training set $\{(x_i, y_i)\}_{i=1}^{6} = \{(-10,5),(-8,5),(-3,4),(-1,3),(2,2),(8,2)\}$, predict the test dataset $\{7, 9\}$ using least squares regression.

Step 1: Learning (work out $\mathbf{w}^*$)

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

$$\mathbf{X} = \begin{bmatrix} 1 & -10 \\ 1 & -8 \\ 1 & -3 \\ 1 & -1 \\ 1 & 2 \\ 1 & 8 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 5 \\ 5 \\ 4 \\ 3 \\ 2 \\ 2 \end{bmatrix}$$

Since $\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 6 & -12 \\ -12 & 242 \end{bmatrix}$ is invertible

$$= \begin{bmatrix} 6 & -12 \\ -12 & 242 \end{bmatrix}^{-1} \begin{bmatrix} 1 & -10 \\ 1 & -8 \\ 1 & -3 \\ 1 & -1 \\ 1 & 2 \\ 1 & 8 \end{bmatrix}^T \begin{bmatrix} 5 \\ 5 \\ 4 \\ 3 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 3.1055 \\ -0.1972 \end{bmatrix}$$

Step 2: Prediction

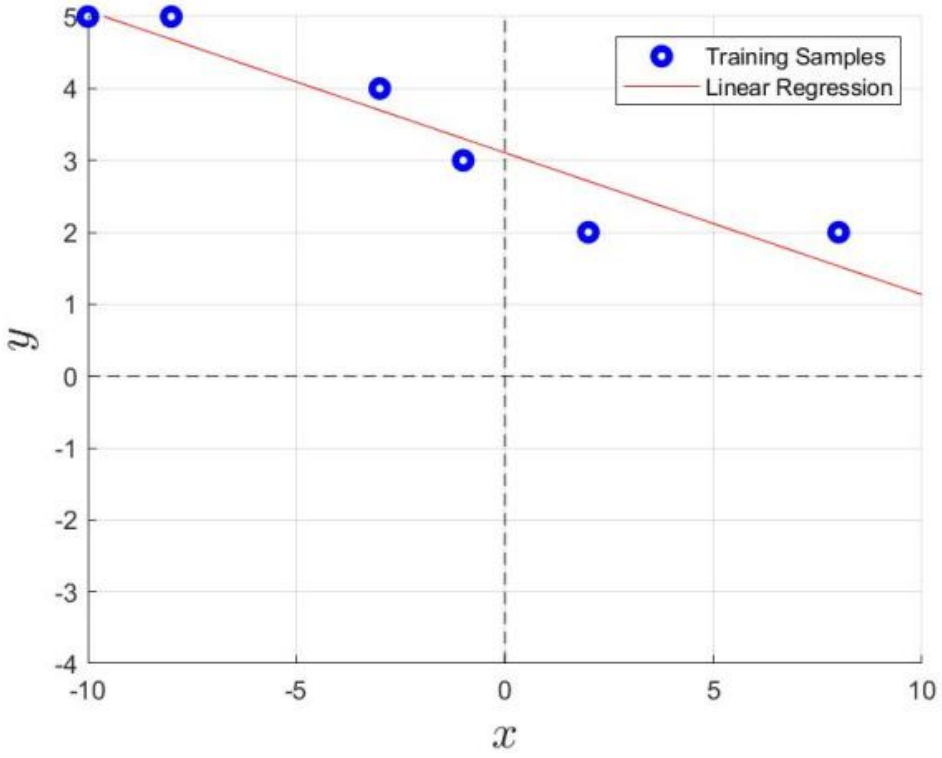
$$\mathbf{X}_t = \begin{bmatrix} 1 & 7 \\ 1 & 9 \end{bmatrix}$$

$$\mathbf{y}_t = \mathbf{f}_{\mathbf{w}^*}(\mathbf{X}_t) = \mathbf{X}_t\mathbf{w}^* = \begin{bmatrix} 1 & 7 \\ 1 & 9 \end{bmatrix} \begin{bmatrix} 3.1055 \\ -0.1972 \end{bmatrix} = \begin{bmatrix} 1.7248 \\ 1.3303 \end{bmatrix}$$

Linear Regression (Scalar Function)

➢ Linear Regression with bias/offset

Example: Given a training set $\{(x_i, y_i)\}_{i=1}^{6} = \{(-10, 5), (-8, 5), (-3, 4), (-1, 3), (2, 2), (8, 2)\}$, predict the test dataset $\{7, 9\}$ using least squares regression.

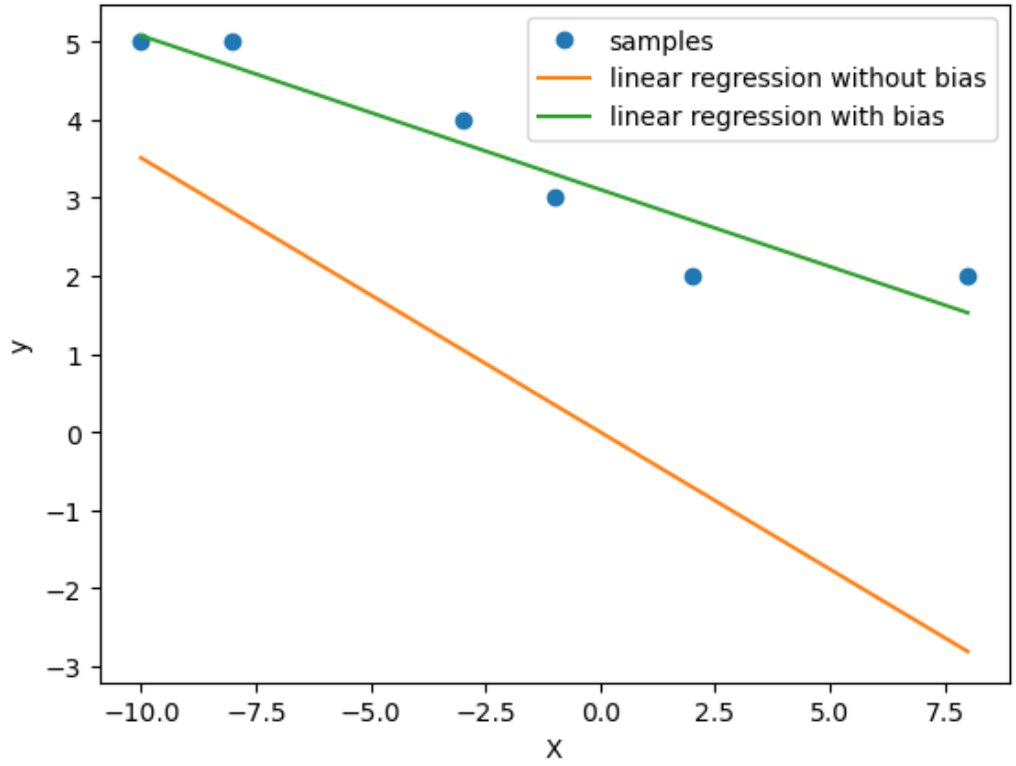


$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1$$
$$= 3.1055 - 0.1972 x_1$$

Linear Regression (Scalar Function)

➢ Comparison

Example: Given a training set $\{(x_i, y_i)\}_{i=1}^{6} = \{(-10, 5), (-8, 5), (-3, 4), (-1, 3), (2, 2), (8, 2)\}$, predict the test dataset $\{7, 9\}$ using least squares regression.



Linear Regression (Vector Function)

➢ Extend to Multiple Outputs

Input: $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_N, \mathbf{y}_N)$,

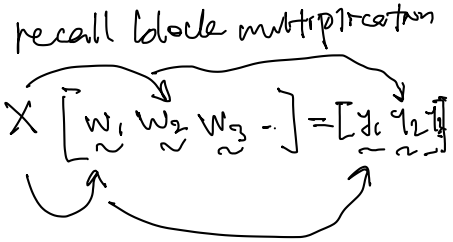
where $\mathbf{x}$ is a $(d+1)$-dimensional feature vector

$\mathbf{y}$ is a h-dimensional target vector

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_h \end{bmatrix}$$

e.g., predict multiple physiological signals

	Age	Weight	Physical activity level	Oxygen saturation	Heart rate	Blood pressure	Respiration rate
1							
…							



Output: $\mathbf{f}_\mathbf{W}(\mathbf{x}) = \mathbf{x}^T\mathbf{W}$, where $\mathbf{W}$ is a matrix of size $(d+1) \times h$

$$\mathbf{W} = \begin{bmatrix} w_{0,1} & \cdots & w_{0,h} \\ \vdots & \ddots & \vdots \\ w_{d,1} & \cdots & w_{d,h} \end{bmatrix}$$

Equivalent to performing Linear Regression for each target output variable (h times)

$$\mathbf{f}_\mathbf{W}(\mathbf{x}) = [f_{\mathbf{w}_1}(\mathbf{x}), \ldots, f_{\mathbf{w}_h}(\mathbf{x})]$$

weight vector $\mathbf{w}_1$ for y_1

weight vector $\mathbf{w}_h$ for y_h

Linear Regression (Vector Function)

Learning

Find the optimal value $\mathbf{W}^*$ which minimizes:

$$J(\mathbf{W}) = \sum_{k=1}^{h}\sum_{i=1}^{N}(f_{\mathbf{w}_k}(\mathbf{x}_i) - y_{i,k})^2$$

$$= \sum_{k=1}^{h}(\mathbf{X}\mathbf{w}_k - \mathbf{y}_k)^T(\mathbf{X}\mathbf{w}_k - \mathbf{y}_k)$$

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & \cdots & x_{N,d} \end{bmatrix}$$

The optimal value $\mathbf{W}^*$ should have **zero gradient:**

$$\frac{dJ(\mathbf{W})}{d\mathbf{W}} = \begin{bmatrix} \frac{\partial J}{\partial w_{0,1}} & \cdots & \frac{\partial J}{\partial w_{0,h}} \\ \vdots & \ddots & \vdots \\ \frac{\partial J}{\partial w_{d,1}} & \cdots & \frac{\partial J}{\partial w_{d,h}} \end{bmatrix} = \mathbf{0} \Rightarrow \frac{dJ(\mathbf{W})}{d\mathbf{w}_k} = \mathbf{0}, k = 1,2,\dots,h$$

$$\frac{dJ(\mathbf{W})}{d\mathbf{w}_1} \quad \frac{dJ(\mathbf{W})}{d\mathbf{w}_h}$$

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

Linear Regression (Vector Function)

➢ Learning

Find the optimal value for $\mathbf{W}^*$ which minimizes:

$$J(\mathbf{W}) = \sum_{k=1}^{h}\sum_{i=1}^{N}(f_{\mathbf{w}_k}(\mathbf{x}_i) - y_{i,k})^2$$

$$= \sum_{k=1}^{h}(\mathbf{X}\mathbf{w}_k - \mathbf{y}_k)^T(\mathbf{X}\mathbf{w}_k - \mathbf{y}_k)$$

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & \cdots & x_{N,d} \end{bmatrix}$$

$$\mathbf{W}^* = [\mathbf{w}_1{}^*, \ldots, \mathbf{w}_h{}^*] = [(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}_1, \ldots, (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}_h]$$

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

$Y = [y_1 y_2 \ldots]$

now multiple vector outputs -

$$\mathbf{Y} = \begin{bmatrix} y_{1,1} & \cdots & y_{1,h} \\ \vdots & \ddots & \vdots \\ y_{N,1} & \cdots & y_{N,h} \end{bmatrix}$$

Is the obtained $\mathbf{W}^*$ a global minimum?

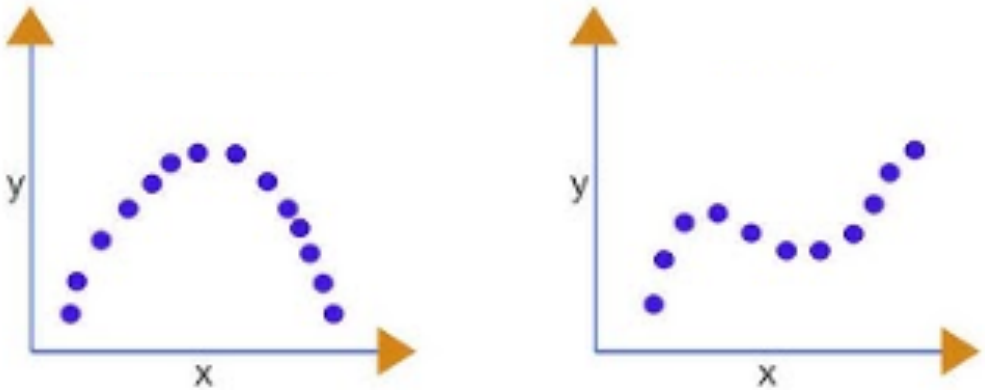
same thing nothing changed.

Linear Regression (Vector Function)

➢ Prediction

$$\mathbf{F}_{\mathbf{W}^*}(\mathbf{X}_t) = \mathbf{X}_t\mathbf{W}^* = \begin{bmatrix} 1 & x_{1,1}^t & \cdots & x_{1,d}^t \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m,1}^t & \cdots & x_{m,d}^t \end{bmatrix} \begin{bmatrix} w_{0,1}^* & \cdots & w_{0,h}^* \\ \vdots & \ddots & \vdots \\ w_{d,1}^* & \cdots & w_{d,h}^* \end{bmatrix}$$

$$= \begin{bmatrix} (\mathbf{x}_1^t)^T\mathbf{w}_1^* & \cdots & (\mathbf{x}_1^t)^T\mathbf{w}_h^* \\ \vdots & \ddots & \vdots \\ (\mathbf{x}_m^t)^T\mathbf{w}_1^* & \cdots & (\mathbf{x}_m^t)^T\mathbf{w}_h^* \end{bmatrix}$$



Polynomial Regression

non-linear relationship between input and output

Polynomial Regression

Polynomial Model

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

1st order (Linear)

2nd order (Quadratic)

3rd order (Cubic)

Order number: Hyperparameter!

e.g., d=1, 3rd order: $f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + w_{11} x_1^2 + w_{111} x_1^3$

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

No. parameters for n-th order polynomial model with d input variables?

$$\binom{n+d}{d} = \frac{(n+d)!}{n!\,d!}$$

No. parameters for *degree n terms* with d input variables?

$$\binom{n+d-1}{d-1} = \frac{(n+d-1)!}{n!\,(d-1)!}$$

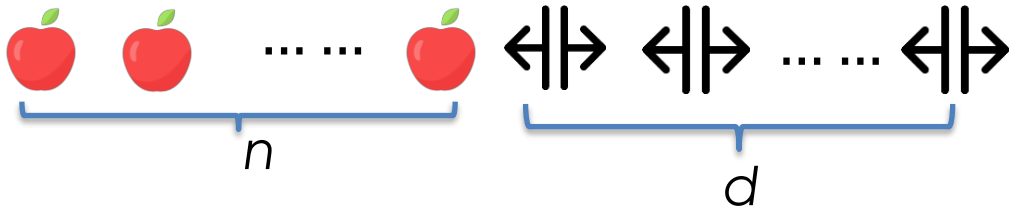
- Proof: total No. paras for *n*-th order polynomial model with *d* input variables is $\binom{n+d}{d} = \frac{(n+d)!}{n!d!}$

$$x_1^{k_1} x_2^{k_2} \dots x_d^{k_d} : k_1 + k_2 + \cdots + k_d \le n$$

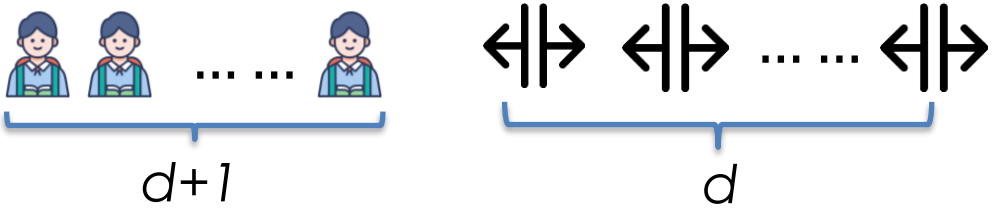
$$\Updownarrow$$

$$k_1 + k_2 + \cdots + k_d + k_{d+1} = n$$

Counting: equivalent to allocating *n* apples to *d*+1 students.



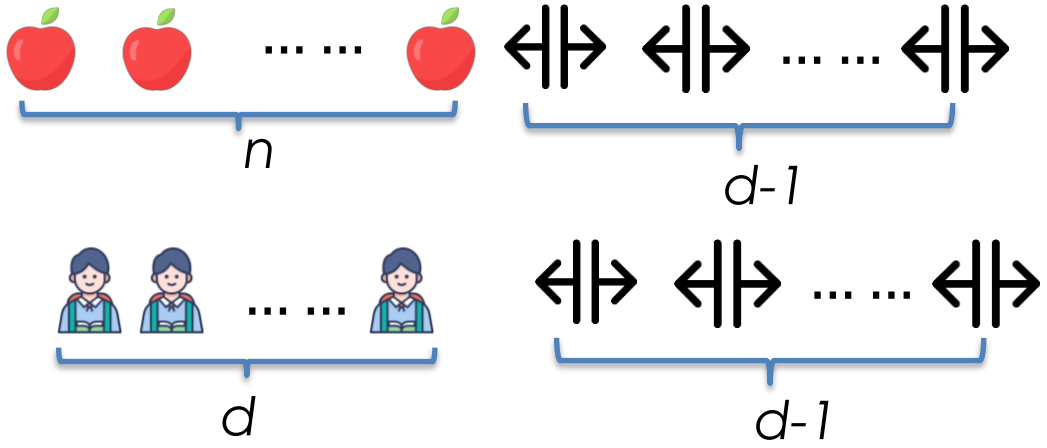
$$\binom{n+d}{d} = \binom{n+d}{n}$$



- Proof: No. paras for *degree n terms* with *d* input variables is $\binom{n+d-1}{d-1} = \frac{(n+d-1)!}{n!(d-1)!}$

$$x_1^{k_1} x_2^{k_2} \dots x_d^{k_d} : k_1 + k_2 + \cdots + k_d = n$$

Counting: equivalent to allocating *n* apples to *d* students.



$$\binom{n+d-1}{d-1} = \binom{n+d-1}{n}$$

Polynomial Regression

➢ Polynomial Model

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

$$= f_{\mathbf{w}}(\mathbf{p}(\mathbf{x})) = \mathbf{p}^T \mathbf{w}$$

$$\mathbf{p} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \\ \vdots \\ x_i x_j \\ \vdots \\ x_i x_j x_k \\ \vdots \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \\ \vdots \\ w_{ij} \\ \vdots \\ w_{ijk} \\ \vdots \end{bmatrix}$$

0th degree

linear term

2nd degree

3rd degree

Equivalent to:

polynomial transformation

+

linear regression

Polynomial Regression

Learning

Input: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$,

Output: $f_{\mathbf{w}}(\mathbf{x}) = f_{\mathbf{w}}(\mathbf{p}(\mathbf{x})) = \mathbf{p}^T\mathbf{w}$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

Find the optimal value $\mathbf{w}^*$ which minimizes:

same cost function

$$J(\mathbf{w}) = \sum_{i=1}^{N} (f_{\mathbf{w}}(\mathbf{p}(\mathbf{x}_i)) - y_i)^2 = (\mathbf{Pw} - \mathbf{y})^T(\mathbf{Pw} - \mathbf{y})$$

just replace X with P which has all degree terms.

$$\mathbf{P} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,d} & \cdots & x_{1,i}x_{1,j} & \cdots & x_{1,i}x_{1,j}x_{1,k} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{N,1} & \cdots & x_{N,d} & \cdots & x_{N,i}x_{N,j} & \cdots & x_{N,i}x_{N,j}x_{N,k} & \cdots \end{bmatrix}$$

Solution: $\mathbf{w}^* = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T\mathbf{y}$, If $\mathbf{P}^T\mathbf{P}$ is invertible

yes.

global minimum?

Polynomial Regression

➢ Prediction

$$\mathbf{f}_{\mathbf{w}^*}(\mathbf{P}(\mathbf{X}_t)) = \mathbf{P}_t\mathbf{w}^*$$

$$= \begin{bmatrix} 1 & x_{1,1}^t & \cdots & x_{1,d}^t & \cdots & x_{1,i}^t x_{1,j}^t & \cdots & x_{1,i}^t x_{1,j}^t x_{1,k}^t & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{m,1}^t & \cdots & x_{m,d}^t & \cdots & x_{m,i}^t x_{m,j}^t & \cdots & x_{m,i}^t x_{m,j}^t x_{m,k}^t & \cdots \end{bmatrix} \begin{bmatrix} w_0^* \\ w_1^* \\ \vdots \\ w_d^* \\ \vdots \\ w_{ij}^* \\ \vdots \\ w_{ijk}^* \\ \vdots \end{bmatrix}$$

$$= \begin{bmatrix} (\mathbf{p}_1^t)^T\mathbf{w}^* \\ \vdots \\ (\mathbf{p}_m^t)^T\mathbf{w}^* \end{bmatrix}$$

Polynomial Regression

Example: Given a training set $\{(x_i, y_i)\}_{i=1}^{6} = \{(-10,5),(-8,5),(-3,4),(-1,3),(2,2),(8,2)\}$, predict the test dataset $\{7,9\}$ using 3rd order polynomial regression.

Step 0: Polynomial Transformation ($\mathbf{X} \rightarrow \mathbf{P}$)

the only added/trained step. the remaining steps are all the same.

3rd order polynomial model with 1 input variable:

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + w_{11} x_1^2 + w_{111} x_1^3 = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_{11} \\ w_{111} \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} 1 & x_{1,1} & x_{1,1}^2 & x_{1,1}^3 \\ 1 & x_{2,1} & x_{2,1}^2 & x_{2,1}^3 \\ 1 & x_{3,1} & x_{3,1}^2 & x_{3,1}^3 \\ 1 & x_{4,1} & x_{3,1}^2 & x_{4,1}^3 \\ 1 & x_{5,1} & x_{4,1}^2 & x_{5,1}^3 \\ 1 & x_{6,1} & x_{6,1}^2 & x_{6,1}^3 \end{bmatrix} = \begin{bmatrix} 1 & -10 & 100 & -1000 \\ 1 & -8 & 64 & -512 \\ 1 & -3 & 9 & -27 \\ 1 & -1 & 1 & -1 \\ 1 & 2 & 4 & 8 \\ 1 & 8 & 64 & 512 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 5 \\ 5 \\ 4 \\ 3 \\ 2 \\ 2 \end{bmatrix}$$

Polynomial Regression

Example: Given a training set $\{(x_i, y_i)\}_{i=1}^{6} = \{(-10,5),(-8,5),(-3,4),(-1,3),(2,2),(8,2)\}$, predict the test dataset $\{7, 9\}$ using 3rd order polynomial regression.

Step 1: Learning (work out $\mathbf{w}^*$)

Since $\mathbf{P}^T\mathbf{P}$ is invertible, $\mathbf{w}^* = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T\mathbf{y} = \begin{bmatrix} 2.6894 \\ -0.3772 \\ 0.0134 \\ 0.0029 \end{bmatrix}$

Step 2: Prediction

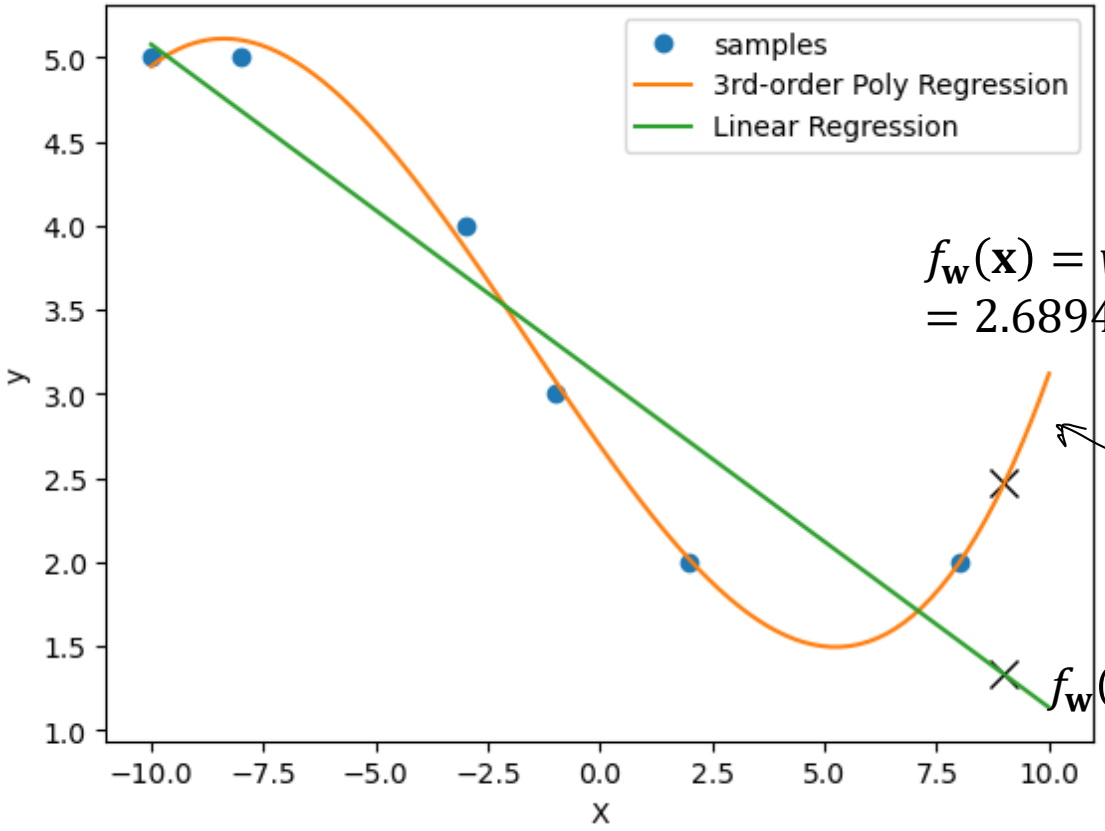
$$\mathbf{P}_t = \begin{bmatrix} 1 & x_{1,1}^t & (x_{1,1}^t)^2 & (x_{1,1}^t)^3 \\ 1 & x_{2,1}^t & (x_{2,1}^t)^2 & (x_{2,1}^t)^3 \end{bmatrix} = \begin{bmatrix} 1 & 7 & 49 & 343 \\ 1 & 9 & 81 & 729 \end{bmatrix}$$

$$\mathbf{y}_t = \mathbf{f}_{\mathbf{w}^*}(\mathbf{P}_t) = \mathbf{P}_t\mathbf{w}^* = \begin{bmatrix} 1 & 7 & 49 & 343 \\ 1 & 9 & 81 & 729 \end{bmatrix} \begin{bmatrix} 2.6894 \\ -0.3772 \\ 0.0134 \\ 0.0029 \end{bmatrix}$$

$$= \begin{bmatrix} 1.6874 \\ 2.4661 \end{bmatrix}$$

Polynomial Regression

Example: Given a training set $\{(x_i, y_i)\}_{i=1}^{6} = \{(-10, 5), (-8, 5), (-3, 4), (-1, 3), (2, 2), (8, 2)\}$, predict the test dataset $\{7, 9\}$ using 3rd order polynomial regression.



$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + w_{11} x_1^2 + w_{111} x_1^3$$
$$= 2.6894 - 0.3772 x_1 + 0.0134 x_1^2 + 0.0029 x_1^3$$

How to know which fits better? ⇒ use validation set!

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 = 3.1055 - 0.1972 x_1$$

Polynomial Regression

➢ Extend to Multiple Outputs

Input: $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_N, \mathbf{y}_N)$,

where $\mathbf{x}$ is a *d*-dimensional feature vector

$\mathbf{y}$ is a *h*-dimensional target vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_h \end{bmatrix}$$

Output: $\mathbf{f_W}(\mathbf{x}) = \mathbf{f_W}(\mathbf{p}(\mathbf{x})) = \mathbf{p}^T\mathbf{W}$,

W become matrix

-Learning: $\mathbf{W}^* = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T\mathbf{Y}$

-Prediction: $\mathbf{F_{W^*}}(\mathbf{P}(\mathbf{X}_t)) = \mathbf{P}_t\mathbf{W}^*$

Ridge Regression

If $\mathbf{X}^T\mathbf{X}$ or $\mathbf{P}^T\mathbf{P}$ is not invertible

Ridge Regression

- **Ridge Regression = Regression + L2 regularization**

  Shrinks the regression coefficients w by imposing a penalty to their size.

  recall EE2211. reduce weights to prevent overfitting.

  go to slide 42.

  Find the optimal value $\mathbf{w}^*$ which minimizes:

$$J(\mathbf{w}) = \sum_{i=1}^{N} (f_{\mathbf{w}}(\mathbf{p}(\mathbf{x}_i)) - y_i)^2 + \lambda \sum_i w_i^2$$

L2 regularization: $\sum_i w_i^2 = \|\mathbf{w}\|_2^2$, squared L2 norm of $\mathbf{w}$

$$= (\mathbf{Pw} - \mathbf{y})^T(\mathbf{Pw} - \mathbf{y}) + \lambda \mathbf{w}^T\mathbf{w}$$

lambda places more weight on Regularisation term. Means minimisation on actual loss function is lesser. Increasing lambda results in greater losses, but smaller bias and variance (W is lesser, but this also means its less accurate)

$\uparrow \lambda = \downarrow w$ magnitude.

(Hyperparameter)

$\lambda > 0$ is a **complexity/regularization parameter** that controls the amount of shrinkage: the larger the value of λ, the greater the amount of shrinkage.

Ridge Regression

➢ Learning

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

Solution: $\frac{dJ(\mathbf{w})}{d\mathbf{w}}=\mathbf{0}$

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

$$2\mathbf{P}^T\mathbf{Pw}-2\mathbf{P}^T\mathbf{y}+2\lambda\mathbf{w}=\mathbf{0}$$

$$\mathbf{P}^T\mathbf{Pw}+\lambda\mathbf{w}=\mathbf{P}^T\mathbf{y}$$

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

Since $\mathbf{P}^T\mathbf{P}+\lambda\mathbf{I}$ is always invertible, $\mathbf{w}^*=(\mathbf{P}^T\mathbf{P}+\lambda\mathbf{I})^{-1}\mathbf{P}^T\mathbf{y}$

Ridge Regression

Proof: $\mathbf{P}^T\mathbf{P} + \lambda\mathbf{I}$ is always invertible when $\lambda > 0$

by contradiction

Suppose $\mathbf{P}^T\mathbf{P} + \lambda\mathbf{I}$ is not invertible.

⇓

Columns/rows in $\mathbf{P}^T\mathbf{P} + \lambda\mathbf{I}$ are linearly dependent

⇓

There exits a vector $\mathbf{a} \neq \mathbf{0}$, s.t., $(\mathbf{P}^T\mathbf{P} + \lambda\mathbf{I})\mathbf{a}=\mathbf{0}$

⇓

$$\mathbf{a}^T(\mathbf{P}^T\mathbf{P} + \lambda\mathbf{I})\mathbf{a}=0$$

⇓

$$\mathbf{a}^T\mathbf{P}^T\mathbf{P}\mathbf{a} + \lambda\mathbf{a}^T\mathbf{a} = 0$$

⇓

$$\|\mathbf{P}\mathbf{a}\|^2 + \lambda\|\mathbf{a}\|^2 = 0$$

However, $\|\mathbf{P}\mathbf{a}\|^2 + \lambda\|\mathbf{a}\|^2 = 0$ only holds if $\|\mathbf{P}\mathbf{a}\|^2 = \|\mathbf{a}\|^2 = 0$

⇓

$$\mathbf{a} = \mathbf{0}$$

⇓

Columns/rows in $\mathbf{P}^T\mathbf{P} + \lambda\mathbf{I}$ are linearly independent ⇒ full rank ⇒ invertible

Ridge Regression

Learning

Geometric Interpretation of Ridge Regression Solution

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

Dual form

Lagrangian Duality

$$\underset{x}{\operatorname{argmin}}\, f(x) + \lambda g(x)$$

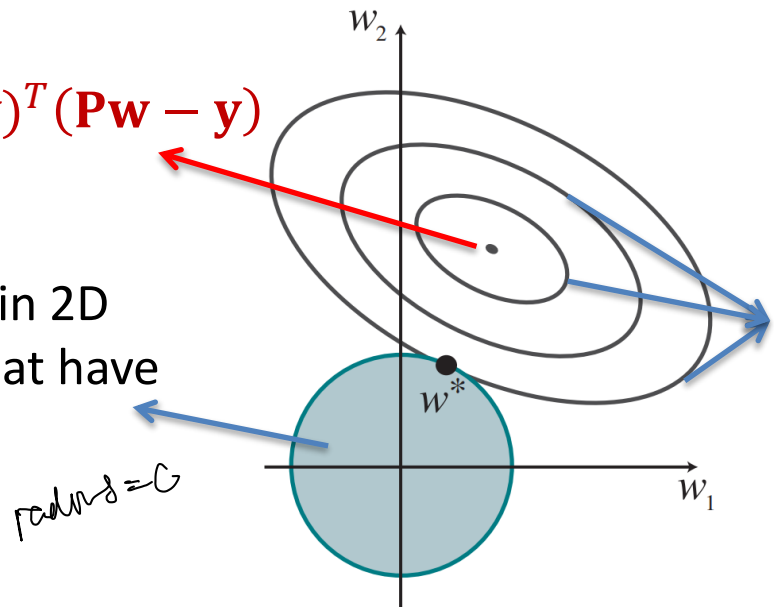
$$\underset{\mathbf{w}}{\operatorname{argmin}} (\mathbf{Pw}-\mathbf{y})^T(\mathbf{Pw}-\mathbf{y}), \text{ subject to } \mathbf{w}^T\mathbf{w} \le c$$

Primal form

$$\underset{x}{\operatorname{argmin}}\, f(x), s.t.\, g(x) \le 0$$

$$\min_{\mathbf{w}} (\mathbf{Pw}-\mathbf{y})^T(\mathbf{Pw}-\mathbf{y})$$

Set of points **w** in 2D weight space that have $\mathbf{w}^T\mathbf{w} \le c$



Contours of $(\mathbf{Pw}-\mathbf{y})^T(\mathbf{Pw}-\mathbf{y})$

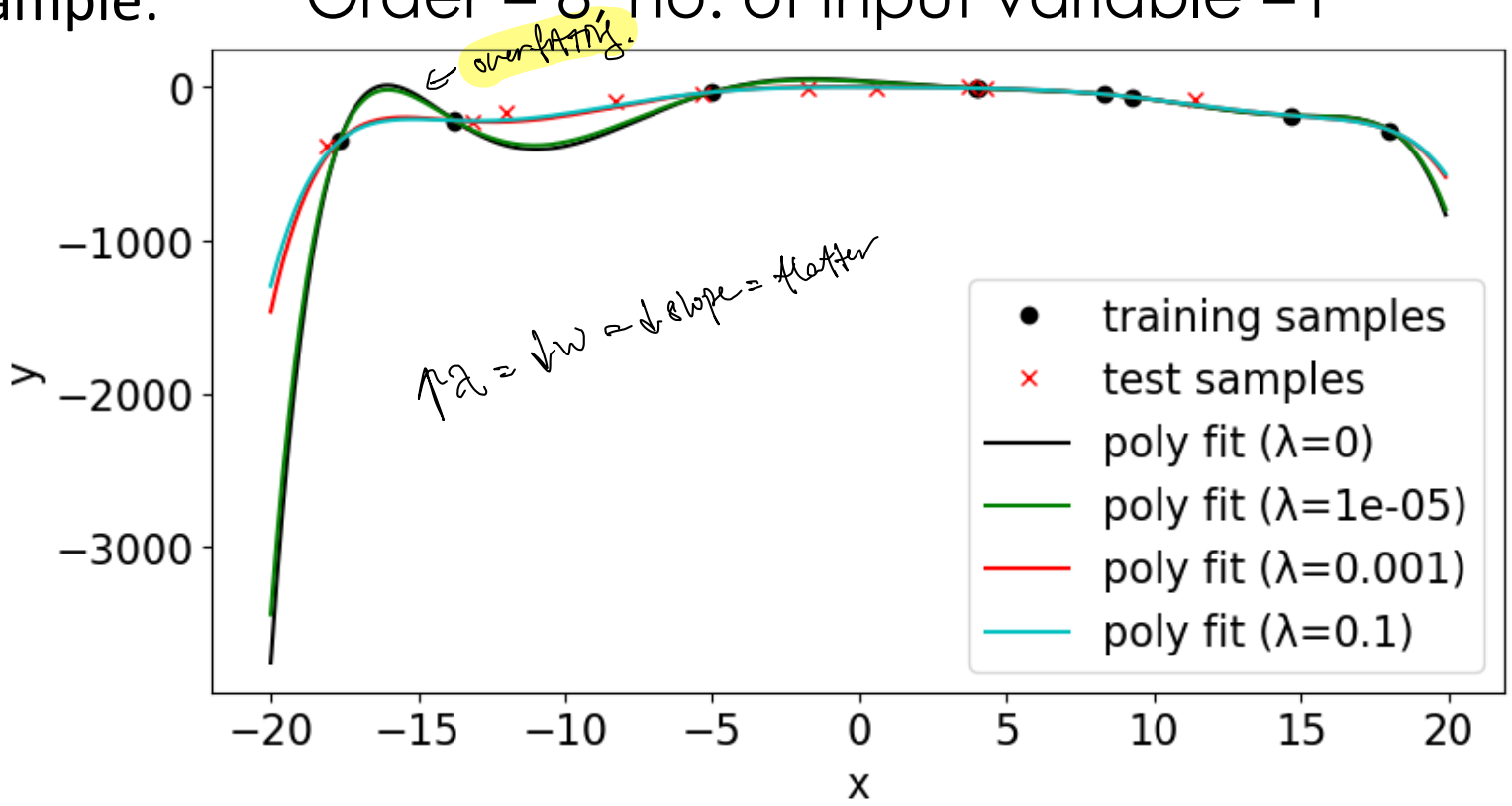
Ridge Regression

➢ Prediction

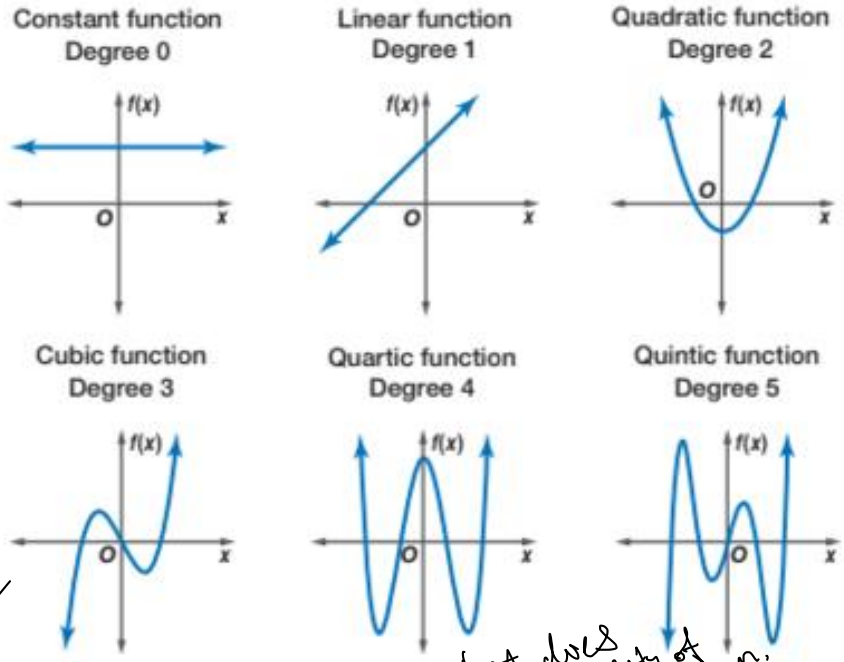
$$\mathbf{f}_{\mathbf{w}^*}(\mathbf{P}(\mathbf{X}_t)) = \mathbf{P}_t\mathbf{w}^*$$

How is weight shrinkage connected with $\mathbf{P}^T\mathbf{P}$ being not invertible?

Example: Order = 8, no. of input variable =1



Polynomial model with different degrees



↓w.

Regularization has similar effect of reducing the polynomial degree/features (flatter).

Flattening
(Fewer ups and downs)

≈

Reducing degree
(Reduce no. polynomial features)

From ① - ⑥ : rank(P) rep. no. of unique features that are just enough to describe the data

polynomial features: derived features created to describe the data

If ⑤, we used features that are combination/dependent on other features.
⇒ highly correlated features (redundancy)
↳ regularization doesn't ↓ poly features, but it suppresses the effect/feature contribution of dependent features ⇒ similar effect as removing redundant poly features

what does invertibility of P imply.

When is $\mathbf{P}^T\mathbf{P}$ not invertible?

When rank(**P**) < no. polynomial features ⑤

① Rank(**P**)=Rank($\mathbf{P}^T\mathbf{P}$)

Can be proved by **rank-nullity theorem**

(Rank(A)+nullity(A)= No. cols)

⑥ unique features of data

⑤ Intrinsic dimension of the feature space

② max no. of rows / col that are linearly independent ⇒ dimension of column space ③ , and each column represents a feature ④

Prove Rank(P)=Rank($\mathbf{P}^T\mathbf{P}$) by rank-nullity theorem

Prove Nullity($\mathbf{P}$)=Nullity($\mathbf{P}^T\mathbf{P}$)

Prove
- if $\mathbf{x}$ satisfies $\mathbf{Px} = \mathbf{0}$, $\mathbf{P}^T\mathbf{Px} = \mathbf{0}$
- and
- if $\mathbf{x}$ satisfies $\mathbf{P}^T\mathbf{Px} = \mathbf{0}$, $\mathbf{Px} = \mathbf{0}$

$$\mathbf{x}^T\mathbf{P}^T\mathbf{Px} = 0$$

$$\|\mathbf{Px}\|^2 = 0$$

$$\mathbf{Px} = \mathbf{0}$$

Ridge Regression

➢ Extend to Multiple Outputs

Input: $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_N, \mathbf{y}_N)$,

where $\mathbf{x}$ is a *d*-dimensional feature vector

$\mathbf{y}$ is a *h*-dimensional target vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_h \end{bmatrix}$$

Output: $\mathbf{f_W}(\mathbf{x}) = \mathbf{f_W}(\mathbf{p}(\mathbf{x})) = \mathbf{p}^T\mathbf{W}$,

-Learning:

Find the optimal values for $\mathbf{W}^*$ which minimizes:

$$J(\mathbf{W}) = \sum_{k=1}^{h} \left(\sum_{i=1}^{N} \left(f_{\mathbf{w}_k}(\mathbf{p}(\mathbf{x}_i)) - y_{i,k} \right)^2 + \lambda \sum_i {w_{i,k}}^2 \right)$$

$$\mathbf{Y} = \begin{bmatrix} y_{1,1} & \cdots & y_{1,h} \\ \vdots & \ddots & \vdots \\ y_{N,1} & \cdots & y_{N,h} \end{bmatrix}$$

$$= \sum_{k=1}^{h} \left((\mathbf{P}\mathbf{w}_k - \mathbf{y}_k)^T (\mathbf{P}\mathbf{w}_k - \mathbf{y}_k) + \lambda {\mathbf{w}_k}^T \mathbf{w}_k \right)$$

now a matrix

now a matrix

Solution in Compact Form: $\mathbf{W}^* = (\mathbf{P}^T\mathbf{P} + \lambda\mathbf{I})^{-1}\mathbf{P}^T\mathbf{Y}$

Ridge Regression

➢ Extend to Multiple Outputs

Input: $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_N, \mathbf{y}_N)$,

where $\mathbf{x}$ is a *d*-dimensional feature vector

$\mathbf{y}$ is a *h*-dimensional target vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_h \end{bmatrix}$$

Output: $\mathbf{f_W}(\mathbf{x}) = \mathbf{f_W}(\mathbf{p}(\mathbf{x})) = \mathbf{p}^T\mathbf{W}$,

-Learning: $\mathbf{W}^* = (\mathbf{P}^T\mathbf{P} + \lambda\mathbf{I})^{-1}\mathbf{P}^T\mathbf{Y}$

-Prediction: $\mathbf{F}_{\mathbf{W}^*}(\mathbf{P}(\mathbf{X}_t)) = \mathbf{P}_t\mathbf{W}^*$

Evaluation Metrics for Regression

Evaluation Metrics for Regression

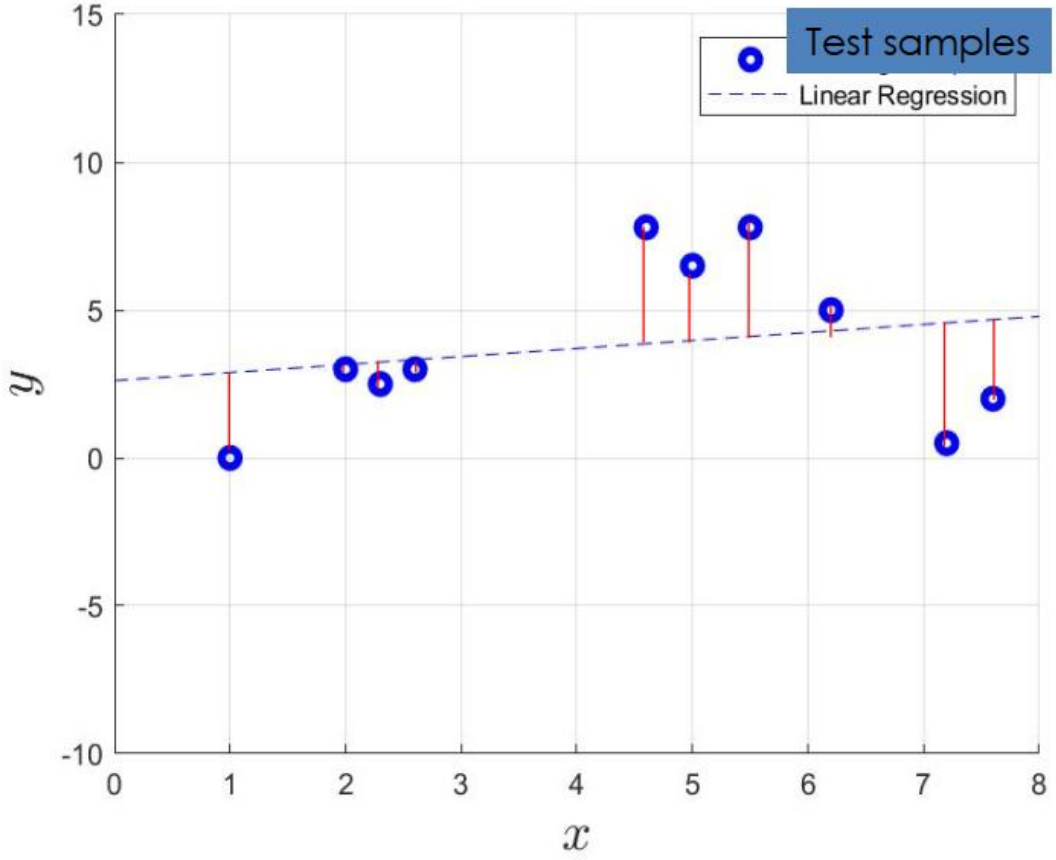
Mean Square Error

$$\left(\text{MSE} = \frac{\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}\right)$$

Mean Absolute Error

$$\left(\text{MAE} = \frac{\Sigma_{i=1}^{n}|y_i - \hat{y}_i|}{n}\right)$$

where y_i denotes the target output and $\hat{y}_i$ denotes the predicted output for sample i.




```
from sklearn.metrics import mean_absolute_error, mean_squared_error

mae = mean_absolute_error(y_true, y_pred)
mse = mean_squared_error(y_true, y_pred)
```

Summary

✓ Linear Regression

Scalar output for each sample

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

-Prediction: $\mathbf{f}_{\mathbf{w}^*}(\mathbf{X}_t) = \mathbf{X}_t\mathbf{w}^*$

Vector output for each sample

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

-Prediction: $\mathbf{F}_{\mathbf{W}^*}(\mathbf{X}_t) = \mathbf{X}_t\mathbf{W}^*$

✓ Polynomial Regression

Scalar output for each sample

-Learning: $\mathbf{w}^* = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T\mathbf{y}$

-Prediction: $\mathbf{f}_{\mathbf{w}^*}(\mathbf{P}(\mathbf{X}_t)) = \mathbf{P}_t\mathbf{w}^*$

Vector output for each sample

-Learning: $\mathbf{W}^* = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T\mathbf{Y}$

-Prediction: $\mathbf{F}_{\mathbf{W}^*}(\mathbf{P}(\mathbf{X}_t)) = \mathbf{P}_t\mathbf{W}^*$

✓ Ridge Regression

Scalar output for each sample

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

-Prediction: $\mathbf{f}_{\mathbf{w}^*}(\mathbf{P}(\mathbf{X}_t)) = \mathbf{P}_t\mathbf{w}^*$

Vector output for each sample

-Learning: $\mathbf{W}^* = (\mathbf{P}^T\mathbf{P} + \lambda\mathbf{I})^{-1}\mathbf{P}^T\mathbf{Y}$

-Prediction: $\mathbf{F}_{\mathbf{W}^*}(\mathbf{P}(\mathbf{X}_t)) = \mathbf{P}_t\mathbf{W}^*$

✓ 2 Evaluation Metrics for Regression Task

mean squared error, mean absolute error

EE2213 Introduction to AI

Lecture 9: Classification

Dr. WANG Si
si.wang@nus.edu.sg

Electrical and Computer Engineering Department
National University of Singapore

OVERVIEW OF COURSE CONTENTS

- **Introduction (Shaojing)**
  - What is AI
  - Applications of AI
  - AI agent
- **Search (Shaojing)**
  - Uninformed search algorithms: breadth-first, depth-first, uniform-cost(Dijkstra's algorithm)
  - Informed search algorithms: greedy best-first, A*
  - Applications
- **Optimisation (Shaojing)**
  - Linear programming
  - Convex problems
  - Applications
- **Machine learning (Wang Si) (Weeks 6-9)**
  - Supervised and unsupervised learning: regression, classification, clustering
  - Neural networks and deep learning
  - Applications
- **Knowledge representation (Wang Si) (Weeks 10-11)**
  - Knowledge Representation and Reasoning
  - Propositional Logic
  - Applications
- **Ethical considerations (Shaojing)**
  - Bias in AI
  - Privacy concerns
  - Societal impact

(Week 10, Monday)

No Class on 20 Oct.! A recording will be uploaded on Canvas.

AGENDA

- We will discuss:
  - Polynomial Regression for Classification
  - Logistic Regression
  - Evaluation Metrics for Classification

At the end of this lecture, you should be able to:

- ✓ Explain the underlying principles of logistic regression
- ✓ Apply the two algorithms for classification
- ✓ Evaluate the classification performance

Recap: Classification Task

Input: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_3, y_N)$

Output: $f(\mathbf{x})$ that predict **categorical** *y* given **x**

Applications:

Spam Detection

Sentiment Analysis

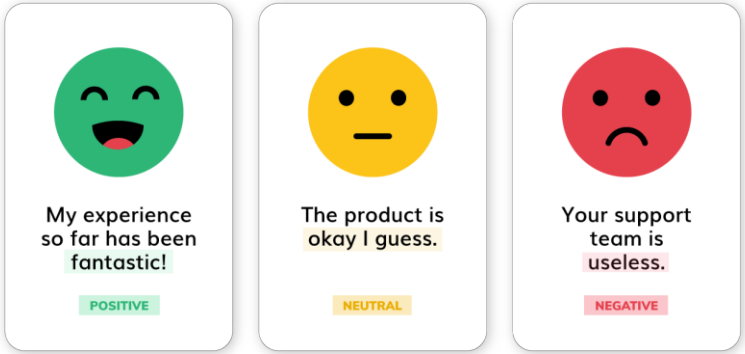
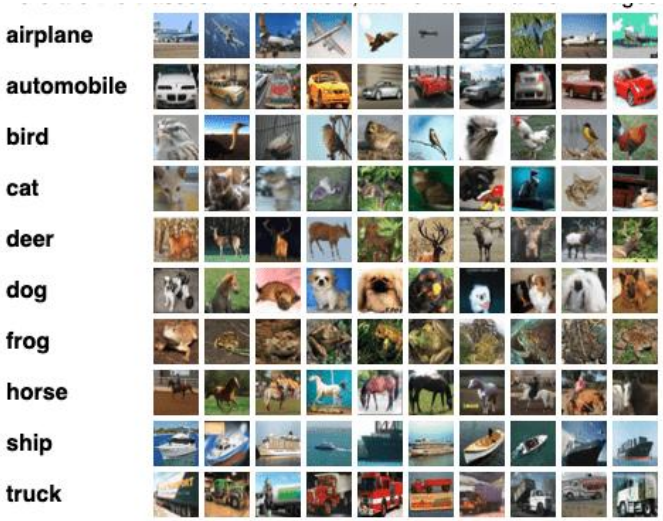
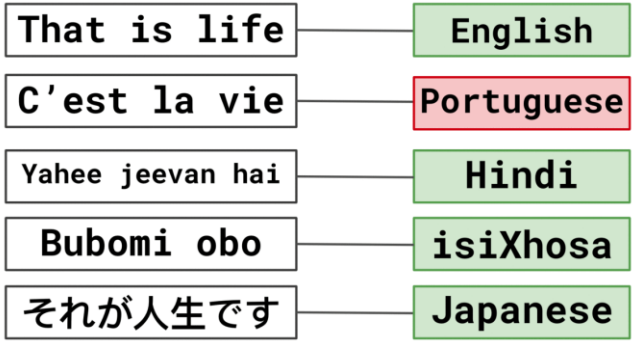


Image Classification





Language Identification

(Linear regression included)

Polynomial Regression for Classification?

Polynomial Regression for Classification?

➢ Binary Classification

Input: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$

Output: $f_{\mathbf{w}}(\mathbf{x}) = f_{\mathbf{w}}(\mathbf{p}(\mathbf{x})) = \text{sign}(\mathbf{p}^T\mathbf{w})$

Threshold function

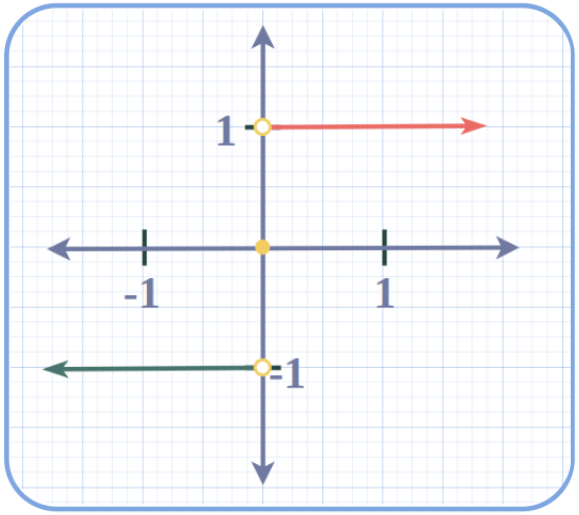
Negative class

Positive class

$y_i \in \{-1, +1\}$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

Graph of Signum Function



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

-Learning: $\mathbf{w}^* = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T\mathbf{y}$ still the same.

-Prediction: $\mathbf{f}_{\mathbf{w}^*}(\mathbf{P}(\mathbf{X}_t)) = \text{sign}\,(\mathbf{P}_t\mathbf{w}^*)$

$(\mathbf{p}_i^t)^T\mathbf{w}^* = \mathbf{0}$? input data = 0?

$$= \begin{bmatrix} \text{sign}((\mathbf{p}_1^t)^T\mathbf{w}^*) \\ \vdots \\ \text{sign}((\mathbf{p}_m^t)^T\mathbf{w}^*) \end{bmatrix}$$

$$\text{sign}(z) = \begin{cases} -1, z < 0 \\ 0, z = 0 \\ +1, z > 0 \end{cases}$$

Polynomial Regression for Classification?

➢ Binary Classification

Input: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$

Output: $f_{\mathbf{w}}(\mathbf{x}) = f_{\mathbf{w}}(\mathbf{p}(\mathbf{x})) = \text{sign}(\mathbf{p}^T\mathbf{w})$

Negative class / Positive class

if we relax → if the relax

$y_i \in \{-1, +1\}$ $\quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$

- **Decision Boundary: $\mathbf{p}^T\mathbf{w} = 0$**

  (a line or a surface separates the two classes)

depends on the threshold function

$$\text{sign}(z) = \begin{cases} -1, z < 0 \\ 0, z = 0 \\ +1, z > 0 \end{cases}$$

What if threshold function

$$th(z) = \begin{cases} -1, z < 0.3 \\ 0, z = 0.3 \\ +1, z > 0.3 \end{cases}$$

$p^T w = 0.3$ is the decision boundary

Polynomial Regression for Classification?

➢ Multi-category Classification

Input: $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_N, \mathbf{y}_N)$

$\mathbf{y}_i$ is *K*-Dimensional one-hot encoded vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

e.g., *K*=3 categories: dog, cat, rabbit.

One-hot encoding

dog=[1,0,0], cat=[0,1,0], rabbit=[0,0,1]

Output: $f_{\mathbf{W}}(\mathbf{x}) = f_{\mathbf{W}}(\mathbf{p}(\mathbf{x})) = \underset{k=1,\ldots,K}{\text{argmax}}(\mathbf{p}^T\mathbf{W})$

look at which k gave the largest value. e.g. [0.1, −0.4, 3] → class 3

How does decision boundary look like?

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

-Learning: $\mathbf{W}^* = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T\mathbf{Y}$

-Prediction: $\mathbf{f}_{\mathbf{W}^*}(\mathbf{P}(\mathbf{X}_t)) = \begin{bmatrix} \underset{k=1,\ldots,K}{\text{argmax}}((\mathbf{p}_1^t)^T\mathbf{W}^*) \\ \vdots \\ \underset{k=1,\ldots,K}{\text{argmax}}((\mathbf{p}_m^t)^T\mathbf{W}^*) \end{bmatrix}$

Polynomial Regression for Classification?

➢ Multi-category Classification

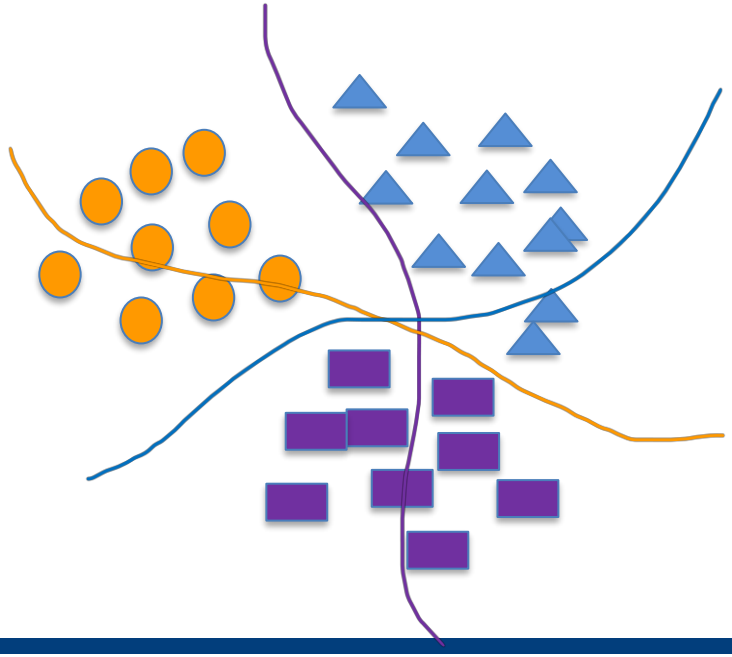
Input: $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_N, \mathbf{y}_N)$

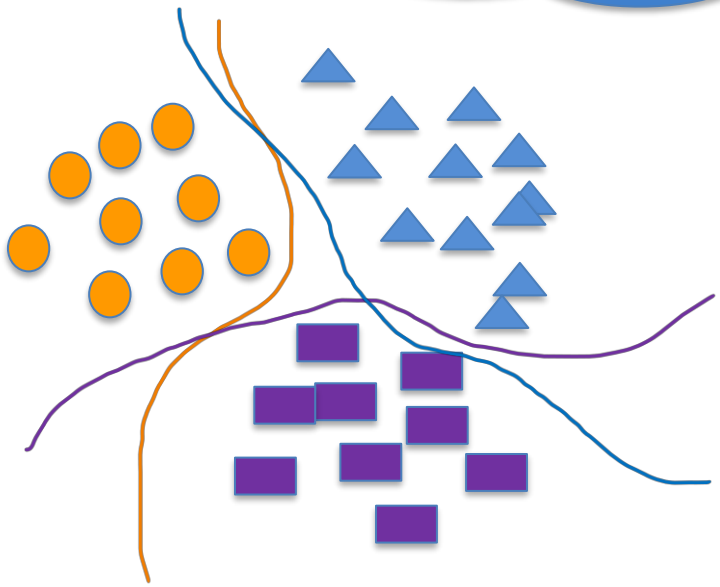
$\mathbf{y}_i$ is *K*-Dimensional one-hot encoded vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

Output: $f_{\mathbf{W}}(\mathbf{x}) = f_{\mathbf{W}}(\mathbf{p}(\mathbf{x})) = \underset{k=1,\ldots,K}{\text{argmax}}(\mathbf{p}^T\mathbf{W})$

How does decision boundary look like?





Polynomial Regression for Classification?

➢ Multi-category Classification

Example: Given a training set $\{(x_i, y_i)\}_{i=1}^{5} = \{(-1, class1), (0, class1), (0.5, class2), (0.3, class3), (0.8, class2)\}$, predict the class label for the test dataset $\{-0.1, 0.4\}$ using 1st order polynomial regression.

2

Is linear regression

Polynomial Regression for Classification?

➢ Multi-category Classification

Example: Given a training set $\{(x_i, y_i)\}_{i=1}^{5} = \{(-1, class1), (0, class1), (0.5, class2), (0.3, class3), (0.8, class2)\}$, predict the class label for the test dataset $\{-0.1, 0.4\}$ using 1st order polynomial regression.

Step 0: Polynomial Transformation ($\mathbf{X} \rightarrow \mathbf{P}$) and One-hot Encoding

1st order polynomial model:

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 = [1 \quad x_1] \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 0.5 \\ 1 & 0.3 \\ 1 & 0.8 \end{bmatrix}$$

$class1: [1,0,0]$
$class2: [0,1,0]$
$class3: [0,0,1]$

$$\mathbf{Y} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Polynomial Regression for Classification?

➢ Multi-category Classification

Example: Given a training set $\{(x_i, y_i)\}_{i=1}^{5} = \{(-1, class1), (0, class1), (0.5, class2), (0.3, class3), (0.8, class2)\}$, predict the class label for the test dataset $\{-0.1, 0.4\}$ using 1st order polynomial regression.

Step 1: Learning (work out $\mathbf{W}^*$)

Since $\mathbf{P}^T\mathbf{P}$ is invertible, $\mathbf{W}^* = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T\mathbf{Y} = \begin{bmatrix} 0.4780 & 0.3333 & 0.1887 \\ -0.6499 & 0.5556 & 0.0943 \end{bmatrix}$

Step 2: Prediction

$$\mathbf{P}_t = \begin{bmatrix} 1 & x_{1,1}^t \\ 1 & x_{2,1}^t \end{bmatrix} = \begin{bmatrix} 1 & -0.1 \\ 1 & 0.4 \end{bmatrix}$$

see the highest value to know which class.

$$\mathbf{Y}_t^* = \mathbf{P}_t\mathbf{W}^* = \begin{bmatrix} 1 & -0.1 \\ 1 & 0.4 \end{bmatrix}\begin{bmatrix} 0.4780 & 0.3333 & 0.1887 \\ -0.6499 & 0.5556 & 0.0943 \end{bmatrix}$$

$$= \begin{bmatrix} 0.5430 & 0.2778 & 0.1792 \\ 0.2180 & 0.5556 & 0.2264 \end{bmatrix} \xrightarrow{\text{argmax}} \begin{bmatrix} 1 \\ 2 \end{bmatrix} \rightarrow \text{class1}, \text{class2}$$

Polynomial Regression for Classification?

➢ Not Recommended for Classification

1. Hard Threshold Function, e.g.,"sign()"

   Completely confident prediction

$$\mathbf{P}_t\mathbf{w}^* = \begin{bmatrix} -5 \\ 10 \\ 0.01 \end{bmatrix} \xrightarrow{\text{sign()}} \hat{\mathbf{y}}_t = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}$$

Near decision boundary

2. $\mathbf{p}^T\mathbf{w}$ is a number between $-\infty$ and $+\infty$.

   The regression model output values are not interpretable

$$\mathbf{P}_t\mathbf{W}^* = \begin{bmatrix} -4 & 4.2 & 2.5 \\ 1 & -0.8 & 0.2 \end{bmatrix}$$

(With **soft** threshold function)

Logistic Regression

-Used for classification task, not regression task

Binomial Logistic Regression

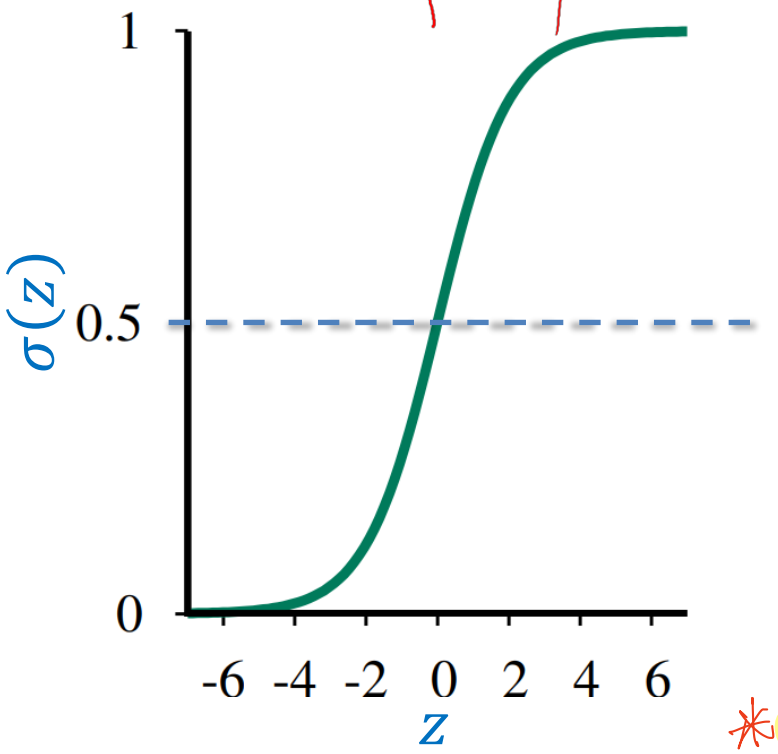
➢ Model (Binary)

Input: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)$, $y_i \in \{0, 1\}$ (Negative class / Positive class)

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}$$

Output: $f_{\mathbf{w}}(\mathbf{x}) = \sigma(\mathbf{x}^T\mathbf{w})$, where $\sigma(z) = \frac{1}{1+e^{-z}}$

$z = x^T w$

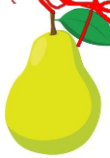


Sigmoid (logistic) function

- **Value range: (0,1)** ➡ **Probability**

asymptotic, cannot exactly 0 or 1

$P(y=1 \mid \mathbf{x})$

e.g.,  VS 

probability of x belong to class 1.

Label (y_i): 0 1

probability of a pear is 0.2 ⇒ higher probability of it being apple

If $f_{\mathbf{w}}(\mathbf{x}_{new}) = 0.2$, $P(? \mid ?) = 0.2$

If $f_{\mathbf{w}}(\mathbf{x}_{new}) = 0.6$, $P(? \mid ?) = ?$

$y = 1$ (True)

* cannot assert, probabilistic. depends on z threshold → most likely a pear.

Binomial Logistic Regression

➢ Learning (Binary)

Find the optimal values for $\mathbf{w}^*$ which minimizes:

$$J(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} Loss(f_{\mathbf{w}}(\mathbf{x}_i), y_i)$$

new cost function.

recall from EE2211 how entropy is calculated (using log/ln), and it represents how pure/impure a prediction is (how many in right class, how many in wrong class)

where $Loss(f_{\mathbf{w}}(\mathbf{x}_i), y_i) = -y_i \ln\left(f_{\mathbf{w}}(\mathbf{x}_i)\right) - (1 - y_i)\ln(1 - f_{\mathbf{w}}(\mathbf{x}_i))$

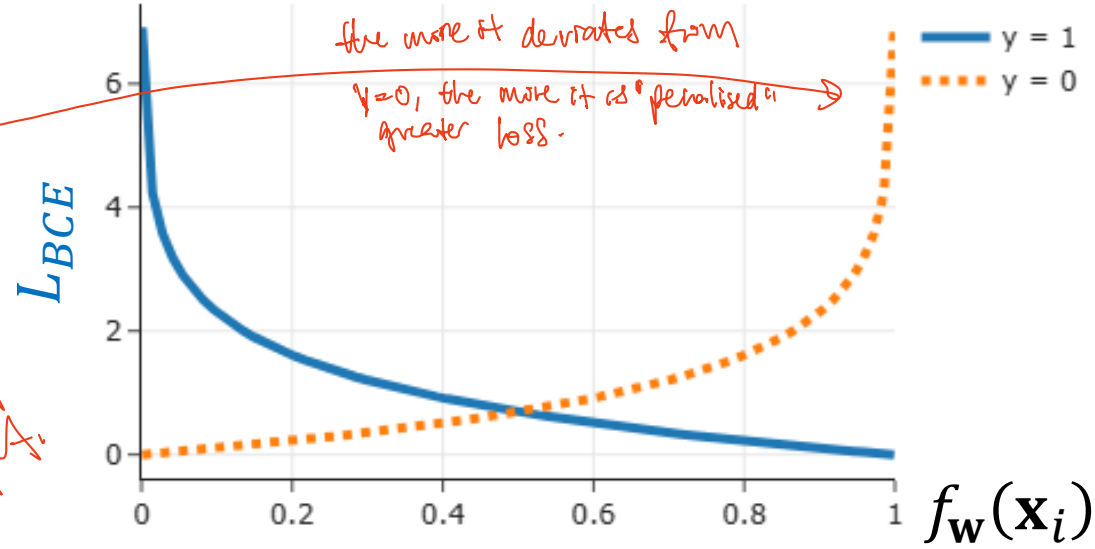
Binary Cross-Entropy Loss

$$L_{BCE}(f_{\mathbf{w}}(\mathbf{x}_i), y_i) = \begin{cases} -\ln\left(1 - f_{\mathbf{w}}(\mathbf{x}_i)\right), & \text{if } y_i = 0 \\ -\ln\left(f_{\mathbf{w}}(\mathbf{x}_i)\right), & \text{if } y_i = 1 \end{cases}$$

gives this

gives this

y_i = target output



Binomial Logistic Regression

➢ Learning (Binary)

$$\underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{N} L_{BCE}(f_{\mathbf{w}}(\mathbf{x}_i), y_i)$$

$$\underset{\mathbf{w}}{\operatorname{argmax}}\, F = \underset{\mathbf{w}}{\operatorname{argmin}}\, -F$$

$$\ln\left(\prod_{i=1}^{N} a_i\right) = \sum_{i=1}^{N} \ln(a_i)$$

Maximum Likelihood Estimation

$$\underset{\mathbf{w}}{\operatorname{argmax}} \ln\left(\prod_{i=1}^{N} f_{\mathbf{w}}(\mathbf{x}_i)^{y_i}(1 - f_{\mathbf{w}}(\mathbf{x}_i))^{(1-y_i)}\right)$$

if $y_i = 0$, $1 - f_{\mathbf{w}}(\mathbf{x}_i)$ belongs-to $P(y_i = 0 \mid \mathbf{x}_i)$

if $y_i = 1$, $f_{\mathbf{w}}(\mathbf{x}_i)$ belongs-to $P(y_i = 1 \mid \mathbf{x}_i)$

(Assumption: data samples are **independent**.)

(Assumption: data samples are **identically distributed**.)

estimated w by maximization over the ln of likelihood of training data to probability of training data belonging to their corresponding target classes.

Binomial Logistic Regression

➢ Learning (Binary)

polynomial regression can just use matrix multiplication to train the weights, but logistic regression need to use gradient descent because we cannot solve the derivative (equate to 0 to get min), ie. no closed form solution. So we need to iteratively move towards the min using steps of the complex gradient function

$$\underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} L_{BCE}(f_{\mathbf{w}}(\mathbf{x}_i), y_i)$$

no closed-form solution!

If you take the derivative and set it to zero, you get:

$$X^\top(\sigma(X\mathbf{w}) - y) = 0$$

This equation is:

✕ **nonlinear in $\mathbf{w}$**

because $\sigma(\cdot)$ wraps a non-linear sigmoid around $X\mathbf{w}$.

There is **no algebraic way to isolate $\mathbf{w}$**.
No matrix inverse trick works here.

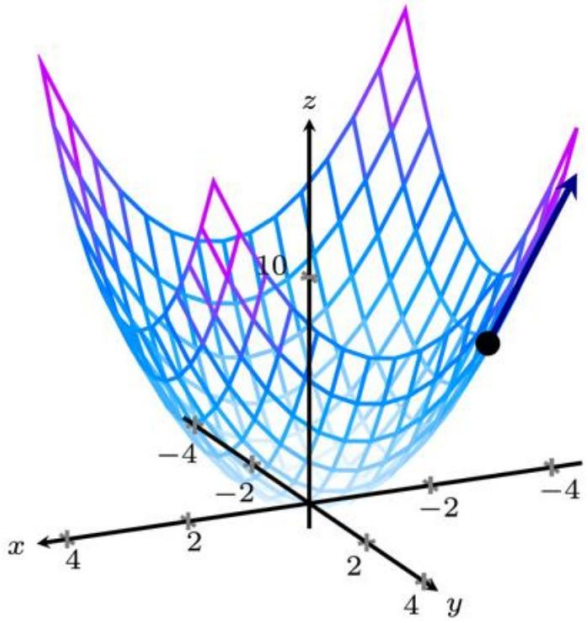
Recall: **Gradient Descent Algorithm!**

Goal: $\underset{\mathbf{w}}{\operatorname{argmin}} J(\mathbf{w})$

Initialize $\mathbf{w}_0$ and learning rate η
while *true* **do**
  $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k - \eta \nabla_{\mathbf{w}} J(\mathbf{w}_k)$
  if *converge* **then**
    return $\mathbf{w}_{k+1}$
  end
end

learning rate.

differentiate again?



Binomial Logistic Regression

➢ Learning (Binary)

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \nabla_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} L_{BCE}(f_{\mathbf{w}}(\mathbf{x}_i), y_i)$$

$$= \frac{1}{N} \sum_{i=1}^{N} \nabla_{\mathbf{w}} L_{BCE}(f_{\mathbf{w}}(\mathbf{x}_i), y_i)$$

$$= \frac{1}{N} \sum_{i=1}^{N} \nabla_{\mathbf{w}} \left(-y_i \ln\left(f_{\mathbf{w}}(\mathbf{x}_i)\right) - (1 - y_i) \ln\left(1 - f_{\mathbf{w}}(\mathbf{x}_i)\right)\right)$$

$$= -\frac{1}{N} \sum_{i=1}^{N} \left(y_i \nabla_{\mathbf{w}} \ln\left(f_{\mathbf{w}}(\mathbf{x}_i)\right) + (1 - y_i) \nabla_{\mathbf{w}} \ln\left(1 - f_{\mathbf{w}}(\mathbf{x}_i)\right)\right)$$

By chain rule:

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

$$\frac{d \ln\left(f_{\mathbf{w}}(\mathbf{x}_i)\right)}{d f_{\mathbf{w}}(\mathbf{x}_i)} \frac{d f_{\mathbf{w}}(\mathbf{x}_i)}{d \mathbf{w}}$$

$$\frac{1}{f_{\mathbf{w}}(\mathbf{x}_i)} \nabla_{\mathbf{w}} f_{\mathbf{w}}(\mathbf{x}_i)$$

$$-\frac{1}{1 - f_{\mathbf{w}}(\mathbf{x}_i)} \nabla_{\mathbf{w}} f_{\mathbf{w}}(\mathbf{x}_i)$$

$$= -\frac{1}{N} \sum_{i=1}^{N} \left(\frac{y_i}{f_{\mathbf{w}}(\mathbf{x}_i)} \nabla_{\mathbf{w}} f_{\mathbf{w}}(\mathbf{x}_i) - \frac{1 - y_i}{1 - f_{\mathbf{w}}(\mathbf{x}_i)} \nabla_{\mathbf{w}} f_{\mathbf{w}}(\mathbf{x}_i)\right)$$

Continue on next page...

Binomial Logistic Regression

➢ Learning (Binary)

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \nabla_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} L_{BCE}(f_{\mathbf{w}}(\mathbf{x}_i), y_i)$$

$$= -\frac{1}{N} \sum_{i=1}^{N} \left(\frac{y_i}{f_{\mathbf{w}}(\mathbf{x}_i)} \nabla_{\mathbf{w}} f_{\mathbf{w}}(\mathbf{x}_i) - \frac{1-y_i}{1-f_{\mathbf{w}}(\mathbf{x}_i)} \nabla_{\mathbf{w}} f_{\mathbf{w}}(\mathbf{x}_i) \right)$$

left with this

$$\nabla_{\mathbf{w}} f_{\mathbf{w}}(\mathbf{x}_i) = \nabla_{\mathbf{w}} \sigma\left(\mathbf{x}_i^T \mathbf{w}\right)$$

$$= \sigma'(z_i)\mathbf{x}_i$$

By chain rule and $\frac{d(\mathbf{b}^T\mathbf{w})}{d\mathbf{w}} = \mathbf{b}$

$$= \sigma(z_i)(1 - \sigma(z_i))\mathbf{x}_i$$

$$= f_{\mathbf{w}}(\mathbf{x}_i)(1 - f_{\mathbf{w}}(\mathbf{x}_i))\mathbf{x}_i$$

once again we matrix-ify to condense the sum of portion.

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & \cdots & x_{N,d} \end{bmatrix}$$

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i (f_{\mathbf{w}}(\mathbf{x}_i) - y_i) = \frac{1}{N} \mathbf{X}^T (\mathbf{f}_{\mathbf{w}}(\mathbf{X}) - \mathbf{y})$$

Binomial Logistic Regression

Proof.

$$\frac{\partial \sigma(a)}{\partial a} = \frac{\partial}{\partial a}\left(\frac{1}{1+\exp(-\beta a)}\right)$$

$$= -\frac{1}{(1+e^{-\beta a})^2}\frac{\partial(1+e^{-\beta a})}{\partial a}$$

$$= \frac{\beta}{(1+e^{-\beta a})^2}e^{-\beta a}$$

$$= \frac{\beta}{(1+e^{-\beta a})^2}(1+e^{-\beta a}-1)$$

$$= \beta\left(\frac{1}{1+e^{-\beta a}} - \frac{1}{(1+e^{-\beta a})^2}\right)$$

$$= \beta\left(\sigma(a) - \sigma^2(a)\right)$$

$$= \beta\sigma(a)(1-\sigma(a))$$

Take $\beta = 1$ for our case

Binomial Logistic Regression

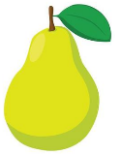
➢ Prediction (Binary)

How can we gradient descent to update w each iterating

$$\mathbf{f}_{\mathbf{w}^*}(\mathbf{X}_t) = \sigma(\mathbf{X}_t\mathbf{w}^*) = \sigma\left(\begin{bmatrix} 1 & x_{1,1}^t & \cdots & x_{1,d}^t \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m,1}^t & \cdots & x_{m,d}^t \end{bmatrix}\begin{bmatrix} w_0^* \\ \vdots \\ w_d^* \end{bmatrix}\right) = \begin{bmatrix} \sigma((\mathbf{x}_1^t)^T\mathbf{w}^*) \\ \vdots \\ \sigma((\mathbf{x}_m^t)^T\mathbf{w}^*) \end{bmatrix}$$

still need this

Each entry is the probability of each test sample belonging to the binary label 1.

e.g.,  VS 

Label: 0 1

To predict class label, set a threshold.

$f_{\mathbf{w}}(\mathbf{x}_{new}) = 0.2$

Probability of $\mathbf{x}_{new}$ belonging to pear class (label 1) is 0.2

$f_{\mathbf{w}}(\mathbf{x}_{new}) = 0.6$

Probability of $\mathbf{x}_{new}$ belonging to pear class (label 1) is 0.6

Binomial Logistic Regression

Example: Given a training set $\{(x_i, y_i)\}_{i=1}^{5} = \{(-1, class1), (0, class1), (0.5, class2), (0.3, class1), (0.8, class2)\}$, predict the probability of {x=-0.1} belonging to *class* 1 using binomial logistic regression.

Binomial Logistic Regression

Example: Given a training set $\{(x_i, y_i)\}_{i=1}^{5} = \{(-1, class1), (0, class1), (0.5, class2), (0.3, class1), (0.8, class2)\}$, predict the probability of {x=-0.1} belonging to *class* 1 using binomial logistic regression.

Step 0: Prepare the training data

Binary encoding of the target output

$class1: 1$ $class2: 0$

$$\mathbf{X} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 0.5 \\ 1 & 0.3 \\ 1 & 0.8 \end{bmatrix}, \mathbf{y} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

up to us what to assign. but since we wants to class 1, we put label 1 to class 1. output will directly be the probability of $x = -0.1$ belongs to class 1

Binomial Logistic Regression

Example: Given a training set $\{(x_i, y_i)\}_{i=1}^{5} = \{(-1, class1), (0, class1), (0.5, class2), (0.3, class1), (0.8, class2)\}$, predict the probability of {x=-0.1} belonging to class 1.

Step 1: Learning (work out $\mathbf{w}^*$)

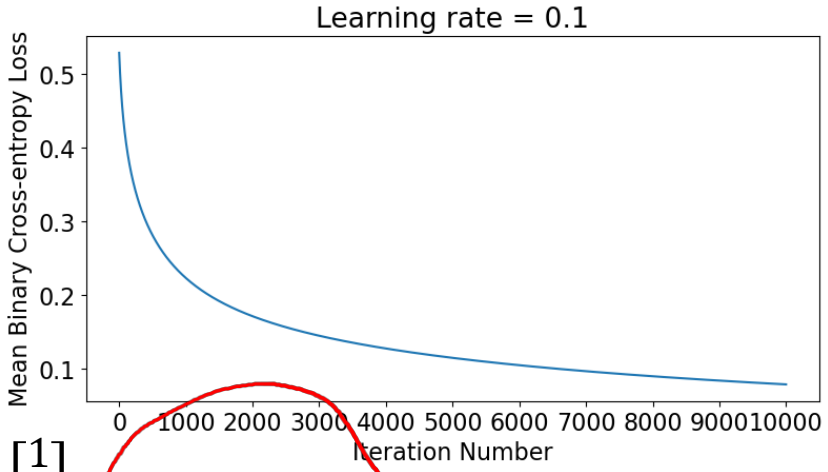
$$\mathbf{w}_{10000} = \begin{bmatrix} 6.0786 \\ -15.3051 \end{bmatrix}$$

Gradient Descent:

-Gradient $\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{N}\mathbf{X}^T(\mathbf{f}_{\mathbf{w}}(\mathbf{X}) - \mathbf{y})$

-Initialize $\mathbf{w}_0 = \begin{bmatrix} 0.1 \\ -1 \end{bmatrix}$, learning rate $\eta = 0.1$

-At each iteration, $\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \nabla_{\mathbf{w}} J(\mathbf{w}_k)$



$$\mathbf{w}_0 = \begin{bmatrix} 0.1 \\ -1 \end{bmatrix}$$

$$\nabla_{\mathbf{w}} J(\mathbf{w}_0) = \frac{1}{5}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -1 & 0 & 0.5 & 0.3 & 0.8 \end{bmatrix}\left(\begin{bmatrix} \frac{1}{1+e^{-\mathbf{x}_1^T\mathbf{w}_0}} \\ \vdots \\ \frac{1}{1+e^{-\mathbf{x}_5^T\mathbf{w}_0}} \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} -0.1083 \\ 0.1102 \end{bmatrix}$$

$$\mathbf{w}_1 = \mathbf{w}_0 - \eta \nabla_{\mathbf{w}} J(\mathbf{w}_0) = \begin{bmatrix} 0.1108 \\ -1.0110 \end{bmatrix}$$

$$\vdots$$

Binomial Logistic Regression

Example: Given a training set $\{(x_i, y_i)\}_{i=1}^{5} = \{(-1, class1), (0, class1), (0.5, class2), (0.3, class1), (0.8, class2)\}$, predict the probability of {x=-0.1} belonging to class 1.

Step 2: Prediction

Augment 1 for the feature vectors

$$\mathbf{x}_t = \begin{bmatrix} 1 \\ -0.1 \end{bmatrix}$$

after 10000 iterations

$$\mathbf{w}_{10000} = \begin{bmatrix} 6.0786 \\ -15.3051 \end{bmatrix}$$

$$y_t^* = \sigma(\mathbf{x}_t^T \mathbf{w}^*)$$

$$= \sigma\left(\begin{bmatrix} 1 & -0.1 \end{bmatrix} \begin{bmatrix} 6.0786 \\ -15.3051 \end{bmatrix}\right)$$

$$= \frac{1}{1 + e^{-7.6091}}$$

$$= 0.9995$$

probability of class 1.

NEED TO KNOW THRESHOLD TO CLASSIFY, or else it is just __% confident that test data belongs to class ___

If the threshold is set to 0.5,

$$i.e., \sigma(z) = \frac{1}{1 + e^{-z}} \begin{cases} > 0.5, class1 \\ < 0.5, class2 \end{cases}$$

$e^{-z} = 1 \Rightarrow z = \mathbf{x}^T\mathbf{w} = 0$

Then, $\frac{1}{1+e^{-(\mathbf{x}^T\mathbf{w}^*)}} = 0.5$ is the decision boundary.

$$\mathbf{x}^T\mathbf{w}^* = 0$$

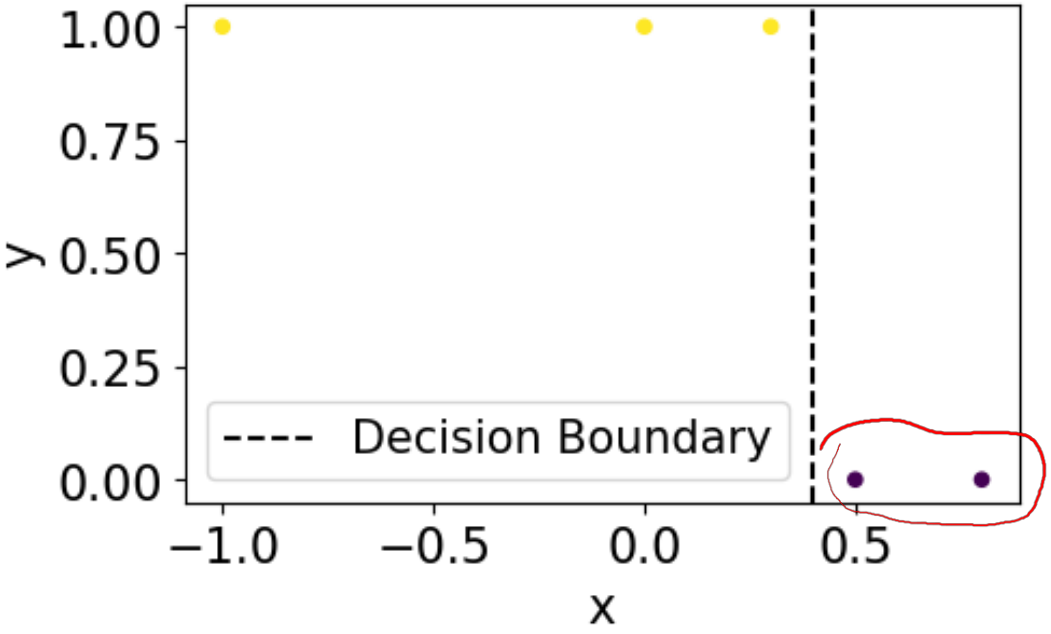
$$\begin{bmatrix} 1 & x_1 \end{bmatrix} \begin{bmatrix} 6.0786 \\ -15.3051 \end{bmatrix} = 0$$

decision boundary

$$x_1 = 0.3971$$

Binomial Logistic Regression

Example: Given a training set $\{(x_i, y_i)\}_{i=1}^{5} = \{(-1, class1), (0, class1), (0.5, class2), (0.3, class1), (0.8, class2)\}$, predict the probability of {x=-0.1} belonging to class 1.



If the threshold is set to 0.5,

$$i.e., \sigma(z) = \frac{1}{1+e^{-z}} \begin{cases} > 0.5, class1 \\ < 0.5, class2 \end{cases}$$

Then, $\frac{1}{1+e^{-(\mathbf{x}^T\mathbf{w}^*)}} = 0.5$ is the decision boundary.

$$\mathbf{x}^T\mathbf{w}^* = 0$$

$$\begin{bmatrix} 1 & x_1 \end{bmatrix} \begin{bmatrix} 6.0786 \\ -15.3051 \end{bmatrix} = 0$$

$$x_1 = 0.3971$$

Multinomial Logistic Regression

➢ Model (Multi-class)

Input: $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_N, \mathbf{y}_N)$,

Target output $\mathbf{y}_i$ is one-hot encoded vector

$$\text{Output: } \mathbf{f_W}(\mathbf{x}) = \boldsymbol{\sigma}(\mathbf{x}^T\mathbf{W}) = \boldsymbol{\sigma}([1 \quad x_1 \quad \cdots \quad x_d] \begin{bmatrix} w_{0,1} & \cdots & w_{0,K} \\ w_{1,1} & \cdots & w_{1,K} \\ \vdots & \ddots & \vdots \\ w_{d,1} & \cdots & w_{d,K} \end{bmatrix})$$

K classes

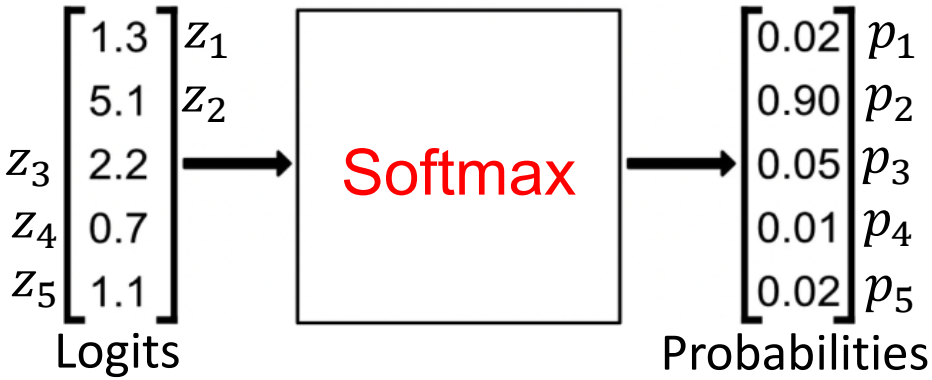
$$= \boldsymbol{\sigma}([\mathbf{x}^T\mathbf{w}_1 \quad \cdots \quad \mathbf{x}^T\mathbf{w}_K])$$

$$= \boldsymbol{\sigma}([z_1 \quad \cdots \quad z_K])$$

$$= \left[\frac{e^{z_1}}{\sum_{j=1}^{K} e^{z_j}} \quad \cdots \quad \frac{e^{z_K}}{\sum_{j=1}^{K} e^{z_j}}\right]$$

where $\boldsymbol{\sigma}(\boldsymbol{z})_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}, k = 1,2, \ldots, K$

is **Softmax** Function



Multinomial Logistic Regression

Learning (Multi-class)

Find the optimal values for $\mathbf{W}^*$ which minimizes:

$$J(\mathbf{W}) = \frac{1}{N}\sum_{i=1}^{N} L_{CCE}(\mathbf{f_W}(\mathbf{x}_i), \mathbf{y}_i)$$

where $L_{CCE}(\mathbf{f_W}(\mathbf{x}_i), \mathbf{y}_i) = -\sum_{k=1}^{K} y_{i,k} \ln(p_{i,k})$.

Categorical cross-entropy loss

$p_{i,k}$: k-th element of output vector $\mathbf{f_W}(\mathbf{x}_i)$

$y_{i,k}$: k-th element of target vector $\mathbf{y}_i$

$$L_{CCE}(\mathbf{f_W}(\mathbf{x}_i), \mathbf{y}_i) = -\ln(p_{i,c}), \text{where } y_{i,c} = 1$$

Only the target/ground-truth class contributes

$\mathbf{x}_i$ from class c, $\mathbf{y}_i = [0, \ldots, 0, 1, 0, \ldots 0]$

$y_{i,1}$ $y_{i,c}$ $y_{i,K}$

Multinomial Logistic Regression

➢ Learning (Multi-class)

$$\underset{\mathbf{W}}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N} L_{CCE}(\mathbf{f}_{\mathbf{W}}(\mathbf{x}_i), \mathbf{y}_i)$$

no closed-form solution!

the sigmoid function makes the loss non linear, derivative leads to equations that cannot be solved algebraically, so cannot equate to zero to get min,

we thus use gradient descent which iteratively goes towards minimum. Still convex

Use Gradient Descent Algorithm!

$$\frac{dJ(\mathbf{W})}{d\mathbf{W}} = \frac{d}{d\mathbf{W}}\left(\frac{1}{N}\sum_{i=1}^{N} L_{CCE}(\mathbf{f}_{\mathbf{W}}(\mathbf{x}_i), \mathbf{y}_i)\right) = \frac{1}{N}\sum_{i=1}^{N} \frac{d}{d\mathbf{W}} L_{CCE}(\mathbf{f}_{\mathbf{W}}(\mathbf{x}_i), \mathbf{y}_i)$$

$$= \frac{1}{N}\sum_{i=1}^{N} \begin{bmatrix} \frac{\partial L_{CCE}(\mathbf{f}_{\mathbf{W}}(\mathbf{x}_i), \mathbf{y}_i)}{\partial w_{0,1}} & \cdots & \frac{\partial L_{CCE}(\mathbf{f}_{\mathbf{W}}(\mathbf{x}_i), \mathbf{y}_i)}{\partial w_{0,K}} \\ \frac{\partial L_{CCE}(\mathbf{f}_{\mathbf{W}}(\mathbf{x}_i), \mathbf{y}_i)}{\partial w_{1,1}} & \cdots & \frac{\partial L_{CCE}(\mathbf{f}_{\mathbf{W}}(\mathbf{x}_i), \mathbf{y}_i)}{\partial w_{1,K}} \\ \vdots & \ddots & \vdots \\ \frac{\partial L_{CCE}(\mathbf{f}_{\mathbf{W}}(\mathbf{x}_i), \mathbf{y}_i)}{\partial w_{d,1}} & \cdots & \frac{\partial L_{CCE}(\mathbf{f}_{\mathbf{W}}(\mathbf{x}_i), \mathbf{y}_i)}{\partial w_{d,K}} \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} w_{0,1} & \cdots & w_{0,K} \\ w_{1,1} & \cdots & w_{1,K} \\ \vdots & \ddots & \vdots \\ w_{d,1} & \cdots & w_{d,K} \end{bmatrix}$$

Need to work out one and the rest are the same.

Continue on next page…

Multinomial Logistic Regression

➢ Learning (Multi-class)

❖ $\frac{\partial L_{CCE}(\mathbf{f_w}(\mathbf{x}_i),\mathbf{y}_i)}{\partial w_{q,g}} = -\frac{\partial \sum_{k=1}^{K} y_{i,k} \ln(p_{i,k})}{\partial w_{q,g}}$

$p_{i,k} = \boldsymbol{\sigma}(\mathbf{z}_i)_k = \frac{e^{z_{i,k}}}{\sum_{j=1}^{K} e^{z_{i,j}}}$

$$= -\sum_{k=1}^{K} y_{i,k} \frac{\partial \ln(p_{i,k})}{\partial w_{q,g}}$$

By chain rule

$$= -\sum_{k=1}^{K} \frac{y_{i,k}}{p_{i,k}} \frac{\partial}{\partial w_{q,g}} \left(\frac{e^{z_{i,k}}}{\sum_{j=1}^{K} e^{z_{i,j}}}\right)$$

By chain rule

$$= -\sum_{k=1}^{K} \frac{y_{i,k}}{p_{i,k}} \frac{\partial}{\partial z_{i,g}} \left(\frac{e^{z_{i,k}}}{\sum_{j=1}^{K} e^{z_{i,j}}}\right) \frac{\partial z_{i,g}}{\partial w_{q,g}}$$

Continue on next page…

Multinomial Logistic Regression

➢ Learning (Multi-class)

- $\frac{\partial}{\partial z_{i,g}}\left(\frac{e^{z_{i,k}}}{\sum_{j=1}^{K} e^{z_{i,j}}}\right)$ $\begin{cases} k = g: e^{z_{i,g}} \\ k \neq g: 0 \end{cases}$ or else constants.

By quotient rule:

$$\frac{d}{dx}\left[\frac{f(x)}{g(x)}\right] = \frac{g(x)f'(x) - f(x)g'(x)}{(g(x))^2}$$

$$= \frac{\left(\sum_{j=1}^{K} e^{z_{i,j}}\right)\frac{\partial e^{z_{i,k}}}{\partial z_{i,g}} - e^{z_{i,k}}\frac{\partial\left(\sum_{j=1}^{K} e^{z_{i,j}}\right)}{\partial z_{i,g}}}{\left(\sum_{j=1}^{K} e^{z_{i,j}}\right)^2} \qquad e^{z_{i,g}}$$

$$= \frac{\left(\sum_{j=1}^{K} e^{z_{i,j}}\right)\delta_{kg} e^{z_{i,g}} - e^{z_{i,k}} e^{z_{i,g}}}{\left(\sum_{j=1}^{K} e^{z_{i,j}}\right)^2}$$

$$\delta_{kg} = \begin{cases} 1, if\ k = g \\ 0. if k \neq g \end{cases}$$

$$= \frac{\delta_{kg} e^{z_{i,g}}}{\sum_{j=1}^{K} e^{z_{i,j}}} - \frac{e^{z_{i,k}}}{\sum_{j=1}^{K} e^{z_{i,j}}} \frac{e^{z_{i,g}}}{\sum_{j=1}^{K} e^{z_{i,j}}}$$

$$= \delta_{kg} p_{i,g} - p_{i,k} p_{i,g}$$

$p_{i,g}$: output of softmax function

$$= (\delta_{kg} - p_{i,k}) p_{i,g}$$

Multinomial Logistic Regression

➢ Learning (Multi-class)

❖ $$\frac{\partial L_{CCE}(\mathbf{f_W}(\mathbf{x}_i),\mathbf{y}_i)}{\partial w_{q,g}} = -\sum_{k=1}^{K} \frac{y_{i,k}}{p_{i,k}} \frac{\partial}{\partial z_{i,g}} \left(\frac{e^{z_{i,k}}}{\sum_{j=1}^{K} e^{z_{i,j}}}\right) \frac{\partial z_{i,g}}{\partial w_{q,g}}$$

$$\frac{\partial z_{i,g}}{\partial w_{q,g}} = \frac{\partial (\mathbf{x}_i{}^T \mathbf{w}_g)}{\partial w_{q,g}}$$

$$= \frac{\partial}{\partial w_{q,g}} (x_{i,0} w_{0,g} + \cdots + x_{i,d} w_{d,g})$$

$$= x_{i,q}$$

$$\mathbf{W} = \begin{bmatrix} w_{0,1} & \cdots & w_{0,K} \\ w_{1,1} & \cdots & w_{1,K} \\ \vdots & \ddots & \vdots \\ w_{d,1} & \cdots & w_{d,K} \end{bmatrix}$$

$$= [\mathbf{w}_1 \quad \cdots \quad \mathbf{w}_K]$$

Multinomial Logistic Regression

➢ Learning (Multi-class)

❖ $$\frac{\partial L_{CCE}(\mathbf{f_W}(\mathbf{x}_i),\mathbf{y}_i)}{\partial w_{q,g}} = -\sum_{k=1}^{K} \frac{y_{i,k}}{p_{i,k}} \frac{\partial}{\partial z_{i,g}} \left(\frac{e^{z_{i,k}}}{\sum_{j=1}^{K} e^{z_{i,j}}}\right) \frac{\partial z_{i,g}}{\partial w_{q,g}}$$

$$= -\sum_{k=1}^{K} \frac{y_{i,k}}{p_{i,k}} (\delta_{kg} - p_{i,k}) p_{i,g} x_{i,q}$$

$$\delta_{kg} = \begin{cases} 1, if\ k = g \\ 0. if k \neq g \end{cases}$$

$$= p_{i,g} x_{i,q} \sum_{k=1}^{K} \frac{y_{i,k}}{p_{i,k}} (p_{i,k} - \delta_{kg})$$

only one term are retained. sum of all entries of one-hot or just one,

$$= p_{i,g} x_{i,q} \left(\sum_{k=1}^{K} y_{i,k} - \frac{y_{i,g}}{p_{i,g}}\right)$$

$$= p_{i,g} x_{i,q} \left(1 - \frac{y_{i,g}}{p_{i,g}}\right)$$

$$= x_{i,q}(p_{i,g} - y_{i,g})$$

Multinomial Logistic Regression

➢ Learning (Multi-class)

$$\frac{\partial L_{CCE}(\mathbf{f_W}(\mathbf{x}_i), \mathbf{y}_i)}{\partial w_{q,g}} = x_{i,q}(p_{i,g} - y_{i,g})$$

❖ $$\frac{dJ(\mathbf{W})}{d\mathbf{W}} = \frac{1}{N}\sum_{i=1}^{N} \frac{d}{d\mathbf{W}} L_{CCE}(\mathbf{f_W}(\mathbf{x}_i), \mathbf{y}_i)$$

$$= \frac{1}{N}\sum_{i=1}^{N} \begin{bmatrix} \frac{\partial L_{CCE}(\mathbf{f_W}(\mathbf{x}_i), \mathbf{y}_i)}{\partial w_{0,1}} & \cdots & \frac{\partial L_{CCE}(\mathbf{f_W}(\mathbf{x}_i), \mathbf{y}_i)}{\partial w_{0,K}} \\ \vdots & \ddots & \vdots \\ \frac{\partial L_{CCE}(\mathbf{f_W}(\mathbf{x}_i), \mathbf{y}_i)}{\partial w_{d,1}} & \cdots & \frac{\partial L_{CCE}(\mathbf{f_W}(\mathbf{x}_i), \mathbf{y}_i)}{\partial w_{d,K}} \end{bmatrix}$$

$$= \frac{1}{N}\sum_{i=1}^{N} \begin{bmatrix} x_{i,0}(p_{i,1} - y_{i,1}) & \cdots & x_{i,0}(p_{i,K} - y_{i,K}) \\ \vdots & \ddots & \vdots \\ x_{i,d}(p_{i,1} - y_{i,1}) & \cdots & x_{i,d}(p_{i,K} - y_{i,K}) \end{bmatrix}$$

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & \cdots & x_{N,d} \end{bmatrix}$$

$\mathbf{Y}$ has a size of *N*x*K*

$$= \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i(\mathbf{f_W}(\mathbf{x}_i) - \mathbf{y}_i) = \frac{1}{N}\mathbf{X}^T(\mathbf{F_W}(\mathbf{X}) - \mathbf{Y})$$

↳ same form! just in matrix and vector.

Exactly the same as binomial Logistic Regression! Why?

Multinomial Logistic Regression

- Softmax()=Logistic(), when K=2

Suppose class 1 is assigned with label 1

$$\boldsymbol{\sigma}(\boldsymbol{z})_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2}}$$

$$= \frac{1}{1 + e^{z_2 - z_1}}$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + e^{-\mathbf{x}^T\mathbf{w}}}$$

$$= \mathrm{P}(\mathrm{y} = 1|\mathbf{x})$$

Let a= $z_1 - z_2 = \mathbf{x}^T(\underbrace{\mathbf{w}_1 - \mathbf{w}_2}_{\mathbf{w}})$

$$\boldsymbol{\sigma}(\boldsymbol{z})_1 = \frac{1}{1 + e^{-a}} = \frac{1}{1 + e^{-\mathbf{x}^T\mathbf{w}}}$$

$$= \mathrm{P}(\mathrm{y} = 1|\mathbf{x})$$

Multinomial Logistic Regression

- $L_{CCE} = L_{BCE}$, when K=2

Suppose class 1 is assigned with label 1

$$L_{CCE} = -\sum_{k=1}^{2} y_k \ln(p_k)$$

$$= -y_1 \ln(p_1) - y_2 \ln(p_2)$$

$$= -y_1 \ln(p_1) - (1 - y_1) \ln(1 - p_1)$$

$$L_{BCE} = -y\ln(p) - (1 - y)\ln(1 - p)$$

Multinomial Logistic Regression

➢ Prediction (Multi-class)

$$\mathbf{F}_{\mathbf{W}^*}(\mathbf{X}_t) = \boldsymbol{\sigma}(\mathbf{X}_t\mathbf{W}^*) = \boldsymbol{\sigma}\left(\begin{bmatrix} 1 & x_{1,1}^t & \cdots & x_{1,d}^t \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m,1}^t & \cdots & x_{m,d}^t \end{bmatrix}\begin{bmatrix} w_{0,1}^* & \cdots & w_{0,K}^* \\ w_{1,1}^* & \cdots & w_{1,K}^* \\ \vdots & \ddots & \vdots \\ w_{d,1}^* & \cdots & w_{d,K}^* \end{bmatrix}\right)$$

$$= \begin{bmatrix} \boldsymbol{\sigma}((\mathbf{x}_1^t)^T\mathbf{W}^*) \\ \vdots \\ \boldsymbol{\sigma}((\mathbf{x}_m^t)^T\mathbf{W}^*) \end{bmatrix}$$

Each row is the probability distribution for each test sample

To predict the class label, output column index that has the largest probability for each test sample.

e.g., 2 test samples, 4 classes

$$\begin{matrix} \text{sample 1} \\ \text{sample 2} \end{matrix}\begin{bmatrix} 0.1 & 0.7 & 0 & 0.2 \\ 0.3 & 0.3 & 0.4 & 0 \end{bmatrix}\begin{matrix} \text{argmax} \\ \text{argmax} \end{matrix}$$

class 2

class 3

Multinomial Logistic Regression

Example: Given a training set $\{(x_i, y_i)\}_{i=1}^{5} = \{(-1, class1), (0, class1), (0.5, class2), (0.3, class3), (0.8, class2)\}$, predict the probability of {x=-0.1} belonging to each class using multinomial logistic regression.

Multinomial Logistic Regression

Example: Given a training set $\{(x_i, y_i)\}_{i=1}^{5} = \{(-1, class1), (0, class1), (0.5, class2), (0.3, class3), (0.8, class2)\}$, predict the probability of {x=-0.1} belonging to each class using multinomial logistic regression.

Step 0: Prepare the training data

One-hot encoding of the target output

$class1$: [1,0,0] $class2$: [0,1,0] $class3$: [0,0,1]

$$\mathbf{X} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 0.5 \\ 1 & 0.3 \\ 1 & 0.8 \end{bmatrix} \qquad \mathbf{Y} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Multinomial Logistic Regression

Example: Given a training set $\{(x_i, y_i)\}_{i=1}^{5} = \{(-1, class1), (0, class1), (0.5, class2), (0.3, class3), (0.8, class2)\}$, predict the probability of {x=-0.1} belonging to each class using multinomial logistic regression.

Step 1: Learning (work out $\mathbf{W}^*$)

$$\mathbf{W}_{10000} = \begin{bmatrix} 6.5540 & -8.5683 & 3.0143 \\ -25.6024 & 27.1366 & -1.9341 \end{bmatrix}$$

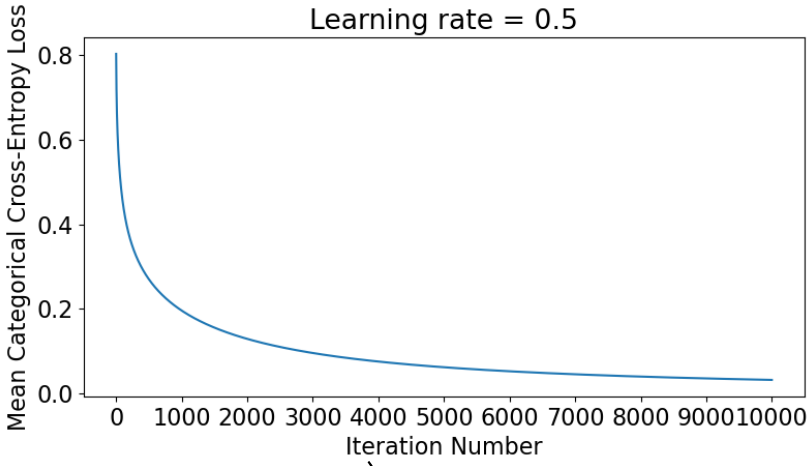
Gradient Descent:

-Gradient $\nabla_{\mathbf{W}} J(\mathbf{W}) = \frac{1}{N}\mathbf{X}^T(\mathbf{F}_{\mathbf{W}}(\mathbf{X}) - \mathbf{Y})$

-Initialize $\mathbf{W}_0 = \begin{bmatrix} 0.5 & 0.3 & 0.2 \\ -1 & 0.5 & 0.1 \end{bmatrix}$,

learning rate $\eta = 0.5$

-At each iteration, $\mathbf{W}_{k+1} = \mathbf{W}_k - \eta \nabla_{\mathbf{W}} J(\mathbf{W}_k)$



$$\mathbf{W}_0 = \begin{bmatrix} 0.5 & 0.3 & 0.2 \\ -1 & 0.5 & 0.1 \end{bmatrix}$$

$$\nabla_{\mathbf{W}} J(\mathbf{W}_0) = \frac{1}{N}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -1 & 0 & 0.5 & 0.3 & 0.8 \end{bmatrix} \left(\begin{bmatrix} \text{Softmax}(\mathbf{x}_1^T \mathbf{W}_0) \\ \vdots \\ \text{Softmax}(\mathbf{x}_5^T \mathbf{W}_0) \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \right)$$

$$\vdots = \begin{bmatrix} -0.0355 & -0.0479 & 0.0834 \\ 0.1321 & -0.1402 & 0.0080 \end{bmatrix}$$

Multinomial Logistic Regression

Example: Given a training set $\{(x_i, y_i)\}_{i=1}^{5} = \{(-1, class1), (0, class1), (0.5, class2), (0.3, class3), (0.8, class2)\}$, predict the probability of {x=-0.1} belonging to each class using multinomial logistic regression.

Step 2: Prediction

Augment 1 for the feature vector

$$\mathbf{x}_t = \begin{bmatrix} 1 \\ -0.1 \end{bmatrix} \qquad \mathbf{W}_{10000} = \begin{bmatrix} 6.5540 & -8.5683 & 3.0143 \\ -25.6024 & 27.1366 & -1.9341 \end{bmatrix}$$

$$y_t^* = \boldsymbol{\sigma}\left(\mathbf{x}_t^T \mathbf{W}^*\right)$$

$$= \boldsymbol{\sigma}\left(\begin{bmatrix} 1 & -0.1 \end{bmatrix} \begin{bmatrix} 6.5540 & -8.5683 & 3.0143 \\ -25.6024 & 27.1366 & -1.9341 \end{bmatrix}\right)$$

$$= \left[\frac{e^{9.1143}}{e^{9.1143} + e^{-11.2820} + e^{3.2077}}, \frac{e^{-11.2820}}{e^{9.1143} + e^{-11.2820} + e^{3.2077}}, \frac{e^{3.2077}}{e^{9.1143} + e^{-11.2820} + e^{3.2077}}\right]$$

$$= [9.9729 \times 10^{-1}, 1.3830 \times 10^{-9}, 2.7142 \times 10^{-3}]$$

class 1 — largest probability. class 2 class 3

Please feel free to provide your thoughts under the Section of "**Discussions**" **in the thread: "Ask Questions Here for Week 7"** on Canvas.

Why Cross-Entropy Loss,

entropy! recall EE2211 Gini, Entropy, it is to test pureness of each classification, and we want to reduce that.

not Squared Error Loss ?

in logistic regression

Evaluation Metrics for Classification

Evaluation Metrics for Classification

Binary Classification

Class1 (label +1 or 1): Positive Class

Class2 (lable -1 or 0): Negative Class

TP: True Positive

FN: False Negative
(i.e., Type II Error)

FP: False Positive
(i.e., Type I Error)

TN: True Negative

Confusion Matrix for Binary Classification

	$\hat{P}$ (predicted)	$\hat{N}$ (predicted)
P (actual)	TP	FN
N (actual)	FP	TN

Recall
TP/(TP+FN)

Precision
TP/(TP+FP)

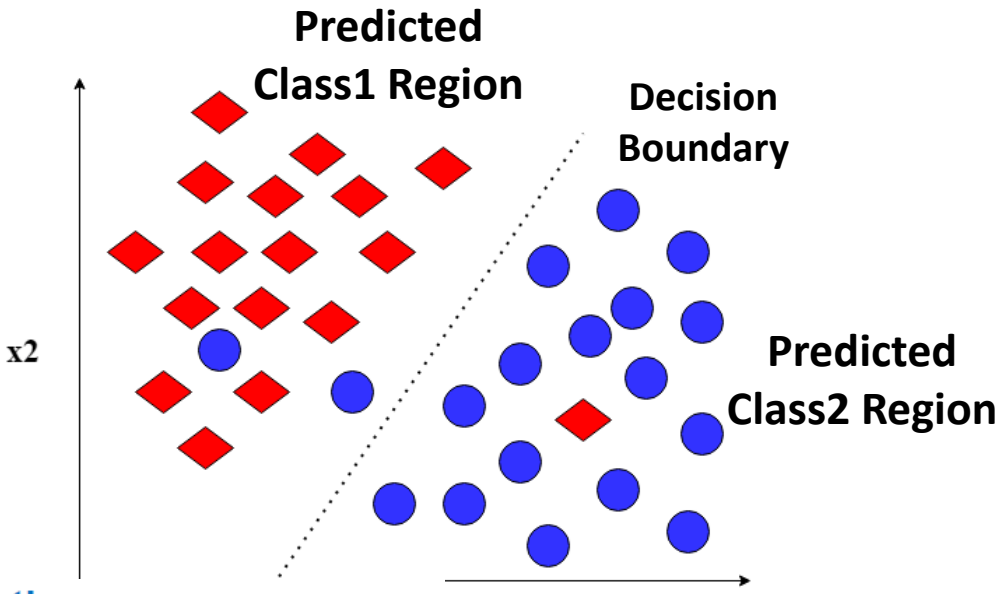
Accuracy
(TP+TN)/(TP+TN+FP+FN)

Evaluation Metrics for Classification

➢ Binary Classification

e.g.,

- Class1: Positive Class
- Class2: Negative Class



Confusion Matrix for Binary Classification

	$\hat{P}$ (predicted)	$\hat{N}$ (predicted)
P (actual)	16 TP	1 FN
N (actual)	2 FP	16 TN

Recall
TP/(TP+FN)

Precision
TP/(TP+FP)

Accuracy
(TP+TN)/(TP+TN+FP+FN)

only binary has these:

TP: True Positive
FN: False Negative
(i.e., Type II Error)
FP: False Positive
(i.e., Type I Error)
TN: True Negative

Evaluation Metrics for Classification

➢ Multicategory Classification

Confusion Matrix for Multicategory Classification

	$P_{\hat{1}}$ (predicted)	$P_{\hat{2}}$ (predicted)		$P_{\hat{C}}$ (predicted)
P_1 (actual)	$P_{1,\hat{1}}$	$P_{1,\hat{2}}$	⋯	$P_{1,\hat{C}}$
P_2 (actual)	$P_{2,\hat{1}}$	$P_{2,\hat{2}}$	⋯	$P_{2,\hat{C}}$
⋮	⋮	⋮	⋱	⋮
P_C (actual)	$P_{C,\hat{1}}$	$P_{C,\hat{2}}$		$P_{C,\hat{C}}$

only accuracy.

diagonal entries are the correct ones.

accuracy = sum of diagonal entries / total sample count

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_true, y_pred)
acc = accuracy_score(y_true, y_pred)
```

Summary

✓ Polynomial Regression for Classification

Binary Classification

-Learning:

$$\mathbf{w}^* = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T\mathbf{y} \quad y_i \in \{-1, +1\}$$

-Prediction:

$$\mathbf{f}_{\mathbf{w}^*}(\mathbf{P}(\mathbf{X}_t)) = \text{sign}(\mathbf{P}_t\mathbf{w}^*)$$

Multi-category Classification

-Learning:

$$\mathbf{W}^* = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T\mathbf{Y}$$

-Prediction:

$$\mathbf{F}_{\mathbf{W}^*}(\mathbf{P}(\mathbf{X}_t)) = \begin{bmatrix} \underset{k=1,\ldots,K}{\text{argmax}}((\mathbf{p}_1^t)^T\mathbf{W}^*) \\ \vdots \\ \underset{k=1,\ldots,K}{\text{argmax}}((\mathbf{p}_m^t)^T\mathbf{W}^*) \end{bmatrix}$$

✓ Logistic Regression

Binary Classification

-Learning: Gradient Descent

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{N}\mathbf{X}^T(\mathbf{f}_{\mathbf{w}}(\mathbf{X}) - \mathbf{y})$$

-Prediction:

$$\mathbf{f}_{\mathbf{w}^*}(\mathbf{X}_t) = \sigma(\mathbf{X}_t\mathbf{w}^*) = \begin{bmatrix} \sigma((\mathbf{x}_1^t)^T\mathbf{w}^*) \\ \vdots \\ \sigma((\mathbf{x}_m^t)^T\mathbf{w}^*) \end{bmatrix}$$

Multi-category Classification

-Learning: Gradient Descent

$$\nabla_{\mathbf{W}} J(\mathbf{W}) = \frac{1}{N}\mathbf{X}^T(\mathbf{F}_{\mathbf{W}}(\mathbf{X}) - \mathbf{Y})$$

-Prediction:

$$\mathbf{F}_{\mathbf{W}^*}(\mathbf{X}_t) = \boldsymbol{\sigma}(\mathbf{X}_t\mathbf{W}^*) = \begin{bmatrix} \boldsymbol{\sigma}((\mathbf{x}_1^t)^T\mathbf{W}^*) \\ \vdots \\ \boldsymbol{\sigma}((\mathbf{x}_m^t)^T\mathbf{W}^*) \end{bmatrix}$$

✓ Evaluation Metrics for Classification Task

binary
1. Precision
2. Recall
3. Accuracy, ⇐ multi-class only this

EE2213 Introduction to AI

Lecture 10: Clustering

Dr. WANG Si
si.wang@nus.edu.sg
Electrical and Computer Engineering Department
National University of Singapore

OVERVIEW OF COURSE CONTENTS

- **Introduction (Shaojing)**
  - What is AI
  - Applications of AI
  - AI agent
- **Search (Shaojing)**
  - Uninformed search algorithms: breadth-first, depth-first, uniform-cost(Dijkstra's algorithm)
  - Informed search algorithms: greedy best-first, A*
  - Applications
- **Optimisation (Shaojing)**
  - Linear programming
  - Convex problems
  - Applications
- **Machine learning (Wang Si) (Weeks 6-9)**
  - Supervised and unsupervised learning: regression, classification, clustering
  - Neural networks and deep learning
  - Applications
- **Knowledge representation (Wang Si) (Weeks 10-11)**
  - Knowledge Representation and Reasoning
  - Propositional Logic
  - Applications
- **Ethical considerations (Shaojing)**
  - Bias in AI
  - Privacy concerns
  - Societal impact

(Week 10, Monday)

No Class on 20 Oct.! A recording will be uploaded on Canvas.

AGENDA

- We will discuss:
  - types of clusters and types of clustering
  - K-means clustering
  - Fuzzy C-means clustering

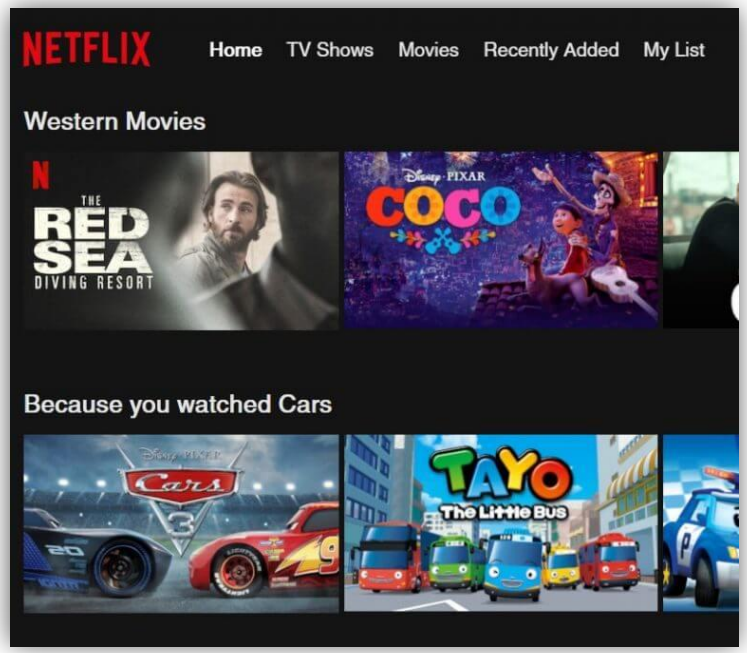
At the end of this lecture, you should be able to:

- ✓ Differentiate between types of clusters and clustering
- ✓ Apply K-means and Fuzzy C-means
- ✓ Describe the limitations of K-means clustering

Recap: Clustering

Groups a set of objects in such a way that objects in the same group (called a **cluster**) are **more similar** (in some sense) to each other than to those in other groups (clusters).

Applications:



Recommender systems

Image Segmentation

Social Network Analysis

Notion of a Cluster can be Ambiguous

How many clusters?

Two Clusters

Four Clusters

Six Clusters

Types of Clusters

-Well-separated Clusters

-Center-based Clusters

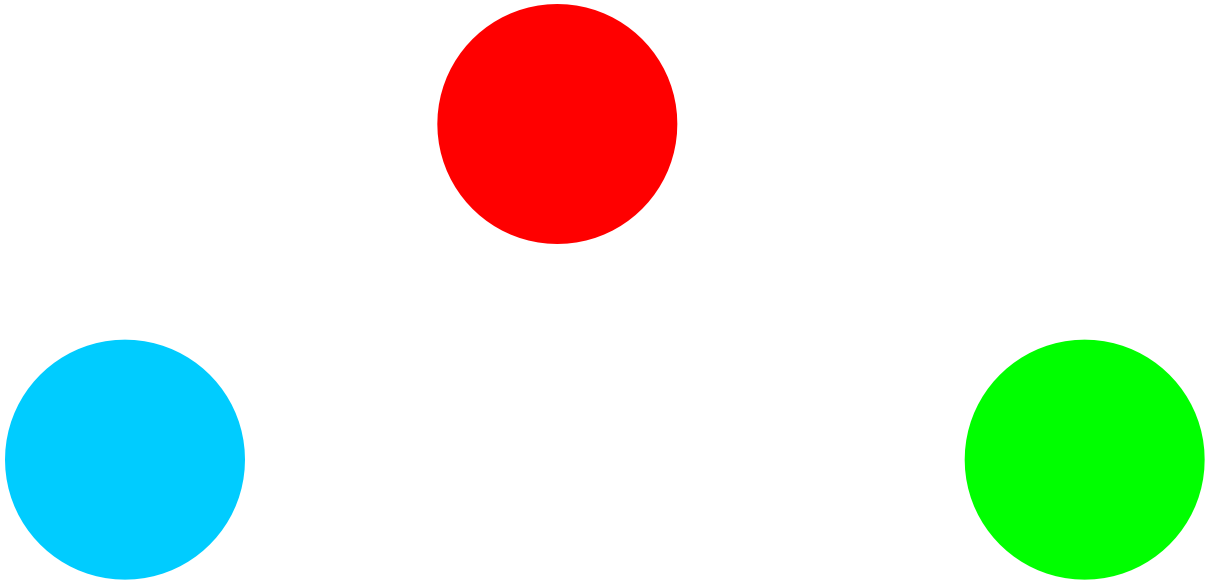
-Contiguous Clusters

-Density-based Clusters

Types of Clusters

- Well-Separated Cluster:
  - A cluster is a set of points such that any point in a cluster is closer (or more similar) to every other point in the cluster than to any point not in the cluster.

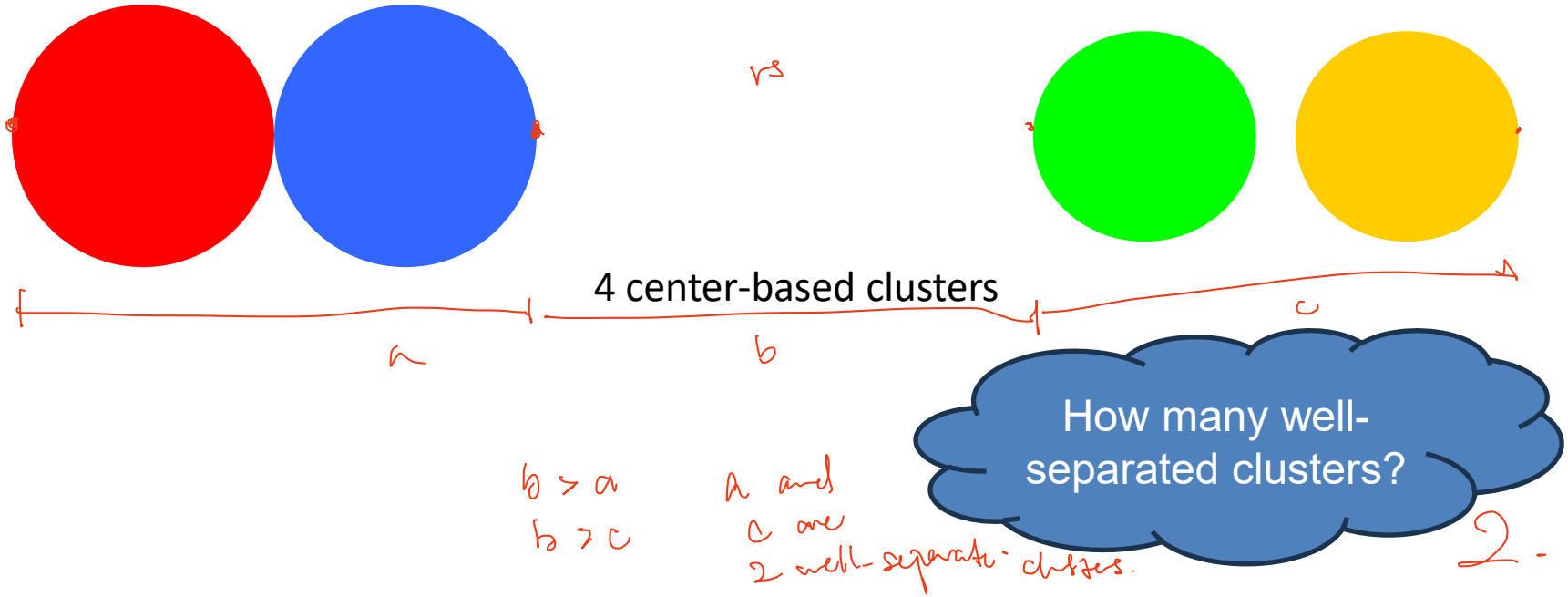
by distance between points



3 well-separated clusters

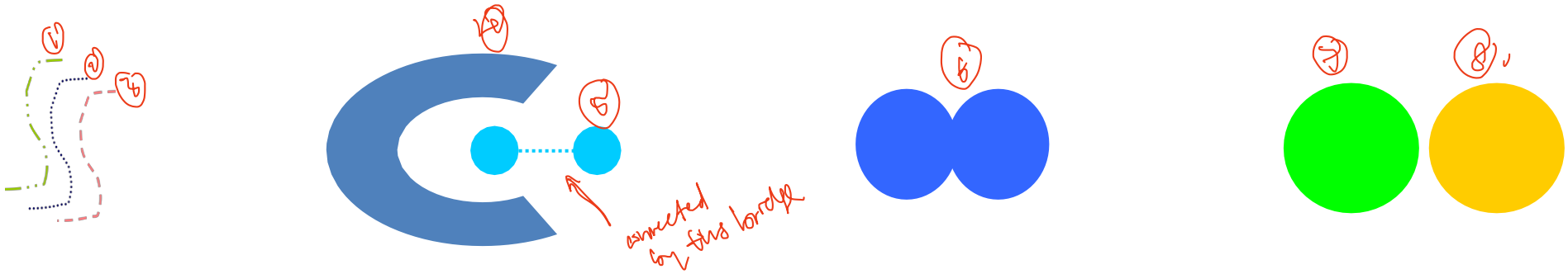
Types of Clusters

- Center-based Cluster:
  - A cluster is a set of objects such that a point in a cluster is closer (more similar) to the "center" of a cluster, than to the center of any other cluster
  - The center of a cluster is often a centroid, the average of all the points in the cluster, or the most "representative" point of a cluster



Types of Clusters

- Contiguous Cluster (Nearest neighbor)
  - A cluster is a set of points such that a point in a cluster is closer (or more similar) to **one or more** other points in the cluster than to any point not in the cluster.



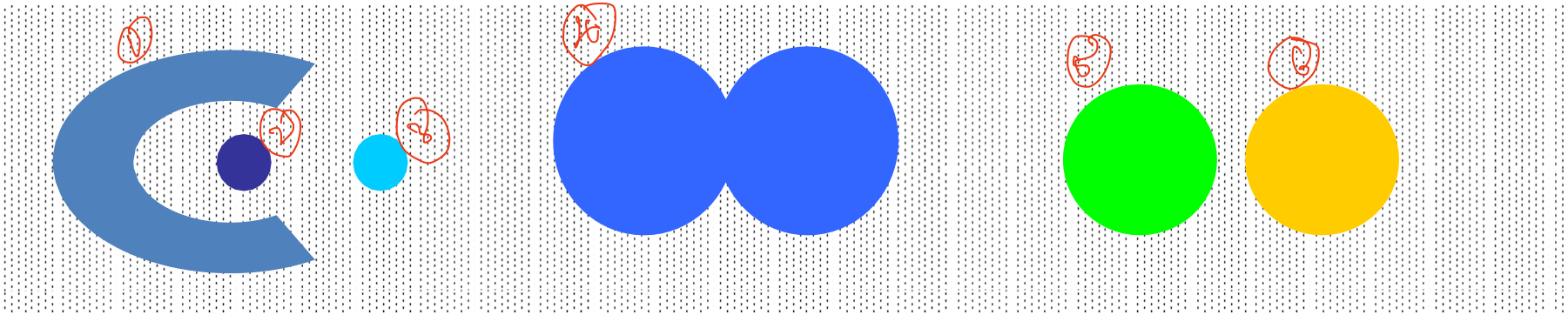
8 contiguous clusters

How many well-separated clusters?

How many center-based clusters?

Types of Clusters

- Density-based Cluster
  - A cluster is a dense region of points, which is separated by low- density regions, from other regions of high density.
  - Used when noise and outliers are present.



6 density-based clusters

Types of Clustering

- Hard Clustering
  - Each data point belongs to a cluster completely or not.
  - e.g., **K-means**, Hierarchical, etc.

- Soft Clustering
  - Each data point belongs to each cluster with a probability or degree of membership.
  - e.g., **Fuzzy C-means**, Gaussian mixture model, etc.

Data Points	Hard	Soft	
	Clusters	P(C1)	P(C2)
A	C1	0.91	0.09
B	C2	0.3	0.7
C	C2	0.17	0.83
D	C1	1	0

K-means Clustering

-Hard Clustering

What type of clusters?

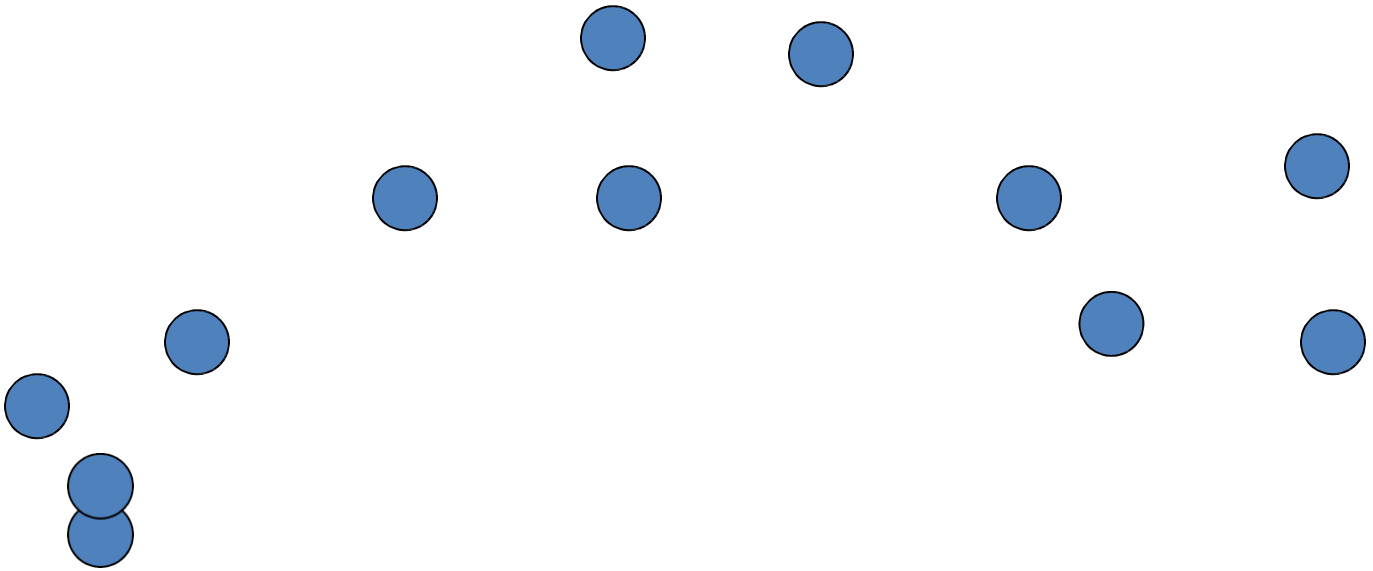
K-means Clustering

center-based clustering

- ✓ Each cluster is associated with a centroid (center point)
- ✓ Each point is assigned to the cluster with the closest centroid

An example: K=3 (we seek 3 clusters)

Centre-based clusters will be produced.

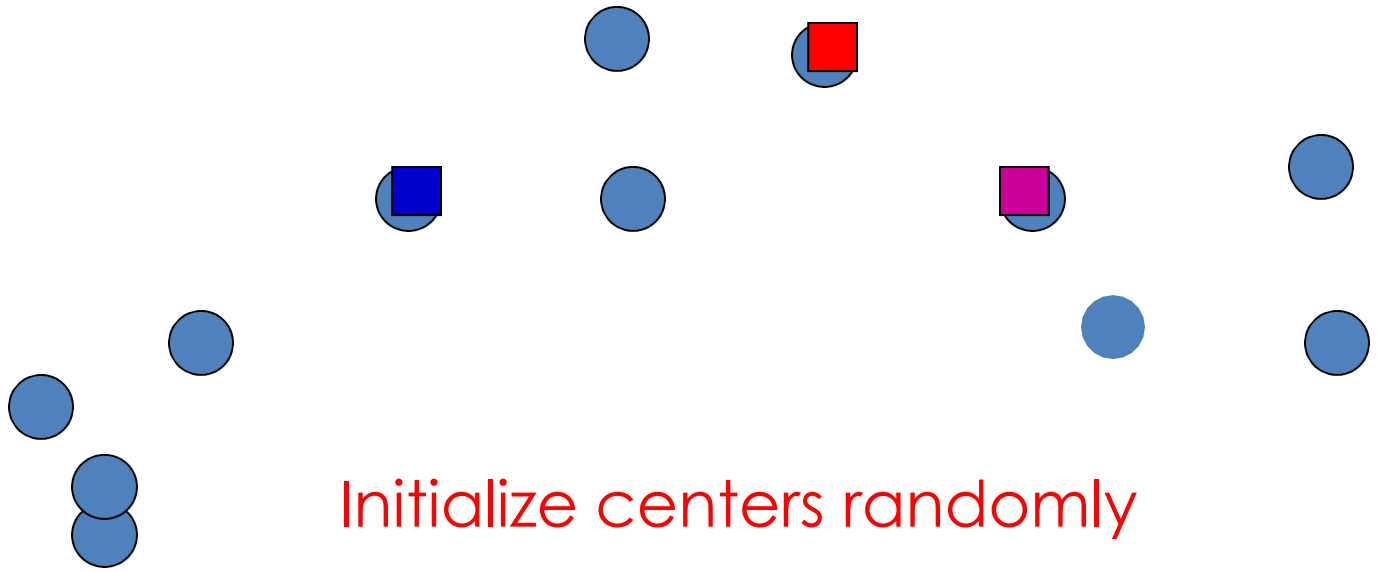


K-means Clustering

- ✓ Each cluster is associated with a centroid (center point)
- ✓ Each point is assigned to the cluster with the closest centroid

An example: K=3 (we seek 3 clusters)

Step 1: Randomly choose 3 points as the centers of the 3 clusters

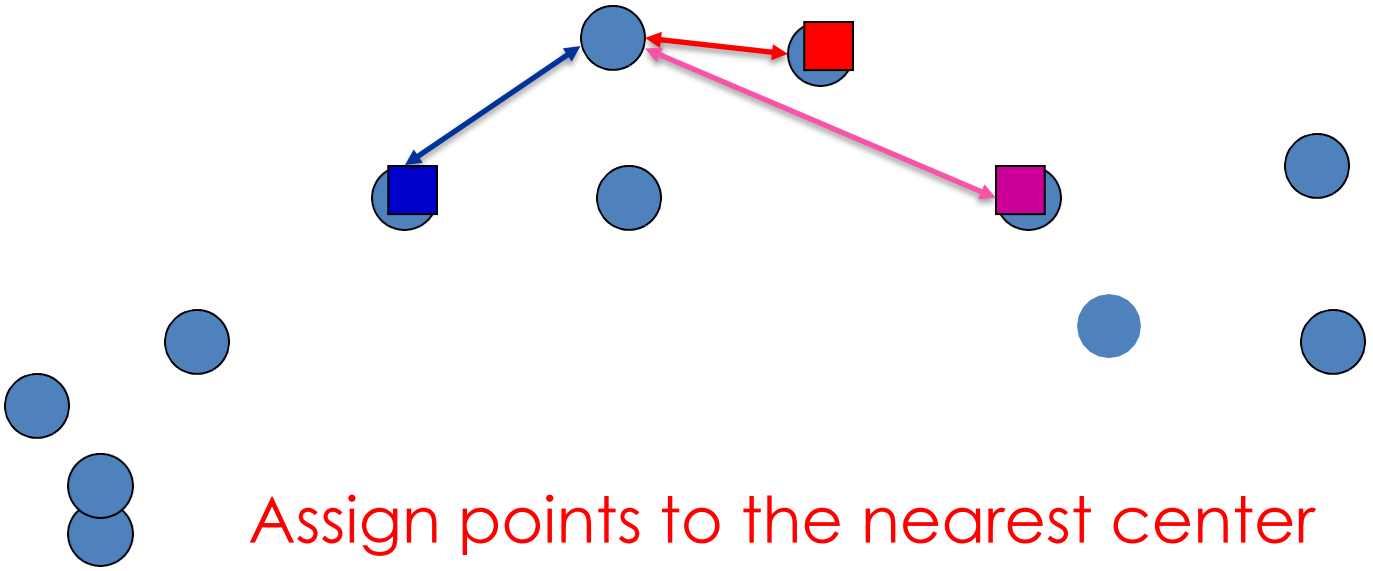


K-means Clustering

- ✓ Each cluster is associated with a centroid (center point)
- ✓ Each point is assigned to the cluster with the closest centroid

An example: K=3 (we seek 3 clusters)

Step 2: For each point, calculate the distance between the point and the 3 centers, and assign it to the nearest one.



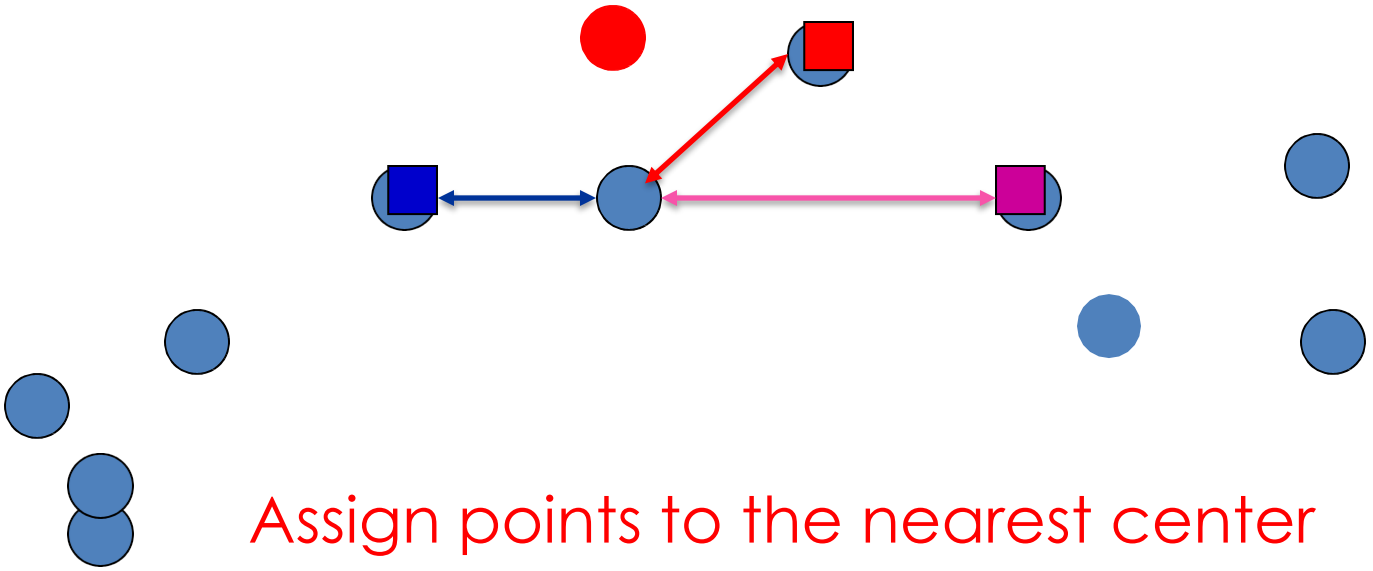
Assign points to the nearest center

K-means Clustering

- ✓ Each cluster is associated with a centroid (center point)
- ✓ Each point is assigned to the cluster with the closest centroid

An example: K=3 (we seek 3 clusters)

Step 2: For each point, calculate the distance between the point and the 3 centers, and assign it to the nearest one.



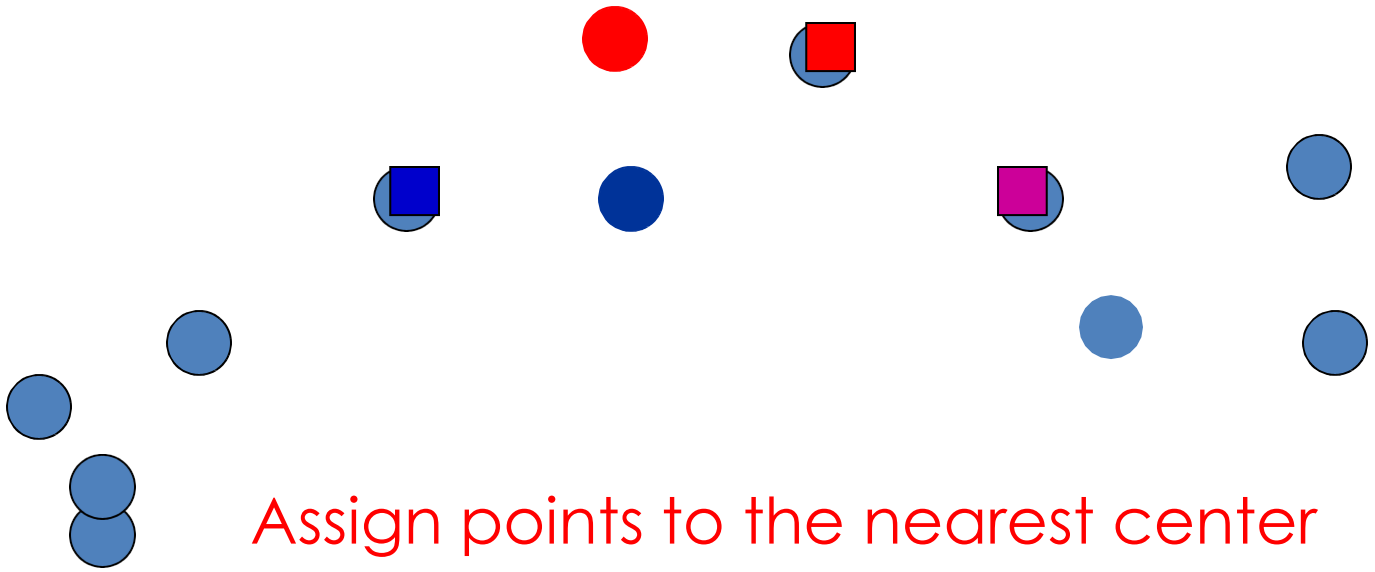
Assign points to the nearest center

K-means Clustering

- ✓ Each cluster is associated with a centroid (center point)
- ✓ Each point is assigned to the cluster with the closest centroid

An example: K=3 (we seek 3 clusters)

Step 2: For each point, calculate the distance between the point and the 3 centers, and assign it to the nearest one.



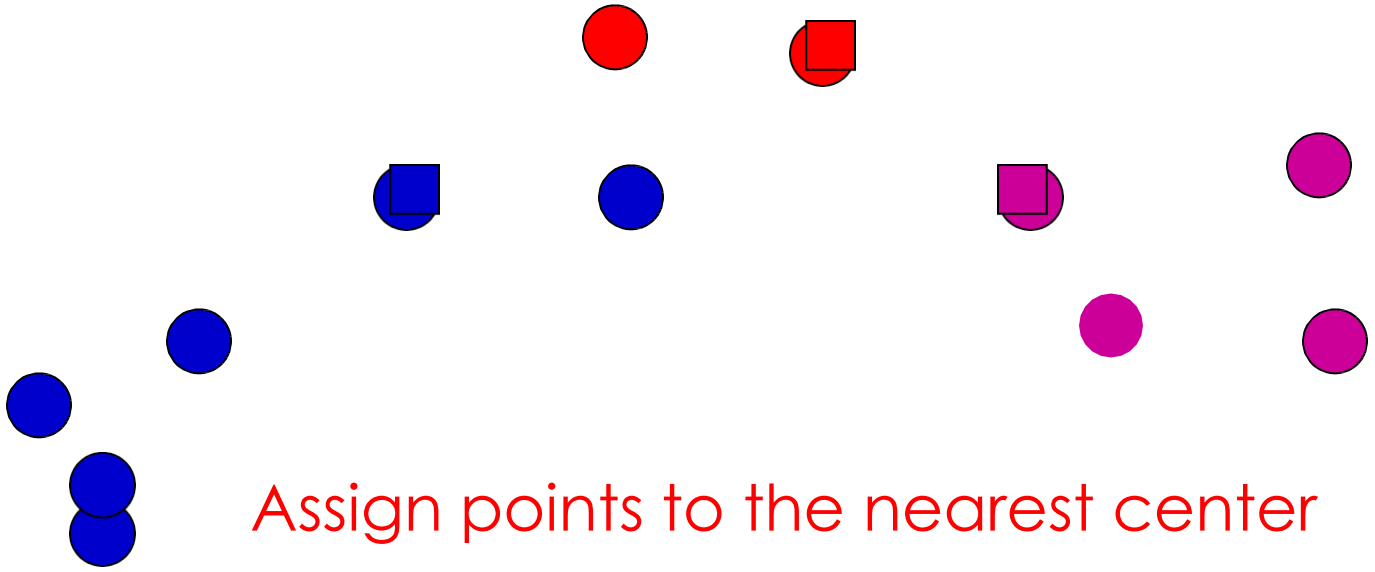
Assign points to the nearest center

K-means Clustering

- ✓ Each cluster is associated with a centroid (center point)
- ✓ Each point is assigned to the cluster with the closest centroid

An example: K=3 (we seek 3 clusters)

Step 2: For each point, calculate the distance between the point and the 3 centers, and assign it to the nearest one.



Assign points to the nearest center

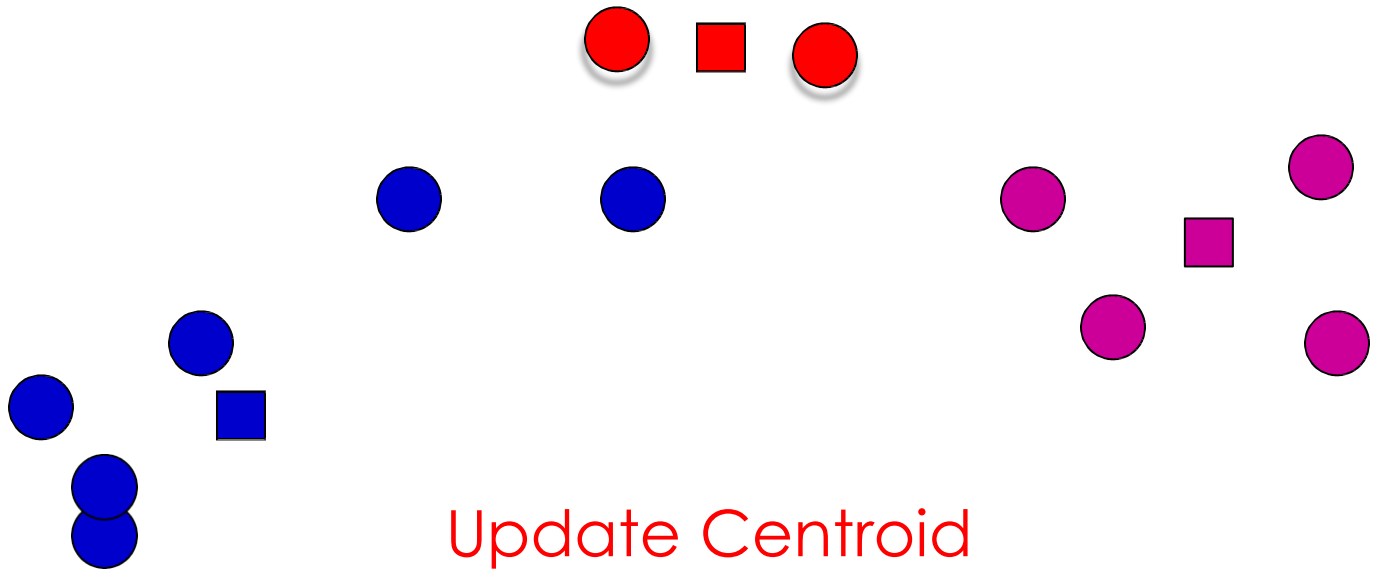
K-means Clustering

- ✓ Each cluster is associated with a centroid (center point)
- ✓ Each point is assigned to the cluster with the closest centroid

An example: K=3 (we seek 3 clusters)

Step 3: Recompute the 3 centers given the current cluster assignments

update each to the centroid of points in new cluster

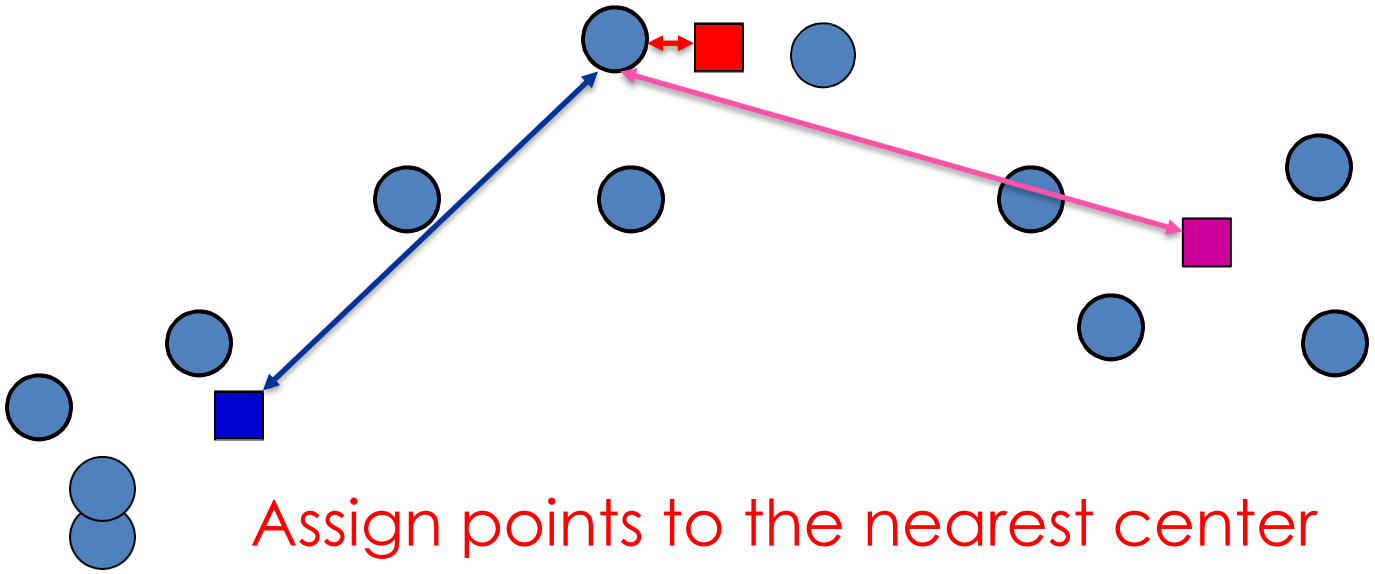


K-means Clustering

- ✓ Each cluster is associated with a centroid (center point)
- ✓ Each point is assigned to the cluster with the closest centroid

An example: K=3 (we seek 3 clusters)

Step 4 (repeat Step 2): For each point, calculate the distance between the point and the updated 3 centers, and assign it to the nearest one.



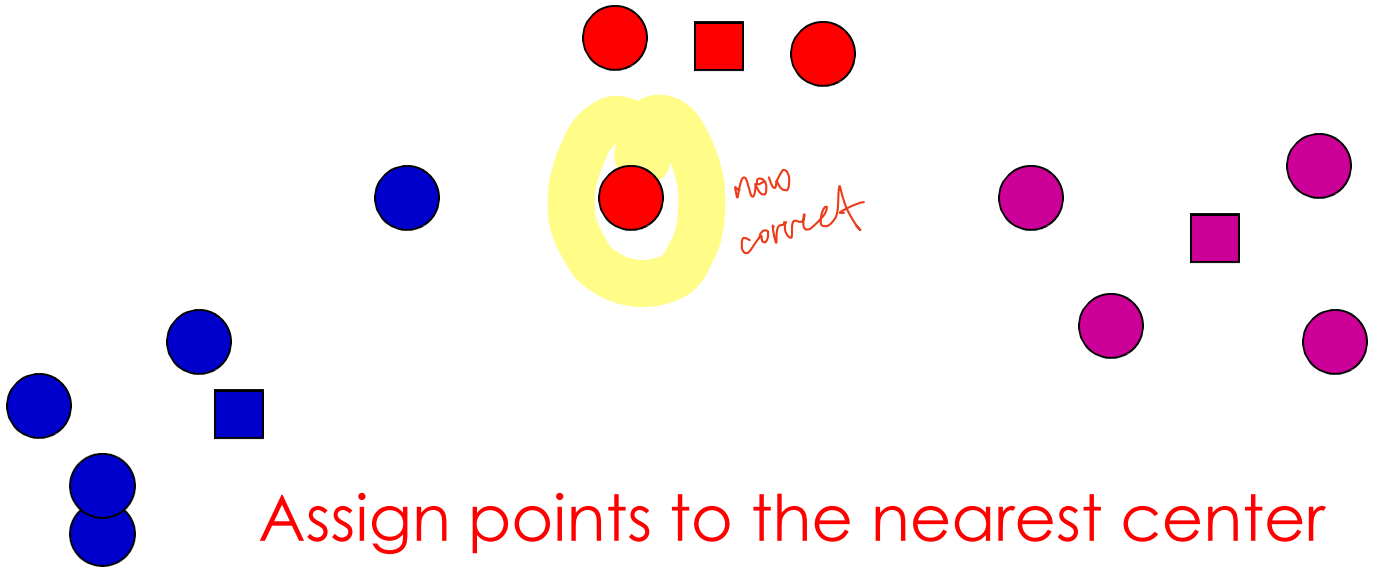
Assign points to the nearest center

K-means Clustering

- ✓ Each cluster is associated with a centroid (center point)
- ✓ Each point is assigned to the cluster with the closest centroid

An example: K=3 (we seek 3 clusters)

Step 4 (repeat Step 2): For each point, calculate the distance between the point and the updated 3 centers, and assign it to the nearest one.



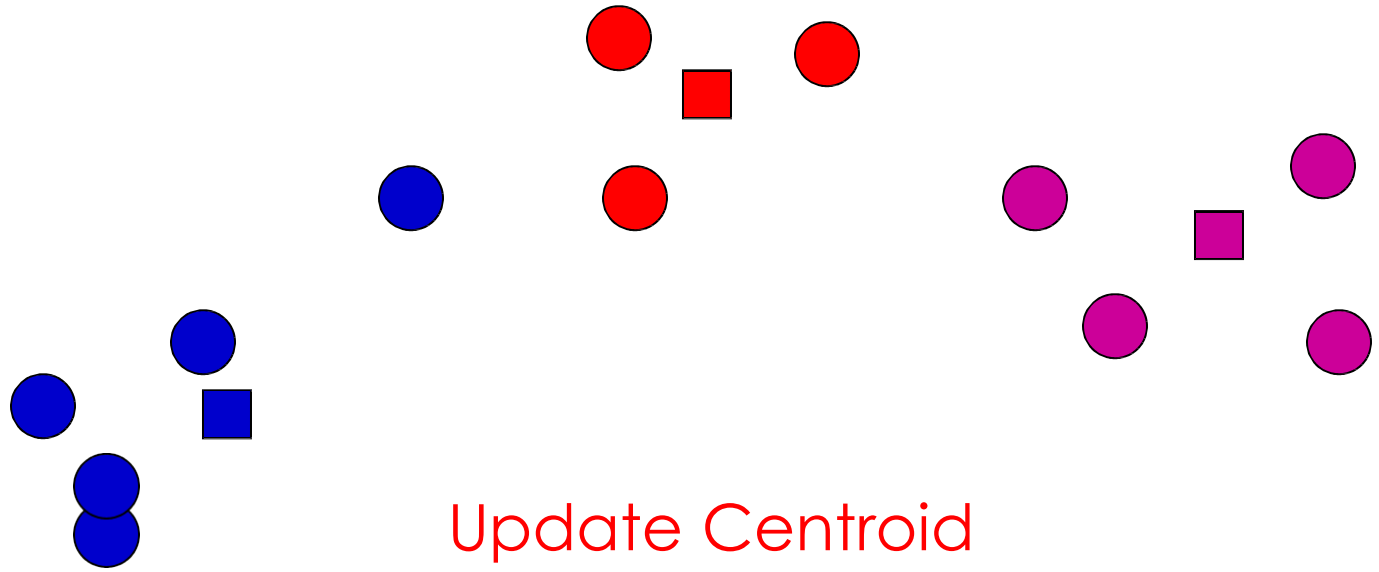
Assign points to the nearest center

K-means Clustering

- ✓ Each cluster is associated with a centroid (center point)
- ✓ Each point is assigned to the cluster with the closest centroid

An example: K=3 (we seek 3 clusters)

again Step 5 (repeat Step 3): Recompute the 3 centers given the current cluster assignments.

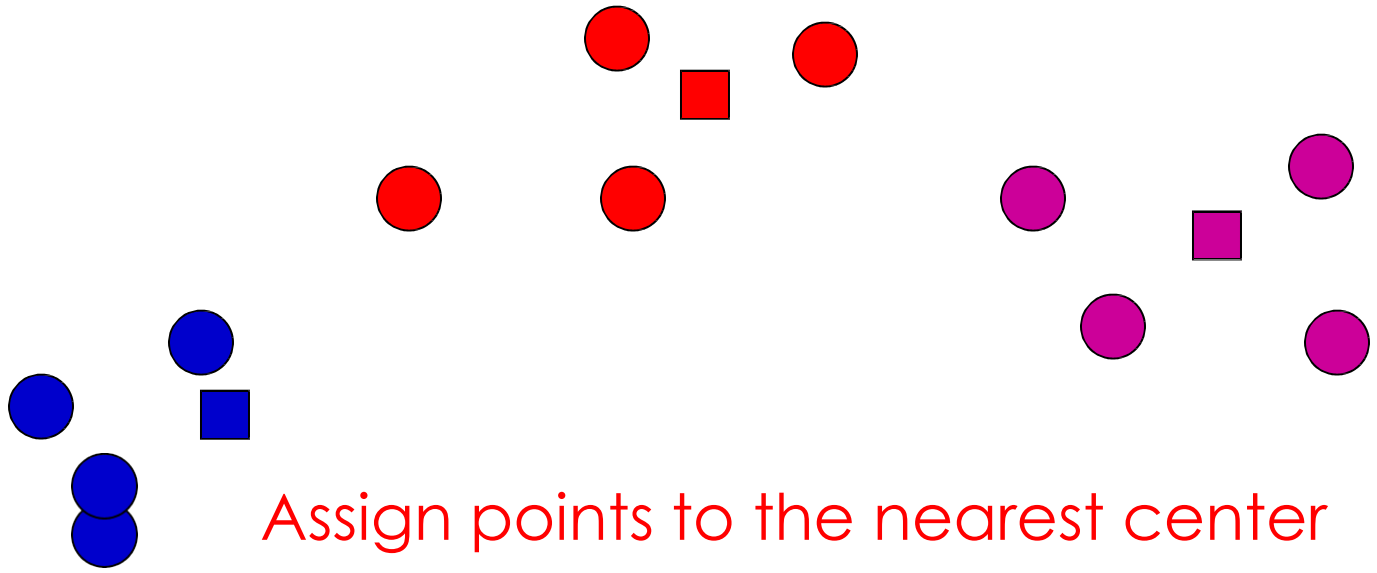


K-means Clustering

- ✓ Each cluster is associated with a centroid (center point)
- ✓ Each point is assigned to the cluster with the closest centroid

An example: K=3 (we seek 3 clusters)

Step 6 (repeat Step 2): For each point, calculate the distance between the point and the updated 3 centers, and assign it to the nearest one.



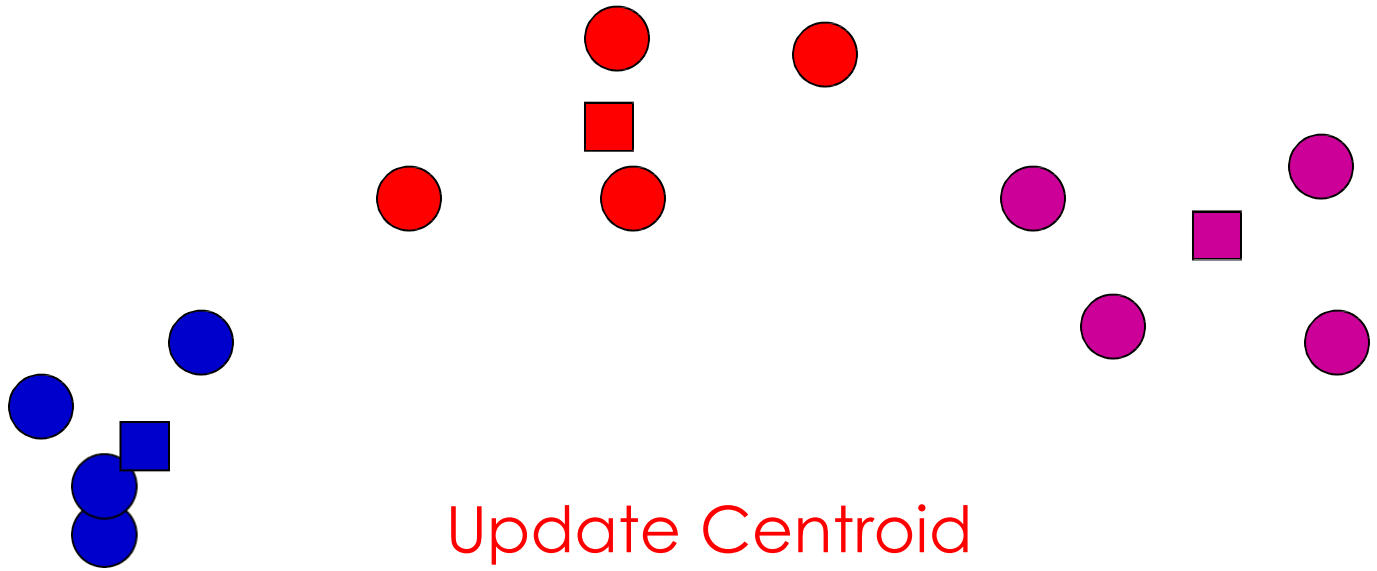
Assign points to the nearest center

K-means Clustering

- ✓ Each cluster is associated with a centroid (center point)
- ✓ Each point is assigned to the cluster with the closest centroid

An example: K=3 (we seek 3 clusters)

Step 7 (repeat Step 3): Recompute the 3 centers given the current cluster assignments.

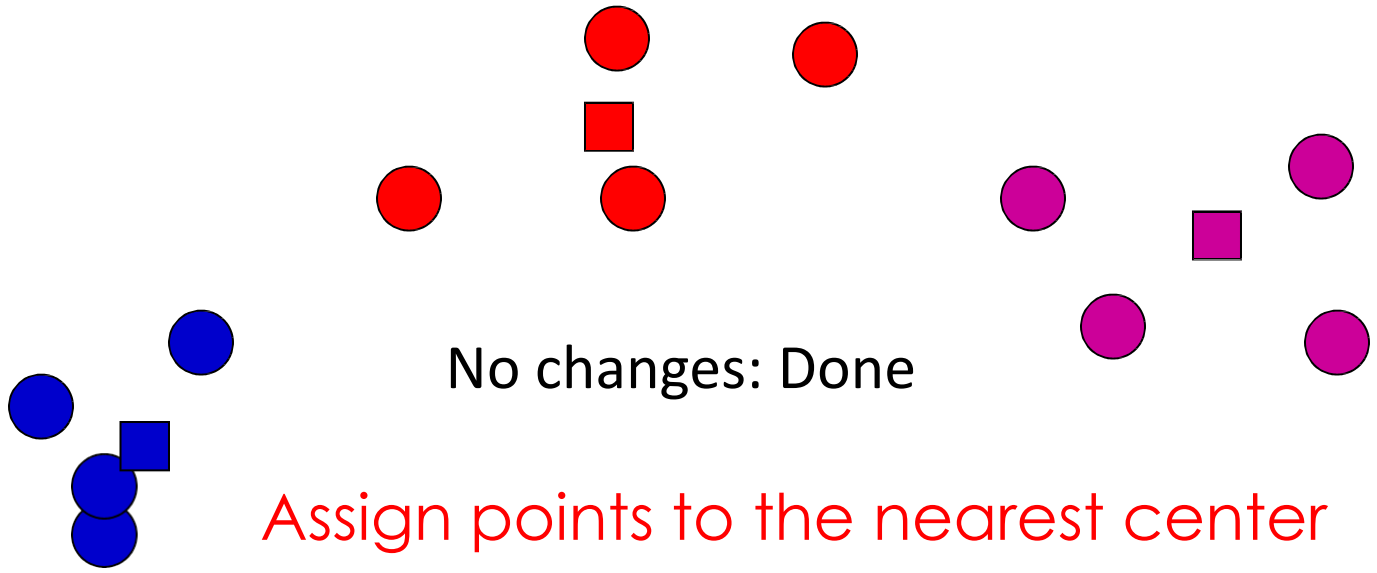


K-means Clustering

- ✓ Each cluster is associated with a centroid (center point)
- ✓ Each point is assigned to the cluster with the closest centroid

An example: K=3 (we seek 3 clusters)

Step 8 (repeat Step 2): For each point, calculate the distance between the point and the updated 3 centers, and assign it to the nearest one.



K-means Clustering

Algorithm

Looping between Assignment and Centroid Update

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

K-means Clustering

➢ Objective Function (within-cluster variance)

Find the optimal values for $\mathbf{W}^*_{N\times K}$ and $\mathbf{C}^*_{K\times d}$ which minimizes:

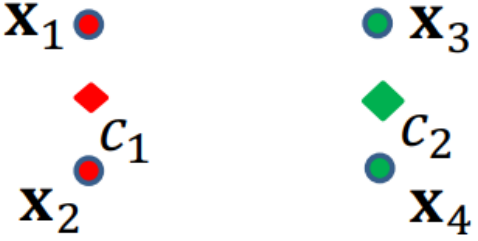
$$J(\mathbf{W},\mathbf{C}) = \sum_{i=1}^{N}\sum_{k=1}^{K} w_{ik}\|\mathbf{x}_i - \boldsymbol{c}_k\|^2$$

Squared Euclidean distance

where $w_{ik} \in \{0,1\}$: $\begin{cases} w_{ik} = 1, if\ \mathbf{x}_i \in cluster\ S_k \\ w_{ik} = 0, \quad else \end{cases}$

$\boldsymbol{c}_k$ (*d-dimensional* vector) is the centroid of cluster S_k

e.g.,



w_{11} =? 1 w_{12} =? 0 w_{21} =? 1 w_{22} =? 0

w_{31} =? 0 w_{32} =? 1 w_{41} =? 0 w_{42} =? 1

$J(\mathbf{W},\mathbf{C})$=?

$\|x_1 - c_1\|^2 + \|x_3 - c_2\|^2 + \|x_2 - c_1\|^2 + \|x_4 - c_2\|^2$

K-means Clustering

➢ Formal Formulation

1. Assignment Step (Fix $\mathbf{C}_{K\times d}$ and update $\mathbf{W}_{N\times K}$)

$$\text{If } \|\mathbf{x}_i - \boldsymbol{c}_k\|^2 < \left\|\mathbf{x}_i - \boldsymbol{c}_{\forall j\neq k}\right\|^2 : w_{ik} = 1$$

$$(\text{else } w_{ik} = 0)$$

$$i = 1, \dots, N; j, k = 1, \dots, K$$

update centroid.

2. Update Step (Fix $\mathbf{W}_{N\times K}$ and update $\mathbf{C}_{K\times d}$)

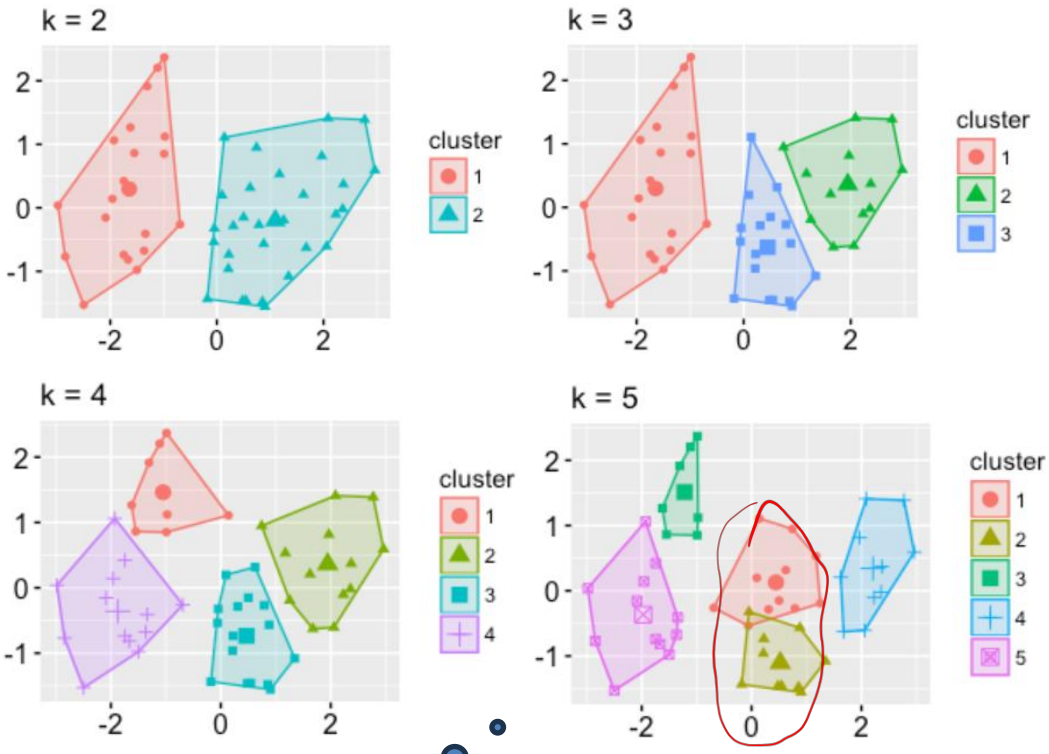
$$\frac{dJ(\mathbf{W},\mathbf{C})}{d\mathbf{C}} = \begin{bmatrix} \frac{\partial \sum_{i=1}^{N}\sum_{k=1}^{K} w_{ik}(\mathbf{x}_i - \boldsymbol{c}_k)^T(\mathbf{x}_i - \boldsymbol{c}_k)}{\partial \boldsymbol{c}_1} \\ \vdots \\ \frac{\partial \sum_{i=1}^{N}\sum_{k=1}^{K} w_{ik}(\mathbf{x}_i - \boldsymbol{c}_k)^T(\mathbf{x}_i - \boldsymbol{c}_k)}{\partial \boldsymbol{c}_K} \end{bmatrix} = \begin{bmatrix} -2\sum_{i=1}^{N} w_{i1}(\mathbf{x}_i - \boldsymbol{c}_1) \\ \vdots \\ -2\sum_{i=1}^{N} w_{iK}(\mathbf{x}_i - \boldsymbol{c}_K) \end{bmatrix} = \mathbf{0}$$

$$\boldsymbol{c}_k = \frac{\sum_{i=1}^{N} w_{ik}\mathbf{x}_i}{\sum_{i=1}^{N} w_{ik}}, k = 1, \dots, K$$

centroid for each cluster.

K-means Clustering: Limitations

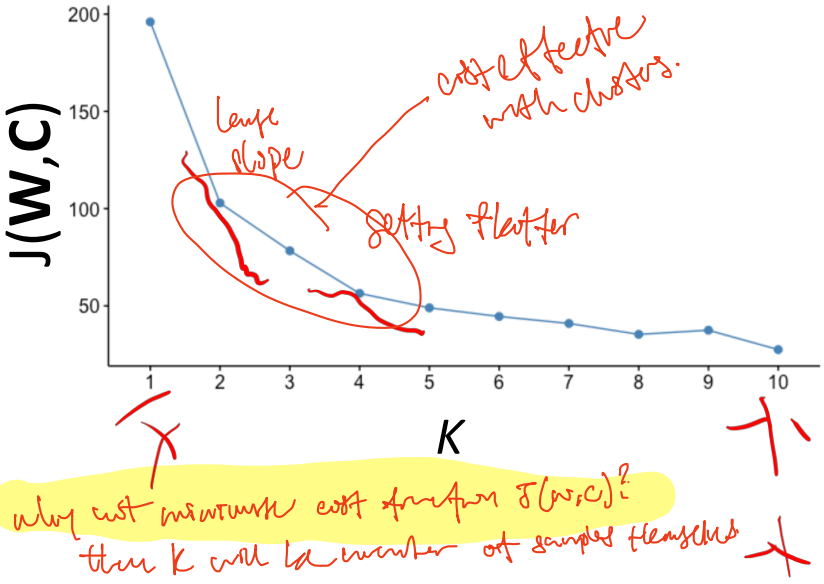
- **Need to predefine *K***



What is the optimal *K*?

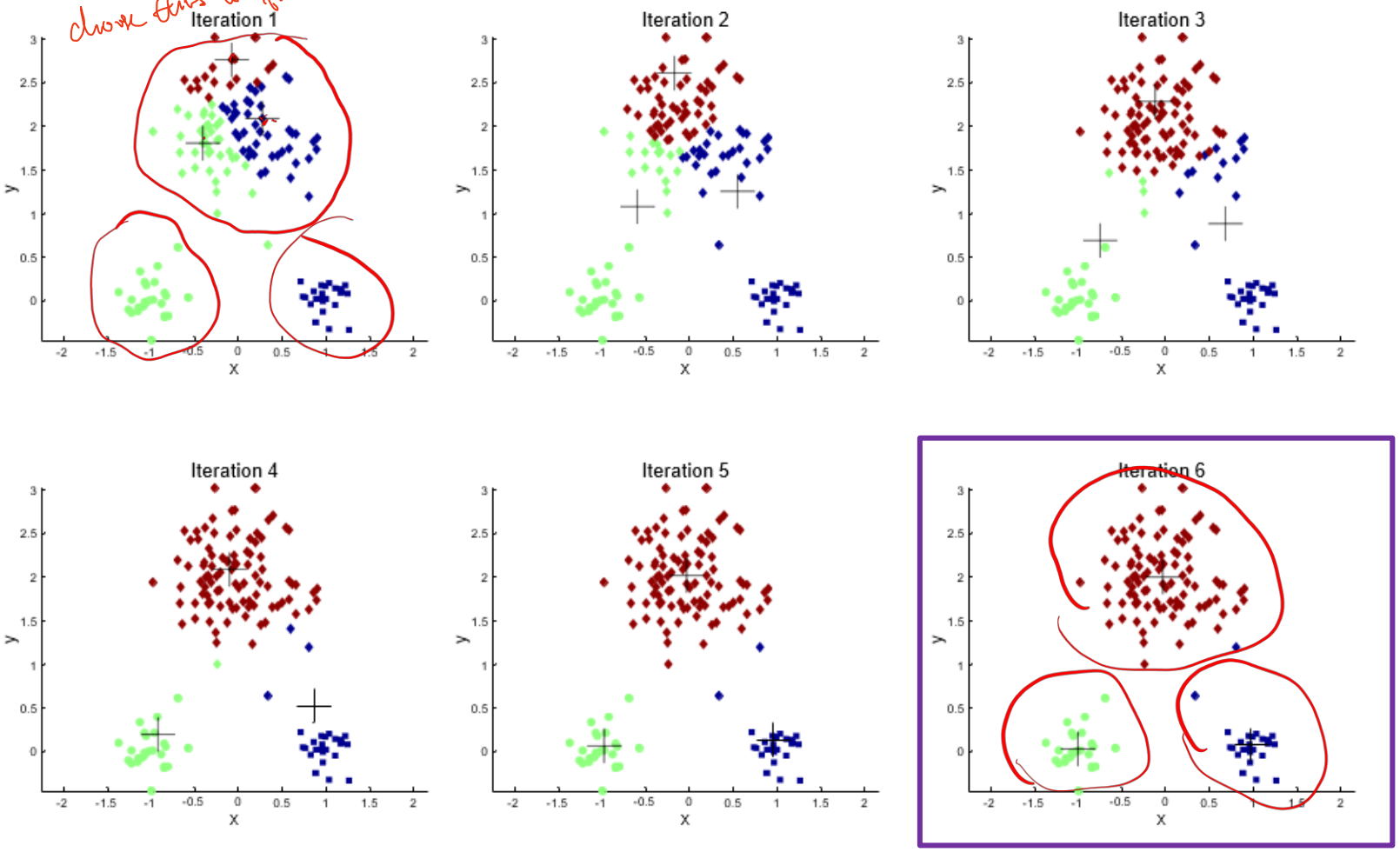
Elbow Method:

1. K-means for different *K*
2. For each *K*, calculate J(**W**,**C**)
3. Plot the curve of J(**W**,**C**) over *K*
4. "bend" or "knee"



K-means Clustering: Limitations

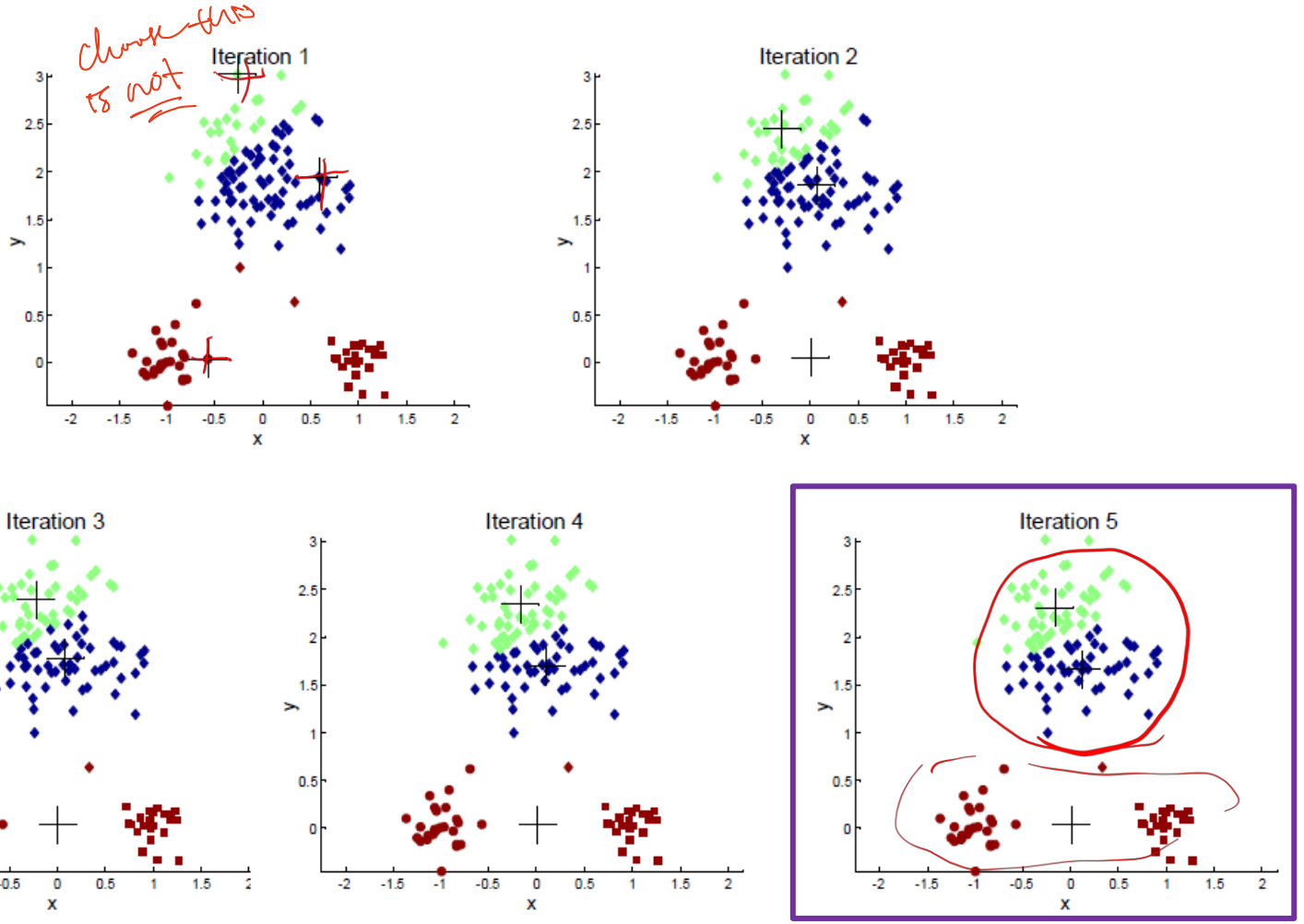
- ***Sensitive to initial centroid selection***



Optimal

K-means Clustering: Limitations

- ***Sensitive to initial centroid selection***

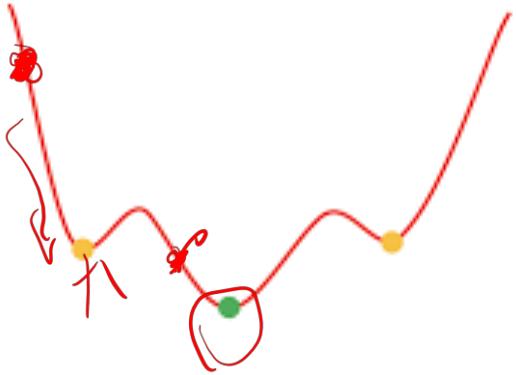


Suboptimal

K-means Clustering: Limitations

- ***Sensitive to initial centroid selection***

J(**W**,**C**) is non-convex



The K-means algorithm always converges.
The K-means algorithm can be configured to use Manhattan Distance.

K-means is not guaranteed to find a global minimum; it finds only local minimum.

↳ which depends on your initial centroids chosen

How to guarantee a global minimum?

Given the number of clusters and data points, compute the cost function for every case possible, minimum weighted sum of errors will be the global minimum.

Computational impracticable, so can try the method in next slide, but not guaranteed, only possible to find.

This is an NPhard problem

K-means Clustering: Limitations

➢ ***Sensitive to initial centroid selection***

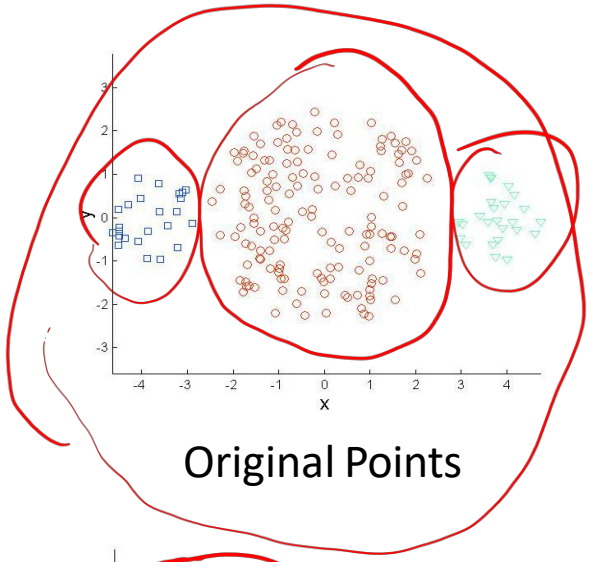
Practically Possible Trials:

- Multiple runs with different random initializations
  - Helps, but probability is not on your side
- Select more than *K* initial centroids and then select among these initial centroids
  - Select most widely separated
- Post-processing
  - Eliminate small clusters that may represent outliers
  - Split 'loose' clusters, i.e., clusters with relatively high J(**W**,**C**)
  - Merge clusters that are 'close' and that have relatively low J(**W**,**C**)

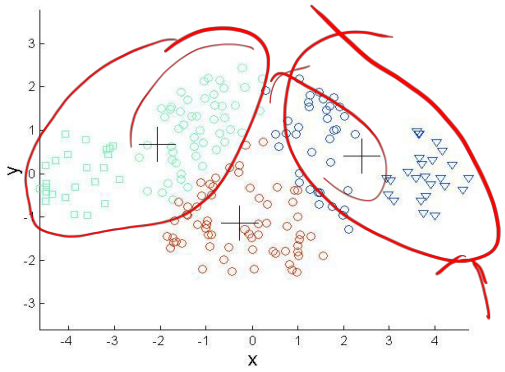
K-Means++: Improved initialization for K-Means

K-means Clustering: Limitations

- ***Assume spherical, equal-sized, similar-density clusters***

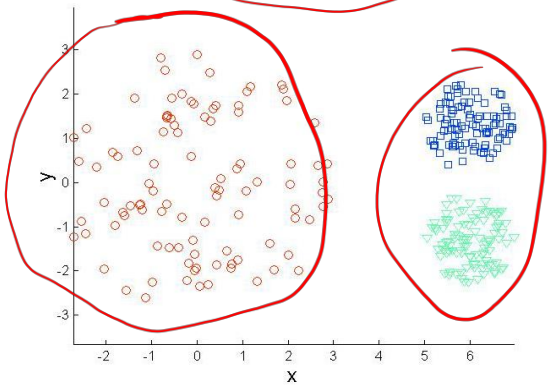


Original Points

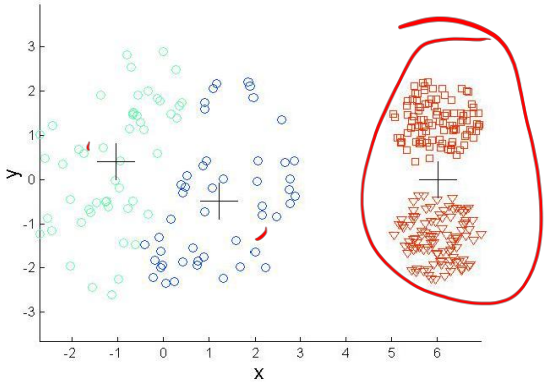


K-means (3 Clusters)

Differ in size!



Original Points



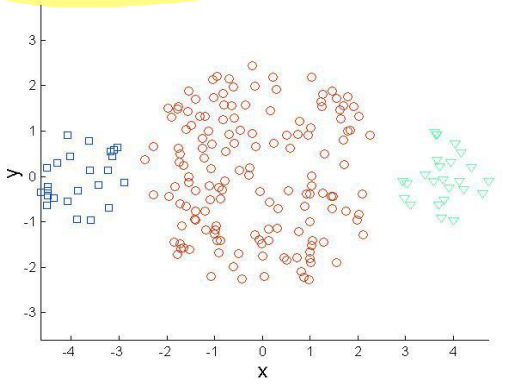
K-means (3 Clusters)

Differ in Density!

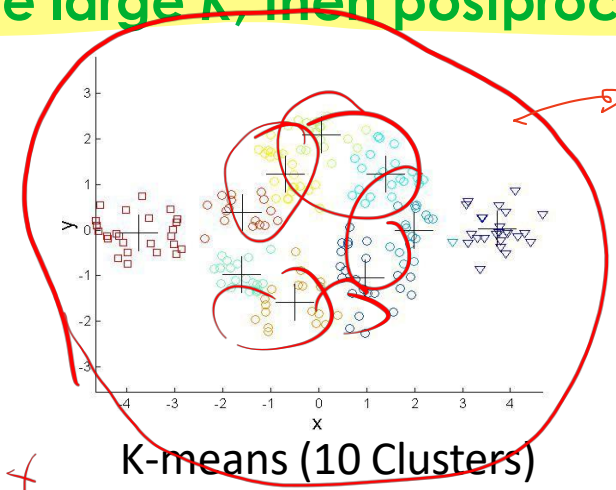
K-means Clustering: Limitations

- ***Assume spherical, equal-sized, similar-density clusters***

Possible Solution: Use large *K*, then postprocess

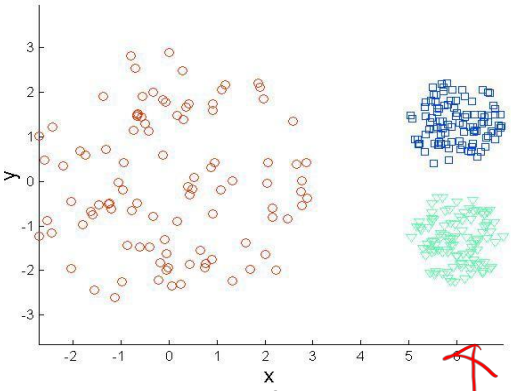


Original Points

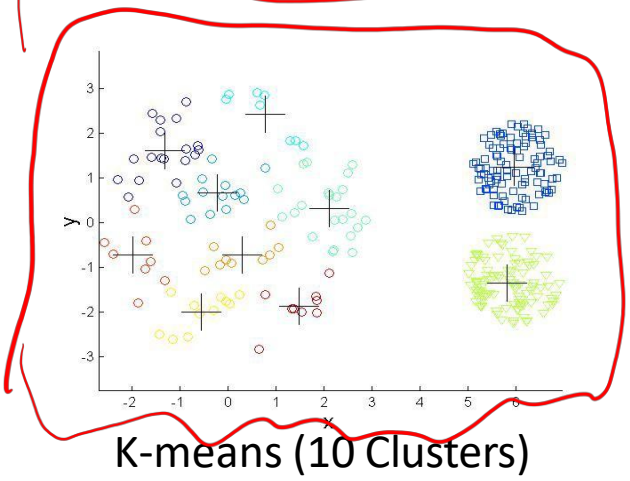


K-means (10 Clusters)

Differ in size



Original Points



K-means (10 Clusters)

Differ in Density

K-means Clustering: Example

Consider the following data points:

Sample ID	1	2	3	4	5	6
Sample	1	2	3	10	11	12

We'd like to group them into two groups by K-means according to Euclidean distance. Assume the initial centroid is Sample 1 for Group A, and Sample 4 for Group B. Upon convergence, what will be the two centres?

Which of the following statements is/are correct?
(a) K-means clustering is suitable for datasets with spiral-shaped clusters.
(b) Given the same set of data points, the number of well-separated clusters that can be formed is always more than the number of center-based clusters that can be formed.
(c) Running K-means clustering multiple times with different random initializations can guarantee reaching the global minimum solution.
(d) Each data point can belong to multiple clusters for K-means clustering.
(e) None
Answer: (e)
(a) is wrong because K-means clustering is suitable for spherical clusters.
(b) is wrong because the number of well-separated clusters that can be formed is always no more than the number of center-based clusters that can be formed, given the same set of data points.
(c) is wrong because it cannot guarantee reaching the global minimum solution. It only increases the likelihood.
(d) is wrong because each data point can belong to only one cluster for K-means clustering. K-means clustering belongs to hard clustering which assigns each data point to exact one cluster.

K-means Clustering: Example

Consider the following data points:

Sample ID	1	2	3	4	5	6
Sample	1	2	3	10	11	12

We'd like to group them into two groups by K-means according to Euclidean distance. Assume the initial centroid is Sample 1 for Group A, and Sample 4 for Group B. Upon convergence, what will be the two centres?

Initialization: $c_A = 1, c_B = 10$

1st iteration:

Assignment: $D(x_1, c_A)=0, D(x_1, c_B)=9$ $x_1 \in group\ A$

$D(x_2, c_A)=1, D(x_2, c_B)=8$ $x_2 \in group\ A$

$D(x_3, c_A)=2, D(x_3, c_B)=7$ $x_3 \in group\ A$

$D(x_4, c_A)=9, D(x_4, c_B)=0$ $x_4 \in group\ B$

$D(x_5, c_A)=10, D(x_5, c_B)=1$ $x_5 \in group\ B$

$D(x_6, c_A)=11, D(x_6, c_B)=2$ $x_6 \in group\ B$

Update:

$$c_A = \frac{x_1 + x_2 + x_3}{3} = 2$$

$$c_B = \frac{x_4 + x_5 + x_6}{3} = 11$$

K-means Clustering: Example

Consider the following data points:

Sample ID	1	2	3	4	5	6
Sample	1	2	3	10	11	12

We'd like to group them into two groups by K-means according to Euclidean distance. Assume the initial centroid is Sample 1 for Group A, and Sample 4 for Group B. Upon convergence, what will be the two centres?

$c_A = 2, c_B = 11$

2nd iteration:

Assignment:

$D(x_1, c_A)=1,\ D(x_1, c_B)=10$ → $x_1 \in group\ A$

$D(x_2, c_A)=0,\ D(x_2, c_B)=9$ → $x_2 \in group\ A$

$D(x_3, c_A)=1,\ D(x_3, c_B)=8$ → $x_3 \in group\ A$

$D(x_4, c_A)=8,\ D(x_4, c_B)=1$ → $x_4 \in group\ B$

$D(x_5, c_A)=9,\ D(x_5, c_B)=0$ → $x_5 \in group\ B$

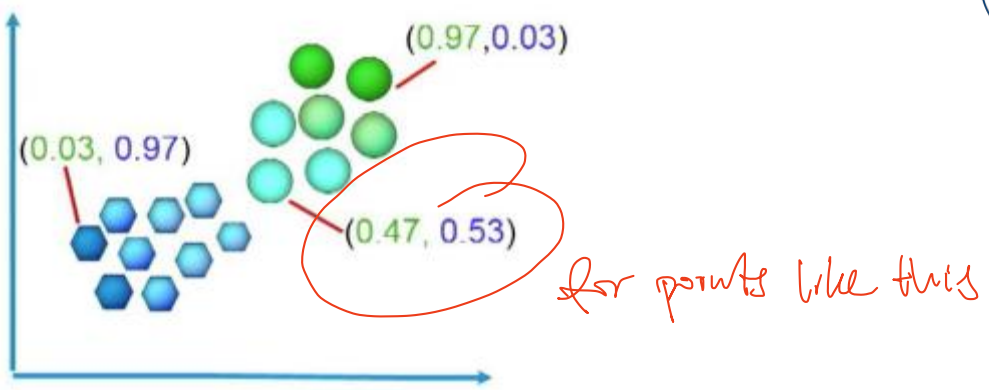
$D(x_6, c_A)=10,\ D(x_6, c_B)=1$ → $x_6 \in group\ B$

Update:

$$c_A = \frac{x_1 + x_2 + x_3}{3} = 2$$

$$c_B = \frac{x_4 + x_5 + x_6}{3} = 11$$

No changes!



Fuzzy C-Means Clustering

-Soft Clustering

Fuzzy C-Means Clustering

➢ Objective Function

Find the optimal values for $\mathbf{W}^*_{N\times C}$ and $\mathbf{C}^*_{C\times d}$ which minimizes:

$$J(\mathbf{W},\mathbf{C}) = \sum_{i=1}^{N}\sum_{k=1}^{C} (w_{ik})^r \|\mathbf{x}_i - \boldsymbol{c}_k\|^2$$

Squared Euclidean distance

where $w_{ik} \in [0,1]$: $w_{ik} = \dfrac{1}{\sum_{j=1}^{C}\left(\frac{\|\mathbf{x}_i-\boldsymbol{c}_k\|}{\|\mathbf{x}_i-\boldsymbol{c}_j\|}\right)^{\frac{2}{r-1}}}$ tells the degree of $\mathbf{x}_i$ belonging to cluster $\boldsymbol{c}_k$

$r > 1$ is the fuzzifier: determines the level of cluster fuzziness.

(usually, $1.25 \le r \le 2$)

In Fuzzy C-Means clustering, the membership degrees of a data point across all clusters always sum to 1.

Fuzzy C-Means Clustering

➢ Algorithm Formulation

Same algorithm,

1. Assignment Step (Fix $\mathbf{C}_{C\times d}$ and update $\mathbf{W}_{N\times C}$)

$$w_{ik} = \frac{1}{\sum_{j=1}^{C}\left(\frac{\|\mathbf{x}_i - \boldsymbol{c}_k\|}{\|\mathbf{x}_i - \boldsymbol{c}_j\|}\right)^{\frac{2}{r-1}}}$$

2. Update Step (Fix $\mathbf{W}_{N\times C}$ and update $\mathbf{C}_{C\times d}$)

$$\frac{dJ(\mathbf{W},\mathbf{C})}{d\mathbf{C}} = \begin{bmatrix} \frac{\partial \sum_{i=1}^{N}\sum_{k=1}^{C}(w_{ik})^r(\mathbf{x}_i - \boldsymbol{c}_k)^T(\mathbf{x}_i - \boldsymbol{c}_k)}{\partial \boldsymbol{c}_1} \\ \vdots \\ \frac{\partial \sum_{i=1}^{N}\sum_{k=1}^{C}(w_{ik})^r(\mathbf{x}_i - \boldsymbol{c}_k)^T(\mathbf{x}_i - \boldsymbol{c}_k)}{\partial \boldsymbol{c}_K} \end{bmatrix} = \begin{bmatrix} -2\sum_{i=1}^{N}(w_{i1})^r(\mathbf{x}_i - \boldsymbol{c}_1) \\ \vdots \\ -2\sum_{i=1}^{N}(w_{iC})^r(\mathbf{x}_i - \boldsymbol{c}_K) \end{bmatrix} = \mathbf{0}$$

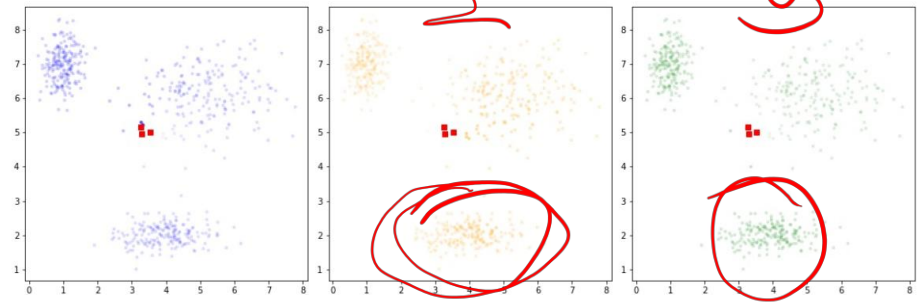
$$\boldsymbol{c}_k = \frac{\sum_{i=1}^{N}(w_{ik})^r\mathbf{x}_i}{\sum_{i=1}^{N}(w_{ik})^r}, k = 1, \ldots, \mathrm{C}$$

only with fixed w_i, the cost function is convex

Fuzzy C-Means Clustering

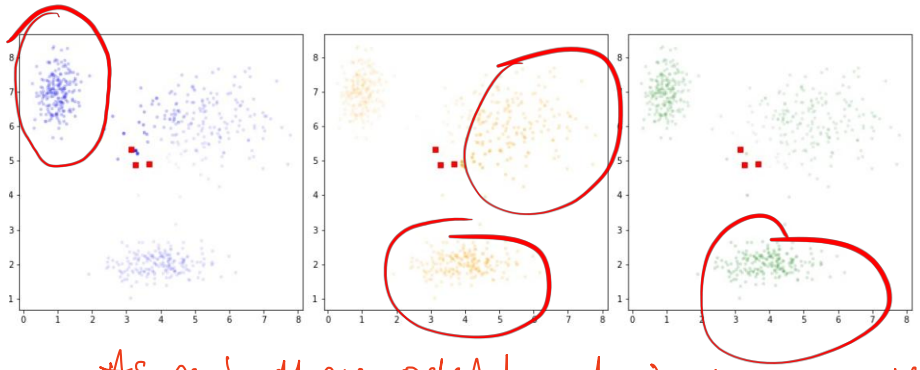
➢ Visualization

Iteration 1



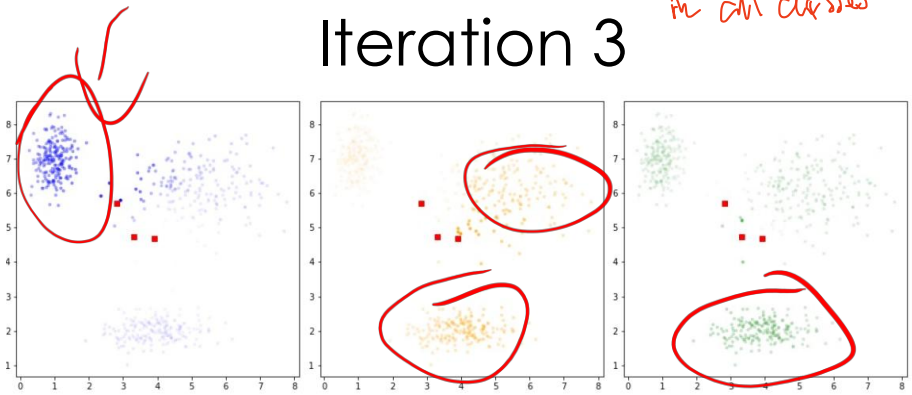
At the start points have membership in all classes

Iteration 2

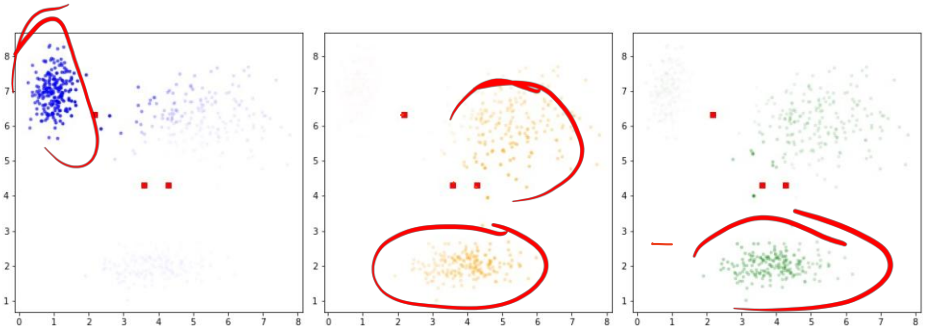


As centroids are selected and w values are updated colours wrt. each class change.

Iteration 3



Iteration 4

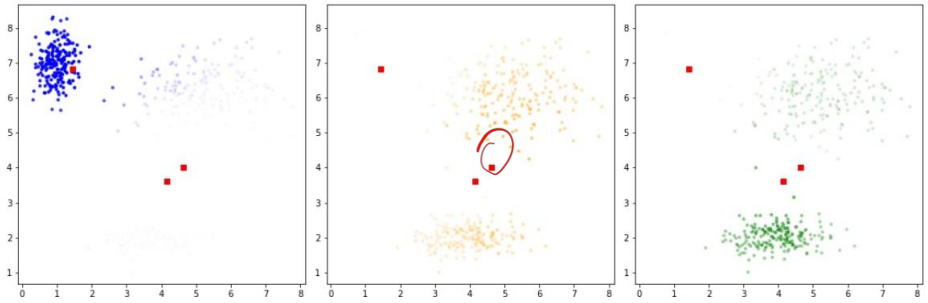


More certain ⇒ stronger membership

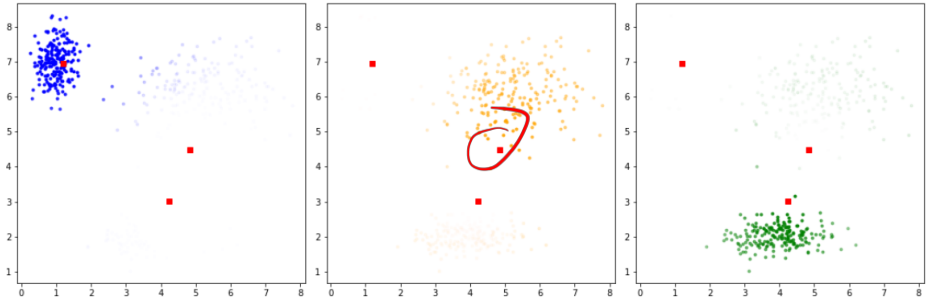
Fuzzy C-Means Clustering

➢ Visualization

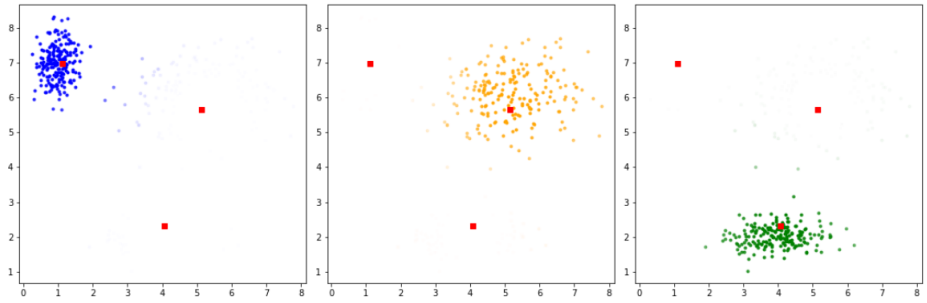
Iteration 5



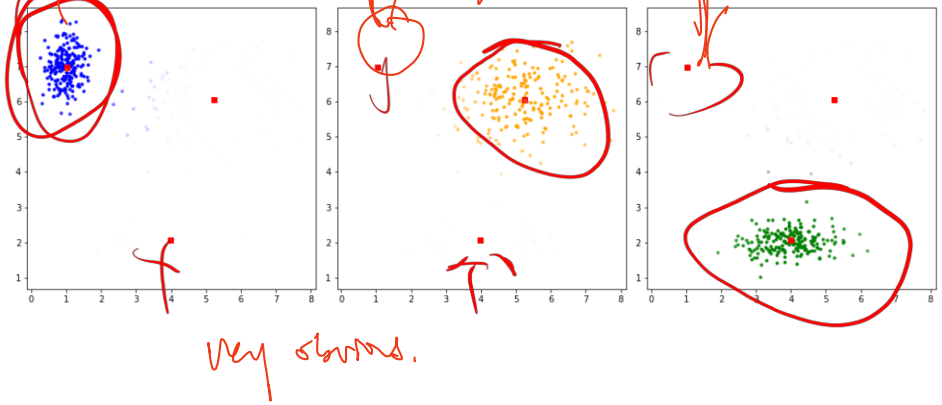
Iteration 6



Iteration 7



Iteration 8



Summary By Quick Quiz

✓ Four Types of Clusters

elbow method to select optimal k

✓ Two Types of Clustering

✓ K-means Clustering

1. Centroid Initialization
2. Assignment (to nearest centroid)
3. Update

$$\boldsymbol{c}_k = \frac{\sum_{i=1}^{N} w_{ik}\mathbf{x}_i}{\sum_{i=1}^{N} w_{ik}}, k = 1, \ldots, K$$

Stop criteria: centroid - centroid < threshold

✓ Fuzzy C-means Clustering

1. Centroid Initialization
2. Assignment

$$w_{ik} = \frac{1}{\sum_{j=1}^{C} \left(\frac{\|\mathbf{x}_i - \boldsymbol{c}_k\|}{\|\mathbf{x}_i - \boldsymbol{c}_j\|} \right)^{\frac{2}{r-1}}}$$

3. Update

$$\boldsymbol{c}_k = \frac{\sum_{i=1}^{N} (w_{ik})^r \mathbf{x}_i}{\sum_{i=1}^{N} (w_{ik})^r}, k = 1, \ldots, C$$

EE2213 Introduction to AI

Lecture 11: Neural Networks

Dr. WANG Si
si.wang@nus.edu.sg
Electrical and Computer Engineering Department
National University of Singapore

OVERVIEW OF COURSE CONTENTS

- **Introduction (Shaojing)**
  - What is AI
  - Applications of AI
  - AI agent
- **Search (Shaojing)**
  - Uninformed search algorithms: breadth-first, depth-first, uniform-cost(Dijkstra's algorithm)
  - Informed search algorithms: greedy best-first, A*
  - Applications
- **Optimisation (Shaojing)**
  - Linear programming
  - Convex problems
  - Applications
- **Machine learning (Wang Si) (Weeks 6-9)**
  - Supervised and unsupervised learning: regression, classification, clustering
  - Neural networks and deep learning
  - Applications
- **Knowledge representation (Wang Si) (Weeks 10-11)**
  - Knowledge Representation and Reasoning
  - Propositional Logic
  - Applications
- **Ethical considerations (Shaojing)**
  - Bias in AI
  - Privacy concerns
  - Societal impact

(Week 10, Monday)

No Class on 20 Oct.! A recording will be uploaded on Canvas.

AGENDA

- We will discuss:
  - Perceptron
  - Multilayer Perceptron (MLP)
  - MLPs for Image Classification

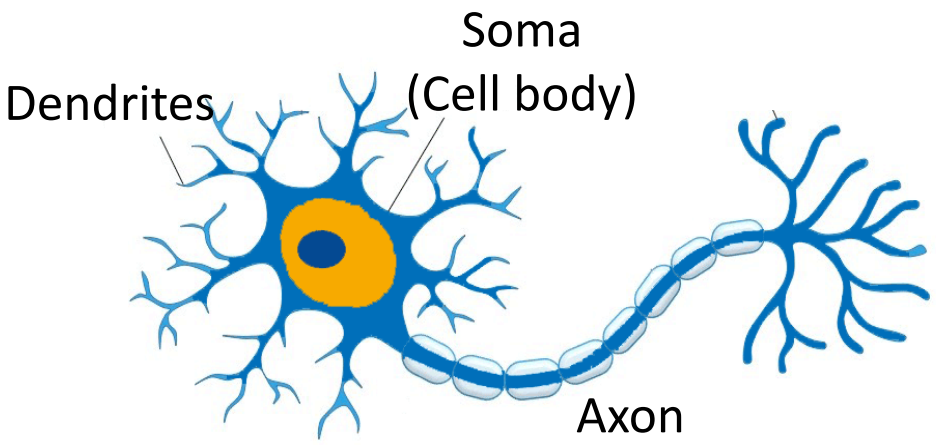
At the end of this lecture, you should be able to:

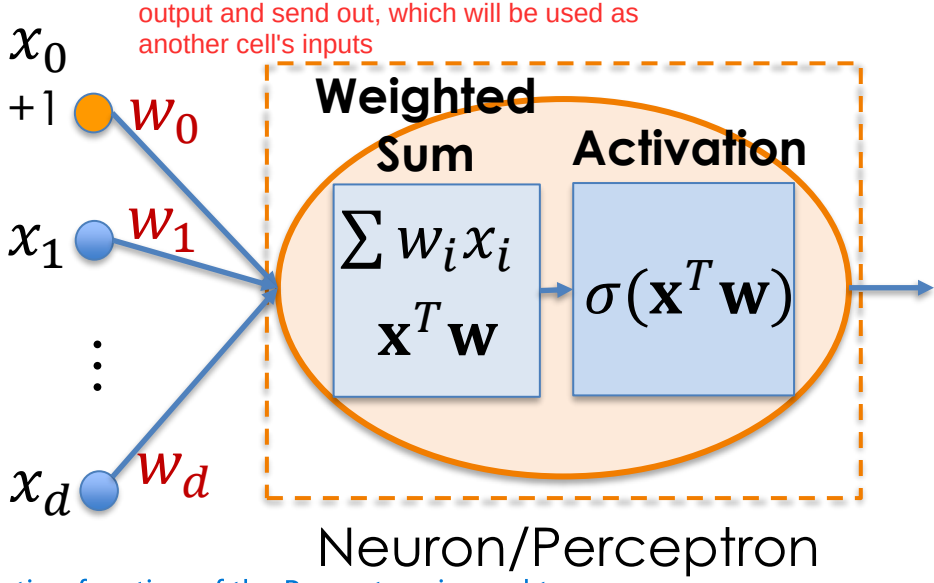
- ✓ Describe perceptron model
- ✓ Explain the architecture and learning process of MLP
- ✓ Apply MLP for image classification and identify the limitations

Perceptron

Biological Neuron & Perceptron

each perceptron takes inputs, run through trained weights, activation function on the output and send out, which will be used as another cell's inputs

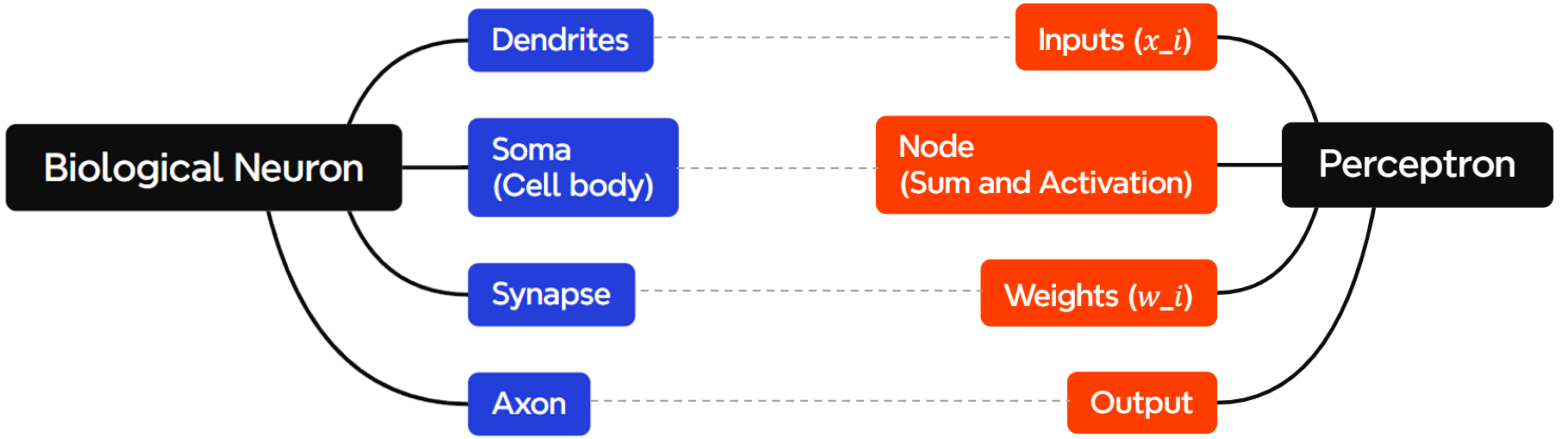




Neuron/Perceptron

The input to the activation function (xTw) of the perceptron is a linear combination of features. relu means not probabilistic output

The activation function of the Perceptron is used to transform the predicted output, not every input feature individually.

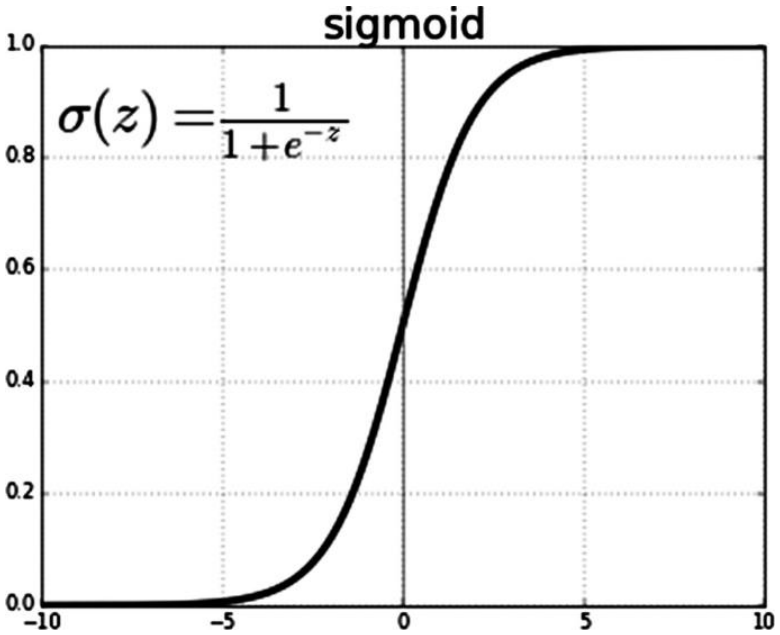


$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}$$

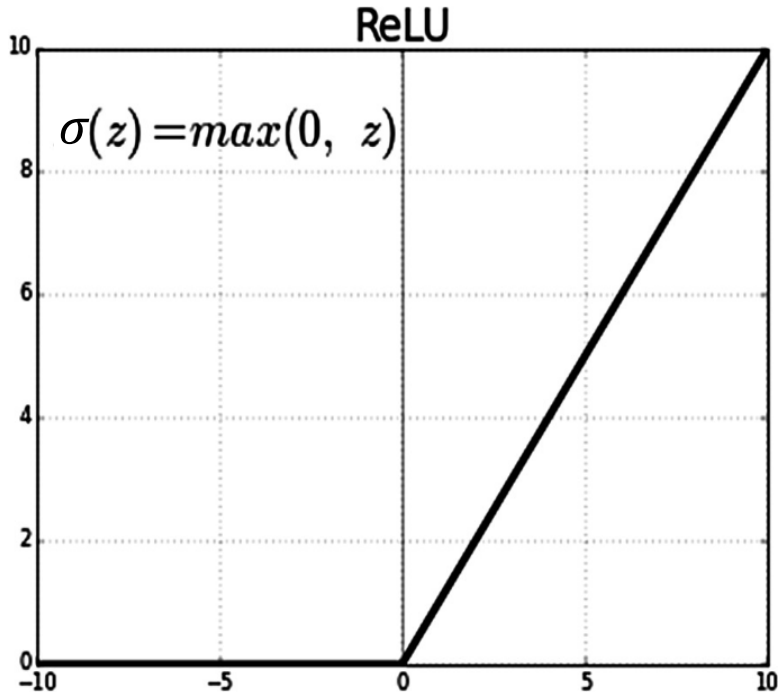
$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$$

Perceptron

➢ Activation Function (Non-Linear)



Type your text



rectified linear unit

if z <0, function is
0, else, z

not probabilistic

What might be the reasons for the difference in performance between the single-layer neural network and the multi-layer neural network?

Solution:

The difference in performance between the single-layer neural network and the multi-layer neural network can be attributed to the fact that the single-layer neural network is a linear classifier, while the multi-layer neural network can learn non-linear decision boundaries. In this case, the multi-layer neural network is able to capture more complex relationships between the input features and the output label, leading to better performance on the validation set.

How might you modify the single-layer neural network to improve its performance, and what are the advantages and disadvantages of doing so?

Solution:

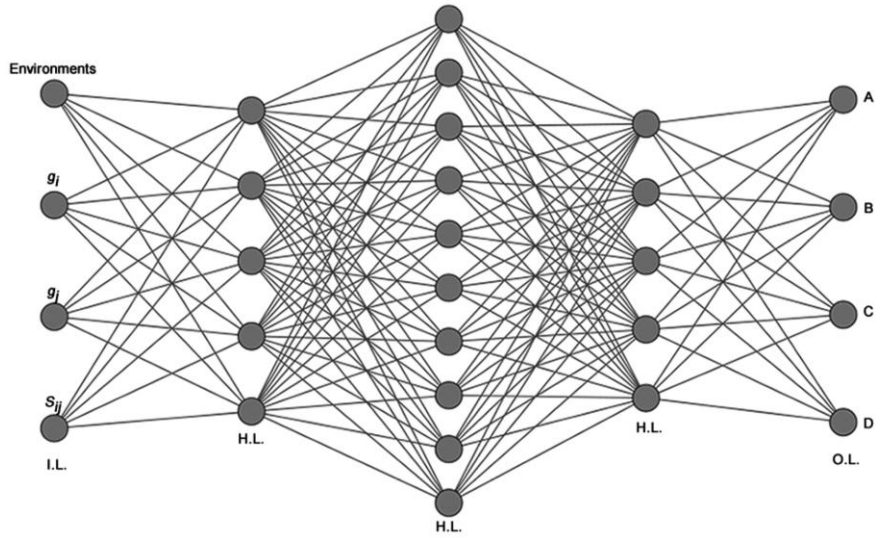
To improve the performance of the single-layer neural network, you could add more features, such as polynomial or interaction terms, to capture non-linear relationships between the input features and the output labels. However, this approach can quickly become computationally expensive and may lead to overfitting if the number of features is too high. Note that the final decision is still based on a linear combination of the transformed features.

What techniques could you use to improve the performance of the multi-layer neural network?

Solution:

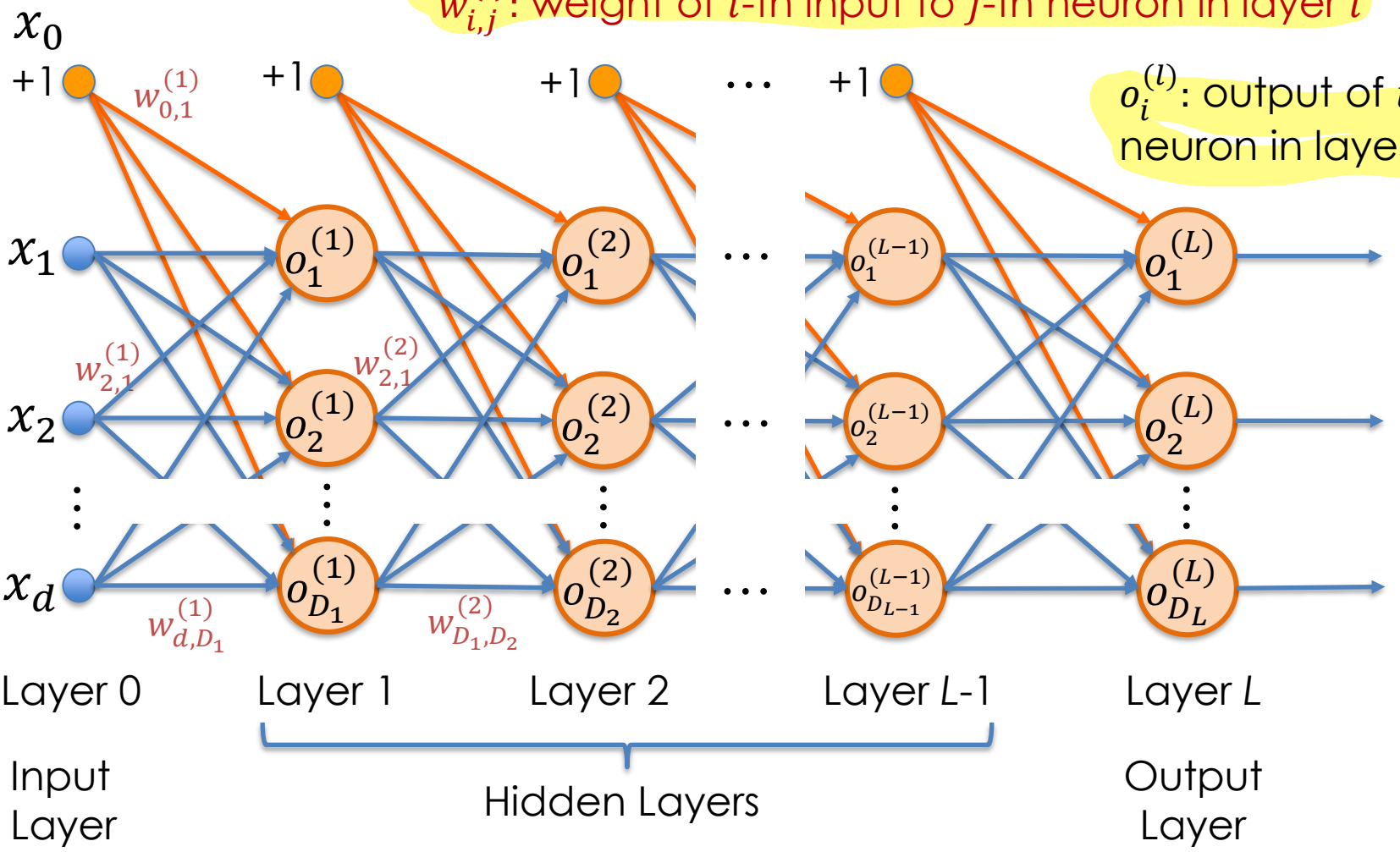
You could try adding more hidden layers (increasing the depth of the network) or increasing the number of neurons in the hidden layer (increasing the width). However, the complexity of the computed function grows exponentially with the depth of an architecture, and increases more slowly with the width. You can find out more in the paper by Eldan and Shamir (2016) on the expressive power of deep neural networks. For further details, see this link. Hence, increasing the depth is generally more efficient. Additionally, while both approaches lead to the multi-layer neural network increasing its capacity to learn more complex patterns, it can also increase the risk of overfitting.

Multilayer Perceptron (MLP)



Multilayer Perceptron (MLP)

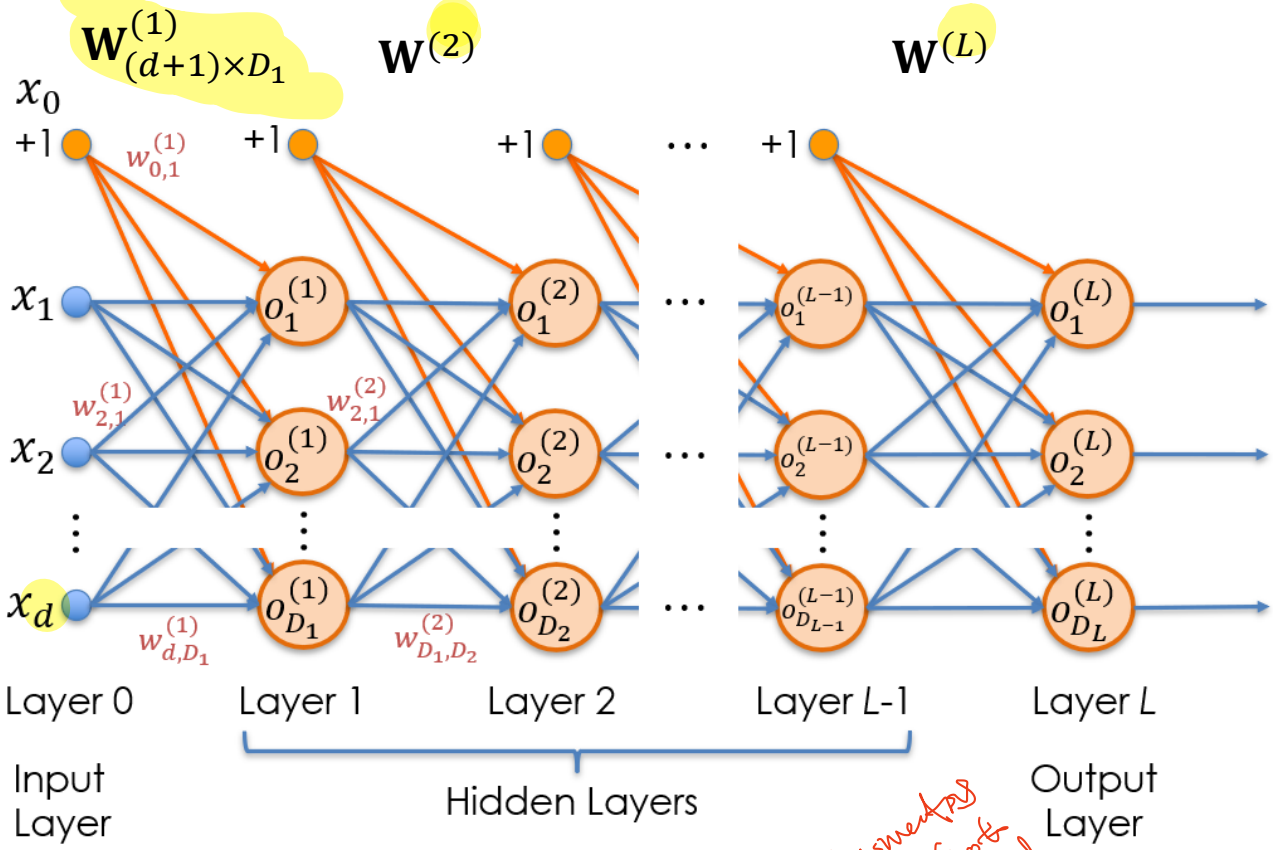
➢ Architecture



Note: D_l denotes the number of neurons in layer l

Multilayer Perceptron (MLP)

➢ Architecture



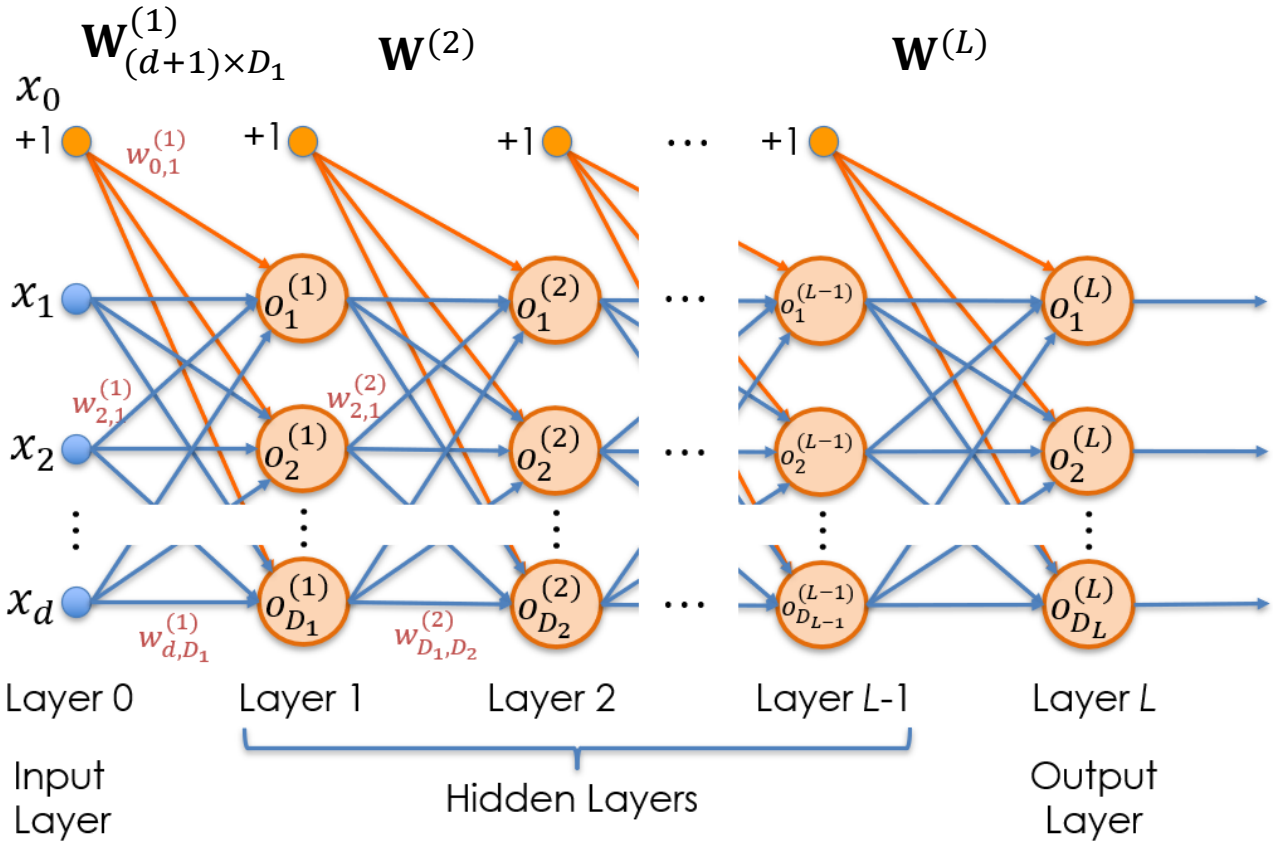
e.g., 3 inputs to layer 3, 2 neurons in layer 3

$$\mathbf{W}^{(l)} = \begin{bmatrix} w^{(l)}_{0,1} & \cdots & w^{(l)}_{0,D_l} \\ \vdots & \ddots & \vdots \\ w^{(l)}_{D_{l-1},1} & \cdots & w^{(l)}_{D_{l-1},D_l} \end{bmatrix} = \begin{bmatrix} \mathbf{w}^{(l)}_1 & \cdots & \mathbf{w}^{(l)}_{D_l} \end{bmatrix}$$

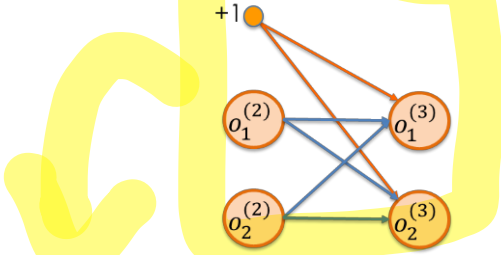
Weight vector for first neuron in layer l

Multilayer Perceptron (MLP)

➢ Architecture



e.g., 3 inputs to layer 3, 2 neurons in layer 3



$$\mathbf{W}^{(3)} = \begin{bmatrix} w^{(3)}_{0,1} & w^{(3)}_{0,2} \\ w^{(3)}_{1,1} & w^{(3)}_{1,2} \\ w^{(3)}_{2,1} & w^{(3)}_{2,2} \end{bmatrix}$$

$\mathbf{f}_{\mathbf{W}}(\mathbf{x}) = Softmax([\mathbf{1}, ReLU([\mathbf{1}, ReLU(\mathbf{x}^T\mathbf{W}^{(1)})]\mathbf{W}^{(2)})]\mathbf{W}^{(3)})$

where $\mathbf{x}$ is the input vector augmented with 1, $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$ and $\mathbf{W}^{(3)}$ are weight matrices. Which of the following statements is/are correct?

(a) This MLP has 3 layers of neurons.
(b) The number of neurons in each layer of this MLP is equal to the number of rows in the weight matrix for that layer.
(c) The number of image classes is equal to the number of columns in $\mathbf{W}^{(3)}$.
(d) The dimension of $\mathbf{x}$ is equal to the dimension of the flattened image.
(e) None

Answer: (a)(c)

(b) is wrong because the number of neurons in each layer of this MLP should be equal to the number of columns in the weight matrix for that layer.

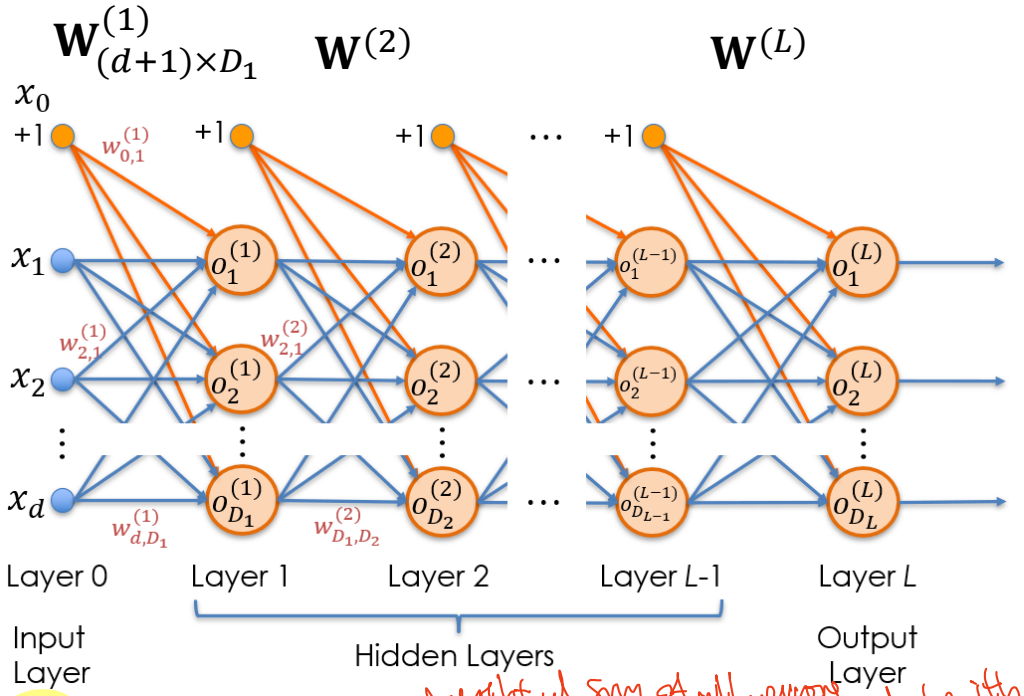
(d) is wrong because the dimension of x is equal to the dimension of the flattened image + 1 (for the bias)

$$\mathbf{W}^{(l)} = \begin{bmatrix} w^{(l)}_{0,1} & \cdots & w^{(l)}_{0,D_l} \\ \vdots & \ddots & \vdots \\ w^{(l)}_{D_{l-1},1} & \cdots & w^{(l)}_{D_{l-1},D_l} \end{bmatrix} = \begin{bmatrix} \mathbf{w}^{(l)}_1 & \cdots & \mathbf{w}^{(l)}_{D_l} \end{bmatrix}$$

Weight vector for first neuron in layer l

Multilayer Perceptron (MLP)

➢ Architecture



$$\mathbf{W}^{(l)} = \begin{bmatrix} w^{(l)}_{0,1} & \cdots & w^{(l)}_{0,D_l} \\ \vdots & \ddots & \vdots \\ w^{(l)}_{D_{l-1},1} & \cdots & w^{(l)}_{D_{l-1},D_l} \end{bmatrix} = \begin{bmatrix} \mathbf{w}^{(l)}_1 & \cdots & \mathbf{w}^{(l)}_{D_l} \end{bmatrix}$$

Weight vector for first neuron in layer l

$$o^{(l)}_i = \sigma^{(l)}\left(\sum_{j=1}^{D_{l-1}} o^{(l-1)}_j w^{(l)}_{j,i} + w^{(l)}_{0,i}\right) = \sigma^{(l)}([1, \mathbf{o}^{(l-1)}]\mathbf{w}^{(l)}_i)$$

$$\mathbf{o}^{(l)} = \boldsymbol{\sigma}^{(l)}([1, \mathbf{o}^{(l-1)}]\mathbf{W}^{(l)})$$

A **MLP** is essentially a **nested function**

$$\mathbf{f}_{\mathbf{W}}(\mathbf{x}) = \boldsymbol{\sigma}^{(L)}([1, \ldots \boldsymbol{\sigma}^{(2)}([1, \boldsymbol{\sigma}^{(1)}(\mathbf{x}^T\mathbf{W}^{(1)})]\mathbf{W}^{(2)}) \ldots]\mathbf{W}^{(L)})$$

Multilayer Perceptron (MLP)

➢ Architecture

Example: A MLP of 2 layers has been constructed as:

$$\mathbf{f_W}(\mathbf{x}) = \boldsymbol{\sigma}\left(\left[1, \boldsymbol{\sigma}\left(\mathbf{x}^T \mathbf{W}^{(1)}\right)\right]\mathbf{W}^{(2)}\right),$$

where $\mathbf{x} = [1,1,3]^T$, $\mathbf{W}^{(1)} = \mathbf{W}^{(2)} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \end{bmatrix}$, $\sigma(z) = \max(0, z)$.

Compute network output $\mathbf{f_W}(\mathbf{x})$.

Multilayer Perceptron (MLP)

Architecture

Example: A MLP of 2 layers has been constructed as:

$$\mathbf{f}_{\mathbf{W}}(\mathbf{x}) = \boldsymbol{\sigma}\left(\left[1, \boldsymbol{\sigma}\left(\mathbf{x}^T \mathbf{W}^{(1)}\right)\right]\mathbf{W}^{(2)}\right),$$

where $\mathbf{x} = [1,1,3]^T$, $\mathbf{W}^{(1)} = \mathbf{W}^{(2)} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \end{bmatrix}$, $\sigma(z) = \max(0, z)$.

Compute network output $\mathbf{f}_{\mathbf{W}}(\mathbf{x})$.

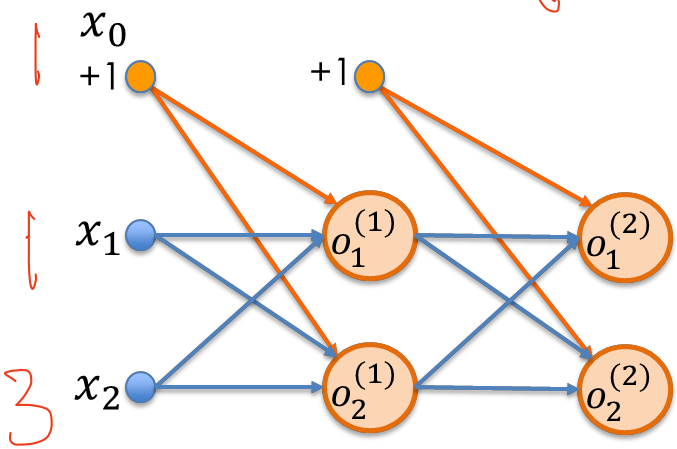
Solution:

$$\boldsymbol{\sigma}\left(\mathbf{x}^T \mathbf{W}^{(1)}\right) = \boldsymbol{\sigma}\left([1,1,3]\begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \end{bmatrix}\right)$$

$$= \boldsymbol{\sigma}([2, -1]) = [2,0]$$

$$\boldsymbol{\sigma}\left(\left[1, \boldsymbol{\sigma}\left(\mathbf{x}^T \mathbf{W}^{(1)}\right)\right]\mathbf{W}^{(2)}\right) = \boldsymbol{\sigma}\left([1,2,0]\begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \end{bmatrix}\right)$$

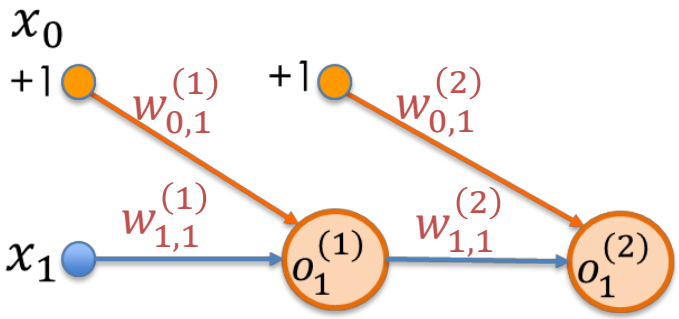
$$= \boldsymbol{\sigma}([-1, -2]) = [0,0]$$



Multilayer Perceptron (MLP)

➢ Architecture

Why non-linear activation function $\sigma(z)$?



If linear/affine activation function:

$$\sigma(z) = az + b$$

$$o_1^{(1)} = \sigma(w_{0,1}^{(1)} + w_{1,1}^{(1)} x_1)$$

$$= a\left(w_{0,1}^{(1)} + w_{1,1}^{(1)} x_1\right) + b$$

$$= \left(aw_{0,1}^{(1)} + b\right) + aw_{1,1}^{(1)} x_1$$

linear: output whatever it receives
affine = add a bias term

$$o_1^{(2)} = \sigma(w_{0,1}^{(2)} + w_{1,1}^{(2)} o_1^{(1)})$$

$$= a\left(w_{0,1}^{(2)} + w_{1,1}^{(2)} o_1^{(1)}\right) + b$$

$$= \left(aw_{0,1}^{(2)} + b\right) + aw_{1,1}^{(2)} o_1^{(1)}$$

$$= \left(aw_{0,1}^{(2)} + b\right) + aw_{1,1}^{(2)}\left(\left(aw_{0,1}^{(1)} + b\right) + aw_{1,1}^{(1)} x_1\right)$$

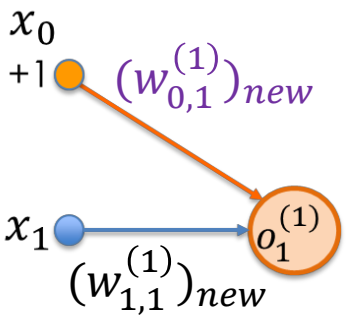
$$= \left(aw_{0,1}^{(2)} + a^2 w_{1,1}^{(2)} w_{0,1}^{(1)} + abw_{1,1}^{(2)} + b\right) + a^2 w_{1,1}^{(2)} w_{1,1}^{(1)} x_1$$

$(w_{0,1}^{(1)})_{new}$

$(w_{1,1}^{(1)})_{new}$

then the whole network is just a linear regression

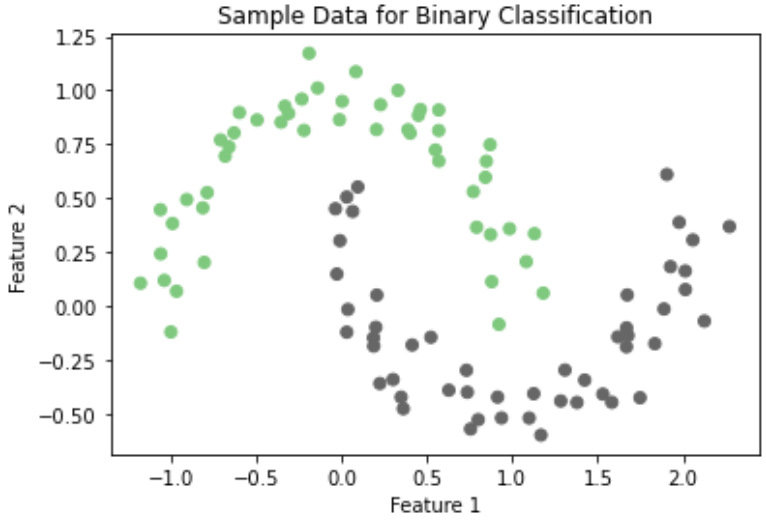
poor performance

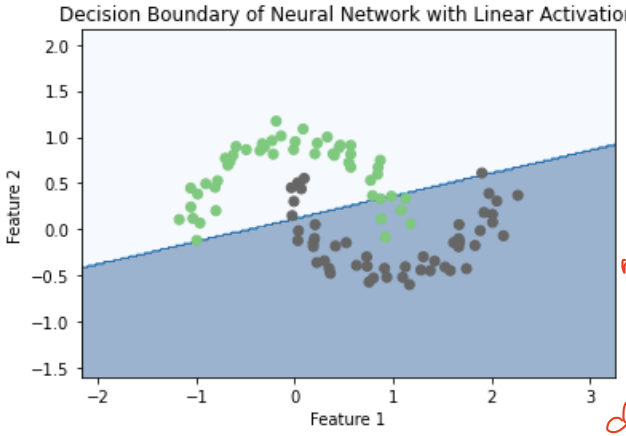


With Linear/affine $\sigma(z)$, MLP=Single-layer Perceptron!

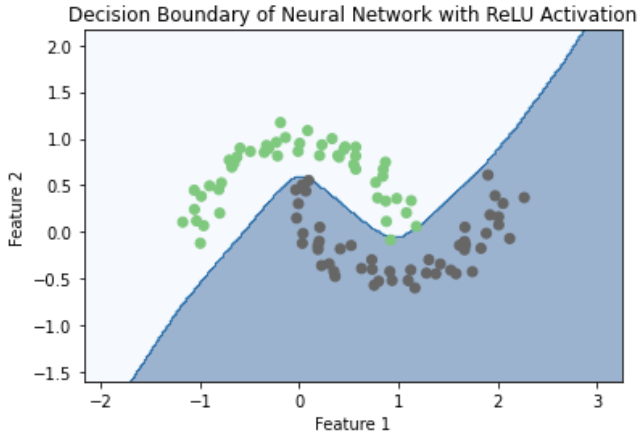
Multilayer Perceptron (MLP)

➢ Visualization of Decision Boundaries





Linear/Affine Activation

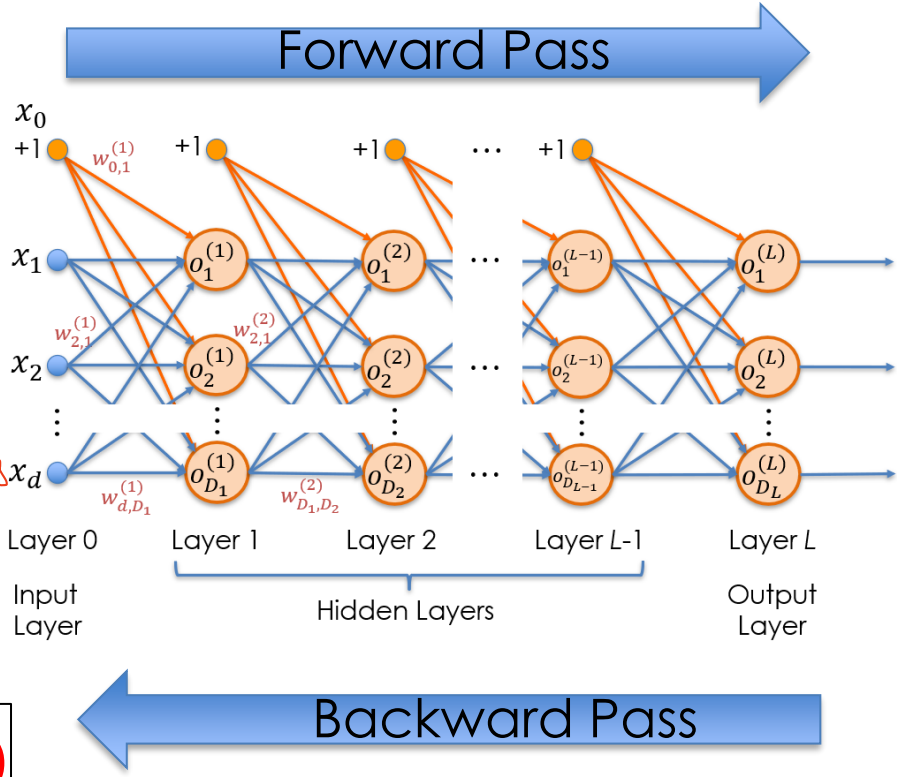


ReLU Activation

Multilayer Perceptron (MLP)

Learning (W)

1. Random Initialization of **W**
2. Forward Pass (fixed **W**)
   i. Compute output responses
3. **Backward Pass** (update **W**)
   i.e., **Backpropagation**
   i. Propagate the errors backward
   ii. Update **W**



$$\mathbf{W}_{new} = \mathbf{W}_{old} - \eta \nabla_{\mathbf{W}} J(\mathbf{W}_{old})$$

In essence, Backpropagation= Gradient Decent!

Multilayer Perceptron (MLP)

➢ Learning (W): Backpropagation

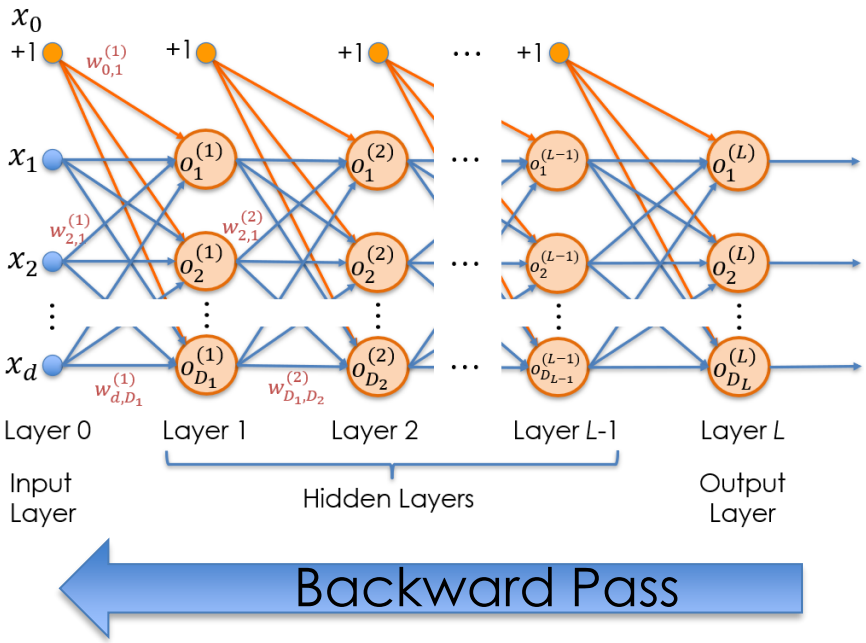
$$\mathbf{W}_{new} = \mathbf{W}_{old} - \eta \nabla_{\mathbf{W}} J(\mathbf{W}_{old})$$

$$\nabla_{\mathbf{W}} J(\mathbf{W}_{old}) = \frac{1}{N} \sum \nabla_{\mathbf{W}} \mathcal{L}(\mathbf{W}_{old})$$

$$\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}} = \frac{\partial \mathcal{L}}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial w_{i,j}^{(l)}}$$

where $z_j^{(l)} = (\mathbf{a}^{(l)})^T \mathbf{w}_j^{(l)}$

$\mathbf{a}^l$: input vector to layer l $= \begin{cases} \mathbf{x}, if\ l = 1 \\ [1, \mathbf{o}^{(l-1)}]^T, else \end{cases}$



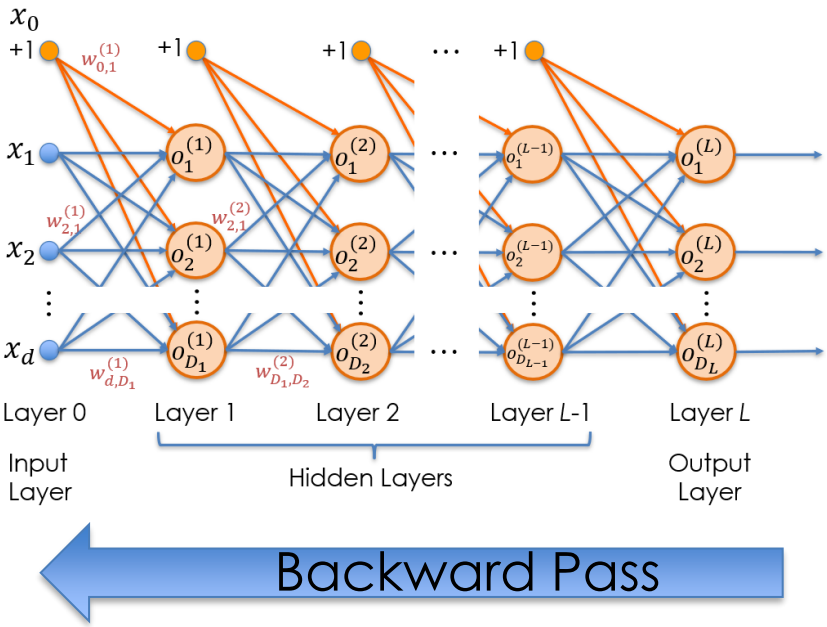
Multilayer Perceptron (MLP)

➢ Learning (W): Backpropagation

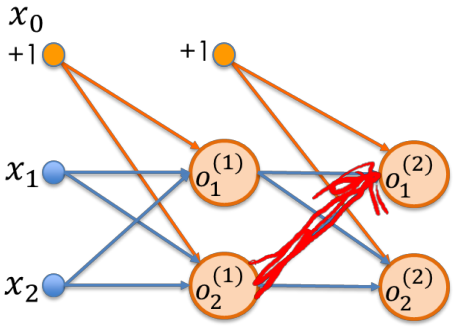
$$\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}} = \frac{\partial \mathcal{L}}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial w_{i,j}^{(l)}}$$

$$\frac{\partial z_j^{(l)}}{\partial w_{i,j}^{(l)}} = \frac{\partial}{\partial w_{i,j}^{(l)}} \left(\sum_m w_{m,j}^{(l)} a_m^{(l)} \right)$$

$$= a_i^{(l)}$$



e.g.,



$$\frac{\partial z_1^{(2)}}{\partial w_{2,1}^{(2)}} = ?$$

$$\frac{\partial z_1^{(1)}}{\partial w_{0,1}^{(1)}} = ?$$

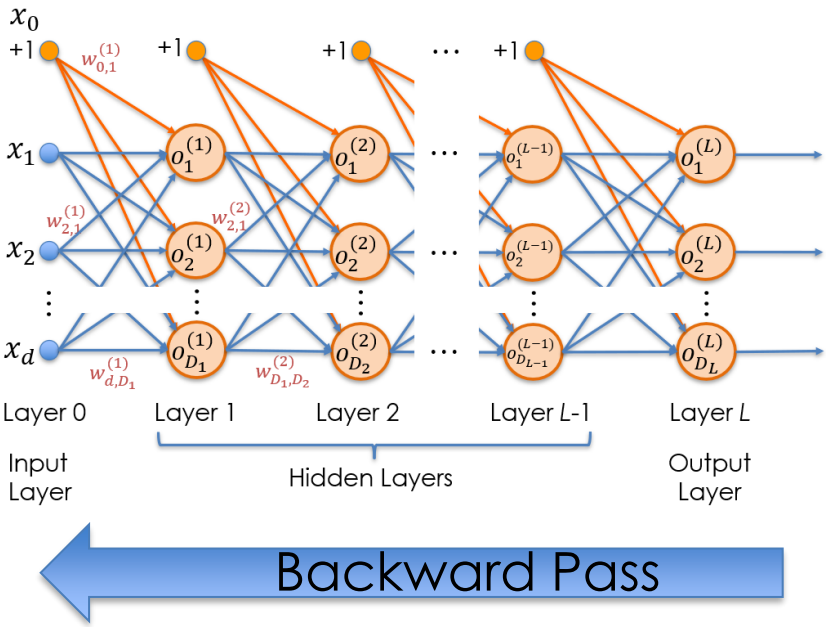
Multilayer Perceptron (MLP)

➢ Learning (W): Backpropagation

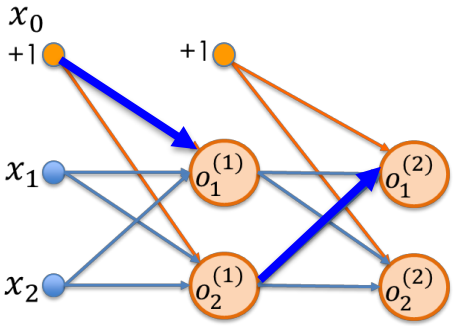
$$\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}} = \frac{\partial \mathcal{L}}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial w_{i,j}^{(l)}}$$

$$\frac{\partial z_j^{(l)}}{\partial w_{i,j}^{(l)}} = \frac{\partial}{\partial w_{i,j}^{(l)}} \left(\sum_m w_{m,j}^{(l)} \, a_m^{(l)} \right)$$

$$= a_i^{(l)}$$



e.g.,



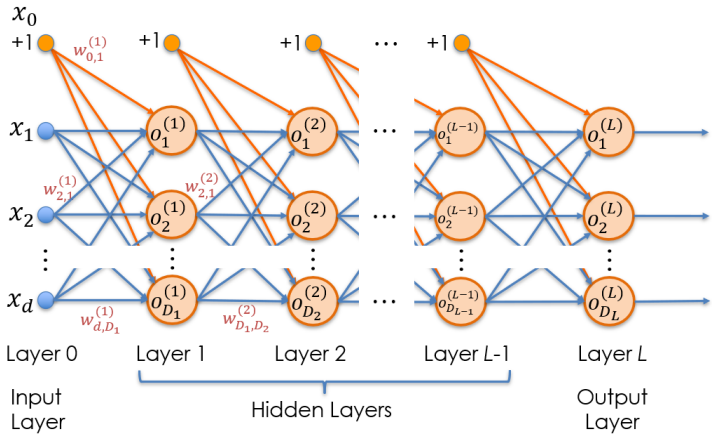
$$\frac{\partial z_1^{(2)}}{\partial w_{2,1}^{(2)}} = o_2^{(1)} \qquad \frac{\partial z_1^{(1)}}{\partial w_{0,1}^{(1)}} = x_0 = 1$$

Multilayer Perceptron (MLP)

➢ Learning (W): Backpropagation

$$\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}} = \frac{\partial \mathcal{L}}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial w_{i,j}^{(l)}}$$

Error term $\epsilon_j^{(l)}$ $\qquad$ $a_i^{(l)}$



Backward Pass

- **If output layer (l =L):**

$$\frac{\partial \mathcal{L}}{\partial z_j^{(L)}} = \frac{\partial \mathcal{L}(o_1^{(L)}, \ldots, o_{D_L}^{(L)})}{\partial z_j^{(L)}} = \sum_{m=1}^{D_L} \frac{\partial \mathcal{L}(o_m^{(L)})}{\partial z_j^{(L)}}$$

$$= \sum_{m=1}^{D_L} \frac{\partial \mathcal{L}(o_m^{(L)})}{\partial o_m^{(L)}} \frac{\partial o_m^{(L)}}{\partial z_j^{(L)}} = \sum_{m=1}^{D_L} \mathcal{L}'(o_m^{(L)}) \sigma^{(L)\prime}(z_j^{(L)})_m$$

$$\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}} = \sum_{m=1}^{D_L} \mathcal{L}'(o_m^{(L)}) \sigma^{(L)\prime}(z_j^{(L)})_m \, a_i^{(L)}$$

Multilayer Perceptron (MLP)

➢ Learning (W): Backpropagation

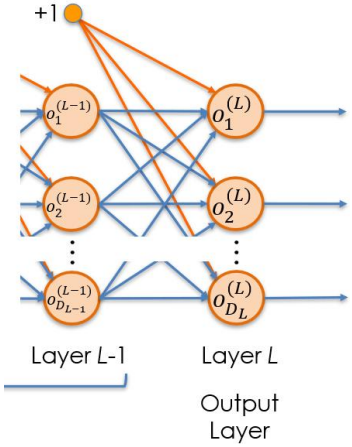
If output layer (l =L):

$$\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}} = \sum_{m=1}^{D_L} \mathcal{L}'(o_m^{(L)}) \sigma^{(L)\prime}(z_j^{(L)})_m \, a_i^{(L)}$$

e.g., If $\boldsymbol{\sigma}^{(L)}(\mathbf{z})_k = \frac{e^{z_k^{(L)}}}{\sum_{s=1}^{D_L} e^{z_s^{(L)}}}$, $\mathcal{L}(\mathbf{W}) = -\sum_{k=1}^{D_L} y_k \ln(p_k)$. What is $\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(L)}}$?

Softmax

Categorical Cross Entropy



Multilayer Perceptron (MLP)

➢ Learning (W): Backpropagation

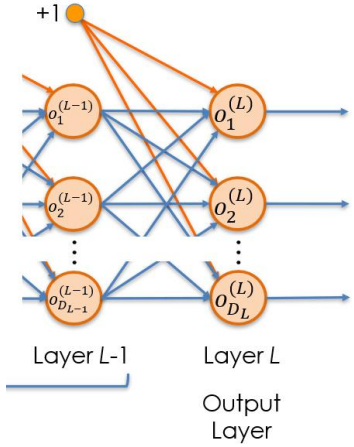
If output layer (l =L):

$$\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}} = \sum_{m=1}^{D_L} \mathcal{L}'(o_m^{(L)}) \sigma^{(L)\prime}(z_j^{(L)})_m \, a_i^{(L)}$$

e.g., If $\boldsymbol{\sigma}^{(L)}(\mathbf{z})_k = \frac{e^{z_k^{(L)}}}{\sum_{s=1}^{D_L} e^{z_s^{(L)}}}$, $\mathcal{L}(\mathbf{W}) = -\sum_{k=1}^{D_L} y_k \ln(p_k)$. What is $\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(L)}}$?

Softmax

Categorical Cross Entropy



$$\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(L)}} = \epsilon_j^{(L)} a_i^{(L)} = a_i^{(L)} (o_j^{(L)} - y_j)$$

a is activation output, e is error term

$$\epsilon_j^{(L)} = \sum_{m=1}^{D_L} \mathcal{L}'(o_m^{(L)}) \boldsymbol{\sigma}^{(L)\prime}(z_j^{(L)})_m$$

$$= \sum_{m=1}^{D_L} -\frac{y_m}{o_m^{(L)}} (\delta_{mj} - o_m^{(L)}) o_j^{(L)}$$

$$= o_j^{(L)} - y_j$$

$$\mathcal{L}'\left(o_m^{(L)}\right) = -\sum_{k=1}^{D_L} y_k \frac{d(\ln(o_k^{(L)}))}{d(o_m^{(L)})} = -\frac{y_m}{o_m^{(L)}}$$

$$\boldsymbol{\sigma}^{(L)\prime}(z_j^{(L)})_m = \frac{d}{d(z_j^{(L)})} \left(\frac{e^{z_m^{(L)}}}{\sum_{s=1}^{D_L} e^{z_s^{(L)}}} \right) = (\delta_{mj} - o_m^{(L)}) o_j^{(L)}$$

$$\delta_{mj} = \begin{cases} 1, if \ m = j \\ 0. if m \neq j \end{cases}$$

Multilayer Perceptron (MLP)

➢ Learning (W): Backpropagation

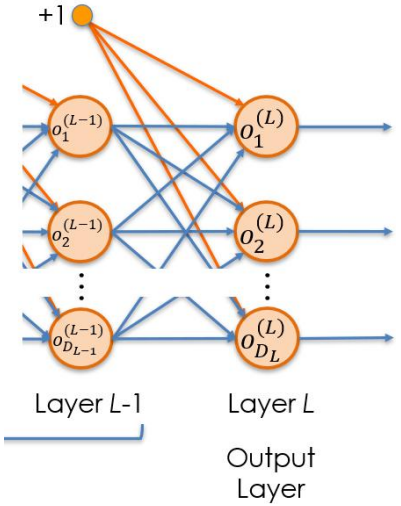
If output layer (l =L):

$$\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}} = \sum_{m=1}^{D_L} \mathcal{L}'(o_m^{(L)}) \sigma^{(L)\prime}(z_j^{(L)})_m \, a_i^{(L)}$$

e.g., If $\boldsymbol{\sigma}^{(L)}(\mathbf{z})_k = \frac{e^{z_k^{(L)}}}{\sum_{s=1}^{D_L} e^{z_s^{(L)}}}$, $\mathcal{L}(\mathbf{W}) = -\sum_{k=1}^{D_L} y_k \ln(p_k)$. What is $\frac{d\mathcal{L}}{d\mathbf{W}^{(L)}}$?

Softmax

Categorical Cross Entropy



Multilayer Perceptron (MLP)

➢ Learning (W): Backpropagation

If output layer (l =L):

$$\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}} = \sum_{m=1}^{D_L} \mathcal{L}'(o_m^{(L)}) \boldsymbol{\sigma}^{(L)\prime}(z_j^{(L)})_m \, a_i^{(L)}$$

e.g., If $\boldsymbol{\sigma}^{(L)}(\mathbf{z})_k = \frac{e^{z_k^{(L)}}}{\sum_{s=1}^{D_L} e^{z_s^{(L)}}}$, $\mathcal{L}(\mathbf{W}) = -\sum_{k=1}^{D_L} y_k \ln(p_k)$. What is $\frac{d\mathcal{L}}{d\mathbf{W}^{(L)}}$?

Softmax — same as no. of inputs

Categorical Cross Entropy — same size as no. of output neurons.

$$\frac{d\mathcal{L}}{d\mathbf{W}^{(L)}} = \mathbf{a}^{(L)} \boldsymbol{\epsilon}^{(L)} = \mathbf{a}^{(L)} (\mathbf{o}^{(L)} - \mathbf{y}) = \mathbf{a}^{(L)} (\mathbf{f}_{\mathbf{W}}(\mathbf{x}) - \mathbf{y})$$

need to check if size of output is correct

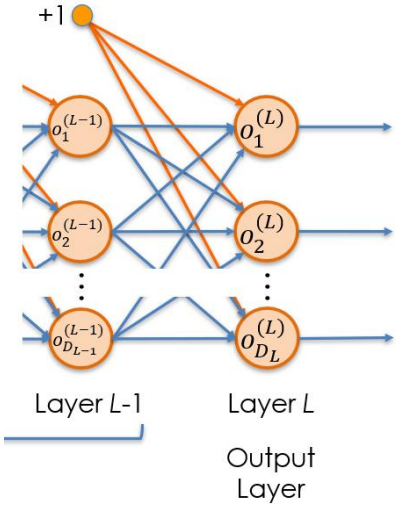
Size: $(D_{L-1}+1) \times D_L$

Size: $(D_{L-1}+1) \times 1$

Size: $1 \times D_L$

same dimension as W_l

$W_{(L+1)\times D}$



$$\frac{dJ}{d\mathbf{W}^{(L)}} = \frac{1}{N} \sum \frac{d\mathcal{L}}{d\mathbf{W}^{(L)}} = \frac{1}{N} (\mathbf{A}^{(L)})^T (\mathbf{F}_{\mathbf{W}}(\mathbf{X}) - \mathbf{Y})$$

mean of loss function

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & \cdots & x_{N,d} \end{bmatrix}$$

$\mathbf{Y}$ has a size of $N \times D_L$

$\mathbf{A}^{(L)}$ has a size of $N \times (D_{L-1}+1)$

Multilayer Perceptron (MLP)

➢ Learning (W): Backpropagation

$$\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}} = \frac{\partial \mathcal{L}}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial w_{i,j}^{(l)}}$$

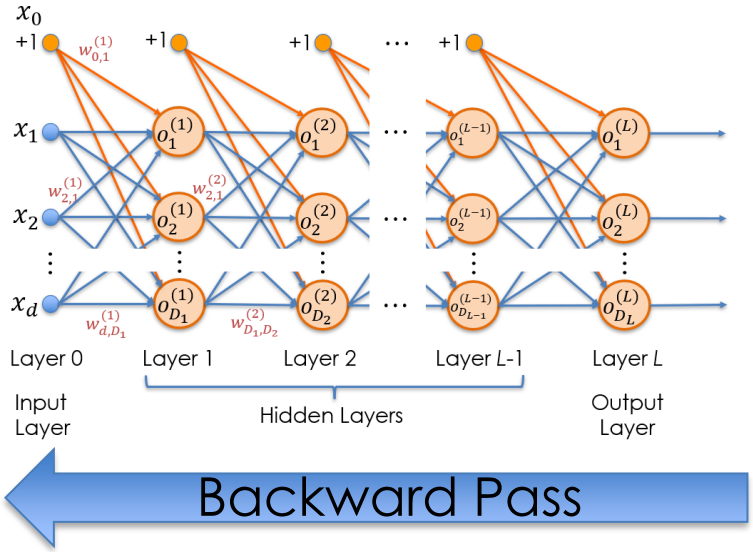
$\epsilon_j^{(l)}$ $\quad$ $a_i^{(l)}$

- **If hidden layer ($l < L$):**

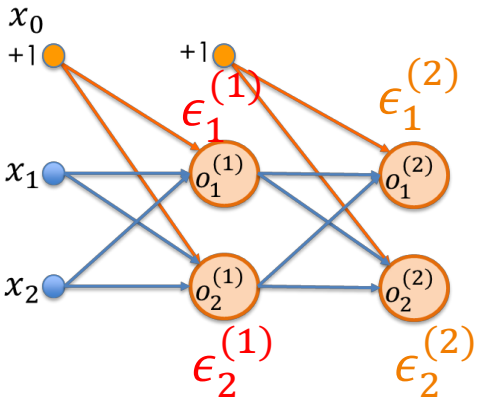
$$\frac{\partial \mathcal{L}}{\partial z_j^{(l)}} = \frac{\partial \mathcal{L}(z_1^{(l+1)}, \dots, z_{D_{l+1}}^{(l+1)})}{\partial z_j^{(l)}} = \sum_{m=1}^{D_{l+1}} \frac{\partial \mathcal{L}(z_m^{(l+1)})}{\partial z_j^{(l)}}$$

$$= \sum_{m=1}^{D_{l+1}} \frac{\partial \mathcal{L}(z_m^{(l+1)})}{\partial z_m^{(l+1)}} \frac{\partial z_m^{(l+1)}}{\partial z_j^{(l)}} = \sum_{m=1}^{D_{l+1}} \epsilon_m^{(l+1)} \frac{\partial z_m^{(l+1)}}{\partial o_j^{(l)}} \frac{\partial o_j^{(l)}}{\partial z_j^{(l)}}$$

$$= \sigma^{(l)\prime}(z_j^{(l)}) \sum_{m=1}^{D_{l+1}} \epsilon_m^{(l+1)} w_{j,m}^{(l+1)}$$



e.g.,



$\epsilon_1^{(1)} = ?$

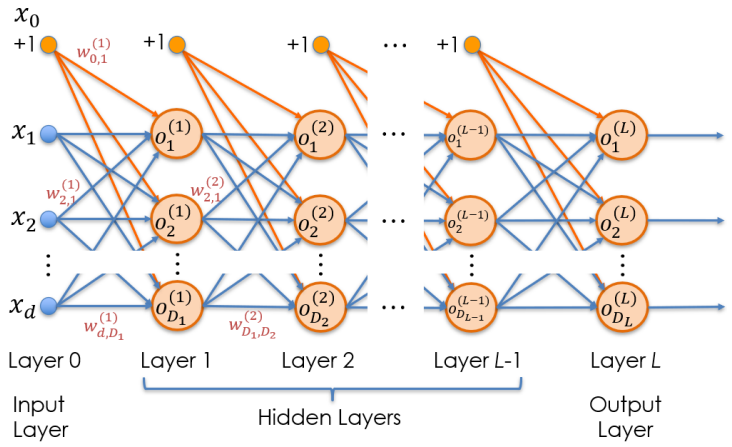
$\epsilon_2^{(1)} = ?$

Multilayer Perceptron (MLP)

➢ Learning (W): Backpropagation

$$\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}} = \frac{\partial \mathcal{L}}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial w_{i,j}^{(l)}}$$

$\epsilon_j^{(l)}$ $a_i^{(l)}$



- **If hidden layer ($l < L$):**

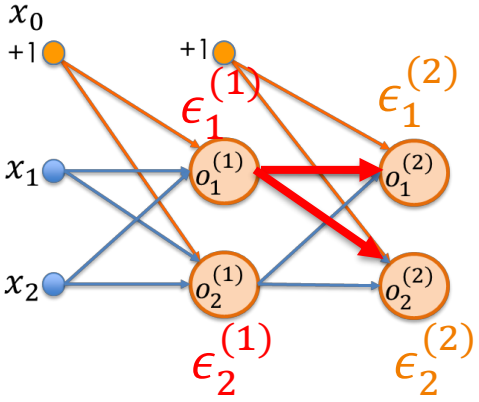
$$\frac{\partial \mathcal{L}}{\partial z_j^{(l)}} = \frac{\partial \mathcal{L}(z_1^{(l+1)}, \ldots, z_{D_{l+1}}^{(l+1)})}{\partial z_j^{(l)}} = \sum_{m=1}^{D_{l+1}} \frac{\partial \mathcal{L}(z_m^{(l+1)})}{\partial z_j^{(l)}}$$

$$= \sum_{m=1}^{D_{l+1}} \frac{\partial \mathcal{L}(z_m^{(l+1)})}{\partial z_m^{(l+1)}} \frac{\partial z_m^{(l+1)}}{\partial z_j^{(l)}} = \sum_{m=1}^{D_{l+1}} \epsilon_m^{(l+1)} \frac{\partial z_m^{(l+1)}}{\partial o_j^{(l)}} \frac{\partial o_j^{(l)}}{\partial z_j^{(l)}}$$

$$= \sigma^{(l)\prime}(z_j^{(l)}) \sum_{m=1}^{D_{l+1}} \epsilon_m^{(l+1)} w_{j,m}^{(l+1)}$$

Backward Pass

e.g.,



$$\epsilon_1^{(1)} = \sigma^{(1)\prime} \times (w_{1,1}^{(2)} \epsilon_1^{(2)} + w_{1,2}^{(2)} \epsilon_2^{(2)})$$

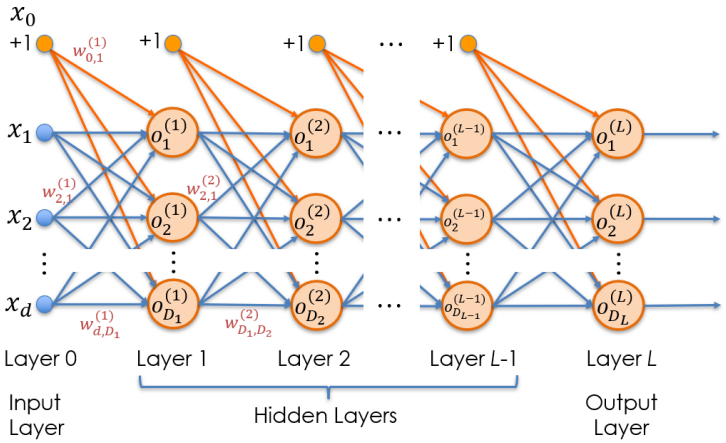
Multilayer Perceptron (MLP)

➢ Learning (W): Backpropagation

If hidden layer ($l < L$):

$$\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}} = \sigma^{(l)\prime}(z_j^{(l)}) \sum_{m=1}^{D_{l+1}} \epsilon_m^{(l+1)} w_{j,m}^{(l+1)} a_i^{(l)}$$

e.g., If $\underbrace{\sigma^{(l)}(z) = \max(0, z)}_{\text{ReLU}}$, what is $\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}}$?



Multilayer Perceptron (MLP)

➢ Learning (W): Backpropagation

If hidden layer ($l < L$):

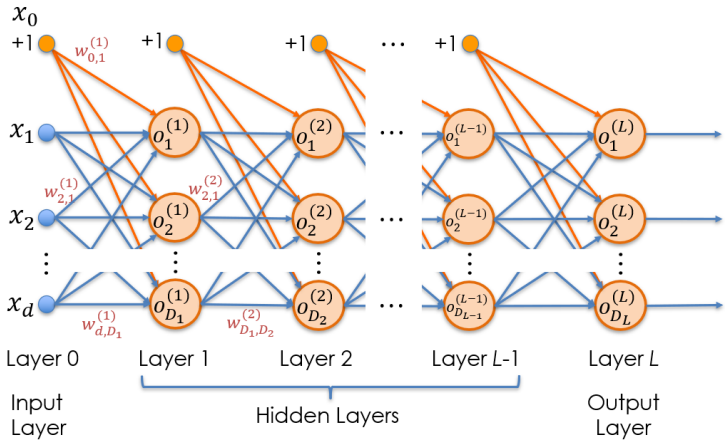
$$\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}} = \sigma^{(l)\prime}(z_j^{(l)}) \sum_{m=1}^{D_{l+1}} \epsilon_m^{(l+1)} w_{j,m}^{(l+1)} a_i^{(l)}$$

e.g., If $\underbrace{\sigma^{(l)}(z) = \max(0, z)}_{\text{ReLU}}$, what is $\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}}$?

$$\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}} = \epsilon_j^{(l)} a_i^{(l)} = \delta_{z_j^{(l)}} \boldsymbol{\epsilon}^{(l+1)} (\mathbf{w}_{j,:}^{(l+1)})^T a_i^{(l)}$$

$$\epsilon_j^{(l)} = \sigma^{(l)\prime}(z_j^{(l)}) \sum_{m=1}^{D_{l+1}} \epsilon_m^{(l+1)} w_{j,m}^{(l+1)}$$

$$= \delta_{z_j^{(l)}} \underbrace{\boldsymbol{\epsilon}^{(l+1)}}_{\text{Size: } 1 \times D_{l+1}} \underbrace{(\mathbf{w}_{j,:}^{(l+1)})^T}_{\text{Size: } D_{l+1} \times 1}$$



$$\sigma^{(l)\prime}\left(z_j^{(l)}\right) = \frac{d(\max(0, z_j^{(l)}))}{d(z_j^{(l)})} = \begin{cases} 0, if\, z_j^{(l)} < 0 \\ 1, if\, z_j^{(l)} > 0 \end{cases}$$

$$\delta_{z_j^{(l)}} = \begin{cases} 0, if\, z_j^{(l)} < 0 \\ 1, if\, z_j^{(l)} > 0 \end{cases}$$

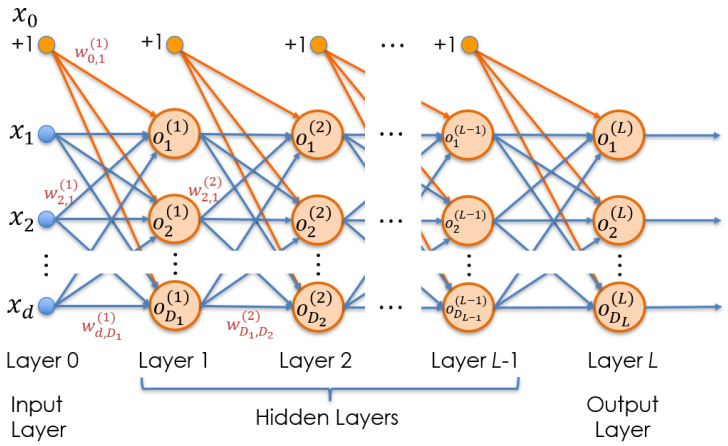
Multilayer Perceptron (MLP)

➢ Learning (W): Backpropagation

If hidden layer ($l < L$):

$$\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}} = \sigma^{(l)\prime}(z_j^{(l)}) \sum_{m=1}^{D_{l+1}} \epsilon_m^{(l+1)} w_{j,m}^{(l+1)} a_i^{(l)}$$

e.g., If $\underbrace{\sigma^{(l)}(z) = \max(0, z)}_{\text{ReLU}}$, what is $\frac{d\mathcal{L}}{d\mathbf{W}^{(l)}}$?



Multilayer Perceptron (MLP)

➢ Learning (W): Backpropagation

If hidden layer ($l < L$):

$$\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}} = \sigma^{(l)\prime}(z_j^{(l)}) \sum_{m=1}^{D_{l+1}} \epsilon_m^{(l+1)} w_{j,m}^{(l+1)} a_i^{(l)}$$

e.g., If $\sigma^{(l)}(z) = \max(0, z)$, what is $\frac{d\mathcal{L}}{d\mathbf{W}^{(l)}}$?

ReLU

$$\mathbf{W}^{(l+1)} = \begin{bmatrix} w_{0,1}^{(l+1)} & \cdots & w_{0,D_{l+1}}^{(l+1)} \\ \vdots & \ddots & \vdots \\ w_{D_l,1}^{(l+1)} & \cdots & w_{D_l,D_{l+1}}^{(l+1)} \end{bmatrix}$$

Size: $(D_{l-1}+1) \times D_l$ — Size: $1 \times D_l$ — Size: $1 \times D_{l+1}$

$$\frac{d\mathcal{L}}{d\mathbf{W}^{(l)}} = \mathbf{a}^{(l)} \boldsymbol{\epsilon}^{(l)} = \mathbf{a}^{(l)} \left(\boldsymbol{\delta}_{\mathbf{Z}^{(l)}} * \left(\boldsymbol{\epsilon}^{(l+1)} \left(\mathbf{W}_{1:D_l,:}^{(l+1)}\right)^T\right)\right)$$

Size: $(D_{l-1}+1) \times 1$

Size: $D_{l+1} \times D_l$

- $\mathbf{A}^{(l)}$ has a size of $N \times (D_{l-1}+1)$
- $\boldsymbol{\delta}_{\mathbf{Z}^{(l)}}$ has a size of $N \times D_l$
- $\mathbf{E}^{(l+1)}$ has a size of $N \times D_{l+1}$

$$\frac{dJ}{d\mathbf{W}^{(l)}} = \frac{1}{N} \sum \frac{d\mathcal{L}}{d\mathbf{W}^{(l)}} = \frac{1}{N} \left(\mathbf{A}^{(l)}\right)^T \left(\boldsymbol{\delta}_{\mathbf{Z}^{(l)}} * \left(\mathbf{E}^{(l+1)} \left(\mathbf{W}_{1:D_l,:}^{(l+1)}\right)^T\right)\right)$$

Multilayer Perceptron (MLP)

➢ Learning (W): Backpropagation

If output layer (l =L):

$$\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}} = \sum_{m=1}^{D_L} \mathcal{L}'(o_m^{(L)}) \sigma^{(L)\prime}(z_j^{(L)})_m \, a_i^{(L)}$$

Summary:

If softmax activation, categorical cross entropy loss:

$$\frac{d\mathcal{L}}{d\mathbf{W}^{(L)}} = \mathbf{a}^{(L)} \underbrace{(\mathbf{f_W}(\mathbf{x}) - \mathbf{y})}_{\boldsymbol{\epsilon}^{(L)}}, \qquad \frac{dJ}{d\mathbf{W}^{(L)}} = \frac{1}{N} (\mathbf{A}^{(L)})^T \underbrace{(\mathbf{F_W}(\mathbf{X}) - \mathbf{Y})}_{\mathbf{E}^{(L)}}$$

If hidden layer (l <L):

$$\frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}} = \sigma^{(l)\prime}(z_j^{(l)}) \sum_{m=1}^{D_{l+1}} \epsilon_m^{(l+1)} w_{j,m}^{(l+1)} \, a_i^{(l)}$$

Why?softmax is used for multi-class classification, takes raw scores and outputs into probabilities for class prediction, and softmax forces every hidden vector to sum to 1 and be positive, which is not what we want. Can use sigmoid here for binary classification.

HIdden layers use non-linear activations like relu to learn representations of the input, must use non-linear activation to avoid collapsing into just linear regression

If ReLU activation:

$$\frac{d\mathcal{L}}{d\mathbf{W}^{(l)}} = \mathbf{a}^{(l)} (\boldsymbol{\delta}_{\mathbf{z}^{(l)}} * (\boldsymbol{\epsilon}^{(l+1)} (\mathbf{W}_{1:D_l,:}^{(l+1)})^T))$$

$$\frac{dJ}{d\mathbf{W}^{(l)}} = \frac{1}{N} (\mathbf{A}^{(l)})^T (\boldsymbol{\delta}_{\mathbf{Z}^{(l)}} * (\mathbf{E}^{(l+1)} (\mathbf{W}_{1:D_l,:}^{(l+1)})^T))$$

Multilayer Perceptron (MLP)

➢ Learning (W): Backpropagation

Example: A MLP of 2 layers has been constructed as:

$$\mathbf{f}_{\mathbf{W}}(\mathbf{x}) = \boldsymbol{\sigma}^{(2)}\left(\left[\mathbf{1}, \boldsymbol{\sigma}^{(1)}\left(\mathbf{x}^T\mathbf{W}^{(1)}\right)\right]\mathbf{W}^{(2)}\right),$$

where $\sigma^{(1)}$ is **ReLU**, and $\boldsymbol{\sigma}^{(2)}$ is the **Softmax**.

Training data: $\mathbf{X} = \begin{bmatrix} 1 & 1 & 3.0 \\ 1 & 2 & 2.5 \end{bmatrix}, \mathbf{Y} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Loss function: **Categorical Cross Entropy**

Weight initialization: $\mathbf{W}^{(1)}(0) = \mathbf{W}^{(2)}(0) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \end{bmatrix}$

What are $\mathbf{W}^{(1)}(1)$ and $\mathbf{W}^{(2)}(1)$ after 1st iteration of weight updates if learning rate η=0.1 ?

Multilayer Perceptron (MLP)

➢ Learning (W): Backpropagation

Example: A MLP of 2 layers has been constructed as:

$$\mathbf{f}_{\mathbf{W}}(\mathbf{x}) = \boldsymbol{\sigma}^{(2)}\left(\left[\mathbf{1}, \boldsymbol{\sigma}^{(1)}\left(\mathbf{x}^T\mathbf{W}^{(1)}\right)\right]\mathbf{W}^{(2)}\right),$$

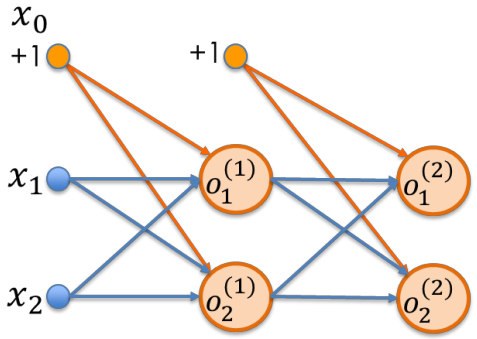
where $\sigma^{(1)}$ is **ReLU**, and $\boldsymbol{\sigma}^{(2)}$ is the **Softmax**.

Training data: $\mathbf{X} = \begin{bmatrix} 1 & 1 & 3.0 \\ 1 & 2 & 2.5 \end{bmatrix}, \mathbf{Y} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Loss function: **Categorical Cross Entropy**

Weight initialization: $\mathbf{W}^{(1)}(0) = \mathbf{W}^{(2)}(0) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \end{bmatrix}$

What are $\mathbf{W}^{(1)}(1)$ and $\mathbf{W}^{(2)}(1)$ after 1st iteration of weight updates if learning rate η=0.1 ?



Step 1: Forward Pass

$$\mathbf{F}_{\mathbf{W}(0)}(\mathbf{X}) = \boldsymbol{\sigma}^{(2)}\left(\left[\mathbf{1}, \boldsymbol{\sigma}^{(1)}\left(\mathbf{X}\mathbf{W}^{(1)}(0)\right)\right]\mathbf{W}^{(2)}(0)\right)$$

z output: $\mathbf{X}\mathbf{W}^{(1)}(0)$

pre activation z output: $\begin{bmatrix} -1 & -2 \\ -1 & -1.5 \end{bmatrix}$

activation output: $\begin{bmatrix} 2 & 0 \\ 1.5 & 0 \end{bmatrix}$

$$= \begin{bmatrix} 0.7311 & 0.2689 \\ 0.6225 & 0.3775 \end{bmatrix}$$

Multilayer Perceptron (MLP)

➢ Learning (W): Backpropagation

Example: A MLP of 2 layers has been constructed as:

$$\mathbf{f}_{\mathbf{W}}(\mathbf{x}) = \boldsymbol{\sigma}^{(2)}\left(\left[\mathbf{1}, \boldsymbol{\sigma}^{(1)}\left(\mathbf{x}^T\mathbf{W}^{(1)}\right)\right]\mathbf{W}^{(2)}\right),$$

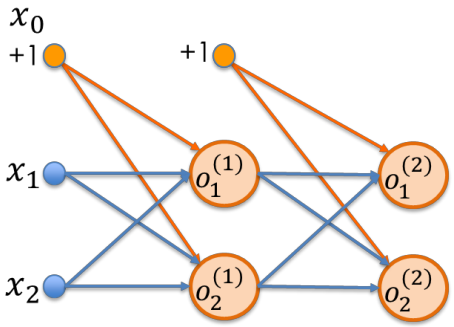
where $\sigma^{(1)}$ is ReLU, and $\boldsymbol{\sigma}^{(2)}$ is the Softmax.

Training data: $\mathbf{X} = \begin{bmatrix} 1 & 1 & 3.0 \\ 1 & 2 & 2.5 \end{bmatrix}, \mathbf{Y} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Loss function: Categorical Cross Entropy

Weight initialization: $\mathbf{W}^{(1)}(0) = \mathbf{W}^{(2)}(0) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \end{bmatrix}$

What are $\mathbf{W}^{(1)}(1)$ and $\mathbf{W}^{(2)}(1)$ after 1st iteration of weight updates if learning rate η=0.1 ?



Step 2: Backward Pass

Output layer:

$$\frac{dJ(\mathbf{W}^{(2)}(0))}{d\mathbf{W}^{(2)}} = \frac{1}{N}(\mathbf{A}^{(2)})^T(\underbrace{\mathbf{F}_{\mathbf{W}(\mathbf{0})}(\mathbf{X}) - \mathbf{Y}}_{\mathbf{E}^{(2)}})$$

A is activation output

E is the error term

$$\mathbf{E}^{(2)} = \begin{bmatrix} -0.2689 & 0.2689 \\ 0.6225 & -0.6225 \end{bmatrix}$$

$$\mathbf{A}^{(2)} = \begin{bmatrix} 1 & 2 & 0 \\ 1 & 1.5 & 0 \end{bmatrix}$$

$$\frac{dJ(\mathbf{W}^{(2)}(0))}{d\mathbf{W}^{(2)}} = \begin{bmatrix} 0.1768 & -0.1768 \\ 0.1979 & -0.1979 \\ 0 & 0 \end{bmatrix}$$

$\mathbf{W}^{(2)}$ update:

$$\mathbf{W}^{(2)}(1) = \mathbf{W}^{(2)}(0) - \eta\frac{dJ(\mathbf{W}^{(2)}(0))}{d\mathbf{W}^{(2)}} = \begin{bmatrix} -1.0177 & 0.0177 \\ -0.0198 & -0.9802 \\ 1 & 0 \end{bmatrix}$$

Multilayer Perceptron (MLP)

➢ Learning (W): Backpropagation

Example: A MLP of 2 layers has been constructed as:

$$\mathbf{f}_{\mathbf{W}}(\mathbf{x}) = \boldsymbol{\sigma}^{(2)}\left(\left[\mathbf{1}, \boldsymbol{\sigma}^{(1)}\left(\mathbf{x}^T\mathbf{W}^{(1)}\right)\right]\mathbf{W}^{(2)}\right),$$

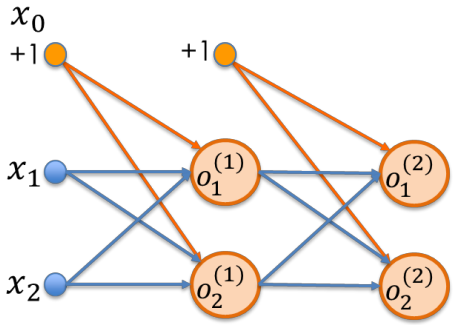
where $\sigma^{(1)}$ is ReLU, and $\boldsymbol{\sigma}^{(2)}$ is the Softmax.

Training data: $\mathbf{X} = \begin{bmatrix} 1 & 1 & 3.0 \\ 1 & 2 & 2.5 \end{bmatrix}, \mathbf{Y} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Loss function: Categorical Cross Entropy

Weight initialization: $\mathbf{W}^{(1)}(0) = \mathbf{W}^{(2)}(0) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \end{bmatrix}$

What are $\mathbf{W}^{(1)}(1)$ and $\mathbf{W}^{(2)}(1)$ after 1st iteration of weight updates if learning rate η=0.1 ?



Step 2: Backward Pass

Hidden layer: $\dfrac{dJ(\mathbf{W}^{(1)}(0))}{d\mathbf{W}^{(1)}}$

$$= \frac{1}{N}(\mathbf{A}^{(1)})^T \underbrace{(\boldsymbol{\delta}_{\mathbf{Z}^{(1)}} * (\mathbf{E}^{(2)}(\mathbf{W}^{(2)}_{1:2,:}(0))^T))}_{\mathbf{E}^{(1)}}$$

$$\mathbf{A}^{(1)} = \mathbf{X} = \begin{bmatrix} 1 & 1 & 3.0 \\ 1 & 2 & 2.5 \end{bmatrix}$$

$$\mathbf{E}^{(1)} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} * (\mathbf{E}^{(2)} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix})$$

$$= \begin{bmatrix} -0.2689 & 0 \\ 0.6225 & 0 \end{bmatrix}$$

$$\frac{dJ(\mathbf{W}^{(1)}(0))}{d\mathbf{W}^{(1)}} = \begin{bmatrix} 0.1768 & 0 \\ 0.4880 & 0 \\ 0.3747 & 0 \end{bmatrix}$$

$\mathbf{W}^{(1)}$ update:

$$\mathbf{W}^{(1)}(1) = \mathbf{W}^{(1)}(0) - \eta \frac{dJ(\mathbf{W}^{(1)}(0))}{d\mathbf{W}^{(1)}} = \begin{bmatrix} -1.0177 & 0 \\ -0.0488 & -1 \\ 0.9625 & 0 \end{bmatrix}$$

Multilayer Perceptron (MLP)

➢ Learning (W): Backpropagation

Example: A MLP of 2 layers has been constructed as:

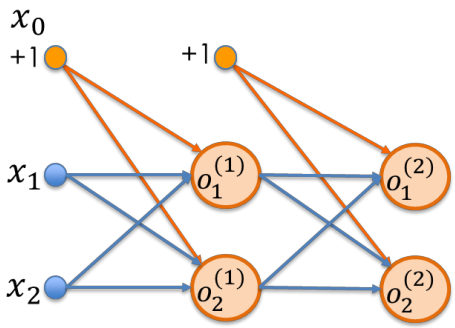
$$\mathbf{f}_{\mathbf{W}}(\mathbf{x}) = \boldsymbol{\sigma}^{(2)}\left(\left[\mathbf{1}, \boldsymbol{\sigma}^{(1)}\left(\mathbf{x}^T\mathbf{W}^{(1)}\right)\right]\mathbf{W}^{(2)}\right),$$

where $\sigma^{(1)}$ is ReLU, and $\boldsymbol{\sigma}^{(2)}$ is the Softmax.

Training data: $\mathbf{X} = \begin{bmatrix} 1 & 1 & 3.0 \\ 1 & 2 & 2.5 \end{bmatrix}, \mathbf{Y} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Loss function: Categorical Cross Entropy

Weight initialization: $\mathbf{W}^{(1)}(0) = \mathbf{W}^{(2)}(0) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \end{bmatrix}$



3rd Iteration…

4th Iteration…

⋮

2nd Iteration:

Step 1: Forward Pass

$$\mathbf{F}_{\mathbf{W}(1)}(\mathbf{X}) = \boldsymbol{\sigma}^{(2)}\left(\left[\mathbf{1}, \boldsymbol{\sigma}^{(1)}\left(\mathbf{X}\mathbf{W}^{(1)}(1)\right)\right]\mathbf{W}^{(2)}(1)\right)$$

Step 2: Backward Pass

✓ **Output layer:**

$$\frac{dJ(\mathbf{W}^{(2)}(1))}{d\mathbf{W}^{(2)}} = \frac{1}{N}(\mathbf{A}^{(2)})^T(\mathbf{F}_{\mathbf{W}(1)}(\mathbf{X}) - \mathbf{Y})$$

$$\mathbf{W}^{(2)}(2) = \mathbf{W}^{(2)}(1) - \eta\frac{dJ(\mathbf{W}^{(2)}(1))}{d\mathbf{W}^{(2)}}$$

✓ **Hidden layer:**

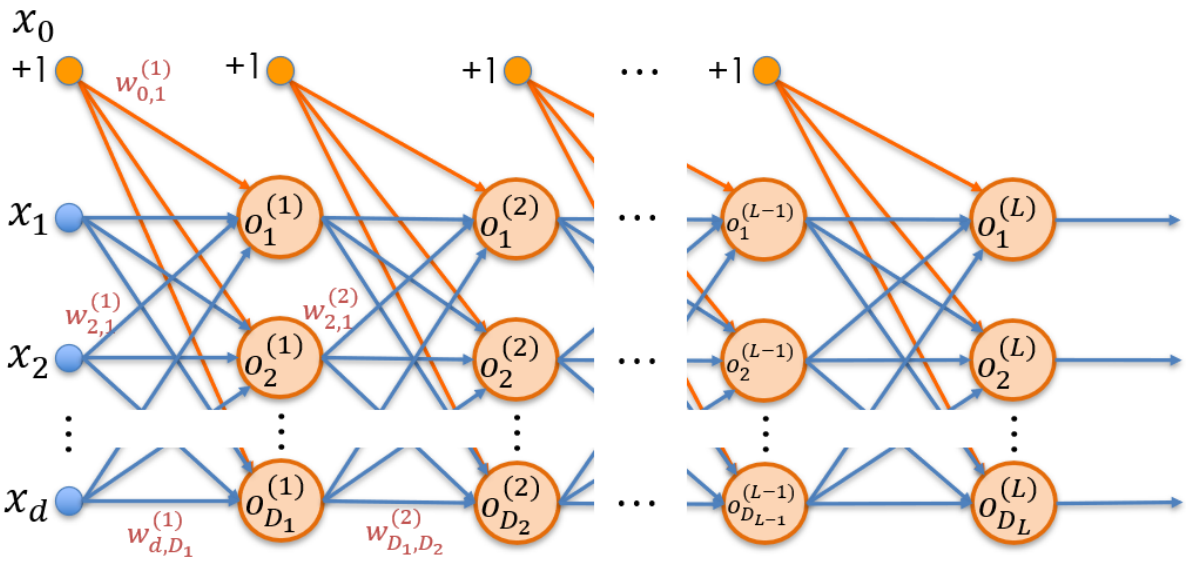
$$\frac{dJ(\mathbf{W}^{(1)}(1))}{d\mathbf{W}^{(1)}} = \frac{1}{N}(\mathbf{A}^{(1)})^T(\boldsymbol{\delta}_{\mathbf{Z}^{(1)}} * (\mathbf{E}^{(2)}(\mathbf{W}^{(2)}_{1:2,:}(1))^T))$$

$$\mathbf{W}^{(1)}(2) = \mathbf{W}^{(1)}(1) - \eta\frac{dJ(\mathbf{W}^{(1)}(1))}{d\mathbf{W}^{(1)}}$$

Multilayer Perceptron (MLP)

➢ Prediction (Forward Pass)

$$\mathbf{x}_{new} = \begin{bmatrix} 0.2 \\ 0.3 \\ 0.4 \end{bmatrix}$$



$$\mathbf{f}_{\mathbf{W}}(\mathbf{x}_{new}) = \begin{bmatrix} 0.2 \\ 0.1 \\ 0.7 \end{bmatrix}$$

Fixed **W**, Compute $\mathbf{f}_{\mathbf{W}}(\mathbf{x}_{new})$

MLP for Image Classification

MLP for Image Classification

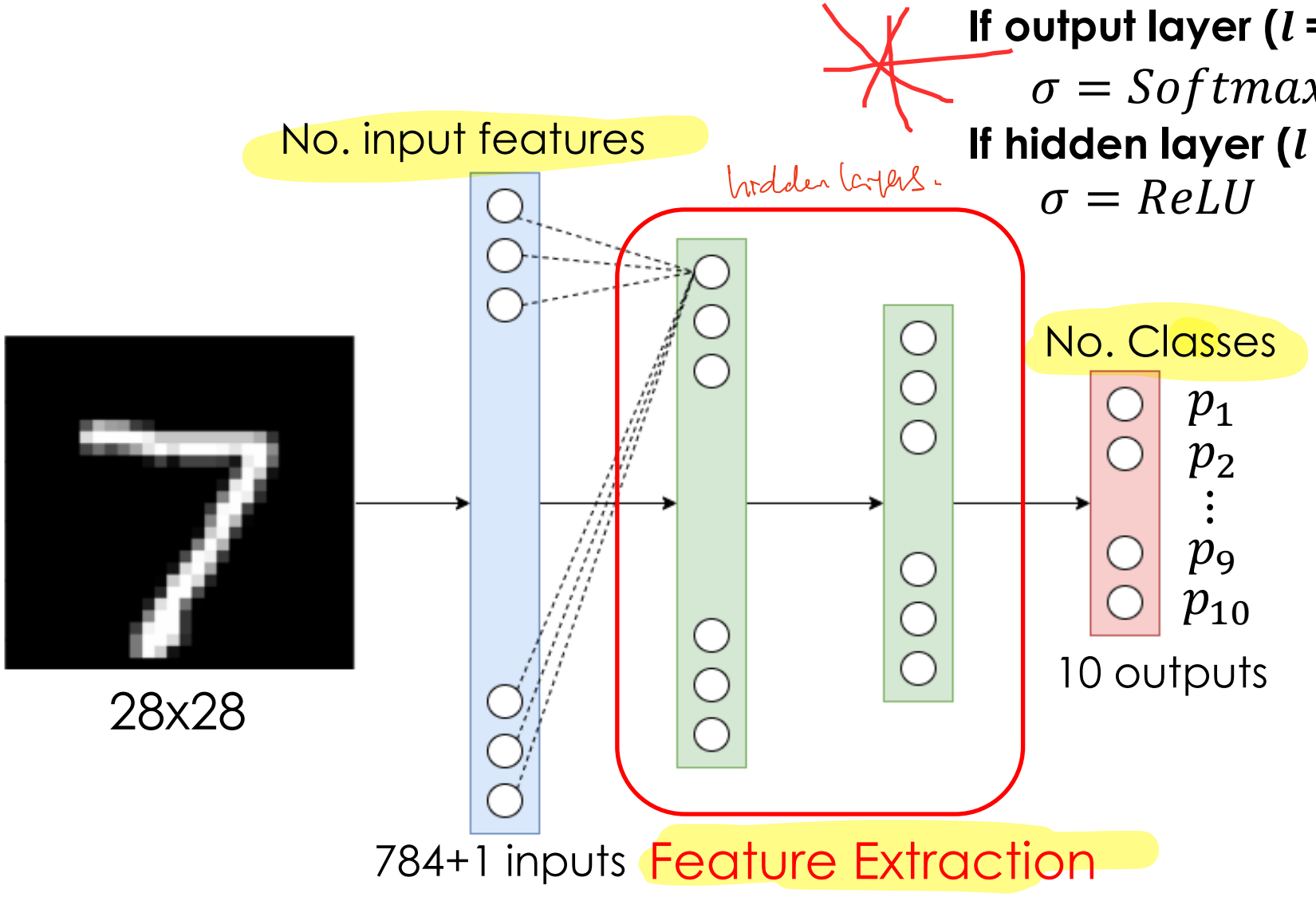
- **Image is stored as a matrix in a computer**

0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

0 2 15 0 0 11 10 0 0 0 0 9 9 0 0 0
0 0 0 4 60 157 236 255 255 177 95 61 32 0 0 29
0 10 16 119 238 255 244 245 243 250 249 255 222 103 10 0
0 14 170 255 255 244 254 255 253 245 255 249 253 251 124 1
2 98 255 228 255 251 254 211 141 116 122 215 251 238 255 49
13 217 243 255 155 33 226 52 2 0 10 13 232 255 255 36
16 229 252 254 49 12 0 0 7 7 0 70 237 252 235 62
6 141 245 255 212 25 11 9 3 0 115 236 243 255 137 0
0 87 252 250 248 215 60 0 1 121 252 255 248 144 6 0
0 13 113 255 255 245 255 182 181 248 252 242 208 36 0 19
1 0 5 117 251 255 241 255 247 255 241 162 17 0 7 0
0 0 0 4 58 251 255 246 254 253 255 120 11 0 1 0
0 0 4 97 255 255 255 248 252 255 244 255 182 10 0 4
0 22 206 252 246 251 241 100 24 113 255 245 255 194 9 0
0 111 255 242 255 158 24 0 0 6 39 255 232 230 56 0
0 218 251 250 137 7 11 0 0 0 2 62 255 250 125 3
0 173 255 255 101 9 20 0 13 3 13 182 251 245 61 0
0 107 251 241 255 230 98 55 19 118 217 248 253 255 52 4
0 18 146 250 255 247 255 255 255 249 255 240 255 129 0 5
0 0 23 113 215 255 250 248 255 255 248 248 118 14 12 0
0 0 6 1 0 52 153 233 255 252 147 37 0 0 4 1
0 0 5 5 0 0 0 0 0 14 1 0 6 6 0 0

Pixel: smallest unit of a digital image

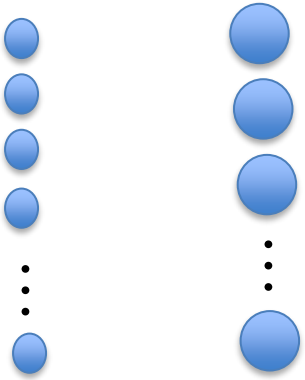
MLP for Image Classification



MLP for Image Classification

➢ Limitations

200x200



How many input features?

How many parameters to learn for the first layer?

10K neurons in the first layer

40001x10000≈400 millions

Parameter explosion

Flattening

Loss of Spatial Structure

≫Convolutional Neural Network!

Summary

- ✓ Perceptron
- ✓ Multilayer Perceptron (MLP)
  - **-Learning (W)**
    1. Random Initialization of **W**
    2. Forward Pass (fixed **W**)
    3. Backward Pass (update **W**)

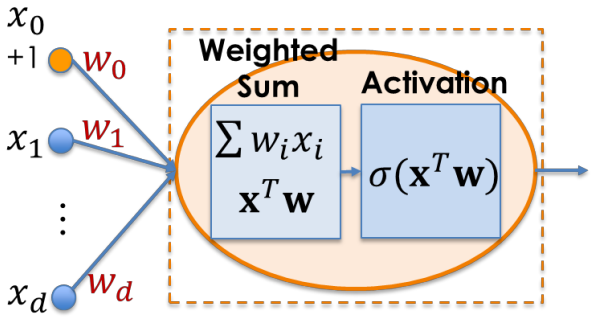
$$\mathbf{W}_{new} = \mathbf{W}_{old} - \eta \nabla_{\mathbf{W}} J(\mathbf{W}_{old})$$

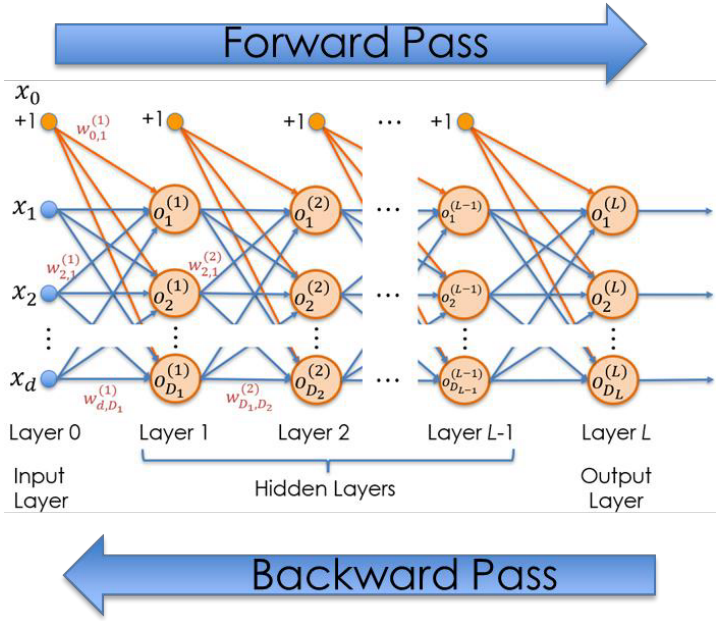
  - **-Prediction (Forward Pass)**

$$\mathbf{f}_{\mathbf{W}}(\mathbf{x}_{new})$$

- ✓ MLP for Image Classification

  Limitations





THE END

NATIONAL UNIVERSITY OF SINGAPORE
DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

ACADEMIC YEAR 2025-2026
SEMESTER 1

EE2213: Introduction to AI

TUTORIAL 6: L7-L9

Question 1 (Lecture 7)

What is the relationship between Machine Learning (ML) and Artificial Intelligence (AI)?

Answer: ML is a subset of AI

Question 2 (Lecture 7)

Suppose you are working on a machine learning algorithm to navigate a robot through a room from start to goal while avoiding obstacles within some time, according to previous runs through the room with sensor feedback (e.g., successful and failed attempts). What are the Task, Performance, and Experience involved according to the definition of machine learning.

Answer:

Task (T): Navigate the robot through a room from start to goal while avoiding obstacles

Performance (P): Success rate of reaching the goal without collision, time taken to reach the goal

Experience (E): Previous runs through the room with sensor feedback.

Question 3 (Lecture 7)

Suppose you are working on a machine learning algorithm that groups customers into market segments based on purchasing behavior.

(i) What type of task is it?

(a) Regression

(b) Classification

(c) Clustering

(d) None of these

Answer: (c)

(ii) If the data you have collected involve millions of attributes, what would you do?

Answer: Feature Selection or Feature Extraction

Question 4 (Lecture 7)

You are given a set of data for supervised learning. A sample block of data looks like this:

$$\begin{array}{lllll} 0.0012 & 20123 & 1.545 & 12.47 & 1 \\ 0.0034 & 29087 & 1.908 & 20.66 & -1 \\ 0.0029 & 19456 & 2.109 & 27.09 & 1 \\ 0.0013 & 32098 & 1.009 & 18.76 & 1 \\ 0.0088 & 25908 & 1.398 & 30.08 & -1 \end{array}$$

Each row corresponds to a sample data measurement with 4 input features and 1 response.

(i) What kind of undesired effect can you anticipate if this set of raw data is used for

learning?

Answer: Those features with very large values may overshadow those with very small values.

(ii) How can the data be preprocessed to handle this issue?

Answer: We can either use min-max or z-score normalization to resolve the problem.

Question 5 (Lecture 7)

Please find the synthetic data provided in the "synthetic-data.xlsx". You may need to install openpyxl library to read .xlsx files. Read the dataset by "from pandas import read_excel". Use Python to perform the following tasks.

(i) Print the summary statistics of this dataset. (You may use ".describe()" or ".info()")
(ii) Check and locate if any missing values or duplicates. (You may use ".isnull()" and ".duplicated()")

Answer: See TUT6-Q5.ipynb

Question 6 (Lecture 8)

Given the following dataset for training:

Sample No.	x	y
1	1.6	3.1
2	1.5	3.4
3	1.0	2.9
4	1.1	3.0
5	1.2	3.8
6	0.9	2.9

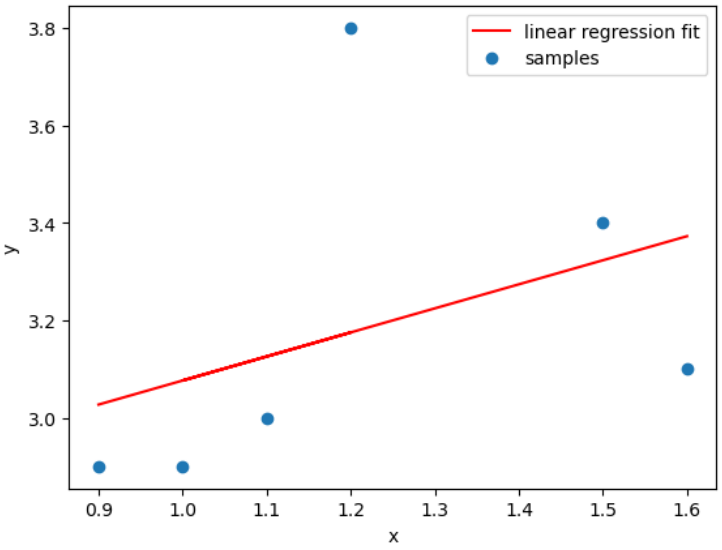
(i) Perform a linear regression, sketch the result of line fitting, calculate the mean squared error for the training set, and predict a test point {x=0.5}. 0.0890

(ii) Perform a 3rd-order polynomial regression, sketch the result of line fitting, calculate the mean squared error for the training set, and predict the same test point {x=0.5}. 0.03024

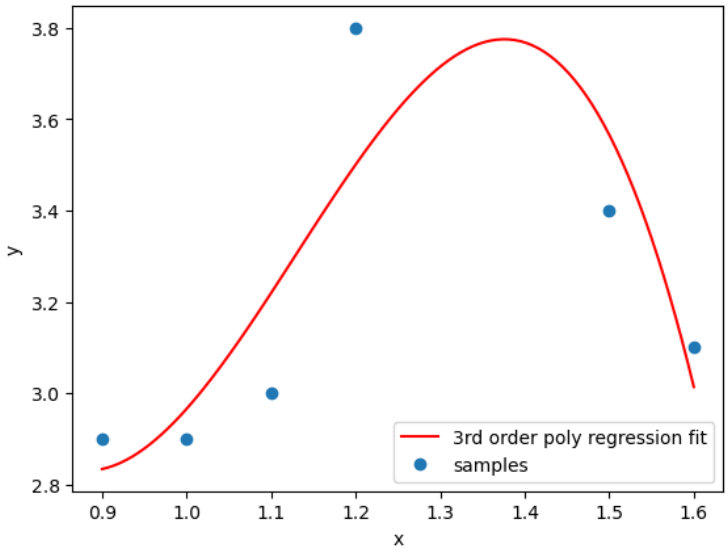
(iii) Does it have a unique 6th-order polynomial regression solution? If so, proceed to solved it. Otherwise, apply ridge regression with different regularization parameter λ:[0.0001, 0.01, 1].

Answer: See TUT6_Q6.ipynb

(i) $w = \begin{bmatrix} 2.5828 \\ 0.4936 \end{bmatrix}, MSE = 0.08896, y_t = 2.8296$



(ii) $w = \begin{bmatrix} 22.6841 \\ -57.2428 \\ 53.2438 \\ -15.7193 \end{bmatrix}, MSE = 0.03024, y_t = 5.4088$



(iii) No, because $\mathrm{P}^T\mathrm{P}$ is not invertible.

When $\lambda = 0.0001$: $MSE = 0.0330, y_t = 2.8114$

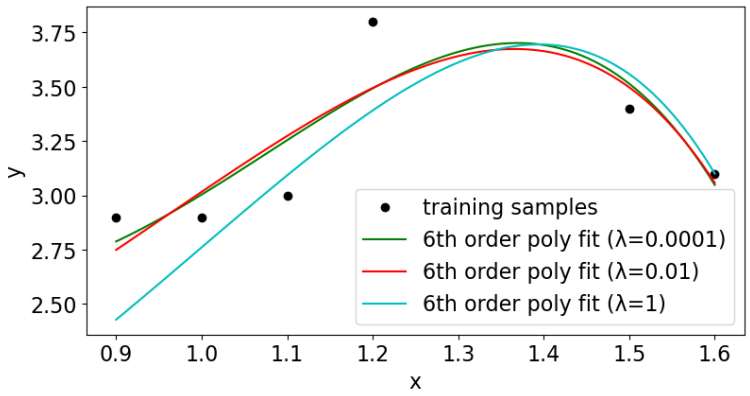
$w = [4.6439, -4.0828, -1.3906, 3.7044, 3.2392, -3.8597, 0.7490]^T,$

When $\lambda = 0.01$: $MSE = 0.0360, y_t = 1.8481$

$w = [1.2471, 0.8001, 0.5515, 0.4051, 0.2553, 0.0220, -0.2649]^T,$

When $\lambda = 1$: $MSE = 0.0739, y_t = 1.3768$,

$w = [0.7708, 0.7170, 0.6438, 0.5342, 0.3618, 0.0864, -0.3535]^T$



Question 7 (Lecture 8)

The California Housing Dataset is used for predicting the median house value for a given district in California, based on features from the 1990 U.S. Census. Get the dataset by "from sklearn.datasets import fetch_california_housing" and load the dataset as DataFrame by "fetch_california_housing(as_frame=True)". Use Python to perform the following tasks.

(i) Split the database into three sets: 60% of samples for training, 20% of samples for validation, and 20% of samples for testing. Hint: you might want to utilize "from sklearn.model_selection import train_test_split" for the splitting.

(ii) Use Pearson's correlation as a feature selection metric to select the top 2 features that have the highest absolute correlations with the target output. Use the selected top 2 features to construct the training, validation and test datasets. Hint: you might want to utilize ".corr()" for calculating Pearson's correlation coefficients.

(iii) Apply the polynomial model from orders 1 to 6 to the dataset generated from part (ii) without L2 regularization unless $\mathbf{P}^T\mathbf{P}$ is not invertible (then, use $\lambda = 0.00001$). Plot Mean Squared Errors (MSE) over polynomial orders for both training and the validation sets. Which polynomial order provides the best MSE in the training and validation sets? Which polynomial order should be selected? Use the selected polynomial order to evaluate the MSE for the test set. Hint: you might want to utilize "from sklearn.metrics import mean_squared_error" for calculating MSE.

(iv) Use regularization parameter λ =1 for all polynomial orders and repeat the same analyses. Compare the plots of (iii) and (iv). What do you see?

Answer: See TUT6_Q7.ipynb

Note: using different "random_state" in "train_test_split" may have different results. The following answers are obtained using "random_state=42"

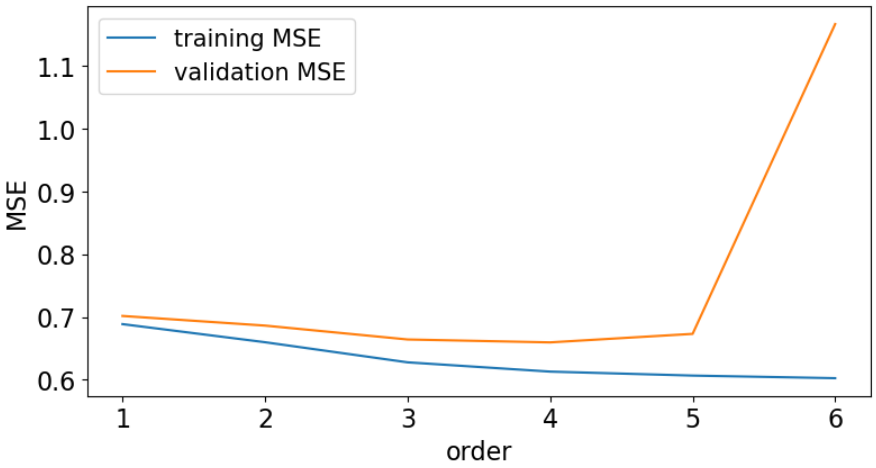
(ii) Top 2 features are "MedInc", "AveRooms"

(iii) MSE for training set: [0.6888, 0.6599, 0.6279, 0.6132, 0.6068, **0.6027**], order 6

MSE for validation set: [0.7017, 0.6864, 0.6643, **0.6597**, 0.6732, 1.1663], order 4

Order 4 should be selected according to the validation set.

MSE for test set using order 4 is 4.4618.

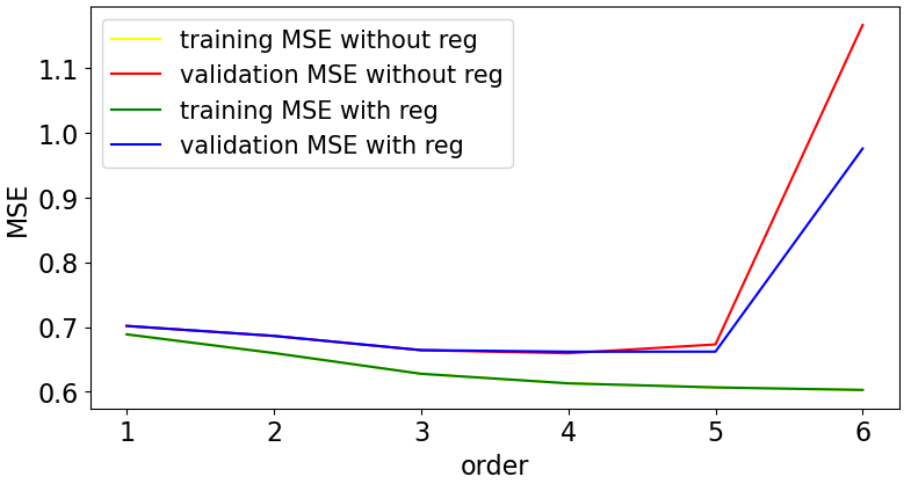


(iv) MSE for training set: [0.6888, 0.6599, 0.6279, 0.6132, 0.6069, **0.6030**], order 6

MSE for validation set: [0.7017, 0.6864, 0.6645, **0.6620**, 0.6621, 0.9755], order 4

Order 4 should be selected according to the validation set.

MSE for test set using order 4 is 4.4618.



Regularization helps generalization (prevent overfitting) for higher polynomial order.

Question 8 (Lecture 9)

The Breast Cancer Wisconsin Diagnostic Dataset is used for classifying tumors as either malignant (labelled 0) or benign (labelled 1), based on digitized measurements of breast mass cell nuclei. Get the dataset by "from sklearn.datasets import load_breast_cancer" and load the dataset as DataFrame by "load_breast_cancer (as_frame=True)". Use Python to perform the following tasks.

(i) Split the database into three sets: 80% of samples for training, 10% of samples for validation, and 10% of samples for testing. Hint: you might want to utilize "from sklearn.model_selection import train_test_split" for the splitting.

(ii) Use Pearson's correlation as a feature selection metrics. First, select features that have

absolute correlations greater than 0.5. Second, among the selected features, remove features that have absolute correlations with each other greater than 0.9. Use the selected features to construct the training, validation and test datasets. Hint: you might want to utilize ".corr()" for calculating Pearson's correlation coefficients.

(iii) Apply linear regression for classification to the dataset generated in part (ii) without regularization unless $\mathbf{X}^T\mathbf{X}$ is not invertible (then, use $\lambda = 0.00001$). The threshold is set to 0.5 for class prediction. Compute the training, validation and test accuracies. Demonstrate the confusion matrix for the test set. Hint: you might want to utilize "from sklearn.metrics import accuracy_score, confusion_matrix" for evaluation metrics.

(iv) Apply logistic regression to the dataset generated from part (ii) without regularization. The threshold is set to 0.5 for class prediction. The number of iterations for training is set to 20000. Try different learning rates: 0.1, 0.01 and 0.001. Plot the cost over training iterations for each learning rate. Which learning rate provides the best accuracy in the training and the validation sets? Which learning rate should be selected? Use the selected learning rate to compute the accuracy and confusion matrix for the test set.

Answer: See TUT6_Q8.ipynb

Note: using different "random_state" in "train_test_split" may have different results. The following answers are obtained using "random_state=42".

(ii) Final selected features: 'mean radius', 'mean compactness', 'mean concavity', 'radius error', 'worst compactness', 'worst concavity', 'worst concave points'

(iii) Training accuracy is 0.9341; validation accuracy is 0.9123; test accuracy is 0.9474.

Confusion matrix for test set is $\begin{bmatrix} 14 & 2 \\ 1 & 40 \end{bmatrix}$.

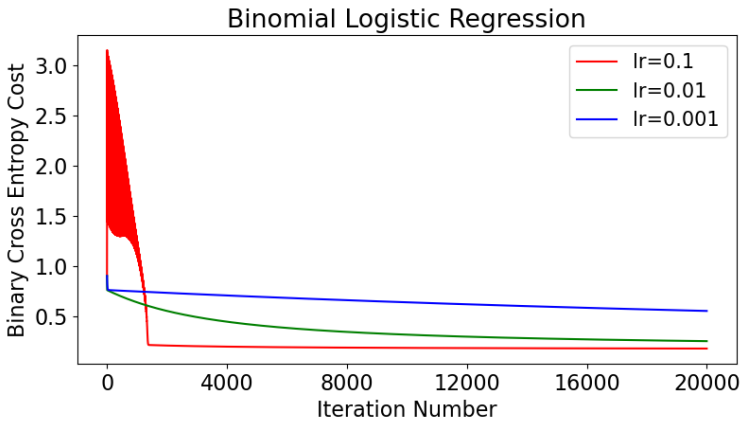
(iv) Training accuracy: [**0.9187**, 0.9143, 0.8308]; learning rate 0.1.

Validation accuracy: [**0.9298**, 0.9123, 0.7895]; learning rate 0.1.

Learning rate 0.1 should be selected according to the validation accuracy.

Test accuracy using learning rate 0.1: 0.9825.

Confusion matrix for test set is $\begin{bmatrix} 15 & 1 \\ 0 & 41 \end{bmatrix}$



Question 9 (Lecture 9)

The Iris dataset is used to classify 3 species of iris flower: Setosa, Versicolor, and Virginica. Get the dataset by "from skelarn.datasets import load_iris". Use Python to perform the following tasks.

(i) Split the database into three sets: 80% of samples for training, 10% of samples for validation, and 10% of samples for testing. Hint: you might want to utilize "from sklearn.model_selection import train_test_split" for the splitting.

(ii) Construct the target output using one-hot encoding. Hint: you might want to utilize "from sklearn.preprocessing import OneHotEncoder".

(iii) Apply linear regression for classification to the dataset generated in part (ii) without regularization unless $\mathbf{X}^T\mathbf{X}$ is not invertible (then, use $\lambda = 0.00001$). Compute the training, validation and test accuracies. Demonstrate the confusion matrix for the test set. Hint: you might want to utilize "from sklearn.metrics import accuracy_score, confusion_matrix" for evaluation metrics.

(iv) Perform logistic regression for classification with different learning rates: 0.1, 0.01 and 0.001. The number of iterations for training is set to 20000. Plot the cost over training iterations for each learning rate. Which learning rate provides the best accuracy in the training and the validation sets? Which learning rate should be selected? Use the selected learning rate to compute the accuracy and confusion matrix for the test set.

Answer: See TUT6_Q9.ipynb

Note: using different "random_state" in "train_test_split" may have different results. The following answers are obtained using "random_state=42".

(iii) Training accuracy: 0.8417; Validation accuracy: 0.8; Test accuracy: 0.9333

Confusion matrix for test set is $\begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 1 \\ 0 & 0 & 4 \end{bmatrix}$.

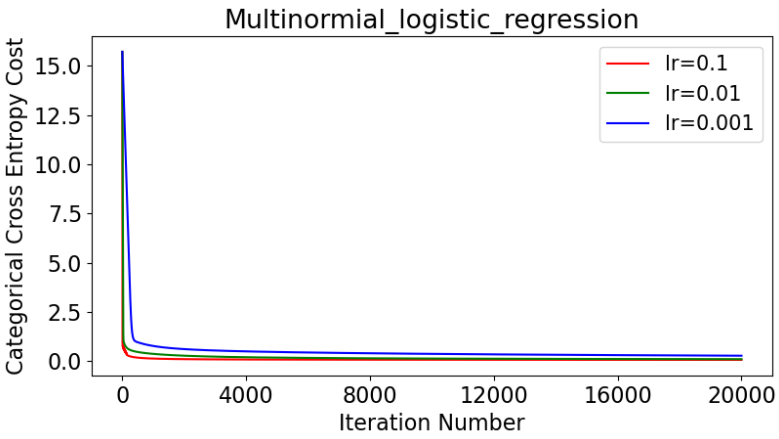
(iv) Training accuracy: [0.975, 0.975, 0.975]

Validation accuracy: [1, 1, 1]

Learning rate 0.1 should be selected as it converges the fastest although both validation and training accuracy are the same for all the learning rates.

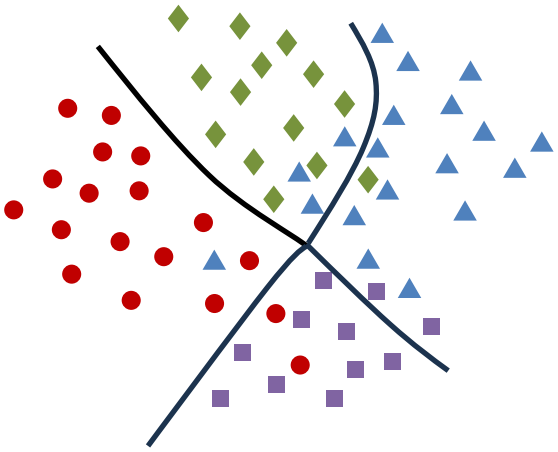
Test accuracy for learning rate 0.1: 0.9333

Confusion matrix for test set is: $\begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 1 \\ 0 & 0 & 4 \end{bmatrix}$



Question 10 (Lecture 9)

Tabulate the confusion matrix for the following classification problem. The class-1, class-2, class-3, and class-4 data points are indicated by circles, diamonds, triangles, and squares.



Answer:

	$P_{\hat{1}}$	$P_{\hat{2}}$	$P_{\hat{3}}$	$P_{\hat{4}}$
P_1 (circles)	16	0	0	2
P_2 (diamonds)	0	13	1	0
P_3 (triangles)	1	3	15	0
P_4 (squares)	0	0	2	9

NATIONAL UNIVERSITY OF SINGAPORE
DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

ACADEMIC YEAR 2025-2026
SEMESTER 1

EE2213: Introduction to AI

TUTORIAL 7: L10-L11

Question 1 (Lecture 10)

What type of cluster does the K-means clustering method form?

(a) Well-separated clusters

(b) Center-based clusters

(c) Contiguous clusters

(d) Density-based clusters

Answer: (b)

Question 2 (Lecture 10)

The K-means clustering method can only find local optima.

(a) True

(b) False

Answer: (a). Because the cost function is non-convex.

Question 3 (Lecture 10)

In order to determine an optimal K from a possible set of K values for K-means clustering, Elbow method which selects the K with the minimum within-cluster variance can be applied.

(a) True

(b) False

Answer: (b). Should choose K corresponding to "bend" or "knee", instead of the minimum within-cluster variance.

Question 4 (Lecture 10)

Prove that the membership degree sums to 1 for each data point in Fuzzy C-means clustering.

Answer:

Need to prove $\sum_{k=1}^{C} w_{ik} = 1$

$$\sum_{k=1}^{C} w_{ik} = \sum_{k=1}^{C} \frac{1}{\sum_{j=1}^{C} \left(\frac{\|\boldsymbol{x}_i - \boldsymbol{c}_k\|}{\|\boldsymbol{x}_i - \boldsymbol{c}_j\|}\right)^{\frac{2}{r-1}}} = \sum_{k=1}^{C} \frac{1}{\sum_{j=1}^{C} \frac{(\|\boldsymbol{x}_i - \boldsymbol{c}_k\|)^{\frac{2}{r-1}}}{(\|\boldsymbol{x}_i - \boldsymbol{c}_j\|)^{\frac{2}{r-1}}}}$$

$$= \sum_{k=1}^{C} \frac{1}{(\|\boldsymbol{x}_i - \boldsymbol{c}_k\|)^{\frac{2}{r-1}} \sum_{j=1}^{C} \frac{1}{(\|\boldsymbol{x}_i - \boldsymbol{c}_j\|)^{\frac{2}{r-1}}}}$$

$$= \sum_{k=1}^{C} \frac{1}{(\|\mathbf{x}_i - \boldsymbol{c}_k\|)^{\frac{2}{r-1}}} \frac{1}{\sum_{j=1}^{C} \frac{1}{(\|\mathbf{x}_i - \boldsymbol{c}_j\|)^{\frac{2}{r-1}}}}$$

$$= \sum_{k=1}^{C} \frac{1}{(\|\boldsymbol{x}_i - \boldsymbol{c}_k\|)^{\frac{2}{r-1}}} \frac{1}{\sum_{k=1}^{C} \frac{1}{(\|\boldsymbol{x}_i - \boldsymbol{c}_k\|)^{\frac{2}{r-1}}}} = 1$$

Question 5 (Lecture 10)

Consider the following data points: $\mathbf{x}_1 = \begin{bmatrix}0\\0\end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix}1\\0\end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix}0\\1\end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix}1\\1\end{bmatrix}$. Both the K-means and Fuzzy C-means clustering are initialized with centers at $\mathbf{x}_1$ and $\mathbf{x}_2$.

(i) What are the two cluster centers found by K-means clustering upon convergence, or after a maximum of 100 iterations?

(ii) What are the two cluster centers found by Fuzzy C-means clustering upon convergence, or after a maximum of 100 iterations? Assume fuzzier r=2.

(0.5, 0.5)
(1, 0.5)
(0.0449, 0.5)
(0.9551, 0.5)

Answer: See TUT_Q5.ipynb

(i) Initialization: $\mathbf{c}_1 = \begin{bmatrix}0\\0\end{bmatrix}, \mathbf{c}_2 = \begin{bmatrix}1\\0\end{bmatrix}$

1st iteration: Assignment:

$$\text{dist}(\mathbf{x}_1, \mathbf{c}_1) = 0, \text{dist}(\mathbf{x}_1, \mathbf{c}_2) = 1, \mathbf{x}_1 \epsilon \text{Cluster 1}$$
$$\text{dist}(\mathbf{x}_2, \mathbf{c}_1) = 1, \text{dist}(\mathbf{x}_2, \mathbf{c}_2) = 0, \mathbf{x}_2 \epsilon \text{Cluster 2}$$
$$\text{dist}(\mathbf{x}_3, \mathbf{c}_1) = 1, \text{dist}(\mathbf{x}_3, \mathbf{c}_2) = 1.41, \mathbf{x}_3 \epsilon \text{Cluster 1}$$
$$\text{dist}(\mathbf{x}_4, \mathbf{c}_1) = 1.41, \text{dist}(\mathbf{x}_4, \mathbf{c}_2) = 1, \mathbf{x}_4 \epsilon \text{Cluster 2}$$

Centroid Update: $\mathbf{c}_1 = \frac{1}{2}(\mathbf{x}_1 + \mathbf{x}_3) = \begin{bmatrix}0\\0.5\end{bmatrix}, \mathbf{c}_2 = \frac{1}{2}(\mathbf{x}_2 + \mathbf{x}_4) = \begin{bmatrix}1\\0.5\end{bmatrix}$

2nd iteration: Assignment:

$$\text{dist}(\mathbf{x}_1, \boldsymbol{\mu}_1) = 0.5, \text{dist}(\mathbf{x}_1, \boldsymbol{\mu}_2) = 1.12, \mathbf{x}_1 \epsilon \text{Cluster 1}$$
$$\text{dist}(\mathbf{x}_2, \boldsymbol{\mu}_1) = 1.12, \text{dist}(\mathbf{x}_2, \boldsymbol{\mu}_2) = 0.5, \mathbf{x}_2 \epsilon \text{Cluster 2}$$
$$\text{dist}(\mathbf{x}_3, \boldsymbol{\mu}_1) = 0.5, \text{dist}(\mathbf{x}_3, \boldsymbol{\mu}_2) = 1.12, \mathbf{x}_3 \epsilon \text{Cluster 1}$$
$$\text{dist}(\mathbf{x}_4, \boldsymbol{\mu}_1) = 1.12, \text{dist}(\mathbf{x}_4, \boldsymbol{\mu}_2) = 0.5, \mathbf{x}_4 \epsilon \text{Cluster 2}$$

Cluster assignment is unchanged. Converged.

Two cluster centers found: $\mathbf{c}_1 = \begin{bmatrix}0\\0.5\end{bmatrix}, \mathbf{c}_2 = \begin{bmatrix}1\\0.5\end{bmatrix}$

(ii) Two cluster centers found: $\mathbf{c}_1 = \begin{bmatrix}0.0449\\0.4999\end{bmatrix}, \mathbf{c}_2 = \begin{bmatrix}0.9551\\0.4999\end{bmatrix}$

Membership matrix $\mathbf{W} = \begin{bmatrix} 0.8219 & 0.1781 \\ 0.1781 & 0.8219 \\ 0.8217 & 0.1783 \\ 0.1783 & 0.8217 \end{bmatrix}$

Question 6 (Lecture 10)

Load the wine dataset by "from sklearn.datasets import load_wine". Assume that the class labels are not given. Apply K-means clustering algorithm to group all the data with different K from 2 to 10. Which K is the best according to the Elbow method?

Answer: See TUT7_Q6.ipynb

Choose the K with "bend" or "knee" (Abrupt change in the slop): either 3 or 4.

Question 7 (Lecture 11)

In neural networks, what is the purpose of introducing nonlinear activation functions like sigmoid and ReLU?

(a) Ensure the output values to be between 0 and 1.

(b) Enable the model to learn complex patterns.

(c) Speed up the gradient calculation in backpropagation.

Answer: (b)

Question 8 (Lecture 11)

A multi-layer perceptron (MLP) of 2 layers has been constructed as:

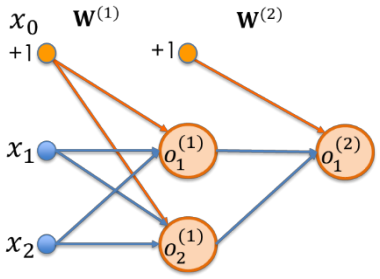
$$f_{\mathbf{W}}(\mathbf{x}) = [1, \sigma(\mathbf{x}^T\mathbf{W}^{(1)})]\mathbf{W}^{(2)}$$

where σ is the ReLU function, $\mathbf{W}^{(1)} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \\ 1 & -1 \end{bmatrix}, \mathbf{W}^{(2)} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$.

Suppose that this MLP is used to train a dataset $\mathbf{X} = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 5 & 1 \end{bmatrix}$ with the target $\mathbf{y} = \begin{bmatrix} 0.1 \\ 0.7 \end{bmatrix}$. The cost function is $J(\mathbf{W}) = \frac{1}{N}\sum_{i=1}^{N}(f_{\mathbf{W}}(\mathbf{x}_i) - y_i)^2$ for optimization. The learning rate η is 0.1. What will $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ be after the 1st iteration of backpropagation?

Answer: See TUT_Q8.ipynb

1. Architecture of MLP:



2. Learning:

1st iteration: Step 1: Forward Pass:

$$\mathbf{f}_{\mathbf{W}(0)}(\mathbf{X}) = \left[\mathbf{1}, \boldsymbol{\sigma}^{(1)}\left(\mathbf{X}\mathbf{W}^{(1)}(0)\right)\right]\mathbf{W}^{(2)}(0)$$

$$= \left[\mathbf{1}, ReLU\left(\begin{bmatrix}1 & 2 & 1\\ 1 & 5 & 1\end{bmatrix}\begin{bmatrix}-1 & 0\\ 0 & 1\\ 1 & -1\end{bmatrix}\right)\right]\begin{bmatrix}1\\ -1\\ 1\end{bmatrix}$$

$$= \begin{bmatrix}1 & 0 & 1\\ 1 & 0 & 4\end{bmatrix}\begin{bmatrix}1\\ -1\\ 1\end{bmatrix}$$

$$= \begin{bmatrix}2\\ 5\end{bmatrix}$$

Step 2: Backward Pass:

$$\mathcal{L} = (f_{\mathbf{W}}(\mathbf{x}) - y)^2$$

(1) Output layer:

$$formula\ from\ the\ lecutre\ notes: \frac{\partial \mathcal{L}}{\partial w_{i,j}^{(L)}} = \epsilon_j^{(L)} a_i^{(L)},$$

$$where\ \epsilon_j^{(L)} = \sum_{m=1}^{D_L} \boldsymbol{\mathcal{L}}'(o_m^{(L)})\boldsymbol{\sigma}^{(L)\prime}(z_j^{(L)})_m$$

$$\epsilon_1^{(2)} = \boldsymbol{\mathcal{L}}'\left(o_1^{(2)}\right) = \boldsymbol{\mathcal{L}}'\left(f_{\mathbf{W}}(\mathbf{x})\right) = 2(f_{\mathbf{W}}(\mathbf{x}) - y)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(2)}} = 2(f_{\mathbf{W}}(\mathbf{x}) - y)\mathbf{a}^{(2)}$$

$$\frac{dJ}{d\mathbf{W}^{(2)}} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial \mathcal{L}_i}{\partial \mathbf{W}^{(2)}} = \frac{1}{N}\sum_{i=1}^{2} 2(f_{\mathbf{W}}(\mathbf{x}_i) - y_i)\mathbf{a}^{(2)} = \frac{1}{N}(\mathbf{A}^{(2)})^T 2(\mathbf{f}_{\mathbf{W}}(\mathbf{X}) - \mathbf{y})$$

$$\frac{dJ(\mathbf{W}^{(2)}(0))}{d\mathbf{W}^{(2)}} = \begin{bmatrix}1 & 0 & 1\\ 1 & 0 & 4\end{bmatrix}^T\left(\begin{bmatrix}2\\ 5\end{bmatrix} - \begin{bmatrix}0.1\\ 0.7\end{bmatrix}\right) = \begin{bmatrix}6\\ 0\\ 19.1\end{bmatrix}$$

$$\mathbf{W}^{(2)}(1) = \mathbf{W}^{(2)}(0) - \eta\frac{dJ\left(\mathbf{W}^{(2)}(0)\right)}{d\mathbf{W}^{(2)}} = \begin{bmatrix}1\\ -1\\ 1\end{bmatrix} - 0.1\begin{bmatrix}6\\ 0\\ 19.1\end{bmatrix} = \begin{bmatrix}0.4\\ -1\\ -0.91\end{bmatrix}$$

(2) Hidden layer:

Since it is ReLU activation,

$$\frac{dJ\left(\mathbf{W}^{(1)}(0)\right)}{d\mathbf{W}^{(1)}} = \frac{1}{N}(\mathbf{A}^{(1)})^T\mathbf{E}^{(1)},$$

$$where\ \mathbf{E}^{(1)} = \boldsymbol{\delta}_{\mathbf{Z}^{(1)}} * \left(\mathbf{E}^{(2)}\left(\mathbf{W}^{(2)}_{1:,:}(0)\right)^T\right)$$

$$= \begin{bmatrix}0 & 1\\ 0 & 1\end{bmatrix} * \left(2\left(\begin{bmatrix}2\\5\end{bmatrix} - \begin{bmatrix}0.1\\0.7\end{bmatrix}\right)[-1,1]\right) = \begin{bmatrix}0 & 3.8\\ 0 & 8.6\end{bmatrix},$$

$$\mathbf{A}^{(1)} = \mathbf{X} = \begin{bmatrix}1 & 2 & 1\\ 1 & 5 & 1\end{bmatrix}$$

$$therefore, \frac{dJ\left(\mathbf{W}^{(1)}(0)\right)}{d\mathbf{W}^{(1)}} = \frac{1}{2}\begin{bmatrix}1 & 1\\ 2 & 5\\ 1 & 1\end{bmatrix}\begin{bmatrix}0 & 3.8\\ 0 & 8.6\end{bmatrix} = \begin{bmatrix}0 & 6.2\\ 0 & 25.3\\ 0 & 6.2\end{bmatrix}$$

$$then, \mathbf{W}^{(1)}(1) = \mathbf{W}^{(1)}(0) - \eta\frac{dJ\left(\mathbf{W}^{(1)}(0)\right)}{d\mathbf{W}^{(1)}}$$

$$= \begin{bmatrix}-1 & 0\\ 0 & 1\\ 1 & -1\end{bmatrix} - 0.1\begin{bmatrix}0 & 6.2\\ 0 & 25.3\\ 0 & 6.2\end{bmatrix} = \begin{bmatrix}-1 & -0.62\\ 0 & -1.53\\ 1 & -1.62\end{bmatrix}$$

Question 9 (Lecture 11)

The Digits dataset is a classic, lightweight dataset ideal for testing simple machine learning models like MLPs, especially on CPU. It has 10 classes (digits 0 to 9). Each sample is a grayscale image with a dimension of 8×8. Get the dataset by "from sklean.datasets import load_digits". Use Python to do the following tasks.

(i) Split the database into three sets: 70% of samples for training, 15% of samples for validation, and 15% of samples for testing. Normalize the features using z-score standardization. Hint: you may use "from sklearn.preprocessing import StandardScaler".

(ii) Construct the target output using one-hot encoding.

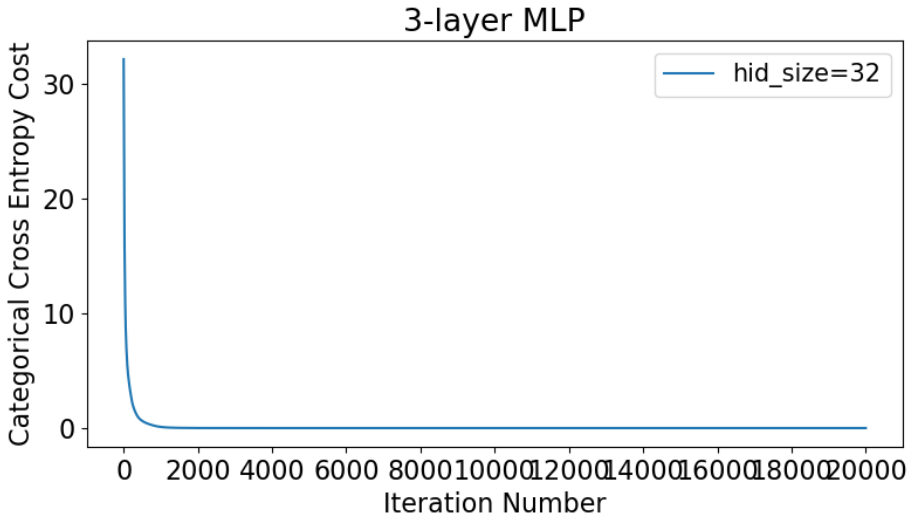
(iii) Use a 3-layer MLP for the digit classification. Suppose that ReLU is the activation function for the hidden layers and Softmax is the activation function for the output layer. Assume that the number of the neurons for each hidden layer is 32. The learning rate is 0.01. The number of iterations for training is 20000. The objective function is $J(\mathbf{W}) = \frac{1}{N}\sum_{i=1}^{N} L_{CCE}(\mathbf{f}_{\mathbf{W}}(\mathbf{x}_i), \mathbf{y}_i)$, where $L_{CCE}(\mathbf{f}_{\mathbf{W}}(\mathbf{x}_i), \mathbf{y}_i)$ is the categorical cross-entropy loss. Plot the cost over number of iterations. Compute the training accuracy and validation accuracy.

(iv) Change the hidden layer size to 36 and do the same thing as in (iii). Which one is better? Use the model with the better hidden layer size to compute the accuracy of the test set.

Answer: See TUT7_Q9

Note: Different initialization of the weight parameters and data splitting may have different results.

(iii) Training accuracy = 1, Validation accuracy = 0.8741



(iv) Training accuracy = 1, Validation accuracy = 0.8259,

Hidden layer size 32 is better according to the validation accuracy.

Test accuracy for model with hidden layer size of 32 = 0.8444.