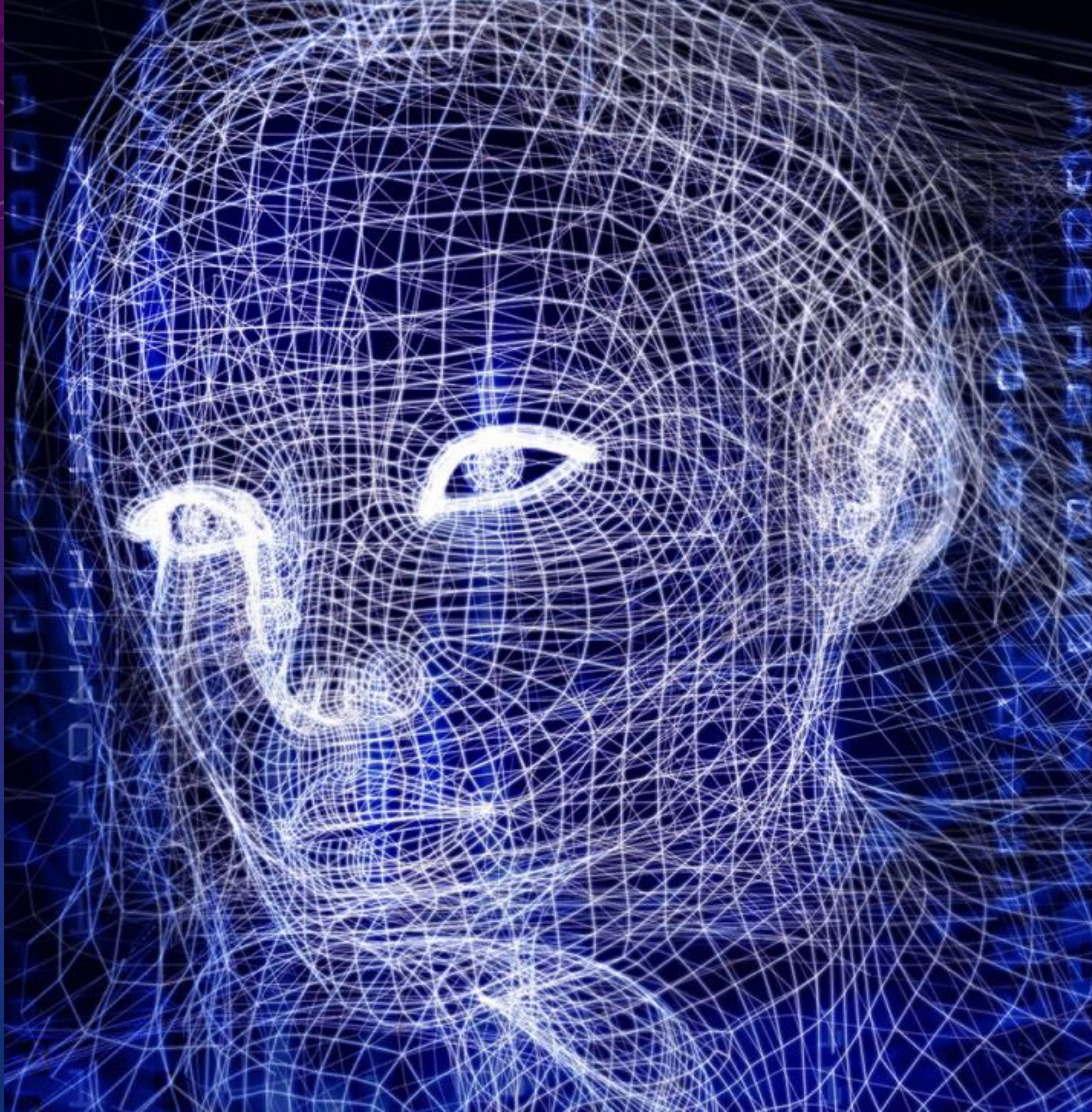
LECTURE SERIES FOR DIGITAL SURVEILLANCE SYSTEMS AND APPLICATION

INTRODUCTION TO DEEP LEARNING

徐繼聖

Gee-Sern Jison Hsu

National Taiwan University of Science and Technology



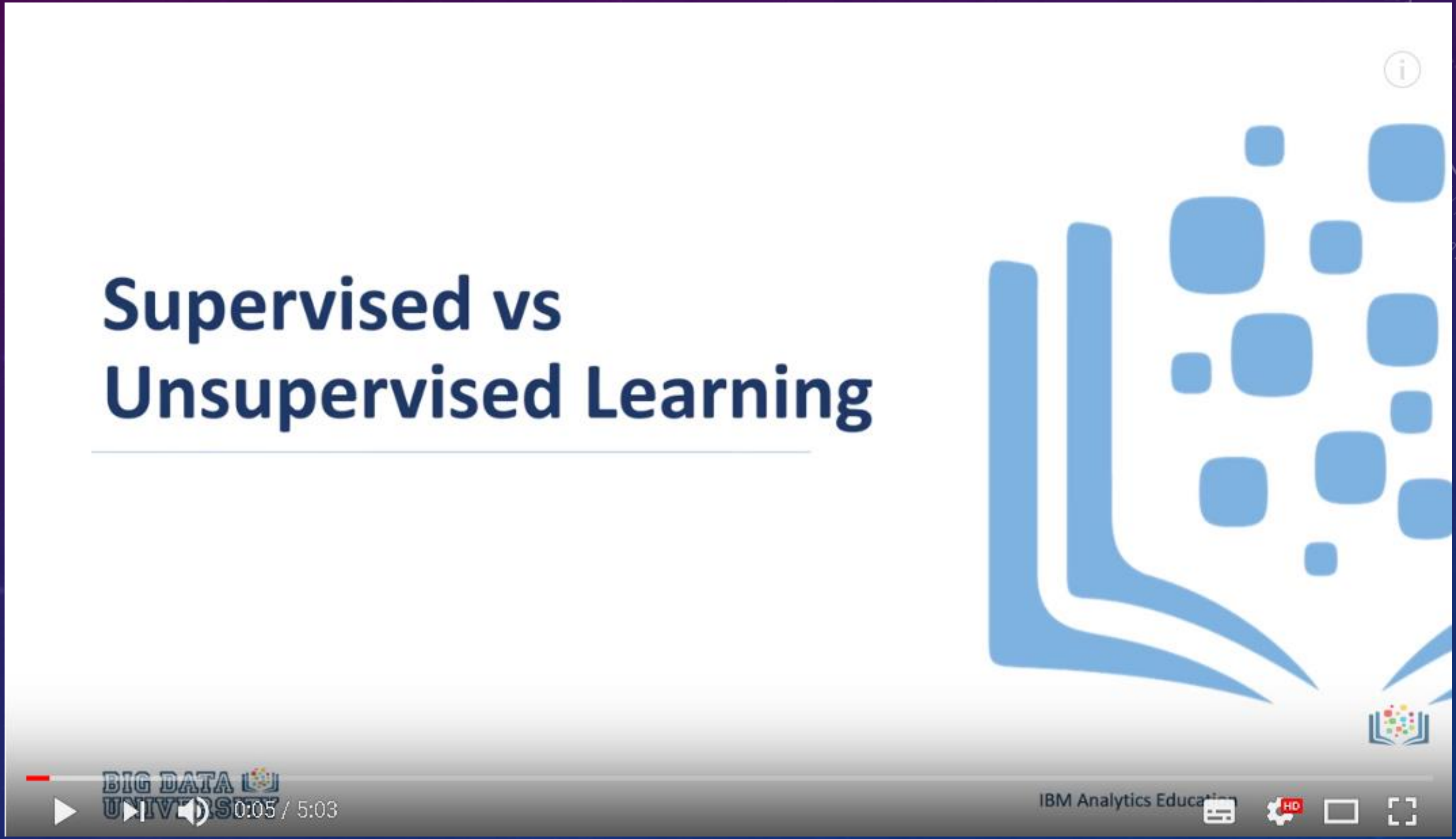
Supervised and Unsupervised Learning

1) Supervised and unsupervised learning
2) Classification models
3) Data for classification
4) Regression models
5) Data for regression
6) Performance measure in precision and recall

Supervised and Unsupervised Learning

Supervised Learning	Unsupervised Learning
• **Regression** Linear Regression, Random Forest, Multi-layer Perceptron • **Classification** Logistic Regression, Decision Tree	• **Clustering** K-means, Birch, Ward Spectral Cluster • **Association** example : Groups of shopper based on their browsing and purchasing histories

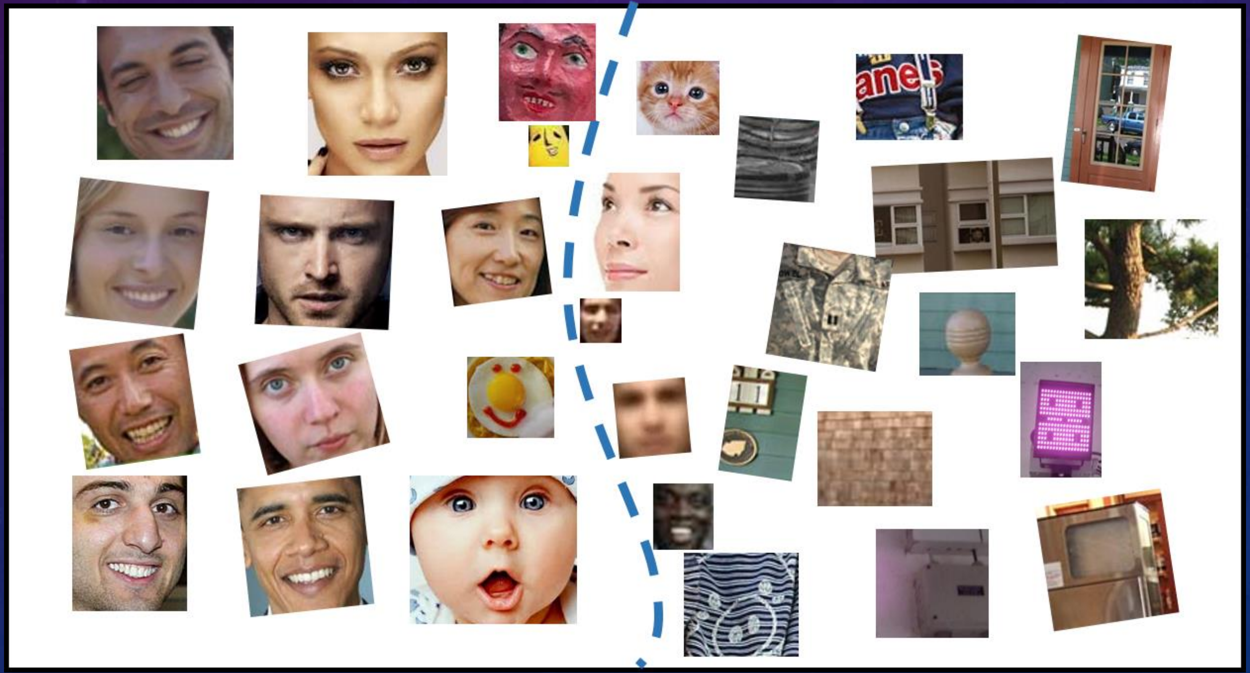
Supervised and Unsupervised Learning



https://www.youtube.com/watch?v=cfj6yaYE86U[5:03]

Classification Models

- Collection of training data
- Model built upon features (or more precisely, on the feature space)
  - Happy face = f (features extracted from a face)
- Model-based prediction when new data is given



High Dimensional
Feature Space

Binary Classification Training Data

	Dimension 1 χ_1	Dimension 2 χ_2	…	y
Example 1	0.95013	0.58279	…	1
Example 2	0.23114	0.4235	…	-1
Example 3	0.8913	0.43291	…	1
Example 4	0.018504	0.76037	…	-1
…	…	…	…	…

Typical Multiple Classification Data

	Dimension 1 χ_1	Dimension 2 χ_2	…	y
Example 1	0.95013	0.58279	…	1
Example 2	0.23114	0.4235	…	5
Example 3	0.8913	0.43291	…	6
Example 4	0.018504	0.76037	…	6
…	…	…	…	…

Regression Models

- Collection of training data
- Regression model built upon feature space
  - Stock value = ***G*** (previous closing, financial indices, profits, revenues, ….)
- Make a prediction given the known features.

TSMC 2330

Regression Learning Data

	Dimension 1 χ_1	Dimension 2 χ_2	…	y
Example 1	0.95013	0.58279	…	0.22
Example 2	0.23114	0.4235	…	-17.34
Example 3	0.8913	0.43291	…	50.1
Example 4	0.018504	0.76037	…	6.2
…	…	…	…	…

Samples of Face Detection

TP = 5 FP = 1 FN = 3

Nine faces in two images.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

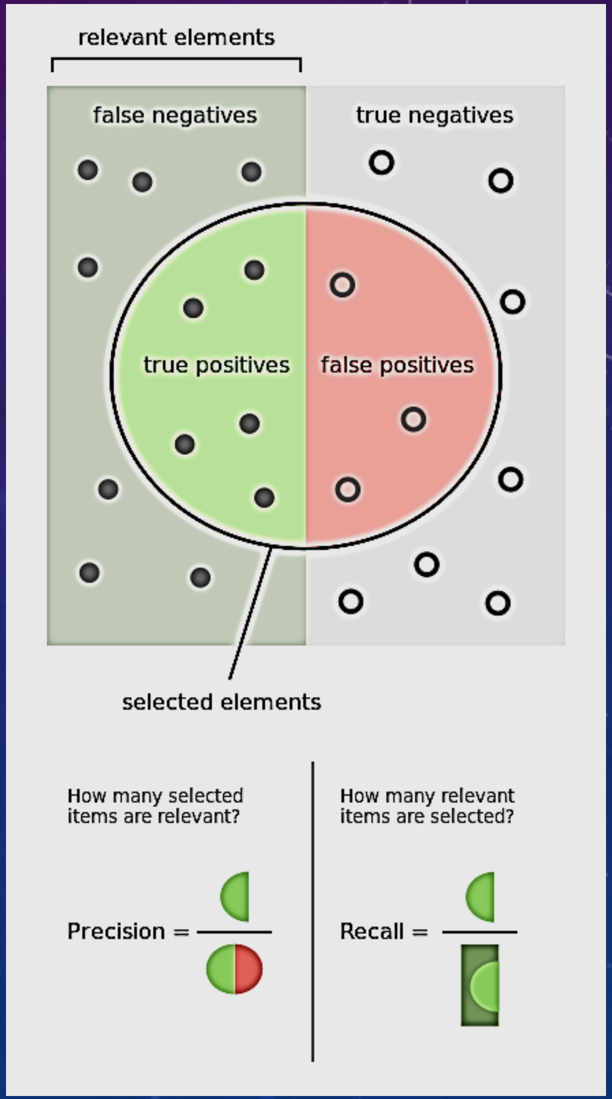
$$\text{Precision} = \frac{5}{5+1} = 0.833$$

$$\text{Recall} = \frac{5}{5+3} = 0.625$$

Precision and Recall

Precision is the fraction of retrieved instances that are relevant, while **Recall** is the fraction of relevant instances that are retrieved.

- True positive (TP) = correctly identified
- False positive (FP) = incorrectly identified
- True negative (TN) = correctly rejected
- False negative (FN) = incorrectly rejected

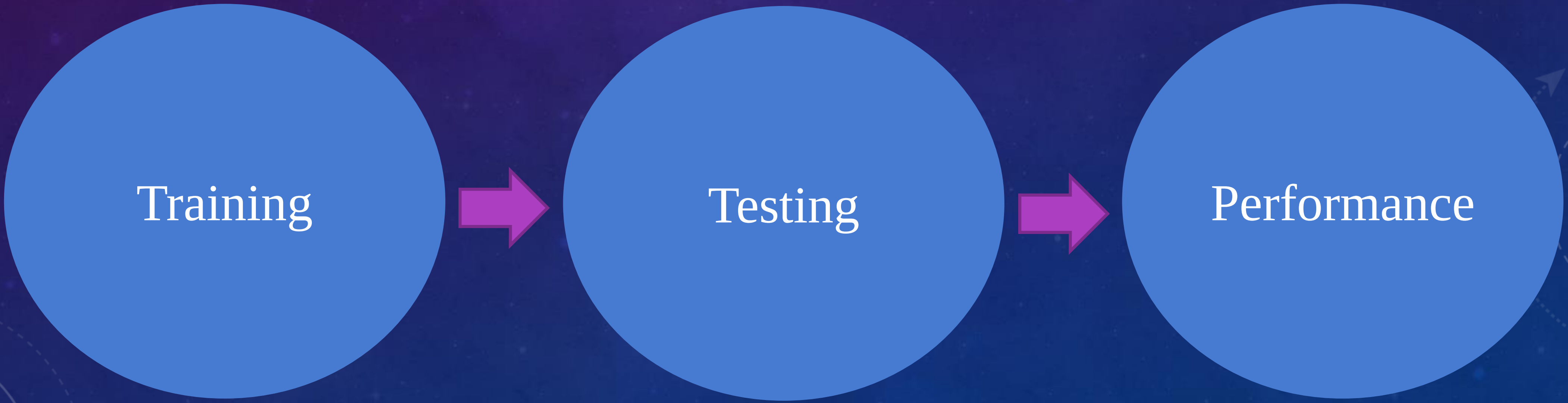


From Wikipedia Precision and Recall

Supervised Learning

Classification and regression consist of:

- Training and Testing Sets
- Models
- Performance on Benchmark Databases



Training and Testing Sets

Training Set

- A set of data made known to a system for building the classification regression model.
- For example, in a face recognition neural network, the face images used to train the network.

Face Images in CASIA-WebFace

Training and Testing Sets

Testing Set

- A set of data unseen to a recognition system.
- For example, the face images to be recognized by the trained face recognition network.

Face Images from IJBA

Linear Regression and Logistic Regression

Andrea Eunbee Jang

https://medium.com/biaslyai/pytorch-linear-and-logistic-regression-models-5c5f0da2cb9

Regression How it Works - Practical Machine Learning Tutorial with Python p.7

https://www.youtube.com/watch?v=V59bYfIomVk [7:56]

Linear Regression

The simplest form of a linear regression problem can be defined by the following sample equation:

$$\hat{Y} = aX + b + \mathrm{e}$$

where,

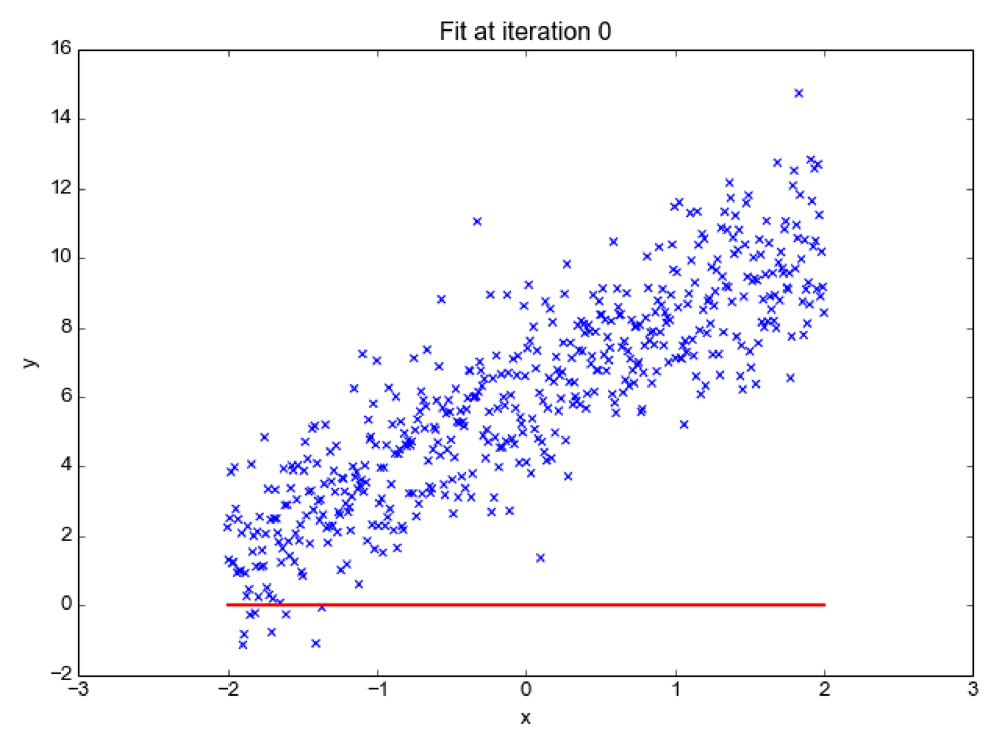
$\hat{Y}$ = Predicted value of Y

X = Independent variable

a = Slope coefficient based on best-fitting line

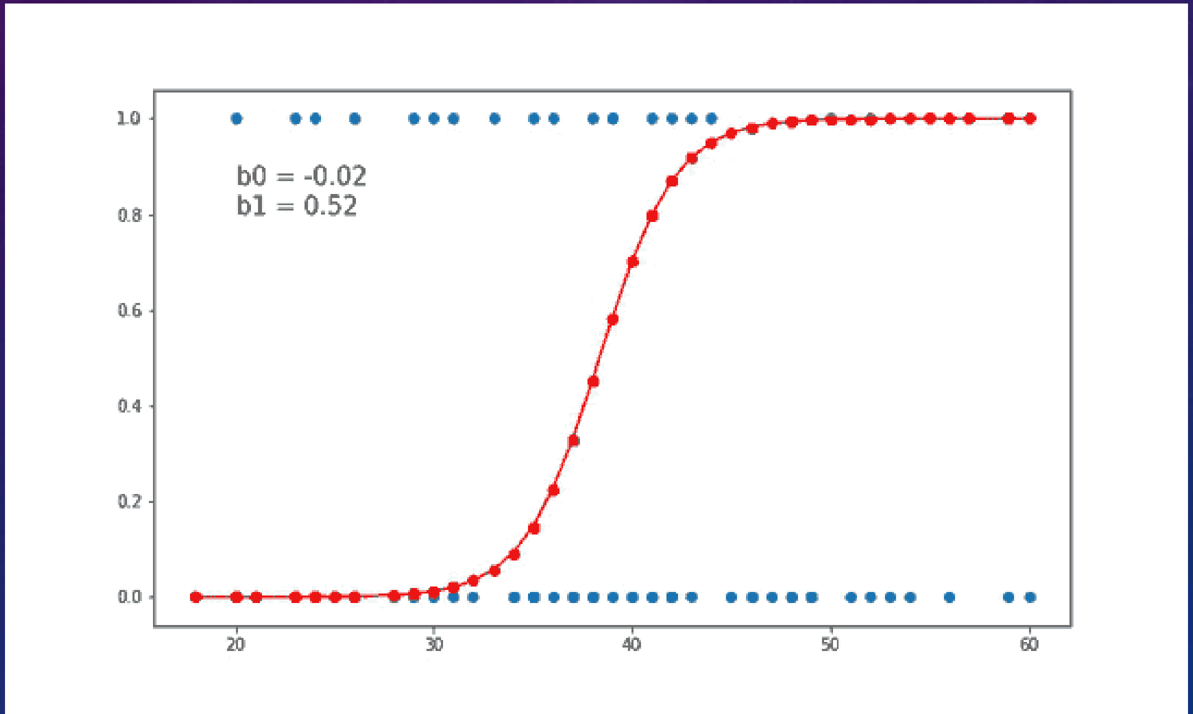
b = Y-axis intercept

e = Noise



Logistic Regression

Logistic regression is predictive analysis. It is used when the dependent variable Y is dichotomous (binary).



Linear Regression

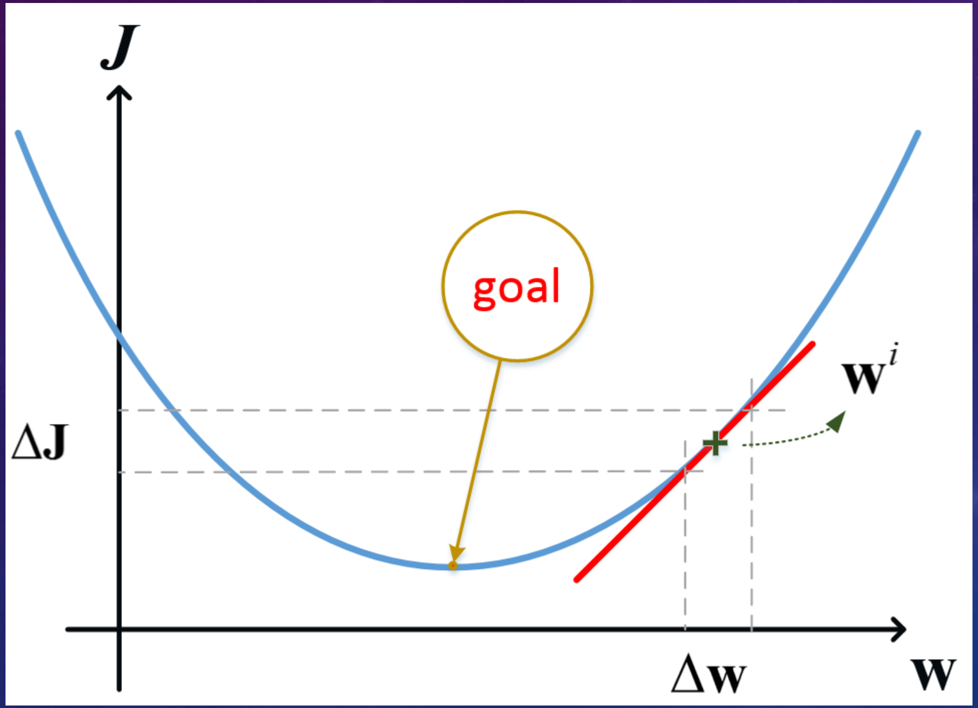
Example Code for Linear Regression

Step:

- Define Model Structure
- Loss Function (Criterion) and Optimizer
- Model Training
- Visualize Linear Regression

Gradient Descent

- Idea : update **w** with the data (***t*** **,** ***y***)
- The update rule :



$$w^{i+1} = w^{i} - \alpha \left(\frac{\partial J}{\partial w^{i}} \right)$$

where α is the learning rate

$$J = (\vec{t} - \vec{y})^{T} (\vec{t} - \vec{y})$$

$$\Rightarrow \frac{\partial \bar{J}}{\partial w} = -(\vec{t} - \vec{y})^{T} \frac{\partial y}{\partial w}$$

Gradient Descent

For this case:

- $J\colon \frac{1}{n}\sum_{i=1}^{n}(T_i - Y_i)^2$
- $Y(Predict) = 0.5x$+1, T : target
- Lr=0.001
- Δ:gradient
- $a_1 = a_0 - \Delta_a * lr$
- $b_1 = b_0 - \Delta_b * lr$

$$\Delta_a = \frac{\partial J}{\partial \mathbf{a}} = \frac{1}{3}\sum_{i=1}^{3} 2(T_i - Y_i) * X_i = \frac{6.27 + 3.872 + 18.26}{3} = 9.4673$$

$$\Delta_b = \frac{\partial J}{\partial \boldsymbol{b}} = \frac{1}{3}\sum_{i=1}^{3} 2(T_i - Y_i) = \frac{1.9 + 0.88 + 3.32}{3} = 2.0333$$

```
Target:[[1.7 ]
 [2.76]
 [2.09]]
Predict:tensor([[2.6500],
        [3.2000],
        [3.7500]], grad_fn=<AddmmBackward>)
Epoch [1/50], Loss: 1.2839, New_a:0.50, New_b:1.00, a_gradient:None, b_gradient:None
Target:[[1.7 ]
 [2.76]
 [2.09]]
Predict:tensor([[2.6167],
        [3.1563],
        [3.6959]], grad_fn=<AddmmBackward>)
Epoch [2/50], Loss: 1.1921, New_a:0.49, New_b:1.00, a_gradient:tensor([[9.4673]]), b_gradient:tensor([2.0333])
```

Please see the coding manual detail at page 2~9

Gradient Descent, How Neural Networks Learn
Deep Learning, Chapter 2

https://www.youtube.com/watch?v=IHZwWFHWa-w[21:00]

Exercise 1-1 : Linear Regression

- Please download 1-1_linear_regression.py and adjust the following parameters:

    Step 1 : Input Data

    x_train= [3.3], [4.4], [5.5], [8.2], [9.4]

    y_train= [1.7], [2.76], [2.09], [5.48], [4.99]

    Step 2 : Epoch

    Step 3 : Learning Rate

- Please upload your code and your comments in MS Word to Moodle.

Neural Network — Feedforward / MLP

Andrea Eunbee Jang

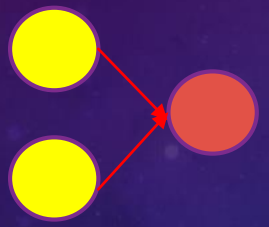
https://medium.com/biaslyai/pytorch-introduction-to-neural-network-feedforward-neural-network-model-e7231cff47cb

Convolutional Neural Networks (CNNs) Explained

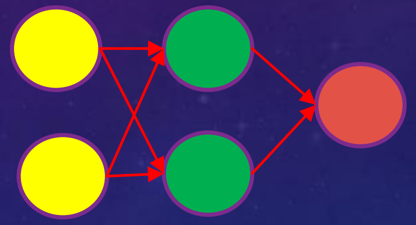
https://www.youtube.com/watch?v=YRhxdVk_sIs&t=62s [8:36]

Neural Network — Feedforward / MLP

Perceptron (P)



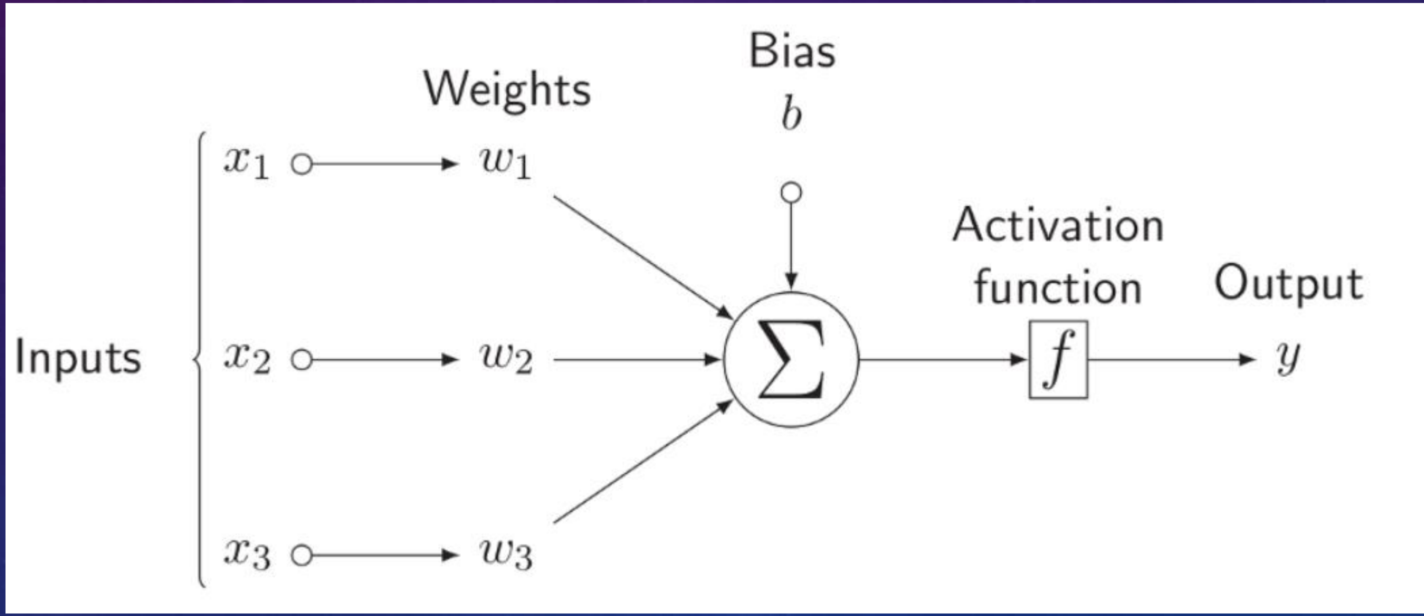
Feed Forward(FF)



Yellow : Input layer
Green : Hidden layer
Orange : Output layer

Single-Layer Perceptron

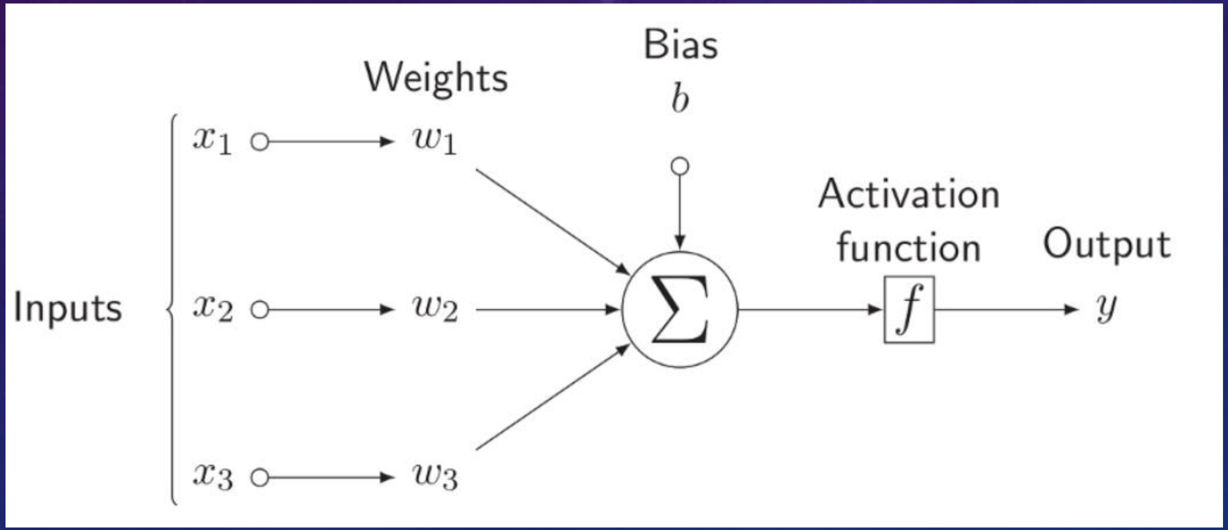
Single-layer perceptron takes data as input and its weights are summed up then an activation function is applied before sent to the output layer.

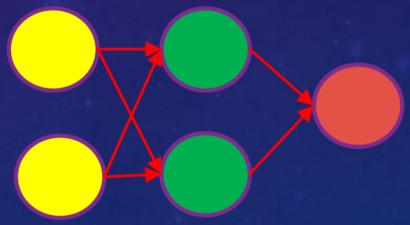


https://tex.stackexchange.com/questions/132444/diagram-of-an-artificial-neural-network

Activation Function

- The activation functions in the neural network introduce the non-linearity to the linear output.
- It defines the output of a layer, given data, meaning it sets the threshold for making the decision of whether to pass the information or not.





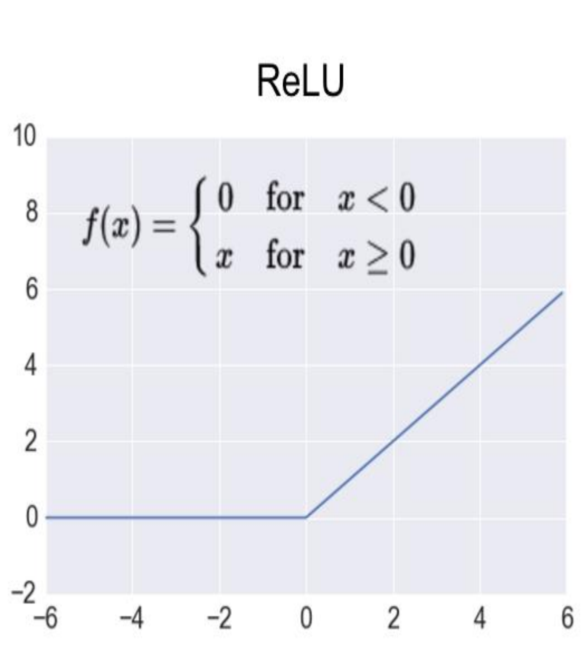
Yellow : Input layer
Green : Hidden layer
Orange : Output layer

https://tex.stackexchange.com/questions/132444/diagram-of-an-artificial-neural-network

Perceptron Model

Example:

```
class Perceptron(torch.nn.Module):
    def __init__(self):
        super(Perceptron, self).__init__()
        self.fc = nn.Linear(1,1)
        self.relu = torch.nn.ReLU()
    def forward(self, x):
        output = self.fc(x)
        output = self.relu(x)
      return output
```



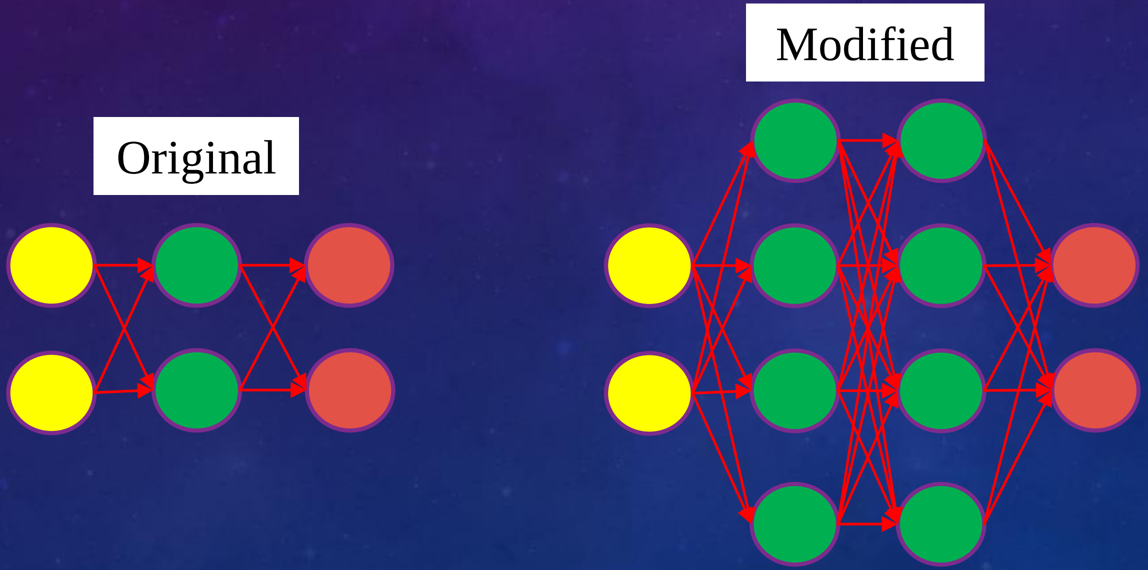
self.fc = outputs linear information

self.relu = makes it non-linear

Please see the coding manual detail at page 10~15

Exercise 1-2 – Build A Neural Network

- Please download 1-2_log_regression_visualize.py which contains a network "Original".
- Please modify the settings in the file so that it can be structured as the "Modified".
- See the configurations below for both networks.
- You also need to add the activation function "ReLU" behind each hidden layer
- Please upload your python code in MS Word to Moodle.



Yellow: Input layer
Green: Hidden layer
Orange: Output layer