Linear Regression and Logistic Regression: Theory and Applications

AAE4011 – Artificial Intelligence for Unmanned Autonomous Systems (UAS)

Dr Weisong Wen
Assistant Professor, Director of PolyU TAS Lab
Department of Aeronautical and Aviation Engineering
The Hong Kong Polytechnic University
Week 6, S2, 2024/2025

Our Teaching Assistants


Zhang Ziqi


Yang Qian


Wang Xin


Qiu Shaoting


Hu Runzhi


Ma Pei

- Role/features of TAs in this course
  - Helper in lab session
  - Expert in AI and coding with Python
  - Experts in UAS, such as drones

- Let’s get to know with each other
  - Short introduction about yourself (if we have enough time)? ☺
  - Who is your Final Year Project supervisor and what is your topic? ☺
  - Why you select this course? ☺

Ground Rules

- ✓ For students:
  - Open mind; speak English; participate activities assigned; ask questions

- ✓ For teachers:
  - Arrive on time; reply emails on time; answer questions related to the subject

- ✓ Be curious, Be inspired, Be motivated, Study further by yourself.

Assessment and Basic Requirement

- Assessment:
  - Homework Assignment (Strictly no late submission) (20%)
  - Mid-Term Quiz/Test (15%, close book)
  - Group Project (Case study, several members in a group) (15%)
  - Final Exam (50%) (Open book)
- Basic requirement:
  - Mathematics on matrix and its calculation
  - Extra time for finish the coding homework based on Python
  - Assurance on the attendance
  - Basic coding skills with Python (expect to learn yourself for extra), one week lecture for basics of Python

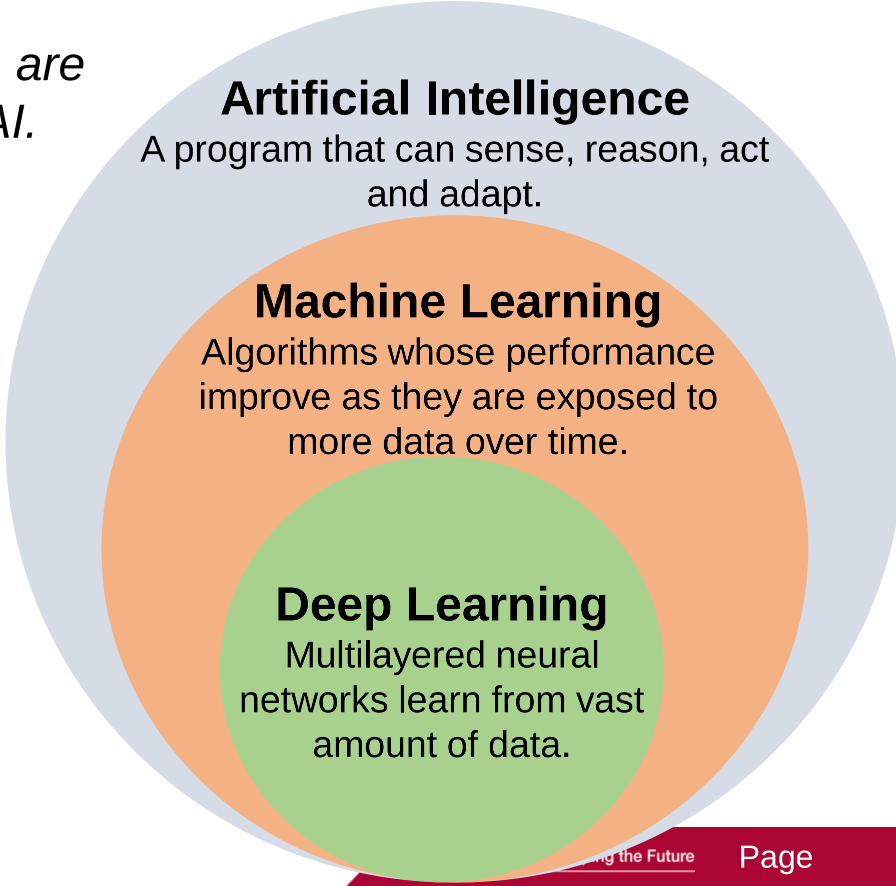
Outline for today

- Regression and classification
- Decision tree and applications
- Random forest and applications

Category of AI

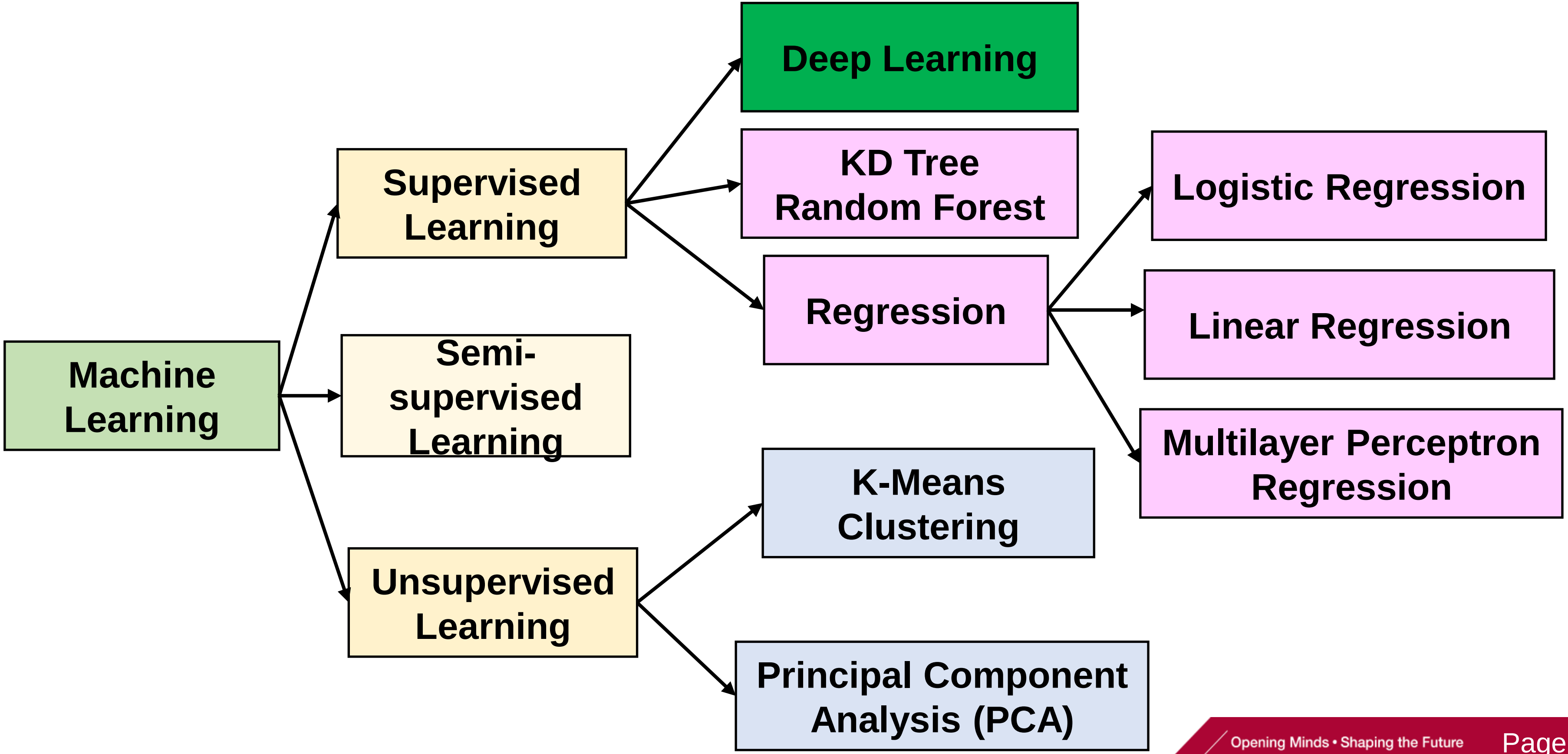
Machine learning (ML) and deep learning (DL) are important branches and core technologies of AI.

- Feature of Deep learning:
  - A science devoted to making machines think and act like humans.
  - Focuses on enabling computers to perform tasks without explicit programming.
  - A subset of machine learning based on artificial neural networks.



ML is a method to achieve AI.
DL is a technique for implementing ML.

Framework of the main categories



Regression Analysis

Regression for classification is an interesting approach where regression techniques are adapted to solve classification problems. While regression is typically used for predicting continuous outcomes, it can be modified to handle discrete class labels. Here's an introduction to how this works:

➢ **Key Concepts**

1. **Regression vs. Classification**:
   - **Regression**: Involves predicting a continuous output. For example, predicting the price of a house.
   - **Classification**: Involves predicting a discrete label. For example, determining whether an email is spam or not.
2. **Using Regression for Classification**:
   - The idea is to use a regression model to predict a continuous score, which is then mapped to a discrete class label.
   - This can be done by setting a threshold. For example, if the regression output is above a certain value, it is classified as one class, otherwise another.
3. **Logistic Regression**:
   - Despite its name, logistic regression is actually a classification algorithm.
   - It uses a logistic function to model the probability that a given input belongs to a particular class.
   - The output is a probability between 0 and 1, which can be thresholded to decide the class label.

Regression Analysis

Determine the quantitative relationship between two or more variables based on data

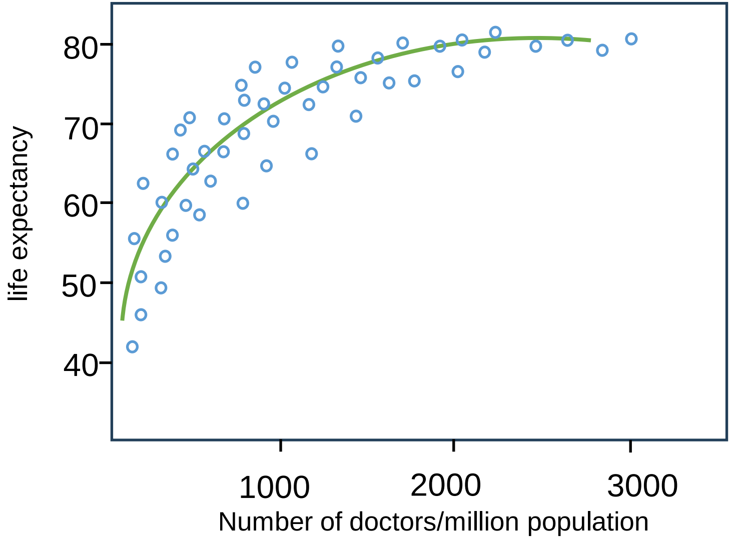
$$y = f(x_1, x_2 \dots x_n)$$

- Regression
  - Variables
    - Single-variable regression: $y = f(x)$
    - Multiple-variable regression: $y = f(x_1, x_2 \dots x_n)$
  - Functional relationships
    - Linear regression: $y = ax + b$
    - Nonlinear regression: $y = ax^2 + bx + c$

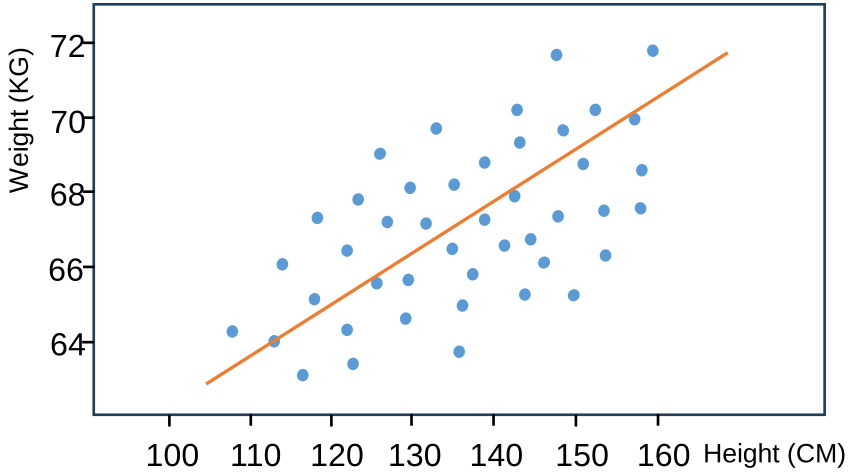
f: the functional relationships
x: the independent variables
y: the dependent variables
a,b,c: coefficients

• What is the Regression Analysis

- ➢ **Number of doctors per million people predicts life expectancy in a region**



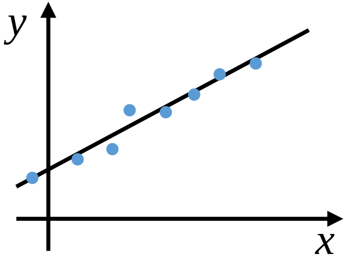
- ➢ **Using Height to Predict Weight**



Linear/Nonlinear Regression

Linear Regression

There is a linear relationship between the variable and the dependent variable



$$y = ax + b$$

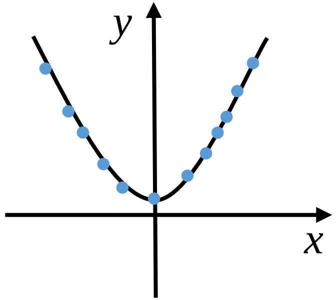
Example of distance and speed：

$$S = v \times t + S_0$$

Machine Learning

Supervised Learning

Nonlinear Regression



$$y = ax^2 + bx + c$$

Example of distance and acceleration：

$$S = a \times t^2 + S_0$$

s: the distance
a: the acceleration
v: the speed
t: the time

• Solving regression problems

Is a 110 Square Meter House for 1.5 Million a Good Investment?

Area	Price
79	404,976
92	948,367
…	…
108	1,049,007
110	?
118	578,142
…	…

- ✓ **Establish the Relationship Between P and A**

$$P = f(A)$$

- ✓ **Predict Price Based on Relationships**

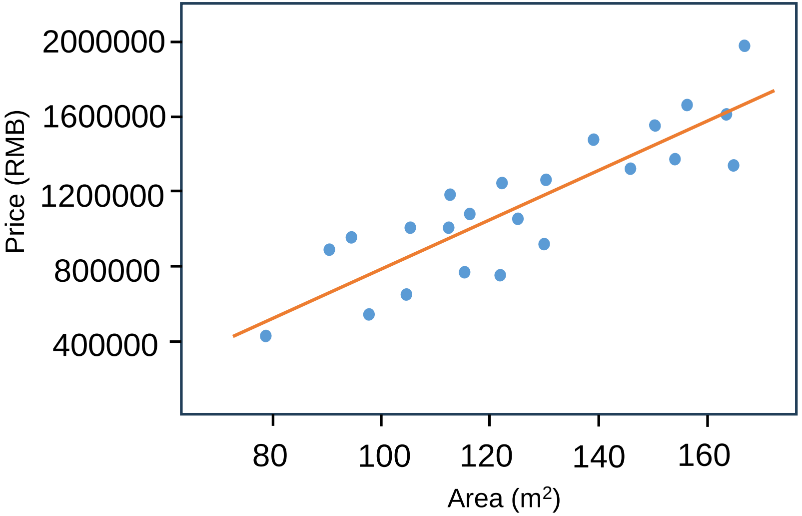
$$P_{(A=110)} = f(110)$$

- ✓ **Evaluate results (for example)**

$$\boxed{P_{(A=110)} >> 1,500,000}$$ ➡ **Yes**

• Solving regression problems

Is a 110 Square Meter House for 1.5 Million a Good Investment?



✓ **Establish the Relationship Between P and A**

$$P = f(A)$$

- **Linear relationship**

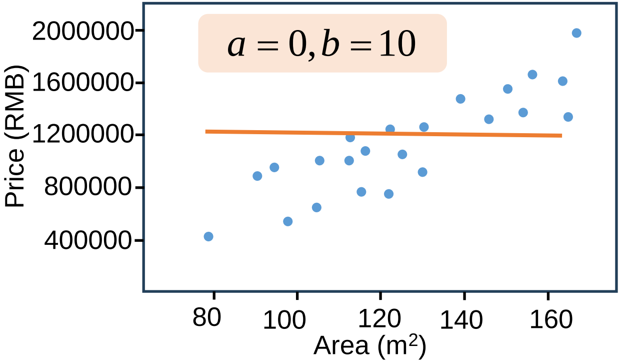
$$y = ax + b$$

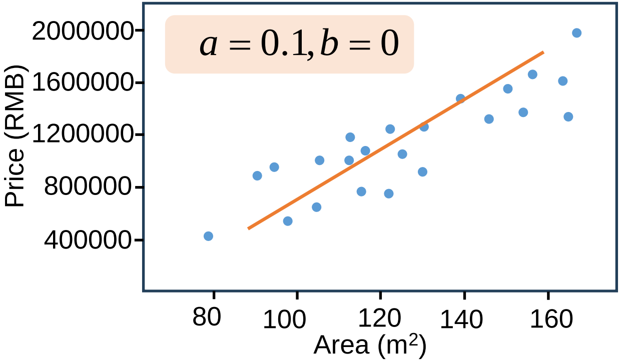
⬇

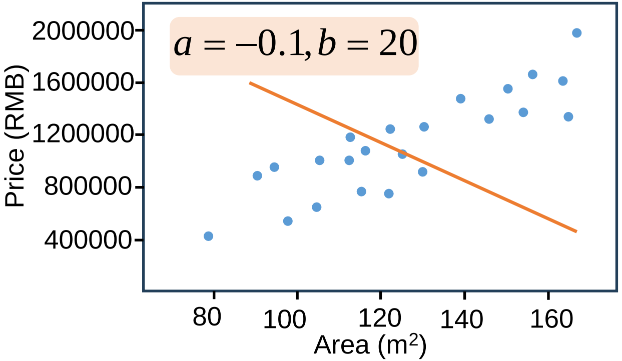
$$a, b?$$

• Solving regression problems

Is a 110 Square Meter House for 1.5 Million a Good Investment?







How to get the most suitable a and b

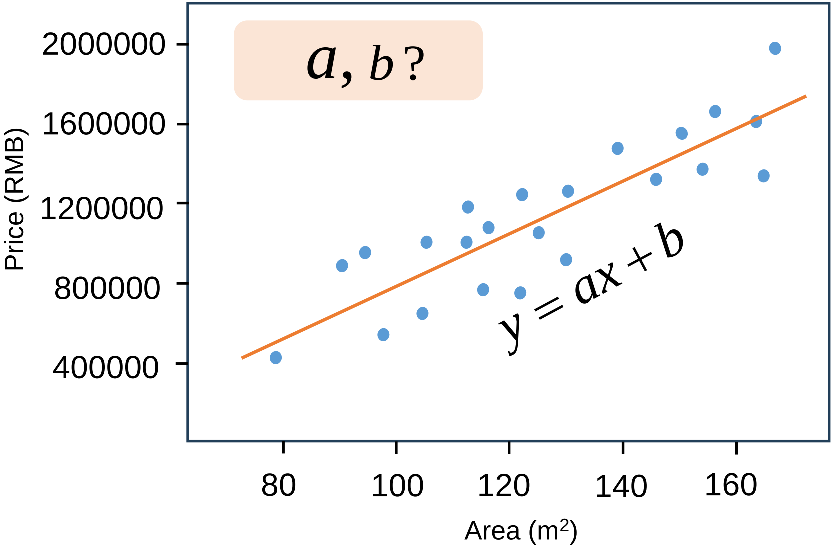
- x : Model input
- y_i : True value
- y_i' : Model output
- m : Sample size

→ y_i' is as close to y_i as possible →

$$minimize\left\{\sum_{i=1}^{m}(y_i'-y_i)^2\right\}$$

• Solving regression problems

Is a 110 Square Meter House for 1.5 Million a Good Investment?



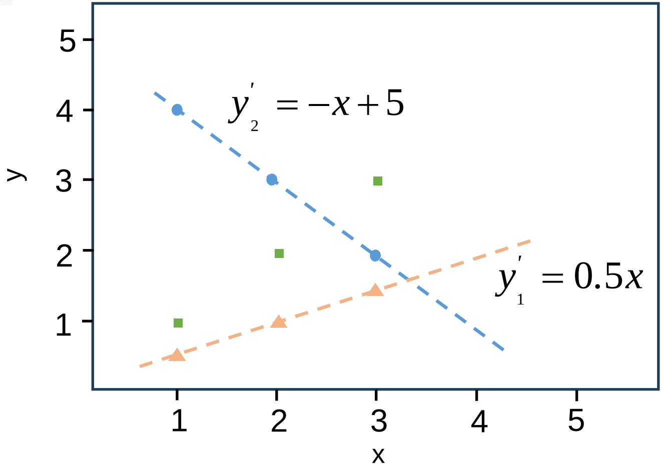
• **Loss Function**

$$minimize\left\{\sum_{i=1}^{m}(y_i^{'} - y_i)^2\right\}$$

$$minmize\left\{\frac{1}{2m}\sum_{i=1}^{m}(y_i^{'} - y_i)^2\right\}$$

• Solving regression problems

Is a 110 Square Meter House for 1.5 Million a Good Investment?



x	y	y_1'	y_2'
1	1	0.5	4
2	2	1	3
3	3	1.5	2

$$J_1 = \frac{1}{2m}\sum_{i=1}^{m}(y_1' - y)^2 = \frac{1}{2\times 3}\times((0.5-1)^2 + (1-2)^2 + (1.5-3)^2) = 0.583$$

$$J_2 = \frac{1}{2m}\sum_{i=1}^{m}(y_2' - y)^2 = \frac{1}{2\times 3}\times((4-1)^2 + (3-2)^2 + (2-3)^2) = 1.83$$

• Solving regression problems

$$min\,mize(\ J\ )$$

$$J=\frac{1}{2m}\sum_{i=1}^{m}(y_i^{'}-y_i)^2=\frac{1}{2m}\sum_{i=1}^{m}(ax_i+b-y_i)^2=g(a,b)$$

✓ **Optimization: Gradient Descent**

- Calculate the gradient of the loss function for each parameter (representing the slope's direction at the present parameter value)
- Find the minimum value (adjust the parameter in the opposite direction of the gradient to diminish the loss function's value.)

$$J=f(\ p\ )$$

Search methods →

$$p_{i+1}=p_i-\alpha\frac{\partial}{\partial p_i}f(\ p_i\ )$$

- **Solving regression problems (do this yourself again)**

Example

$$J = f(P) = 3.5p^2 - 14p + 14$$

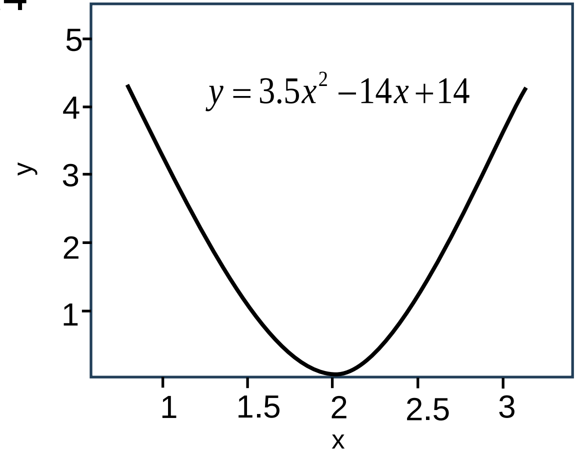
$$p_i = 0.5, \ \alpha = 0.01$$

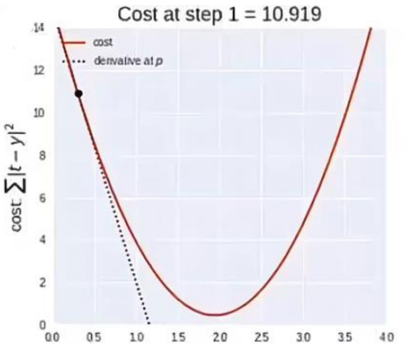
$$p_{i+1} = ?$$

$$\frac{\partial}{\partial p_i} f(p_i) = 7p - 14$$

$$\frac{\partial}{\partial p_i} f(p_i) = -10.5$$

$$p_{i+1} = p_i - \alpha \frac{\partial}{\partial p_i} f(p_i) = 0.5 + 0.105 = 0.605$$





Gradually approaching the minimum point （P=2）

- **Solving regression problems**

$$min\,mize(\,J\,)$$

$$J=\frac{1}{2m}\sum_{i=1}^{m}(y_i^{'}-y_i)^2=\frac{1}{2m}\sum_{i=1}^{m}(ax_i+b-y_i)^2=\boxed{g(a,b)}$$

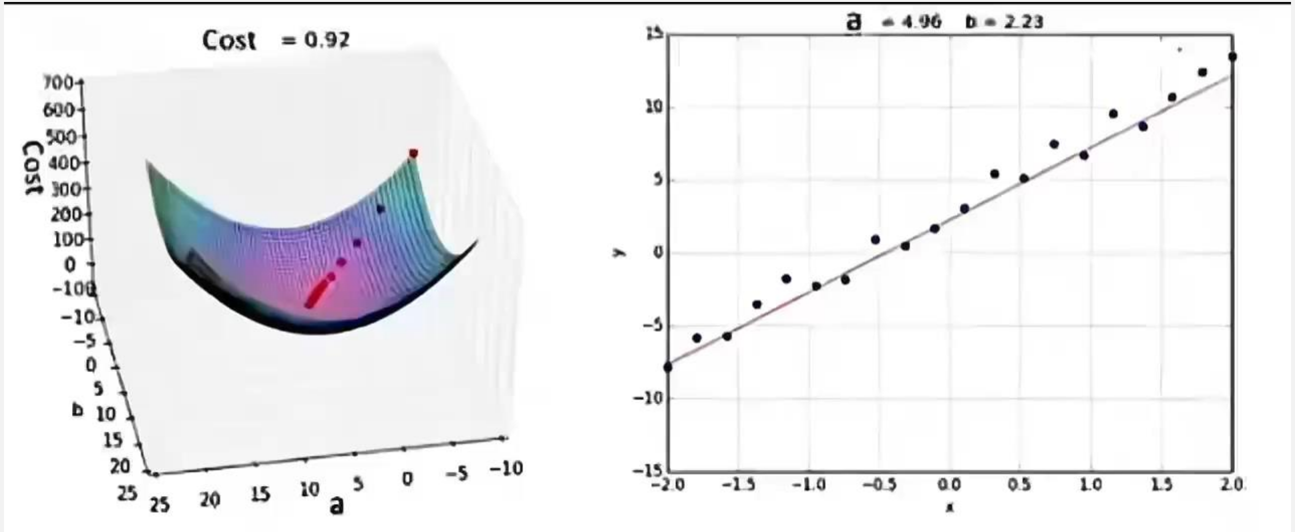
✓ **Continue this iterative process until convergence is achieved**

$$\left\{\begin{array}{c} temp_a=a-\alpha\dfrac{\partial}{\partial a}g(a,b)=a-\alpha\dfrac{1}{m}\sum_{i=1}^{m}(ax_i+b-y_i)x_i \\ temp_b=b-\alpha\dfrac{\partial}{\partial b}g(a,b)=b-\alpha\dfrac{1}{m}\sum_{i=1}^{m}(ax_i+b-y_i) \\ a=temp_a \\ b=temp_b \end{array}\right\}$$

• Solving regression problems

✓ **Function fitting process**

$$Cost = f(a,b) \qquad y = ax + b$$



• Examples and Python code

$$y = ax + b$$

X	y
1	7
2	9
3	11
4	13
5	15
6	17
7	19
8	21
9	23
10	25

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
# Load the dataset
data = pd.read_csv('E:\TEST/***.csv')  # Replace with your dataset file path
# Check the column names in the dataset
print(data.columns)
# Assume the dataset has two columns X and y, prepare the data
X = data['X'].values.reshape(-1, 1)  # Feature variable
y = data['y'].values  # Target variable
# Create and fit the linear regression model
lin_reg = LinearRegression()
lin_reg.fit(X, y)
```

• Examples and Python code

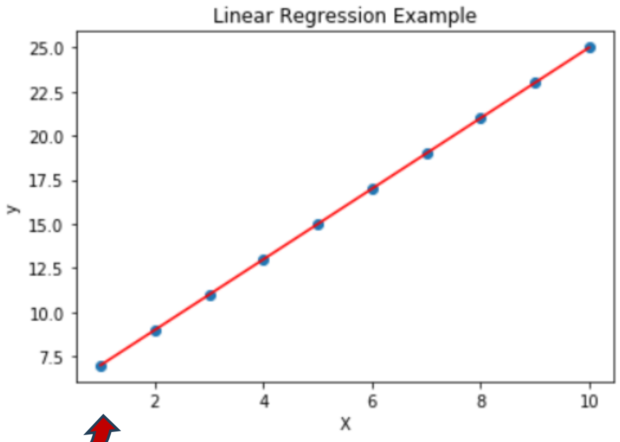
$$y = ax + b$$

X	y
1	7
2	9
3	11
4	13
5	15
6	17
7	19
8	21
9	23
10	25

```
# Extract the regression coefficients and intercept
slope = lin_reg.coef_[0]
intercept = lin_reg.intercept_
# Display the fitted equation
equation = f'y = {slope}x + {intercept}'
print("Fitted linear regression equation:", equation)
# Visualize the data and the fitted line
plt.scatter(X, y)
plt.plot(X, lin_reg.predict(X), color='red')
plt.xlabel('X')
plt.ylabel('y')
plt.title('Linear Regression Example')
plt.show()
```

Results:

```
Index(['X', 'y'], dtype='object')
Fitted linear regression equation: y = 2.0x + 5.0
```



- **Steps of Regression Analysis**

Step 1: Determine the independent variables and dependent variables

Step 2: Determine the model and establish the regression equation

- Generate the loss function
- Gradient descent method to solve parameters

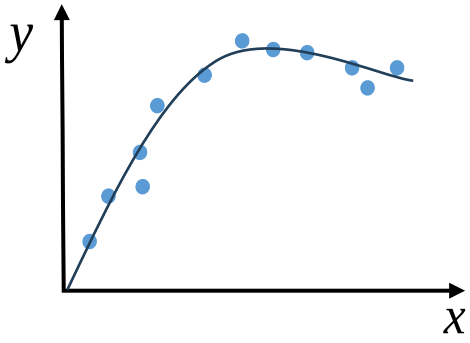
Step 3: Verify the regression equation

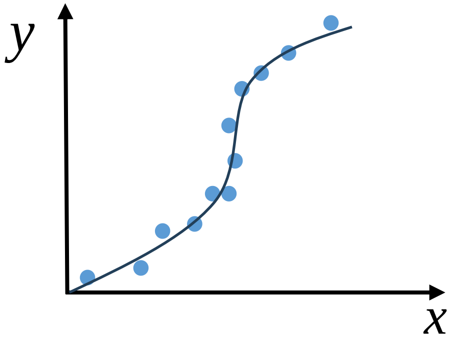
Step 4: Use the regression equation for prediction

• Nonlinear Regression

- ✓ Explore the nonlinear relationship between independent variables and dependent variables
- ✓ Use nonlinear models to describe how the dependent variable changes with the independent variable.

➢ **Nonlinear relationships**



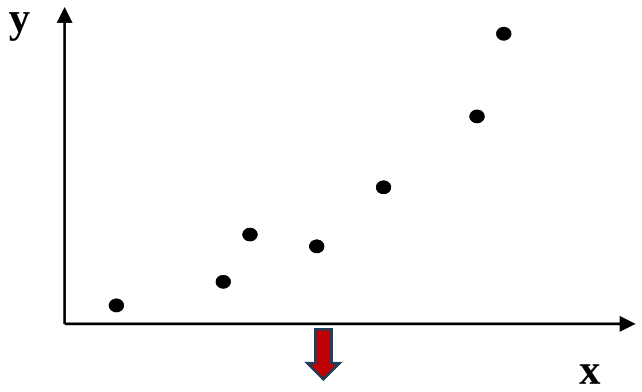


Nonlinear regression **can better fit the data** when the relationship between the independent and dependent variables is **curvilinear or exponential**.

• Stages of Nonlinear Regression Analysis

Step 1: Draw a scatter plot and determine the model of the regression equation

x	x_1	x_2	…	x_n
y	y_1	y_2	…	y_n



$$y = ax^b$$

✓ Regression Model

Stages of Nonlinear Regression Analysis

Step 2: Find the unknown coefficients in the equation and establish the regression equation

1. Convert nonlinear equations to linear equations

$$y = ax^b \xrightarrow{\text{Log both sides}} \lg y = \lg a + b \lg x$$

$$y' = \lg y \quad a' = \lg a \quad x' = \lg x$$

$$\Rightarrow y' = a' + bx'$$

2. Find the unknown coefficients to establish linear equations

3. Convert linear equations to nonlinear equations

✓ As above example $\Rightarrow a = \lg^{-1} a'$

• Converting Nonlinear Regression into a Linear Form

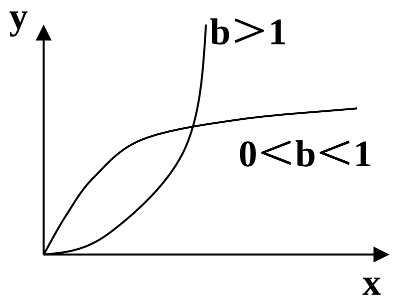
➢ Popular Curve Equation Types and Straightening Methods:

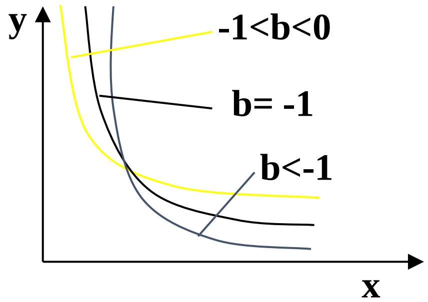
1. Power function

$$y = ax^b \quad (a \neq 0)$$

Log both sides ↓ Straightening

$$\lg y = \lg a + b \lg x$$



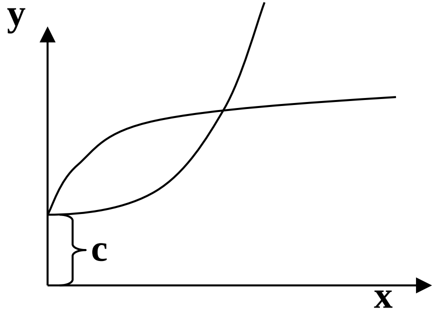


1.2 Power function with constant

$$y = c + ax^b \quad (a \neq 0)$$

Log both sides ↓ Straightening

$$\lg(y - c) = \lg a + b \lg x$$



• Converting Nonlinear Regression into a Linear Form

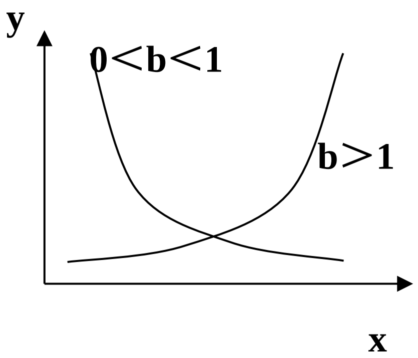
➢ Popular Curve Equation Types and Straightening Methods:

2. Exponential function

$$y = ab^x \quad (b \neq 1)$$

Log both sides ⬇ Straightening

$$\lg y = \lg a + x \lg b$$

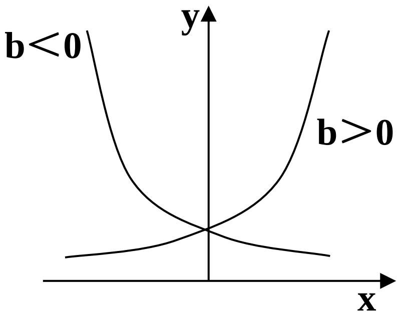


2.1

$$y = ae^{bx}$$

Log both sides ⬇ Straightening

$$\ln y = \ln a + bx$$



• Converting Nonlinear Regression into a Linear Form

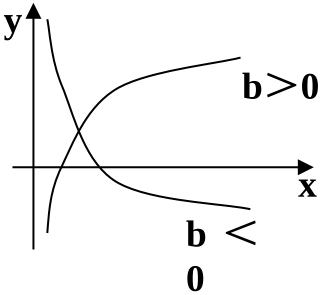
➢ Popular Curve Equation Types and Straightening Methods:

3. Logarithmic function

$$y = a + b\lg x \quad (b \neq 1)$$

Log both sides ⬇ Straightening

$$x' = \lg x$$

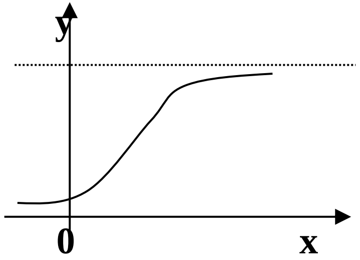


4. S-curve (logistic)

$$y = \frac{c}{1 + ae^{-bx}}$$

Log both sides ⬇ Straightening

$$\frac{c}{y} = 1 + ae^{-bx} \Rightarrow \ln\left(\frac{c}{y} - 1\right) = \ln a - bx$$

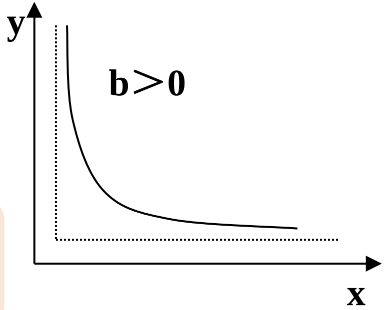


5. Hyperbolic function

$$y = \frac{x}{a + bx}$$

Log both sides ⬇ Straightening

$$\frac{x}{y} = a + bx$$

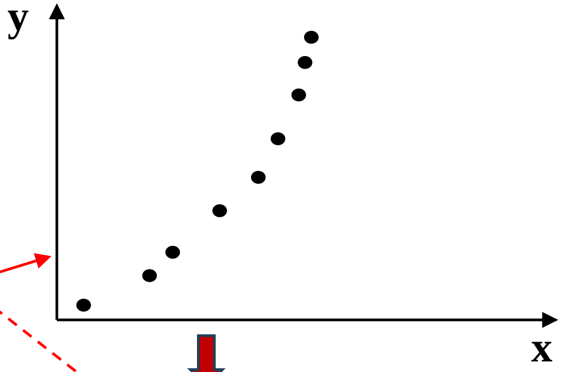


• Solving regression problems

Example **Exploring the Relationship Between Apple Diameter and Weight Over Time**

Diameter (x)	Weight (y)	x′ =lgx	y′ =lgy
2.71	11.49	0.4330	1.0603
3.26	18.68	0.5132	1.2714
3.59	24.07	0.5551	1.3815
4.02	40.10	0.6042	1.6031
4.42	55.70	0.6452	1.7458
4.69	66.92	0.6712	1.8255
4.89	80.55	0.6893	1.9061
4.97	90.96	0.6963	1.9588
5.32	113.40	0.7259	2.0546
5.61	145.90	0.7489	2.1641
5.55	145.90	0.7443	2.1641
5.31	129.40	0.7251	2.1119

1. Draw scatter plot



2. Determine the regression model:

$$y = ax^b$$

Straightening

$$\boxed{\lg y} = \boxed{\lg a} + b\boxed{\lg x} \rightarrow \boxed{y'} = \boxed{a'} + b\boxed{x'}$$

• Solving regression problems

Example

Exploring the Relationship Between Apple Diameter and Weight Over Time

Diameter (x)	Weight (y)	x′ =lgx	y′ =lgy
2.71	11.49	0.4330	1.0603
3.26	18.68	0.5132	1.2714
3.59	24.07	0.5551	1.3815
4.02	40.10	0.6042	1.6031
4.42	55.70	0.6452	1.7458
4.69	66.92	0.6712	1.8255
4.89	80.55	0.6893	1.9061
4.97	90.96	0.6963	1.9588
5.32	113.40	0.7259	2.0546
5.61	145.90	0.7489	2.1641
5.55	145.90	0.7443	2.1641
5.31	129.40	0.7251	2.1119

1. Find the unknown coefficients and establish the equation of the line:

$$\sum x' = 7.7517 \qquad \sum x'^2 = 5.1184$$

$$\sum y' = 21.2472 \qquad \sum y'^2 = 39.1177$$

$$\sum x'y' = 14.1307 \qquad \overline{x'} = 0.6460 \qquad \overline{y'} = 1.7706$$

$$SS_{x'} = \sum x'^2 - \frac{(\sum x')^2}{n} = 5.1184 - \frac{7.7517^2}{12} = 0.1110$$

$$SS_{y'} = \sum y'^2 - \frac{(\sum y')^2}{n} = 39.1177 - \frac{21.2472^2}{12} = 1.4974$$

$$SP_{x'y'} = \sum x'y' - \frac{(\sum x')(\sum y')}{n} = 14.1307 - \frac{7.7517 \times 21.2472}{12} = 0.4055$$

SS: Sum of Squares of deviation from mean
SP: The sum of the deviations of x from the mean multiplied by the deviations of y from the mean

• Solving regression problems

Example

Exploring the Relationship Between Apple Diameter and Weight Over Time

1. Find the unknown coefficients and establish the equation of the line:

$$y = ax^b$$

⇓ Straightening

$$\lg y = \boxed{\lg a} + \boxed{b} \lg x$$

⇓

$$y' = \boxed{a'} + \boxed{b}x'$$

$$b = \frac{SP_{x'y'}}{SS_{x'}} = \frac{0.4055}{0.111} = 3.6532$$

⇓

$$a' = \overline{y'} - b\overline{x'} = 1.7706 - 3.6532 \times 0.646 = -0.5894$$

⇓

$$y' = -0.5894 + 3.6532x'$$

⇓ Restore the equation to the curve equation

$$a' = \lg a = -0.5894$$

⇓

✓ **Regression Model:**

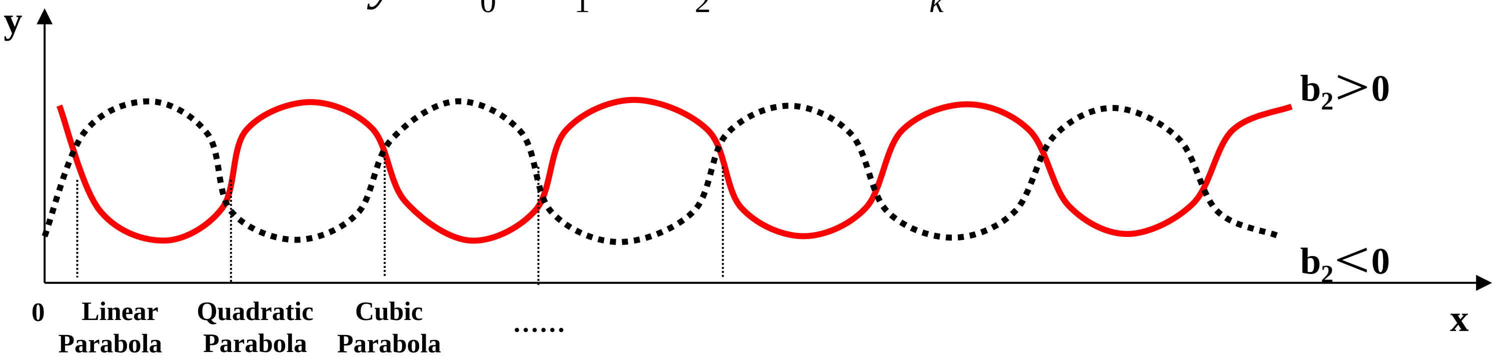
$$\hat{y} = 0.2574x^{3.6532}$$

• Polynomial Regression

➢ Polynomial Regression

Polynomial regression is a regression analysis method that models the relationship between the independent variable x and the dependent variable y as an *m*-th degree polynomial.

$$y = b_0 + b_1 x + b_2 x^2 + \ldots + b_k x^k + \varepsilon$$



- ✓ Linear Parabola: $y = b_0 + \boxed{b_1 x}$
- ✓ Quadratic Parabola: $y = b_0 + b_1 x + \boxed{b_2 x^2}$
- ✓ Cubic Parabola: $y = b_0 + b_1 x + b_2 x^2 + \boxed{b_3 x^3}$

x: the independent variables
y: the dependent variables
$b_{1,\ldots,}b_k$: coefficients
ε : error

Polynomial Regression

Polynomial Regression

✓ Calculation of regression coefficients (**Least squares**)

- For n pairs of data:

$$y_i = b_0 + b_1 x_i + b_2 x_i^2 + b_3 x_i^3$$

- Evaluation Function:

$$\min mize \;\; L(b_0, b_1, b_2, b_3) = \sum_{i=1}^{n} (y_i^{'} - y_i)^2$$

- Least squares solution:

$$\left\{\begin{aligned} \frac{\partial}{\partial b_0} L(b_0, b_1, b_2, b_3) = 0 \\ \frac{\partial}{\partial b_1} L(b_0, b_1, b_2, b_3) = 0 \\ \frac{\partial}{\partial b_2} L(b_0, b_1, b_2, b_3) = 0 \\ \frac{\partial}{\partial b_3} L(b_0, b_1, b_2, b_3) = 0 \end{aligned}\right. \Rightarrow (b_0, b_1, b_2, b_3)$$

x_i, y_i: the variables
$y_i^{'}$: the true value
$b_{1,\dots,} b_k$: coefficients

• Polynomial Regression

➢ Polynomial Regression

✓ Calculation of regression coefficients (**Gradient Descent**)

- For n pairs of data:

$$y_i = b_0 + b_1 x_i + b_2 x_i^2 + b_3 x_i^3$$

⬇

$$b = (b_0, b_1, b_2, b_3)^T \quad x = (x_i, x_i, x_i, x_i)^T$$

⬇

$$y_b(x) = b^T x$$

- Evaluation Function:

$$\min mize \quad J(b) = \frac{1}{2n}\sum_{i=1}^{n}(y_i' - y_i)^2 = \frac{1}{2n}\sum_{i=1}^{n}(y_i' - b^T x_i)^2$$

⬇

$$\frac{\partial J}{\partial b} = \frac{1}{n}\sum_{i=1}^{n}(y_i' - b^T x_i)x_i$$

⬇ As the Gradient Descends, Iterate Continuously.

$$b = b - \alpha \frac{\partial J}{\partial b}$$

x_i, y_i: the variables
y_i': the true value
$b_{1,\dots,}b_k$: coefficients

Examples and Python code

$$y = b_0 + b_1 x + b_2 x^2 + \ldots + b_k x^k + \varepsilon$$

X	y
0.2	3
0.42	6
0.5	8
0.7	8.2
0.9	7.3
1.1	6
1.25	4.5
1.4	4.2
1.58	3.3
1.76	5
1.93	7.5
2.11	9

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# Read the data
data = pd.read_csv('E:\TEST/data_m.csv')  # Please replace 'your_data_file.csv'
with your data file path
X = data['x'].values.reshape(-1, 1)
y = data['y'].values.reshape(-1, 1)
# Define the gradient descent function
def gradient_descent(X, y, degree=3, learning_rate=0.01, n_iterations=80000):
    m = len(X)
    theta = np.random.randn(degree + 1, 1)  # Initial coefficients
    X_b = np.c_[np.ones((m, 1))]
    for d in range(1, degree + 1):
        X_b = np.c_[X_b, X**d]  # Add polynomial features
    for iteration in range(n_iterations):
        gradients = 2/m * X_b.T.dot(X_b.dot(theta) - y)
        theta = theta - learning_rate * gradients
    return theta
```

• Examples and Python code

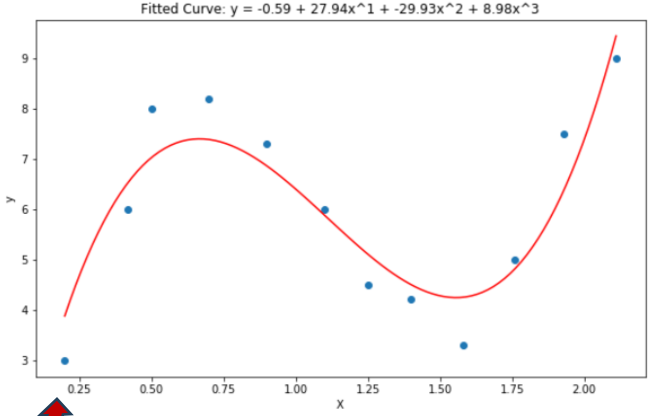
$$y = b_0 + b_1 x + b_2 x^2 + \ldots + b_k x^k + \varepsilon$$

```
# Run the gradient descent algorithm
theta = gradient_descent(X, y, degree=3)
# Display the fitted equation
equation = 'y = {:.2f}'.format(theta[0][0])
for i in range(1, len(theta)):
    equation += ' + {:.2f}x^{}'.format(theta[i][0], i)
print('Fitted equation:', equation)
# Plot the fitted curve
X_new = np.linspace(min(X), max(X), 100).reshape(-1, 1)
X_new_b = np.c_[np.ones((100, 1))]
for d in range(1, 4):
    X_new_b = np.c_[X_new_b, X_new**d]
y_predict = X_new_b.dot(theta)
```

```
plt.figure(figsize=(10, 6))
plt.scatter(X, y)
plt.plot(X_new, y_predict, 'r-')
plt.xlabel('X')
plt.ylabel('y')
plt.title('Fitted Curve: ' + equation)
plt.show()
```

Results:

```
Fitted equation: y = -0.59 + 27.94x^1 + -29.93x^2 + 8.98x^3
```



• What is the Classification

➢ Email classification

- Task: Input email
- Output: Spam/Normal email?

Sender: d%sds@123.com
Kindly **claim the 20,000 bonus**.

●●●

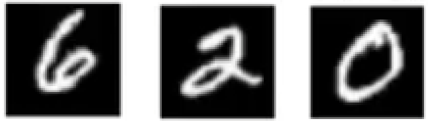
Sender: Allen@gmail.com
Hello, we have a meeting at 2 pm.

➢ Handwritten digit recognition

- Task: Input New handwritten digits
- Output: Predict label

Lable=5 Lable=3 Lable=9

Lable=6 Lable=2 Lable=0



⇩ The computer learns features from a large number of samples to make judgments ⇩

Sender: 8hn%s888@888.com
Hello, Double click to receive gift package.

➡ Normal email?

➡ Lable = ?

Image Source: Google

• What is the Classification

➢ Classification

Using specific features from known samples, ascertain the category to which a new sample belongs

Framework
$$y = f(x_1, x_2 \dots x_n)$$
Identify as category N, $if \;\; y = n$

✓ **Email classification**

Sender contains characters: **%&. . .**

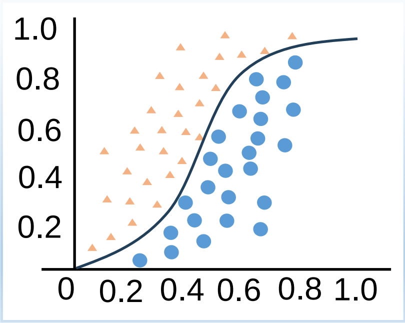
Text contains: **cash, collection, etc.**

Other features

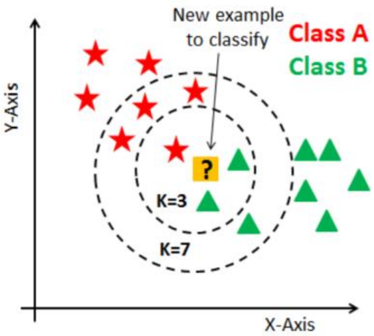
$\Rightarrow$ $y = 0$ $\Rightarrow$ ✓ Spam email

• Classification Method

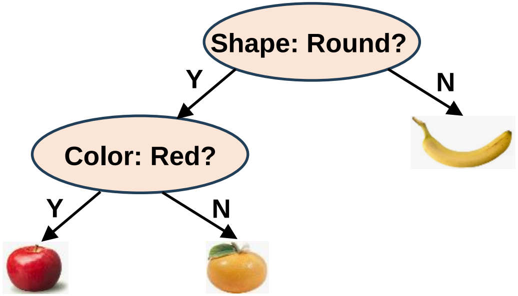
✓ **Logistic regression**



✓ **K-Nearest Neighbor (KNN)**



✓ **Decision Tree**



✓ **Neural Networks**

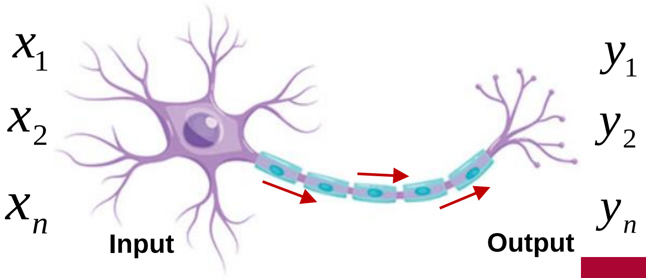
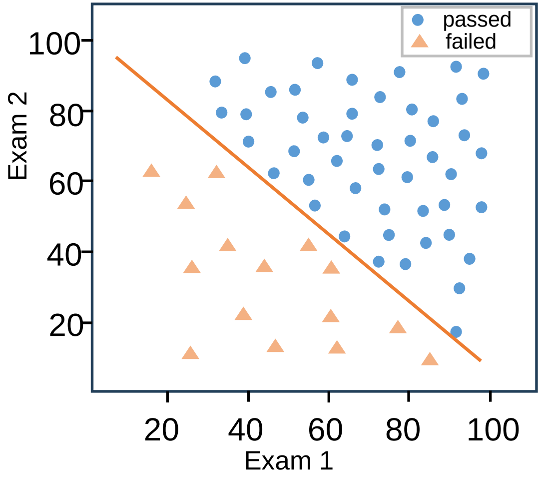


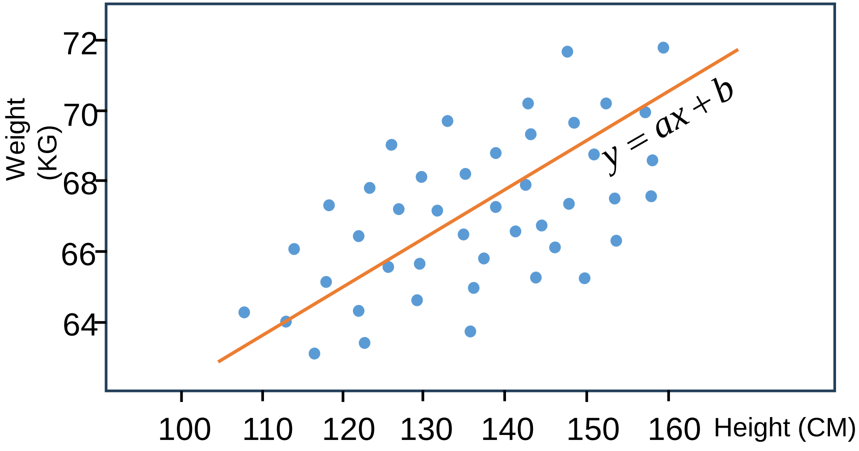
Image Source: Google

• Classification vs. Regression Tasks

✓ Classification



✓ Regression



Classification target: identify the category

Model output: non-continuous label

Regression target: establish a functional relationship

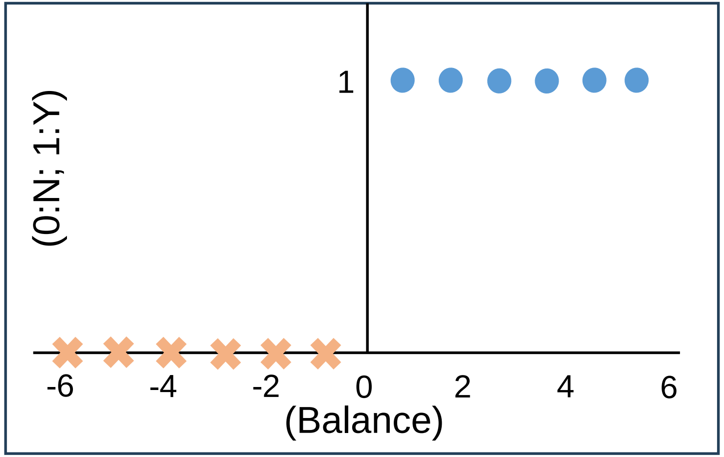
Model output: continuous value

• Solving Classification task

Task: Identify whether Xiao Ming will watch a movie based on his balance?

Balance: 1, 2, 3, 4, 5
Watch a movie (positive sample)
Lable: 1

Balance: -1, -2, -3, -4, -5
Do not watch a movie (negative sample)
Lable: 0



⇩

Framework

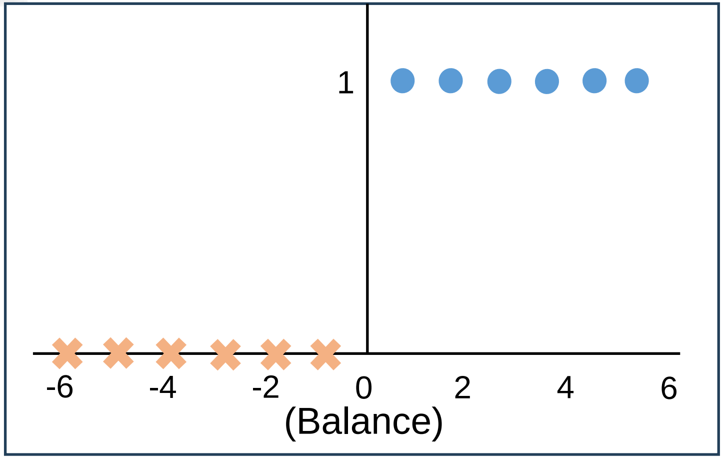
$$y = f(x_1, x_2 \ldots x_n)$$

Identify as category N, $y = n$

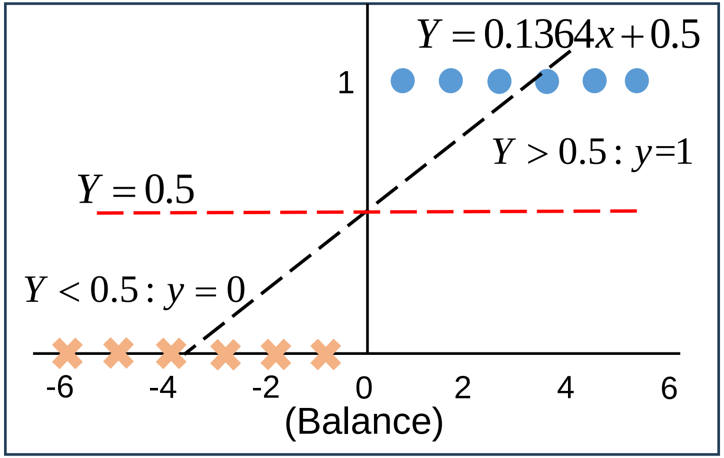
⇨ $y = f(x) \in \{0,1\}$ ⇨ $f(x)$

- **Solving Classification task**

How to get the f(x)



Linear Regression

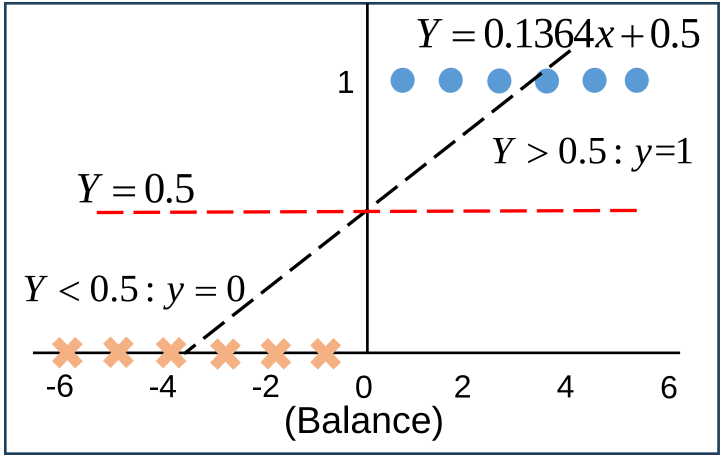


$$(1)Y = 0.1364x + 0.5 \qquad (2)\, y = f(x) = \begin{cases} 1, & Y >= 0.5 \\ 0, & Y < 0.5 \end{cases}$$

• Solving Classification task

How to get the f(x)

$$(1) Y = 0.1364x + 0.5 \qquad (2) y = f(x) = \begin{cases} 1, & Y >= 0.5 \\ 0, & Y < 0.5 \end{cases}$$



$$Y = 0.1364 \times \boxed{(-5)} + 0.5 = -0.182 < 0.5$$

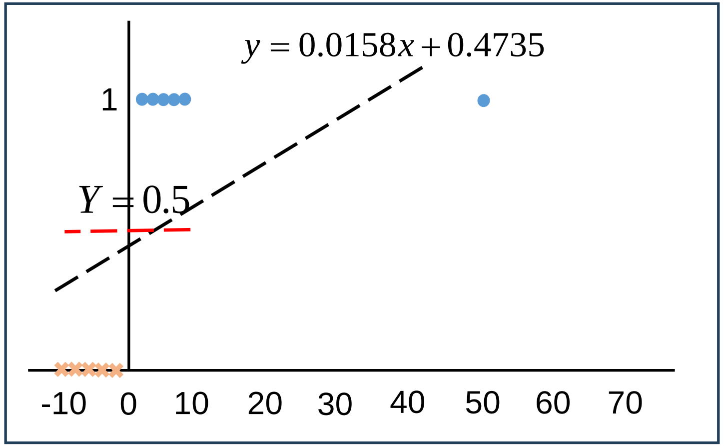
$$\boxed{y(x) = 0}$$

X	Y(X)	y(x)	Y(truth)
-5	-0.18	0	0
-4	-0.05	0	0
-3	0.09	0	0
-2	0.23	0	0
-1	0.36	0	0
1	0.64	1	1
2	0.77	1	1
3	0.91	1	1
4	1.05	1	1
5	1.18	1	1

Linear Regression works well

• Solving Classification task

Drawbacks: **The accuracy decreases as the sample size increases**



X	Y(X)	y(x)	Y(truth)
-5	0.39	0	0
-4	0.41	0	0
-3	0.43	0	0
-2	0.44	0	0
-1	0.46	0	0
1	0.49	$\boxed{0}$	$\boxed{1}$
2	0.51	1	1
3	0.52	1	1
4	0.54	1	1
5	0.55	1	1
50	1.26	1	1

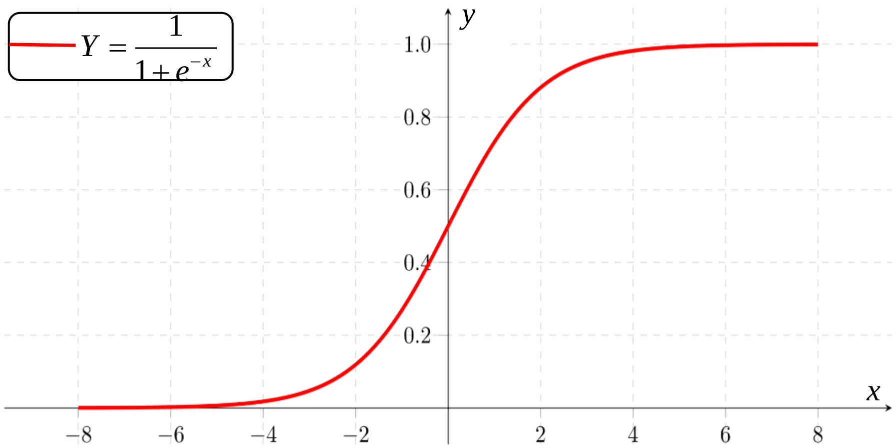
Classification Error

$$y = 0.0158 \times \boxed{1} + 0.4735 = 0.49 < 0.5 \Rightarrow \boxed{y(x) = 0}$$

As X moves further away from the origin, the accuracy of the prediction diminishes.

• Logistic regression

$$(1)\, Y = \frac{1}{1+e^{-x}} \qquad (2)\, y = f(x) = \begin{cases} 1, & Y >= 0.5 \\ 0, & Y < 0.5 \end{cases}$$



X	Y(X)	y(x)	Y(truth)
-5	0.01	0	0
-4	0.02	0	0
-3	0.05	0	0
-2	0.12	0	0
-1	0.27	0	0
1	0.73	0	1
2	0.88	1	1
3	0.95	1	1
4	0.98	1	1
5	0.99	1	1
50	1.00	1	1
1000	1.00	1	1

Using logistic regression to model the data can enhance the effectiveness of the classification task.

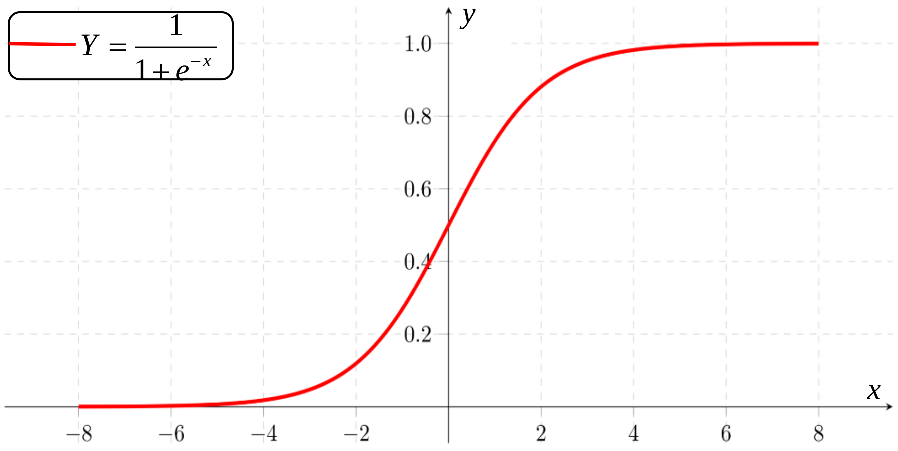
• What is Logistic regression

- ✓ A model for **solving classification problems**.
- ✓ By analyzing the data's characteristics, **the probability of belonging to a specific category is computed to classify it** accordingly.
- ✓ Mainly used in binary classification.

Sigmoid Function:

$$P(x) = \frac{1}{1+e^{-x}}$$

$$y = \begin{cases} 1, & P(x) >= 0.5 \\ 0, & P(x) < 0.5 \end{cases}$$



• Solving Classification task

Task: Identify whether Xiao Ming will watch a movie based on his balance? (Balance: -10; 100)

$$P(x)=\frac{1}{1+e^{-x}}$$

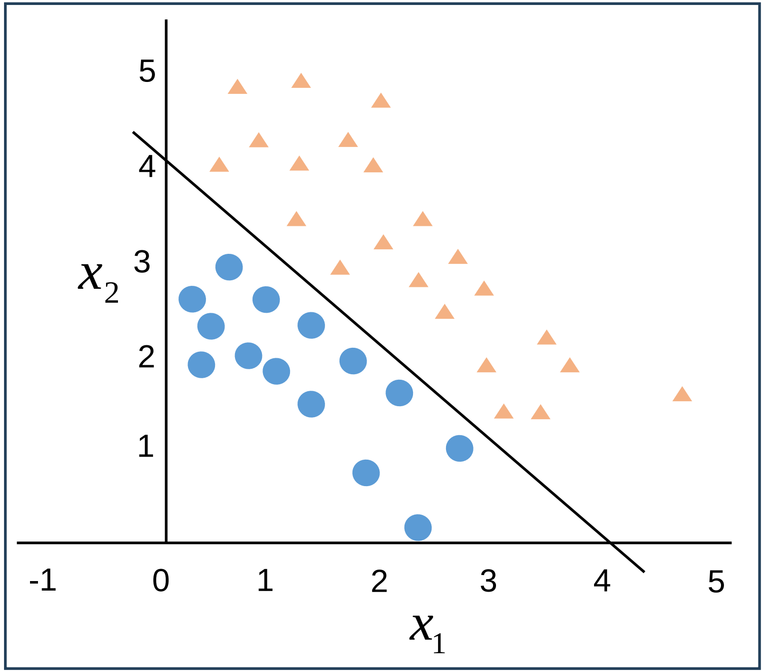
$$y=\begin{cases}1, & P(x)>=0.5\\ 0, & P(x)<0.5\end{cases}$$

$$P(x=-10)=\frac{1}{1+e^{10}}=4.5\times10^{-5}<0.5$$

$$P(x=100)=\frac{1}{1+e^{-100}}=1>0.5$$

Balance:
(-10: Do not watch a movie)
(100: Watch a movie)

• Complex classification problem



Decision Boundary: $-4+\boxed{x_1+x_2}=0$

$$P(x)=\frac{1}{1+e^{-x}}$$

$$P(x)=\frac{1}{1+e^{-g(x)}}$$

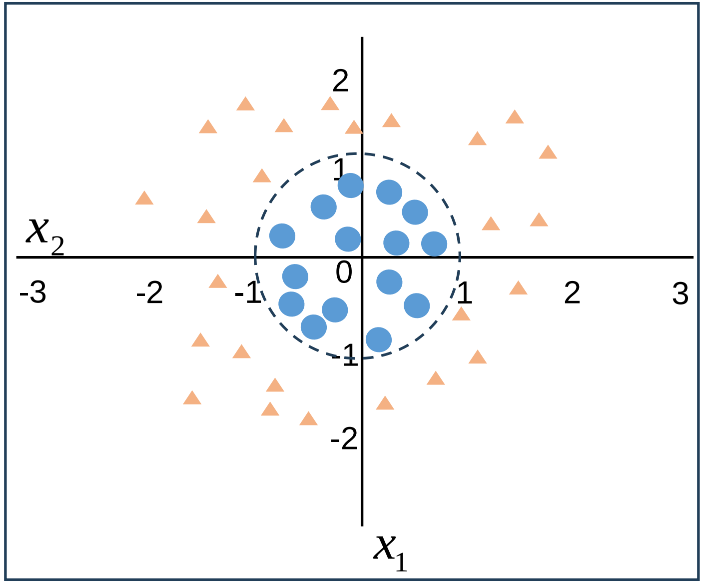
$$g(x)=\boxed{\theta_0+\theta_1 x_1+\theta_2 x_2}$$

$$g(x)=-4+x_1+x_2$$

$g(x)=-4+x_1+x_2>0:$ Triangle

$g(x)=-4+x_1+x_2<0:$ Circle

- **Complex classification problem**



$$P(x) = \frac{1}{1+e^{-g(x)}}$$

$$g(x) = \boxed{\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2}$$

$$g(x) = -1 + x_1^2 + x_2^2$$

$g(x) = -1 + x_1^2 + x_2^2 > 0$: Triangle

$g(x) = -1 + x_1^2 + x_2^2 < 0$: Circle

Decision Boundary: $-1 + \boxed{x_1^2 + x_2^2} = 0$

Logistic regression combined with polynomial boundary functions can solve complex classification problems

Logistic regression solution

- Find the category boundaries based on the training samples

$$P(x) = \frac{1}{1+e^{-g(x)}}$$

$$g(x) = \theta_0 + \theta_1 x_1 + \ldots$$

✓ Find the $\theta_0, \theta_1, \theta_2$

- Solve linear regression and minimize the loss function (J)

$$J = \frac{1}{2m}\sum_{i=1}^{m}(y_i^{'} - y_i)^2$$

Drawbacks: In classification problems, where labels and prediction results are discrete, identifying the exact minimum using this loss function is not achievable.

• Logistic regression solution

- Solve the logistic regression and minimize the loss function (J)

$$J_i = \begin{cases} -\log(P(x_i)) & if \;\; y_i = 1 \\ -\log(1-P(x_i)) & if \;\; y_i = 0 \end{cases}$$

$$\Updownarrow$$

$$J = \frac{1}{m}\sum_{i=1}^{m} J_i = -\frac{1}{m}\left[\sum_{i=1}^{m}(y_i \log(\boxed{P(x_i)}) + (1-y_i)\log(1-P(x_i)))\right]$$

$$P(x) = \frac{1}{1+e^{\boxed{-g(x)}}}$$

$$g(x) = \boxed{\theta_0 + \theta_1 x_1} + \ldots$$

$$\Rightarrow \quad \min(J(\theta))$$

• Logistic regression solution

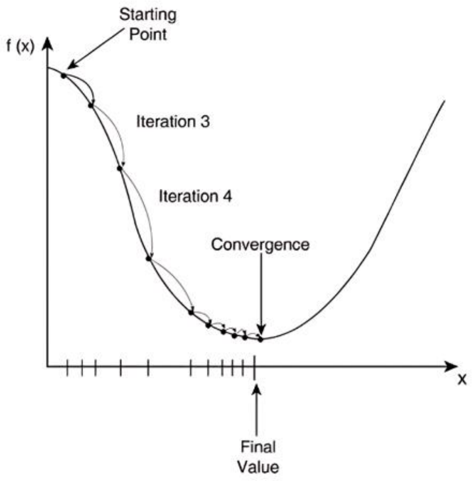
✓ Optimization: Gradient Descent

Calculate the gradient of the loss function for each parameter and find the minimum value.

$$J = f(p)$$

Search methods

$$p_{i+1} = p_i - \alpha \frac{\partial}{\partial p_i} f(p_i)$$



✓ Continue this iterative process until convergence is achieved

$$\begin{cases} temp_{\theta_j} = \theta_j - \alpha \dfrac{\partial}{\partial \theta_j} J(\theta) \\ \theta_j = temp_{\theta_j} \end{cases}$$

• Examples and Python code

When Exam 1 scores 70 and Exam 2 scores 65, predict the likelihood of passing other exams

Exam1	Exam2	Pass
80	75	1
85	90	1
60	55	0
40	30	0
70	65	1
…	…	…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
# Load data, 2 exams and labels（1: pass，0:failed）
data = {
    'Exam1': [80, 85, 60, 40, 70],
    'Exam2': [75, 90, 55, 30, 65],
    'Pass': [1, 1, 0, 0, 1]
}
df = pd.DataFrame(data)
# Prepare features and target variable
X = df[['Exam1', 'Exam2']]
y = df['Pass']
# Create and fit a logistic regression model
log_reg = LogisticRegression()
log_reg.fit(X, y)
```

```
# Make predictions on the entire dataset
predictions = log_reg.predict(X)
# Output the predictions
print("Predictions:", predictions)
# Output model accuracy
accuracy = log_reg.score(X, y)
print("accuracy:", accuracy)
# Plot the decision boundary
x_min, x_max = X.iloc[:, 0].min() - 1, X.iloc[:, 0].max() + 1
y_min, y_max = X.iloc[:, 1].min() - 1, X.iloc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max,
0.1),np.arange(y_min, y_max, 0.1))
Z = log_reg.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, alpha=0.4)
plt.scatter(X['Exam1'], X['Exam2'], c=y, edgecolor='k', s=20)
plt.xlabel('Exam 1 score')
plt.ylabel('Exam 2 score')
```

- **Examples and Python code (please generate the code by yourself)**

```
plt.title('Logistic Regression Decision Boundary')
plt.show()

# Output the model equation
coef = log_reg.coef_[0]
intercept = log_reg.intercept_[0]
print(f"equation: Pass = 1 / (1 + e^(-({coef[0]}*Exam1 +
{coef[1]}*Exam2 + {intercept})))")
```

Results

```
Predictions: [1 1 1 0 1]
accuracy: 0.8
```


```
equation: Pass = 1 / (1 + e^(-(-0.4417113928415653*Exam1 + 0.49128528282026507*Exam2 + -0.1395866043578856)))
```

```
# New data for prediction
y_test = log_reg.predict([[70,65]])
print("passed" if y_test==1 else "failed")
```

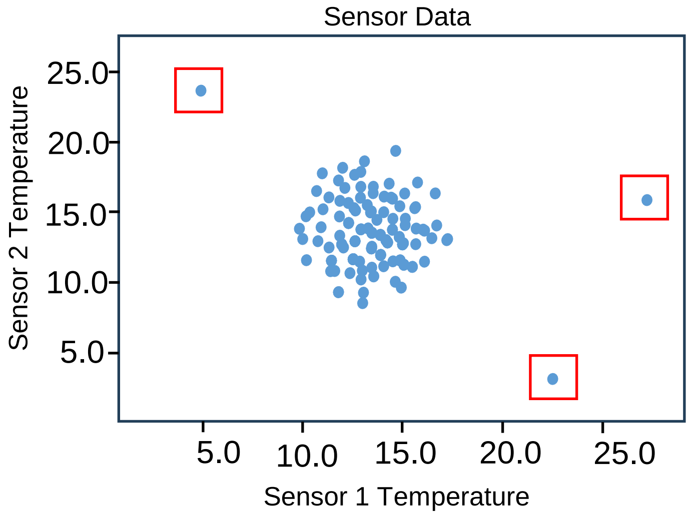
```
y_test = log_reg.predict([[70,65]])
print("passed" if y_test==1 else "failed")
```

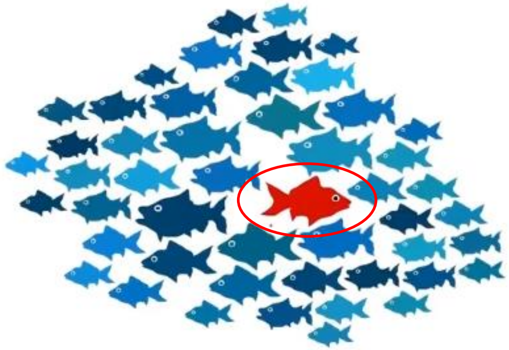
```
passed
```

Anomaly Detection

Task: Automatically monitor abnormal working status of the device based on the data of sensors 1 and 2 on the device

Task: Automatically find abnormal objects in the image



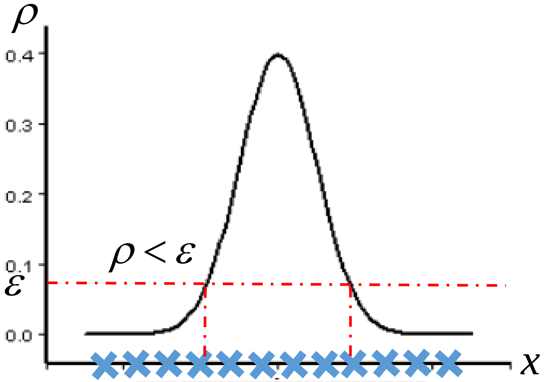


Identify data that does not conform to the expected pattern based on the input data

• Anomaly Detection

➢ One-dimensional dataset

$$\left\{x^{(1)}, x^{(2)}, \ldots x^{(m)}\right\}$$

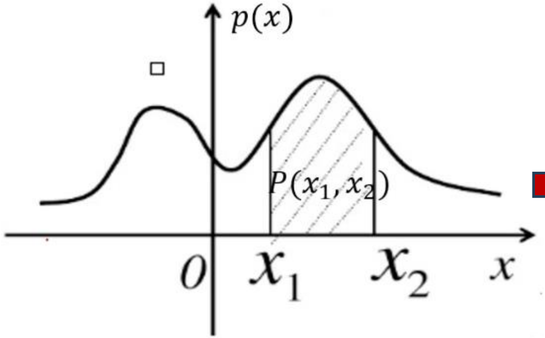


Data distribution

Find low-probability data (events)

➢ Probability density

Probability density is a function that describes the probability of a random variable near a certain value point.



X distribution probability density

The probability of the interval (X1, X2) is:

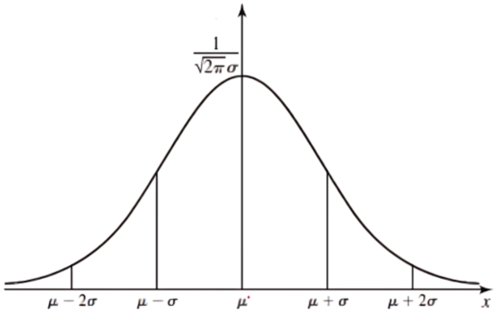
$$P(x_1, x_2) = \int_{x_1}^{x_2} P(x)dx$$

• Anomaly Detection

➢ Gaussian distribution

Probability density function of Gaussian distribution:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



μ : Data Mean σ : Standard deviation

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)},$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - \mu)^2$$

Example:

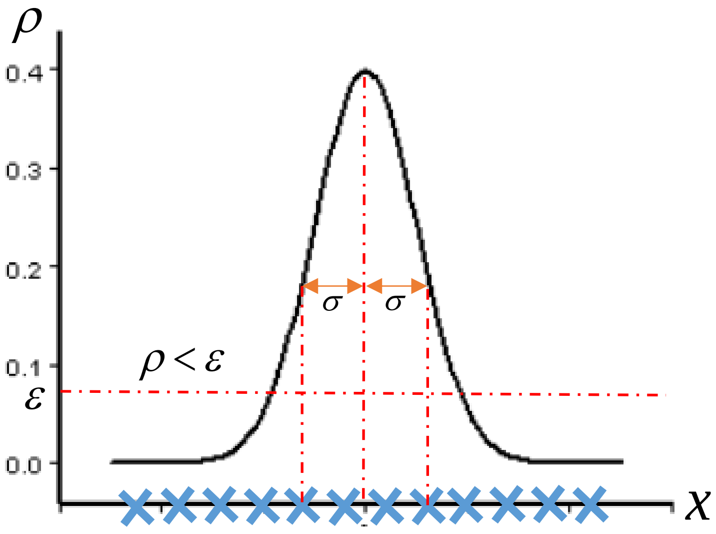
X_1	X_2	X_3	X_4
-1	0	1	2

$$\mu = \frac{1}{4}(-1+0+1+2) = 0.5$$

$$\sigma^2 = \frac{1}{4}[(-1-0.5)^2 + (0-0.5)^2 + (1-0.5)^2 + (2-0.5)^2] = 1.2$$

• Anomaly Detection

➢ Anomaly detection based on Gaussian distribution



Data distribution

- ✓ Calculate the data mean (μ) and standard deviation (σ) ;
- ✓ Calculate the corresponding Gaussian distribution probability density function:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- ✓ Make a judgment based on the probability of the data point

if $P(x) < \varepsilon$ ➡ This point is an abnormal point

• Anomaly Detection

➢ Anomaly detection based on Gaussian distribution

☐ High-dimensional data

$$\begin{Bmatrix} x_1^{(1)}, x_1^{(2)}, \ldots x_1^{(m)} \\ x_2^{(1)}, x_2^{(2)}, \ldots x_2^{(m)} \\ \ldots \\ x_n^{(1)}, x_n^{(2)}, \ldots x_n^{(m)} \end{Bmatrix}$$

- Calculate the data mean ($\mu_1, \mu_2, \ldots, \mu_n$) and standard deviation ($\sigma_1, \sigma_2, \ldots, \sigma_n$) ;

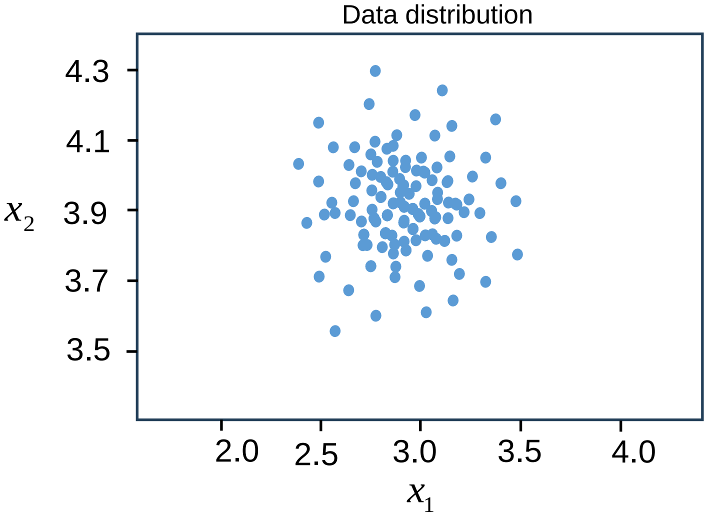
$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}, \qquad \sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m} (x_j^{(i)} - \mu_j)^2$$

- Calculate the probability density function:

$$P(x) = \prod_{j=1}^{n} P(x_j, \mu_j, \sigma_j^2) = \prod_{j=1}^{n} \frac{1}{\sigma_j\sqrt{2\pi}} e^{-\frac{(x_j-\mu_j)^2}{2\sigma_j^2}}$$

Anomaly Detection

Anomaly detection based on Gaussian distribution



- Calculate the probability density function:

$$P(x) = \prod_{j=1}^{n} P(x_j, \mu_j, \sigma_j^{\ 2}) = \prod_{j=1}^{n} \frac{1}{\sigma_j \sqrt{2\pi}} e^{-\frac{(x_j - \mu_j)^2}{2\sigma_j^{\ 2}}}$$

$$P(x_1) = \frac{1}{0.5\sqrt{2\pi}} e^{[-\frac{(3.5-3)^2}{2\times 0.5^2}]} = 0.4839$$

$$P(x_2) = \frac{1}{0.14\sqrt{2\pi}} e^{[-\frac{(3.5-4)^2}{2\times 0.14^2}]} = 0.0048$$

(3.5,3.5) : Anomaly?

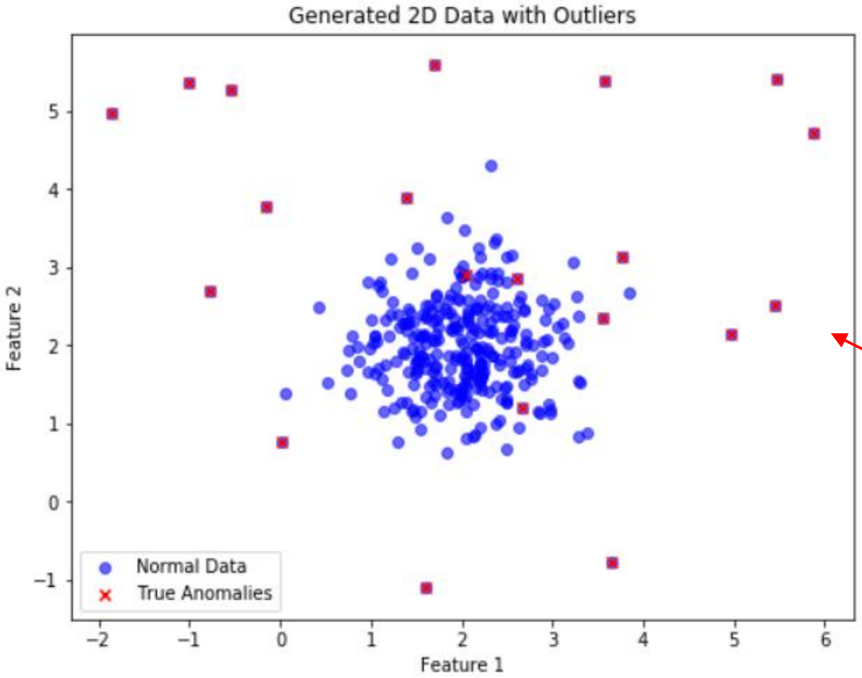
$$\mu_1 = 3, \sigma_1 = 0.5$$
$$\mu_2 = 4, \sigma_2 = 0.14$$
$$\varepsilon = 0.05, x_1 = x_2 = 3.5$$

$$P(x_1, x_2) = P(x_1) \times P(x_2) = 0.0023 < 0.05 = \varepsilon$$

Abnormal point!

• Examples and Python code

" **Anomaly detection** from 300 samples using Gaussian distribution"

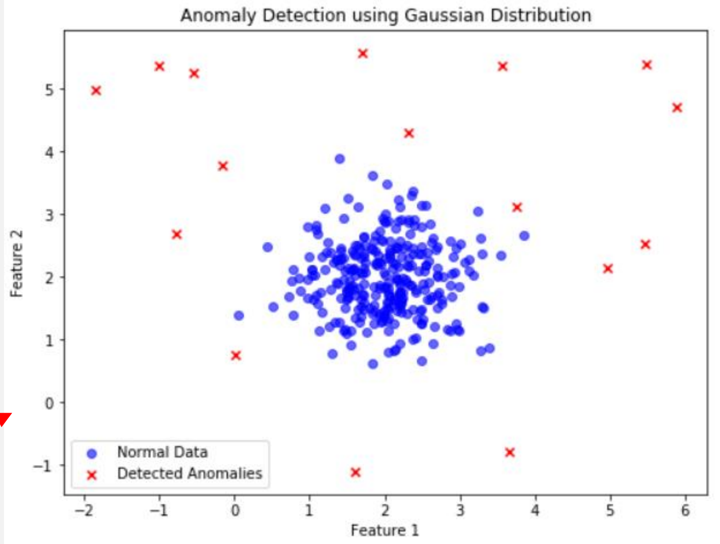



```
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
# Generate 2D data
np.random.seed(42)
n_samples = 300
# Generate normal data (Single-cluster 2D Gaussian distribution)
X_normal = 0.6 * np.random.randn(n_samples, 2) + [2, 2]
# Generate outliers
n_outliers = 20
X_outliers = np.random.uniform(low=-2, high=6, size=(n_outliers, 2))
X = np.vstack((X_normal, X_outliers))
# Visualize original data (Mark true outliers)
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c='blue', label='Normal Data', alpha=0.6)
plt.scatter(X_outliers[:, 0], X_outliers[:, 1], c='red', label='True Anomalies', marker='x')
plt.legend()
plt.title("Generated 2D Data with Outliers")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```

- # Examples and Python code

```
# Compute Gaussian distribution parameters
mu = np.mean(X_normal, axis=0)  # Mean
sigma = np.cov(X_normal.T)  # Covariance matrix
# Compute probability density for each point
rv = multivariate_normal(mean=mu, cov=sigma)
p_values = rv.pdf(X)
# Set threshold for anomalies (Lowest 5% probability as anomalies)
threshold = np.percentile(p_values, 5)  # Compute 5th percentile as threshold
y_pred = p_values < threshold  # Mark anomalies (True means anomaly)

# Visualize anomaly detection results (Mark detected anomalies)
plt.figure(figsize=(8, 6))
plt.scatter(X[~y_pred, 0], X[~y_pred, 1], c='blue', label='Normal Data', alpha=0.6)
plt.scatter(X[y_pred, 0], X[y_pred, 1], c='red', label='Detected Anomalies', marker='x')
plt.legend()
plt.title("Anomaly Detection using Gaussian Distribution")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```

Results



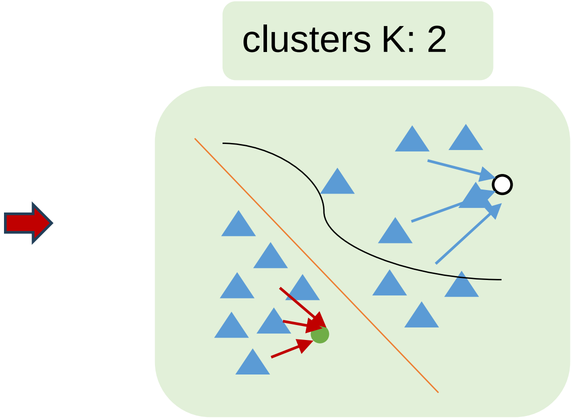
K-Means Analysis

- ✓ K-Means is an unsupervised learning algorithm mainly used for clustering tasks.
- ✓ Its goal is to partition a dataset into K clusters, ensuring that data points within the same cluster are similar, while those in different clusters are as distinct as possible.

Formula:

- The distance between the data and the center of each cluster: $dist(x_i, u_j^t)$
- Classification based on distance: $x_i \in u_{nearest}^t$
- Update the center point: $u_j^{t+1} = \frac{1}{k} \sum_{x_i \in s_j} (x_i)$

s_j: the j^th regional cluster at time t
x_i: Number of points included in the s_j range
y: Points included in the s_j range
u_j^t: Center of j^th region at state t



• K-Means Analysis

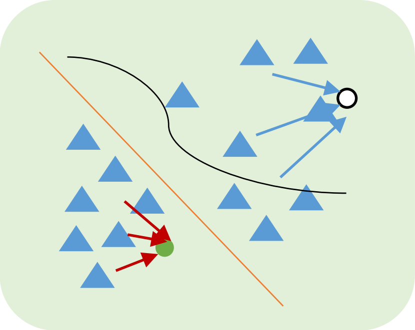
➢ Algorithm Steps

- Select the number of clusters K
- Determine the cluster center
- Determine the category of each point based on the distance from the point to the cluster center
- Update the cluster center based on the data of each category
- Repeat the above steps until convergence (the center point no longer changes)

➢ Advantages

- Simple principle, easy implementation, fast convergence speed
- Few parameters, convenient and practical

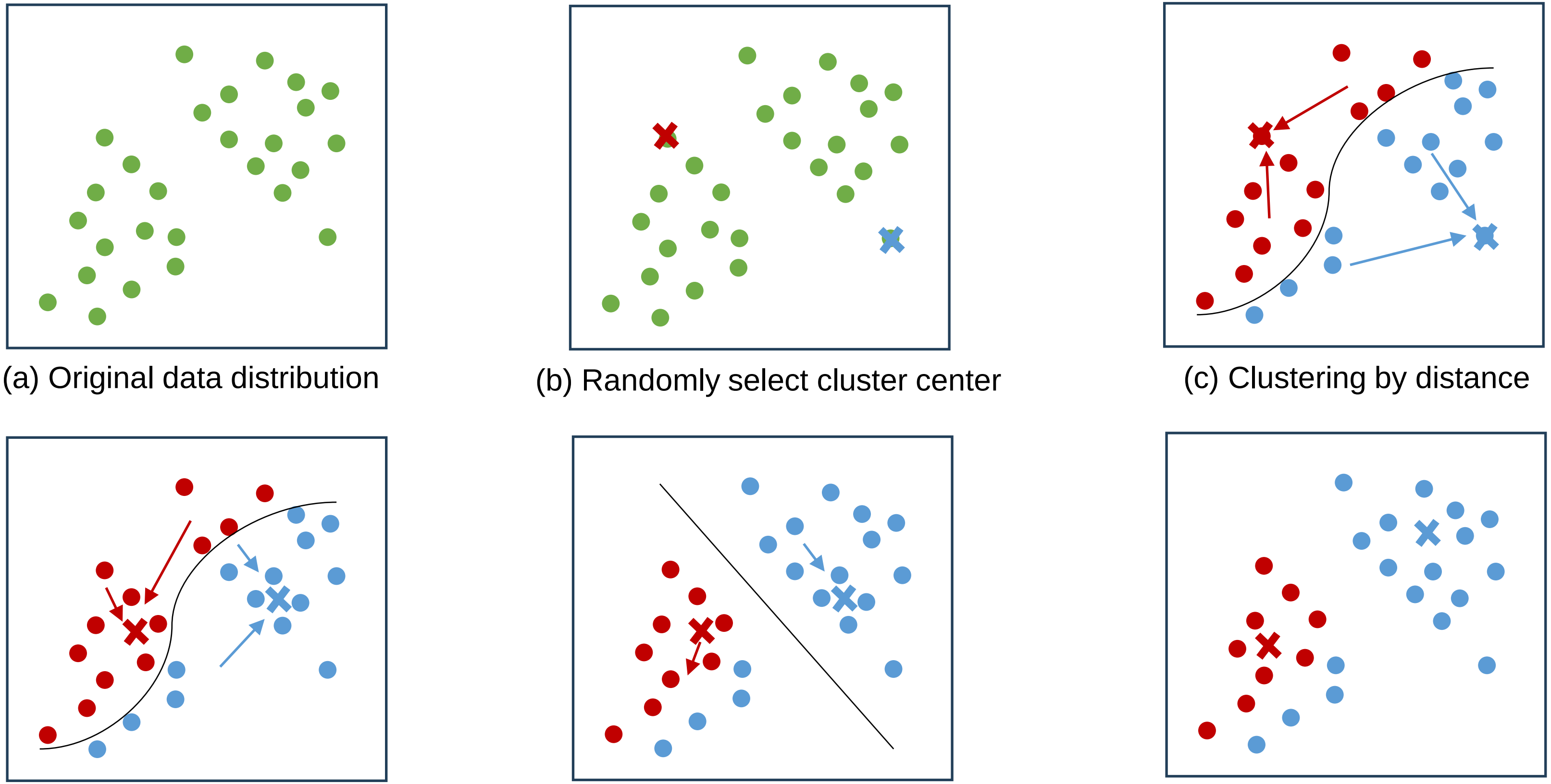
clusters K: 2



➢ Disadvantages

- The number of clusters must be set
- Randomly select the initial cluster center, the result may lack consistency

- **K-Means Analysis**



(a) Original data distribution

(b) Randomly select cluster center

(c) Clustering by distance

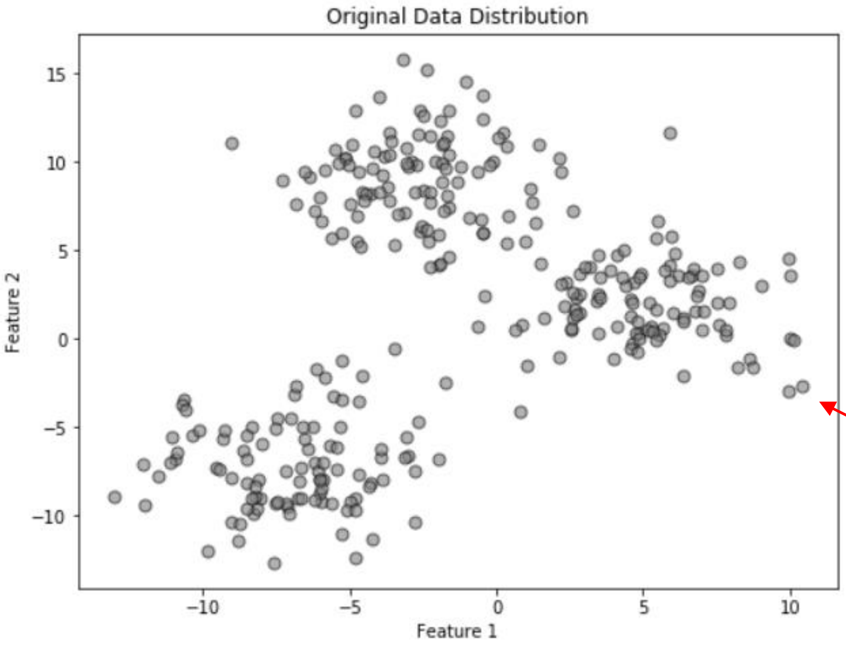
(d) Update center based on clustering

(e) Update cluster based on new distance

(f) Center no longer changes

• Examples and Python code

"Cluster analysis of 300 data using **K-means** method"

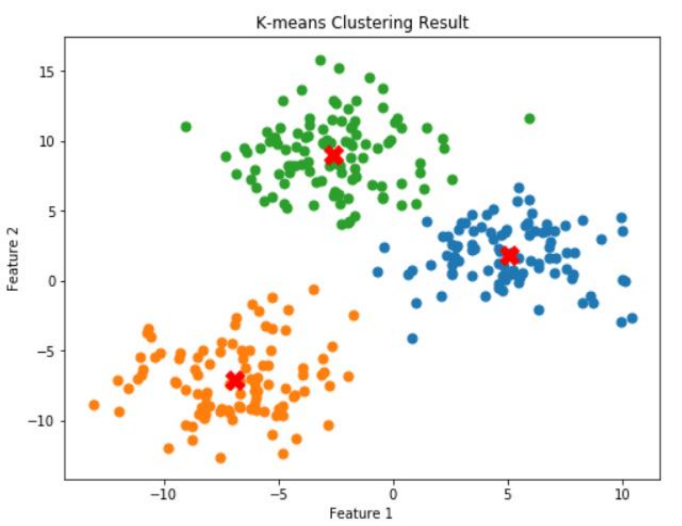



```
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
# Generate synthetic dataset
n_samples = 300  # Number of data points
n_features = 2   # Number of features (2D)
n_clusters = 3   # Number of clusters
random_state = 42 # Random seed for reproducibility
cluster_std = 2.5 # Reduce standard deviation to make clusters denser
# Create data points
X, y = make_blobs(n_samples=n_samples, n_features=n_features,
centers=n_clusters,cluster_std=cluster_std,random_state=random_state)

# Visualize the original dataset
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], s=50, c='gray', marker='o', edgecolors='k', alpha=0.6)
plt.title("Original Data Distribution")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```

Examples and Python code

```
# Apply K-Means clustering
kmeans = KMeans(n_clusters=n_clusters, random_state=random_state)
y_pred = kmeans.fit_predict(X)

# Visualize clustering results
plt.figure(figsize=(8, 6))
for i in range(n_clusters):
    plt.scatter(X[y_pred == i, 0], X[y_pred == i, 1], s=50, label=f'簇 {i} / Cluster {i}')

# Plot cluster centroids
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=200,
c='red', marker='X', label='质心 / Centroids')

plt.title("K-means Clustering Result")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
```

Results



K-Nearest Neighbors (KNN)

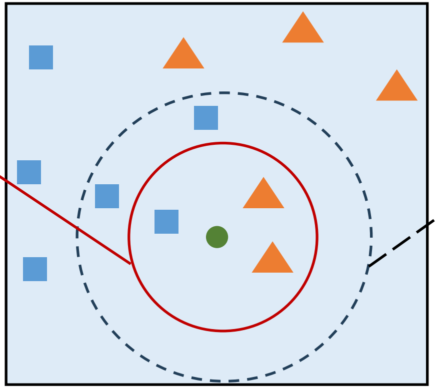
- ✓ KNN is a supervised learning algorithm used for classification and regression tasks.
- ✓ The core idea is that a data point's category is determined by the majority class of its K nearest neighbors.

Example:

K = 3:

The three nearest neighbors of the green dot are two small red triangles and one blue square

The green point category: ▲



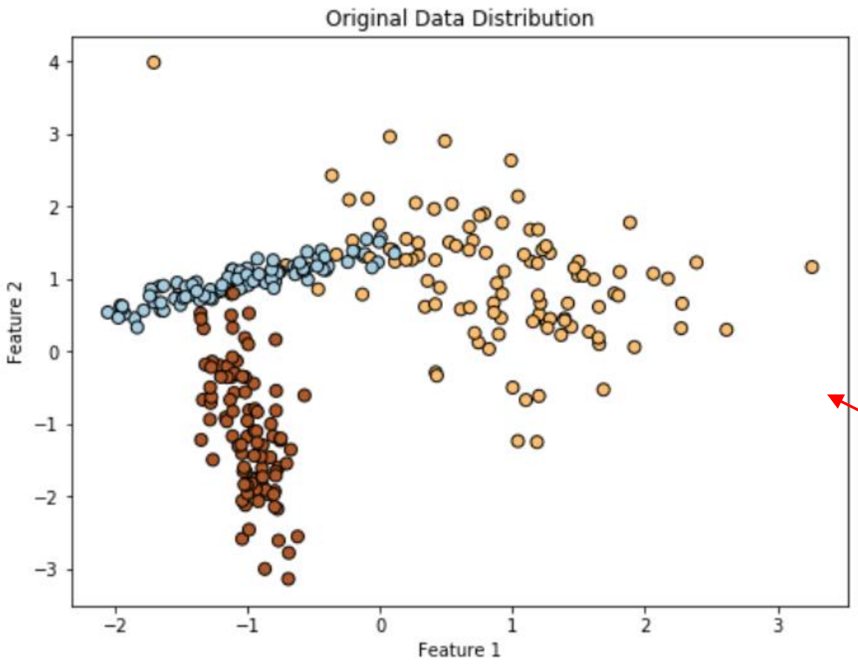
K = 5:

The three nearest neighbors of the green dot are two small red triangles and three blue square

The green point category: ■

- # Examples and Python code

"Cluster analysis of 300 data using **K-Nearest Neighbors** method"




```
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from matplotlib.colors import ListedColormap
# Generate 2D classification dataset
n_samples = 300  # Number of samples
n_features = 2   # Number of features (2D)
n_classes = 3    # Number of classes
random_state = 42  # Random seed
# Generate classification dataset
X, y = make_classification(n_samples=n_samples, n_features=n_features,
n_classes=n_classes, n_clusters_per_class=1, n_redundant=0,
random_state=random_state)
# Visualize the original dataset
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired, edgecolors='k', s=50)
plt.title("Original Data Distribution")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```

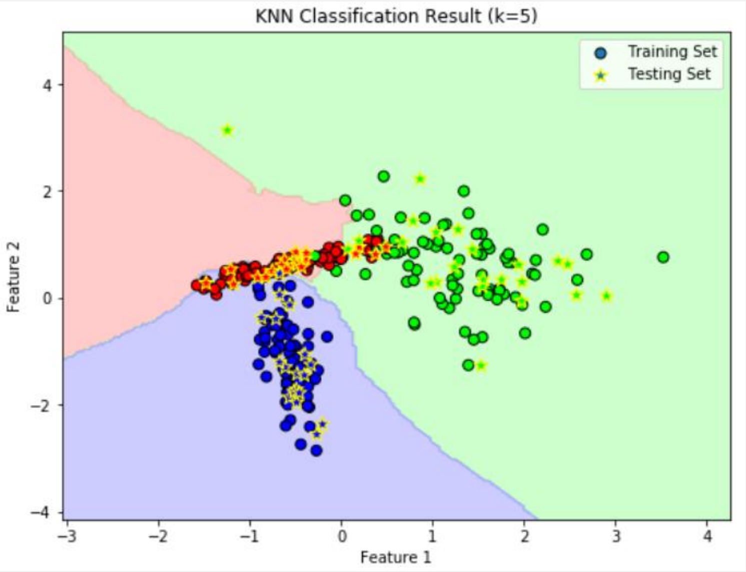
• Examples and Python code

```
# Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=random_state)
# Standardize data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Train KNN classifier
k = 5  # Choose k value
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)
# Predict test set
y_pred = knn.predict(X_test)
# Plot decision boundary
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 200), np.linspace(y_min, y_max, 200))
# Predict classification labels for the grid area
Z = knn.predict(scaler.transform(np.c_[xx.ravel(), yy.ravel()]))
Z = Z.reshape(xx.shape)
```

• Examples and Python code

```
# Visualize classification result
plt.figure(figsize=(8, 6))
cmap_light = ListedColormap(["#FFAAAA", "#AAFFAA", "#AAAAFF"])
cmap_bold = ListedColormap(["#FF0000", "#00FF00", "#0000FF"])

# Plot decision regions
plt.contourf(xx, yy, Z, cmap=cmap_light, alpha=0.6)
# Plot training data points
scatter_train = plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cmap_bold,
edgecolors='k', s=50, label="Training Set")
# Plot testing data points
scatter_test = plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cmap_bold,
edgecolors='yellow', s=100, marker="*", label="Testing Set")
plt.title("KNN Classification Result (k=5)")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.legend(handles=[scatter_train, scatter_test])
plt.show()
```

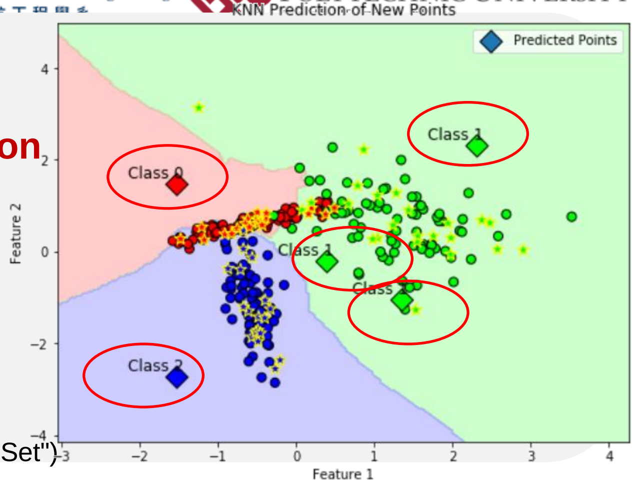
Results



• Examples and Python code

```
# Generate new points for prediction
new_points = np.array([[0, 0], [1, -1],[-2, -3], [-2, 2], [2, 3]])  # New data points
new_points_scaled = scaler.transform(new_points)  # Apply standardization
# Predict the class of new points
new_predictions = knn.predict(new_points_scaled)
# Visualize new predictions
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, cmap=cmap_light, alpha=0.6)  # Plot classification boundary
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cmap_bold, edgecolors='k', s=50, label="Training Set")
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cmap_bold, edgecolors='yellow', s=100, marker="*", label="Testing Set")
# Plot new points and annotate their classes
scatter_new = plt.scatter(new_points_scaled[:, 0], new_points_scaled[:, 1], c=new_predictions, cmap=cmap_bold, edgecolors='black', marker="D", s=150, label="Predicted Points")
# Annotate new points
for i, txt in enumerate(new_predictions):
    plt.annotate(f"Class {txt}", (new_points_scaled[i, 0], new_points_scaled[i, 1]), textcoords="offset points", xytext=(5, 5), ha='right', fontsize=12, color='black')
plt.title("KNN Prediction of New Points")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.legend(handles=[scatter_new])
plt.show()
```

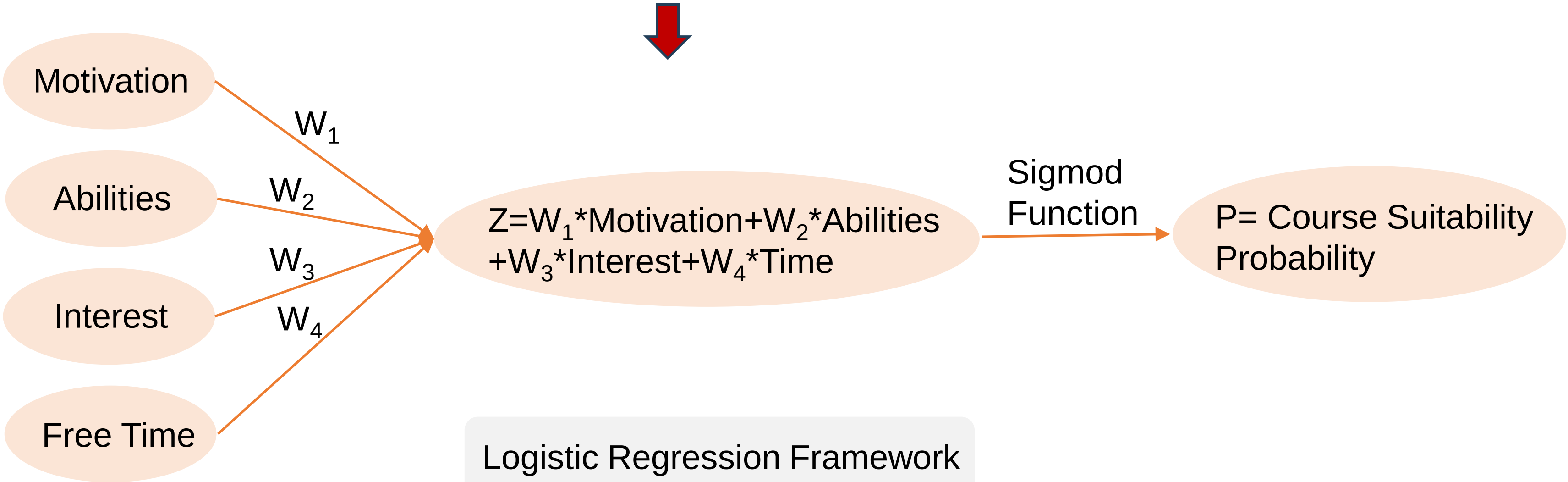
Results: New points for prediction



• Decision Tree

➢ Logistic Regression VS Decision Tree

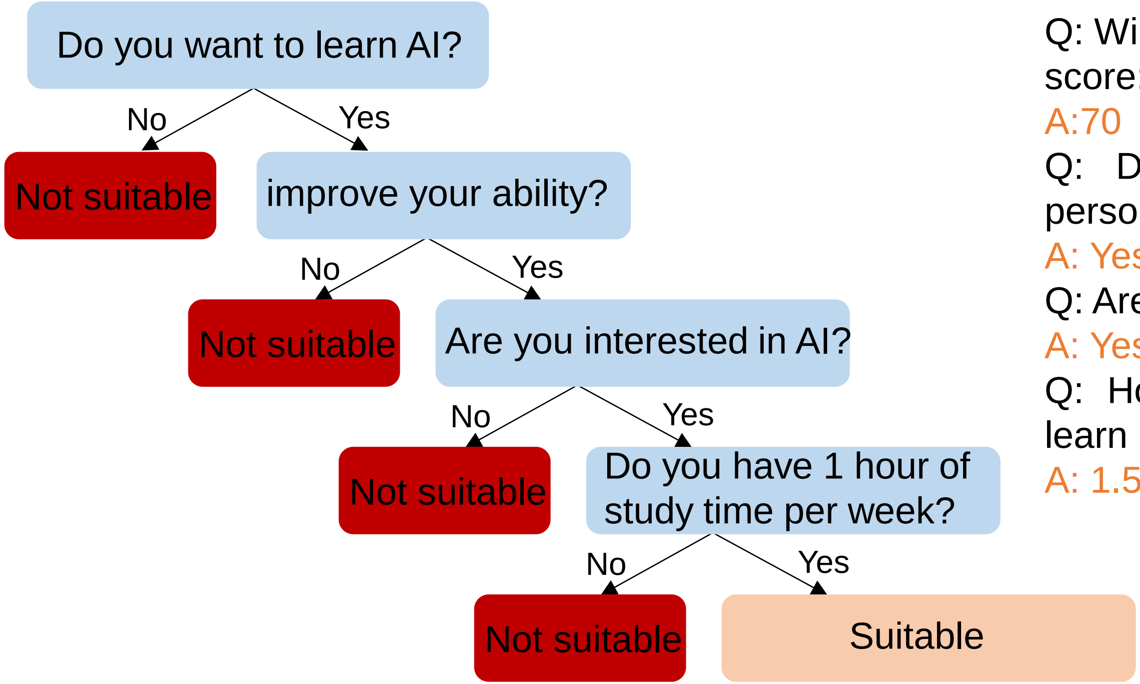
Task: Determine whether the user is suitable for this course based on their learning motivation, willingness to improve their abilities, interest, and free time.



Logistic Regression Framework

• Decision Tree

➢ Logistic Regression VS Decision Tree



Q: Willingness score for learning AI (full score: 100)

A:70

Q: Do you want to improve your personal ability?

A: Yes

Q: Are you interested in AI?

A: Yes

Q: How much time do you have to learn new knowledge per week?

A: 1.5 hours

⬇

Conclusion:
Sign up for study

• Decision Tree

A tree structure for classifying instances, identifying the categories to which the target belongs through multi-tiered evaluations

Essence: Through multi-layer judgment, a set of classification rules are summarized from the training data set

➢ Advantages

- Small amount of calculation, fast operation speed
- Easy to understand, and the importance of each attribute can be clearly viewed

➢ Disadvantages

- Ignoring the correlation between attributes
- When **the sample category distribution is uneven**, it is easy to affect the model performance

Decision Tree

What is the Decision Tree

Assuming the given training dataset:

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

Input instance:	$x = (x_i^1, x_i^2, \ldots, x_i^m)^T$
Number of features:	m
Class label:	$y_i \in \{1, 2, \ldots, K\}$
Sample size:	$i = 1, 2, \ldots, N$

✓ **Objective:**

Build a decision tree model based on the training data set so that it can correctly classify instances.

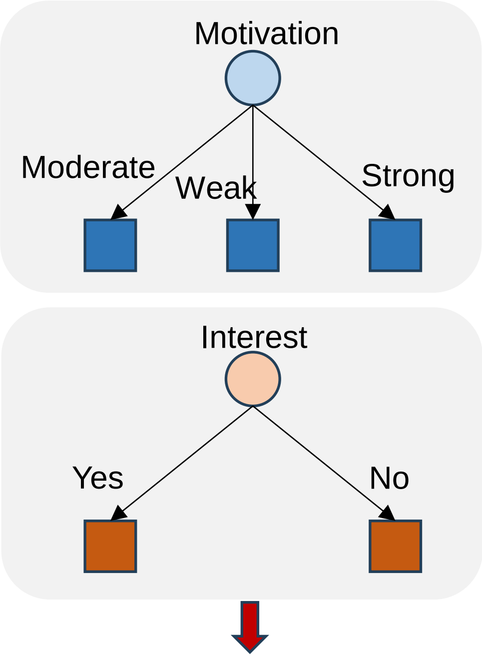
⬇

Key: Feature selection at each node

• Decision Tree

Dataset about the suitability of learning this course:

ID	Motivation	Abilities	Interest	Time	Lable
1	Moderate	No	No	Yes	No
2	Moderate	No	Yes	No	No
3	Strong	Yes	Yes	Yes	Yes
4	Moderate	No	No	Yes	No
5	Moderate	No	No	No	No
6	Moderate	Yes	No	No	No
7	Moderate	Yes	Yes	Yes	Yes
8	Moderate	Yes	Yes	Yes	Yes
9	Strong	Yes	Yes	Yes	Yes
10	Weak	No	No	No	No



Key: Different decision trees based on different features

Decision Tree

Solution method：ID3

The information entropy principle is used to select the attribute with the largest information gain as the classification attribute, and the branches of the decision tree are recursively expanded to complete the construction of the decision tree.

Information entropy is an indicator of the uncertainty of random variables. **The larger the entropy, the greater the uncertainty of the variable.** Assuming that the proportion of the K-th class of samples in the current sample set D is P_k, the information entropy of D is:

$$Ent(D) = -\sum_{k=1}^{|y|} P_K \log_2 P_K$$

$$P_K = 1 \Rightarrow Ent(D) = 0$$

• Decision Tree

➢ Solution method：ID3

According to information entropy, the information gain brought by sample division based on attribute a can be calculated:

$$Gain(D,a) = Ent(D) - \sum_{v=1}^{v} \frac{D^v}{D} Ent(D^v)$$

D: the total number of current samples
D^v: the number of samples of category V
V: the number of categories divided according to attribute a

$Ent(D)$ ➡ Information entropy before division

$\sum_{v=1}^{v} \frac{D^v}{D} Ent(D^v)$ ➡ Information entropy after division

⬇

Goal: The uncertainty of sample distribution after division is as small as possible, that is, the information entropy after division is small and the information gain is large.

• Decision Tree

➢ Solution method：ID3

$$Ent(D) = -\sum_{k=1}^{|y|} P_K \log_2 P_K$$

$$Gain(D,a) = Ent(D) - \sum_{v=1}^{v} \frac{D^v}{D} Ent(D^v)$$

Calculate the Ent and Gain of the attribute of interest:

$$Ent = -(\frac{6}{10}\log_2\frac{6}{10} + \frac{4}{10}\log_2\frac{4}{10}) = 0.971$$

$$\sum_{v=1}^{v}\frac{D^v}{D}Ent(D^v) = \frac{5}{10}(0) + \frac{5}{10}(-\frac{1}{5}\log_2\frac{1}{5} - \frac{4}{5}\log_2\frac{4}{5}) = 0.361$$

$$Gain = 0.971 - 0.361 = 0.61$$

ID	Motivation	Abilities	Interest	Time
Ent	0.6	0.55	0.36	0.55
Gain	0.37	0.42	0.61	0.42

Dataset about the suitability of learning this course:

ID	Motivation	Abilities	Interest	Time	Lable
1	Moderate	No	No	Yes	No
2	Moderate	No	Yes	No	No
3	Strong	Yes	Yes	Yes	Yes
4	Moderate	No	No	Yes	No
5	Moderate	No	No	No	No
6	Moderate	Yes	No	No	No
7	Moderate	Yes	Yes	Yes	Yes
8	Moderate	Yes	Yes	Yes	Yes
9	Strong	Yes	Yes	Yes	Yes
10	Weak	No	No	No	No

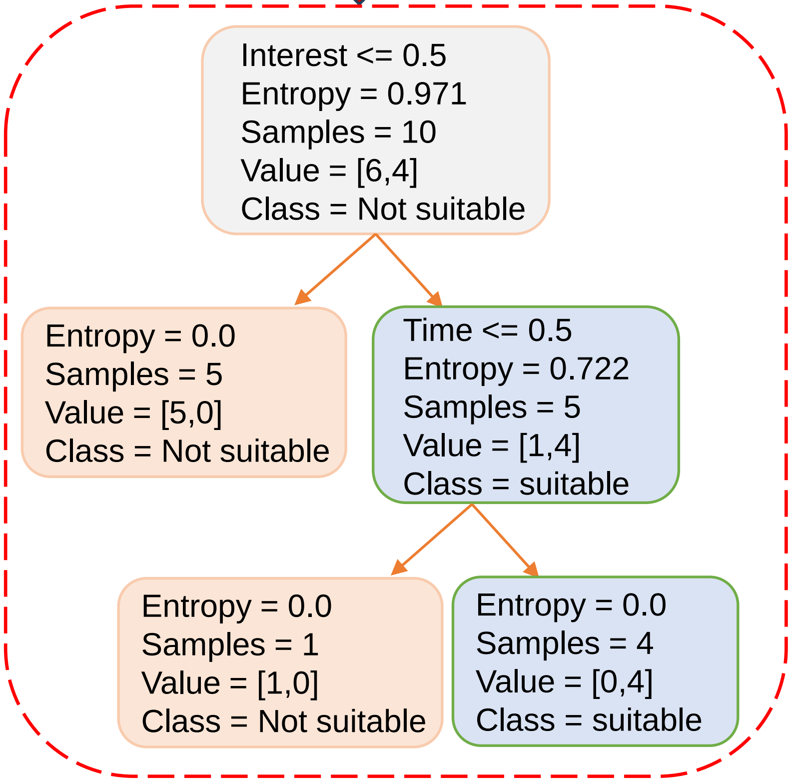
• Decision Tree

➢ Solution method：ID3

Dataset about the suitability of learning this course:

ID	Motivation	Abilities	Interest	Time	Lable
1	Moderate	No	No	Yes	No
2	Moderate	No	Yes	No	No
3	Strong	Yes	Yes	Yes	Yes
4	Moderate	No	No	Yes	No
5	Moderate	No	No	No	No
6	Moderate	Yes	No	No	No
7	Moderate	Yes	Yes	Yes	Yes
8	Moderate	Yes	Yes	Yes	Yes
9	Strong	Yes	Yes	Yes	Yes
10	Weak	No	No	No	No

Decision Tree



• Examples and Python code

Utilize decision trees to identify gender based on attributes like hair type, voice, height, clothing color, and more.

ID	Hair	Voice	Height	Color	Gender
1	Short	Rough	175	Dark	Male
2	Long	Soft	165	Bright	Female
3	Short	Rough	177	Bright	Male
4	Long	Rough	160	Bright	Female
5	Short	Soft	180	Dark	Male
6	Long	Soft	155	Bright	Female
7	Short	Rough	185	Dark	Male
8	Long	Soft	169	Bright	Female

```
# Import necessary libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
# Create a sample dataset
data = {
    'hair': ['short', 'long', 'short', 'long', 'short', 'long', 'short', 'long'],
    'voice': ['rough', 'soft', 'rough', 'rough', 'soft', 'soft', 'rough', 'soft'],
    'height': [175, 165, 177, 160, 180, 155, 185, 169],
    'color': ['dark', 'bright', 'bright', 'bright', 'dark', 'bright', 'dark', 'bright'],
    'gender': ['male', 'female', 'male', 'female', 'male', 'female', 'male', 'female']
}
df = pd.DataFrame(data)
# Convert categorical variables to numerical
df = pd.get_dummies(df, columns=['hair', 'voice', 'color’])
```

Examples and Python code

Utilize decision trees to identify gender based on attributes like hair type, voice, height, clothing color, and more.

```
df = pd.DataFrame(data)
# Convert categorical variables to numerical
df = pd.get_dummies(df, columns=['hair', 'voice', 'color’])

# Save the dataset to an Excel file
df.to_excel('student_data.xlsx', index=False)

# Split data into features and target
X = df.drop('gender', axis=1)
y = df['gender']
```

Height	Gender	Hair_ Long	Hair_ Short	Voice_ Rough	Voice_ Soft	Color_ Bright	Color_ Dark
175	Male	0	1	1	0	0	1
165	Female	1	0	0	1	1	0
177	Male	0	1	1	0	1	0
160	Female	1	0	1	0	1	0
180	Male	0	1	0	1	0	1
155	Female	1	0	0	1	1	0
185	Male	0	1	1	0	0	1
169	Female	1	0	0	1	1	0

• Examples and Python code

Utilize decision trees to identify gender based on attributes like hair type, voice, height, clothing color, and more.

```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create Decision Tree classifier
dt_classifier = DecisionTreeClassifier(random_state=42)

# Fit the model
dt_classifier.fit(X_train, y_train)

# Make predictions
y_pred = dt_classifier.predict(X_test)
```

Results

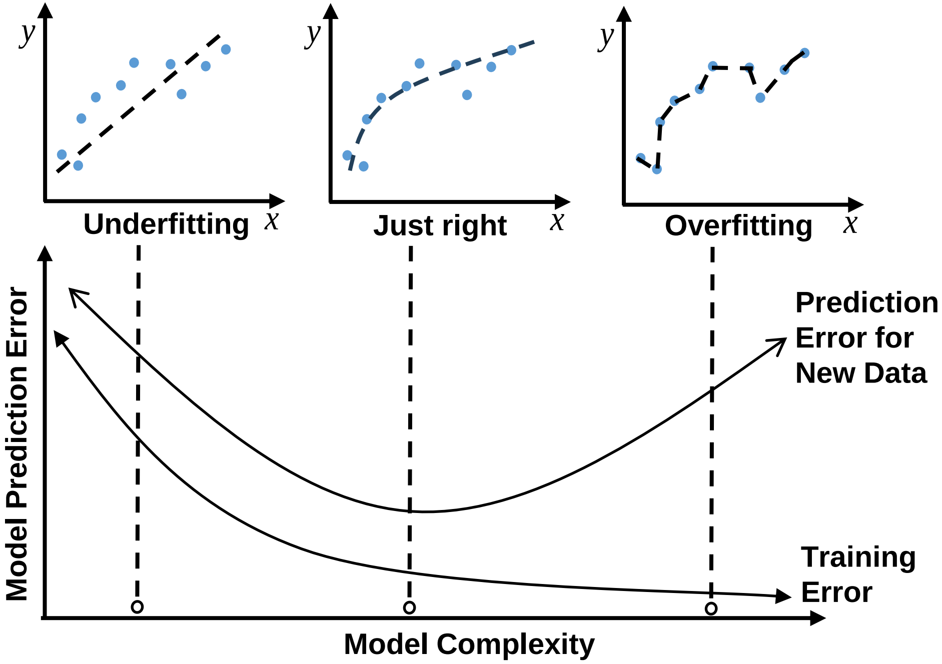
```
# Predict gender for a new student
new_student = pd.DataFrame({
    'hair_short': [1],
    'hair_long': [0],
    'voice_rough': [0],
    'voice_soft': [1],
    'height': [180],
    'color_bright': [0],
    'color_single': [1]
})

prediction = dt_classifier.predict(new_student)
print(f'Predicted Gender: {prediction[0]}')
```

```
Predicted Gender: male
```

• Overfitting vs. Underfitting

➢ Overfitting

- Overfitting occurs when **a model performs well on the training data but poorly on the test data**.
- The model learns **noise and details** in the training data rather than the underlying patterns.



➢ Underfitting

- Underfitting occurs when **a model performs poorly on both training and test data**.
- The model is too simple to capture the underlying patterns in the data.

Q&A

Thank you for your attention

Q&A

Dr Weisong Wen
Assistant Professor at PolyU
If you have any questions or inquiries, please feel free to contact me.
Lab Page: https://polyu-taslab.github.io/team/
Email: welson.wen@polyu.edu.hk