MACHINE LEARNING

Neural Network

RNN

CNN

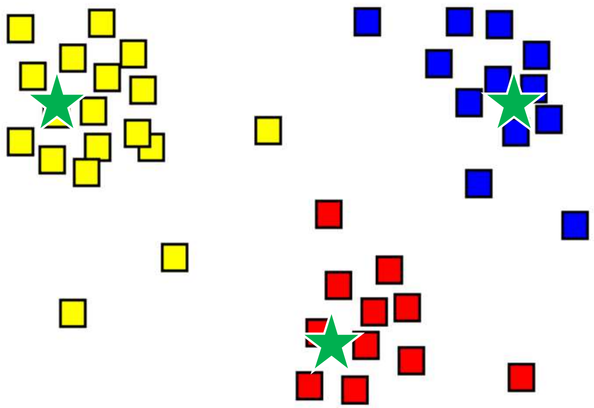
Review

Cha, Eunju

Review

Clustering

– K-mean Clustering

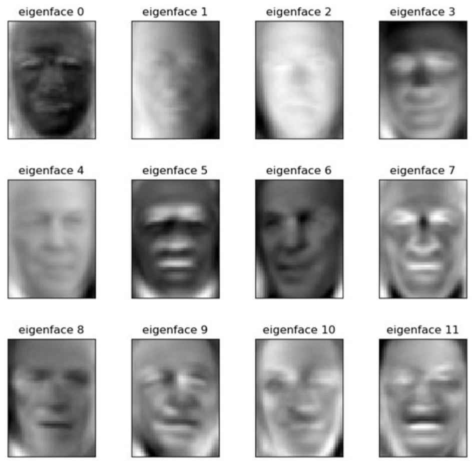


K-means clustering is an unsupervised learning algorithm that groups data points into clusters, gathering similar items together.

The primary objective is to minimize the variance within each cluster so that the items within a cluster are similar to each other and distinctly different from items in other clusters.

Dimension Reduction

– Principal Component Analysis



Dimension reduction is a technique in machine learning and data science that simplifies high-dimensional data by transforming it into a lower-dimensional form, making it easier to handle.

Principal Component Analysis (PCA):

Neural Network

Artificial Neural Network [ANN]

What is ANN?

A computational model **inspired by the way biological neural networks in the human brain process information**.
Utilizes a network of interconnected **artificial neurons (nodes) to model complex relationships** between inputs and outputs.

How does it work?

Mimics the brain's large network of neurons, each neuron in ANN contributes to solving complex learning problems.
While the **human brain** contains approximately **86 billion neurons**, ANNs operate with far fewer neurons, signifying that the creation of an artificial brain is not an immediate risk.

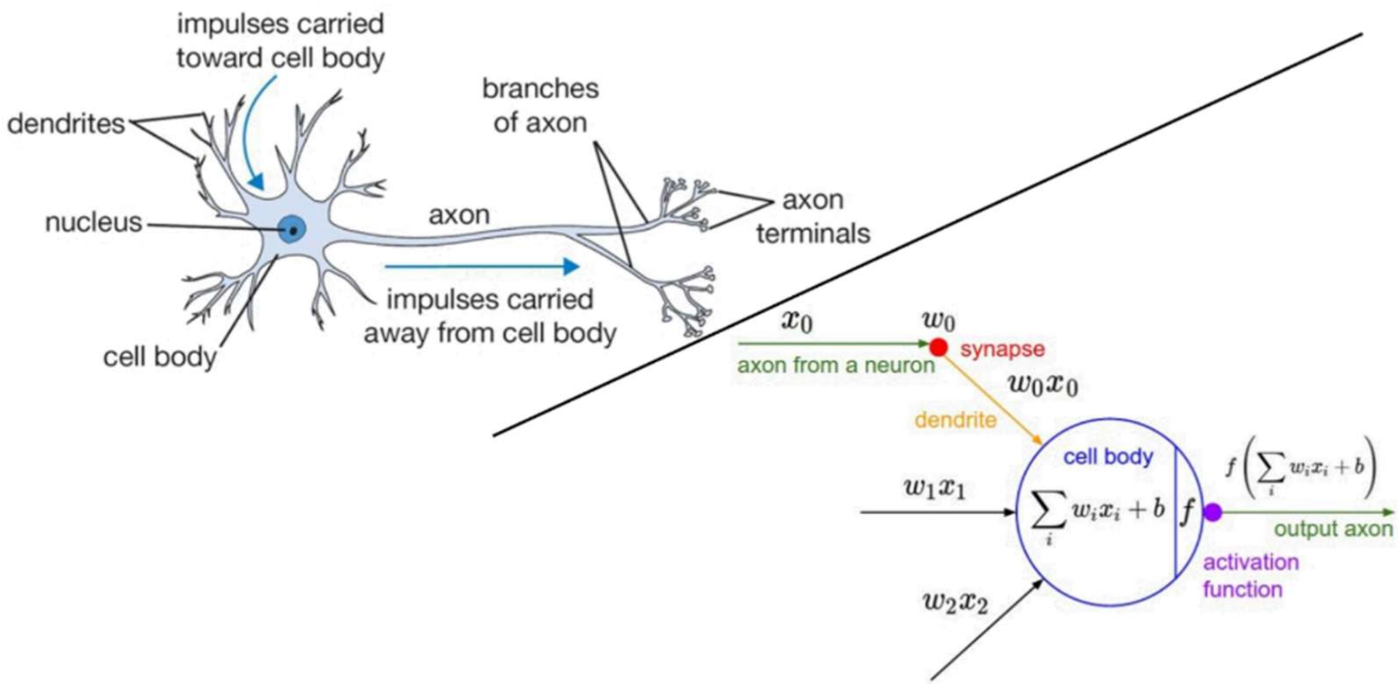
Why ANN?

Capable of representing vast amounts of knowledge through interconnected networks.
Solves a wide range of complex computational tasks, much like the human brain responds to sensory inputs.

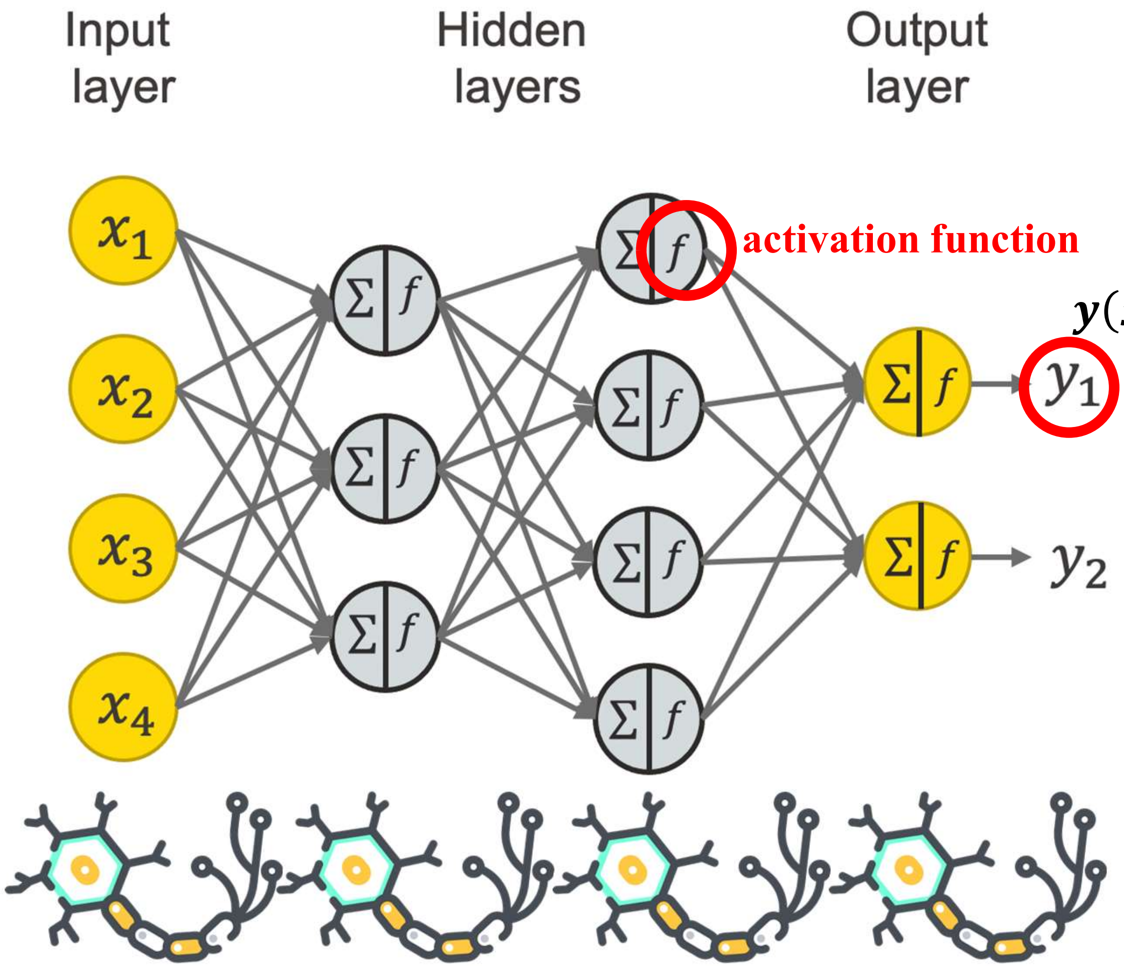
Neural Network

Artificial Neural Network [ANN]

ANN is a versatile learner that **can be applied to nearly any learning task**, such as classification, numerical prediction, and autonomous pattern recognition



Neural Network



$$y(x) = f\left[\sum_{i=1}^{n} w_i x_i\right]$$

Activation Function: Transforms the net input signal of a neuron into a single output signal that can propagate further across the network.

Network Topology: Describes the number of layers and neurons within the model, as well as the pattern of connections between them.

Training Algorithm: Specifies the method for setting connection weights to either inhibit or excite neurons in proportion to the input signals.

Neural Network

1. Input Layer

Role: The input layer **receives raw data**. This could be **images, text, numerical data**, etc.

Features: Each node in this layer represents one feature of the input

2. Hidden Layers

Role: Located between the input and output layers, hidden layers **learn complex patterns and features from the data**.

Features: one or multiple hidden layers, each containing several neurons. Neurons → weights and activation functions.

3. Output Layer

Role: **Produces the final predictions of the model**. #nodes and the activation function in this layer → specific task.

Classification: **#nodes usually matches # classes**, and a softmax activation function might be used.

Regression: typically **one node is used**, and sometimes no activation function is applied.

4. Activation Functions

Purpose: Introduce non-linearity, enabling the neural network **to learn complex patterns**.

Examples: Sigmoid, ReLU (Rectified Linear Unit), tanh (hyperbolic tangent), etc.

5. Learning Process

Feedforward: The process where **data flows from the input layer to the output layer sequentially**.

Backpropagation and Gradient Descent: The method by which the model adjusts its weights. It **calculates how far off the prediction** is from the actual value and **modifies the weights to reduce this error**.

6. Loss Function

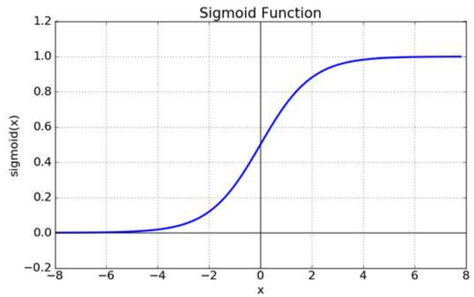
Purpose: **Measures how different the model's predictions are from the actual values**.

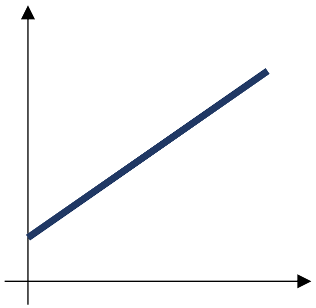
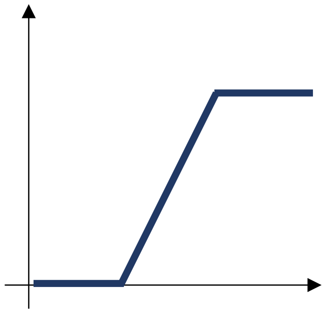
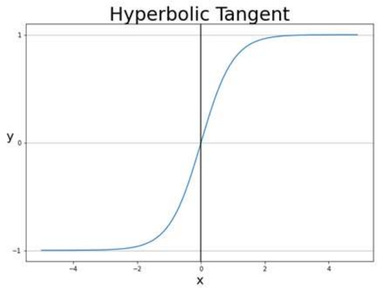
Examples: Mean Squared Error (MSE) for regression tasks, Cross-Entropy for classification tasks.

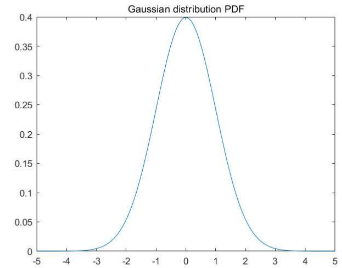
Neural Network

Activation functions

- **Mechanisms** that allow artificial neurons to process **incoming information and transmit** it through the network.
- Just as artificial neurons are modeled after their biological versions, **activation functions are also modeled after natural designs.**
- If the threshold is met, the neuron fires and transmits a signal; if not, it does nothing.
  ➔ Threshold activation function, which only generates an output signal when a specified input threshold is reached.
- **Sigmoid, Linear, Saturated Linear, hyperbolic tangent,** and **Gaussian functions** are also used.
- A linear activation function → Similar to linear regression
- Gaussian activation function → Radial Basis Function Network (RBF Network).





Neural Network

Network Topology

The learning ability of a neural network is attributed to the topology (or pattern and structure) of interconnected neurons.

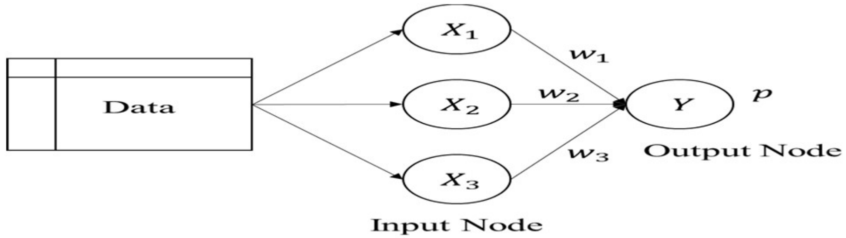
While there are numerous forms of network architectures, they can be distinguished based on three main characteristics:

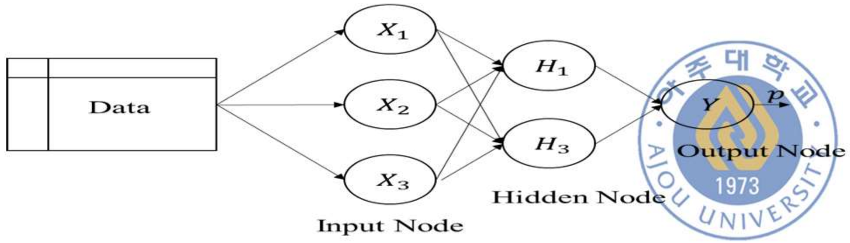
- **The number of layers in the network**
- **Whether the network allows information to move backward [Direction of Information Movement]**
- **The number of nodes in each layer of the network**

Topology determines the complexity of tasks that can be learned by the network.

As networks become larger and more complex, they are able to identify more subtle patterns and complex decision boundaries.

The strength of a network is not only a function of its size but also how its constituent units are arranged





Neural Network

Network Topology

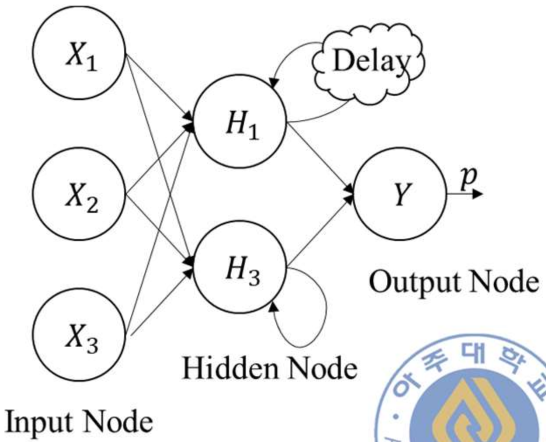
- **Whether the network allows information to move backward [Direction of Information Movement]**

Feedforward Networks

- ***Definition:*** Information travels in a single direction from input to output without looping back.
- ***Flexibility:*** Highly adaptable, can increase complexity by adding more layers and nodes.
- ***Deep Neural Networks:*** Feedforward networks with multiple hidden layers, ideal for deep learning.

Recurrent Networks

- ***Definition:*** Networks with the ability for information to travel backward, allowing loops, mimicking biological neural networks.
- ***Learning Complex Patterns:*** Can recognize time-dependent patterns thanks to backward paths, short-term memory, and delays.
- ***Applications:*** Especially useful for time series analysis, like stock market predictions, speech recognition, and weather forecasting.



Implications

- ***Problem Suitability:*** Each network type is tailored to different problem complexities.
- ***Structural Impact:*** The structure directly affects the network's ability to solve a range of problems.

Neural Network

Network Topology

- **The number of nodes in each layer of the network**

Key Components

- ***Input Nodes:*** Represent each feature of the data
- ***Output Nodes:*** Determine the classes of the modeled outputs
- ***Hidden Nodes:*** Decided by the user, crucial for learning capability

Complexity Adjustment

- Network complexity can be adjusted by the number of nodes in hidden layers
- More complex problems require more nodes and connections

Learning Capability

- ***More neurons:*** Better reflection of the training data
- ***Beware of Overfitting:*** Too many neurons can be risky
- ***Minimum Nodes Usage:*** Maintain adequate performance on the validation dataset

Optimization Strategy

- High learning ability achievable with a few hidden nodes
- Appropriate number of nodes varies with the amount of training data, noise, and task complexity

Neural Network

Artificial Neural Network [ANN]

Advantages

- ***Flexibility:*** Can be applied to various types of data and problems.
- ***Generalization:*** Capable of applying learned patterns to new, unseen data.
- ***Adaptability:*** Can improve performance through learning from real-time data and self-adjustments.
- ***Scalability:*** Easy to increase model complexity by adding more neurons and layers

Disadvantages

- ***Overfitting:*** May perform too well on limited data, reducing generalization to new data.
- ***Opacity:*** Often described as 'black boxes' with internal workings that are difficult to interpret.
- ***Training Time:*** Large datasets and complex networks require significant time to learn.
- ***Data Dependence:*** High performance depends on the availability of large volumes of labeled data.

Neural Network

Extended Models

Artificial Neural Networks (ANNs)

- ***Basics:*** A network of connected neurons for general-purpose learning.
- ***Functionality:*** Processes numerical data for classification or regression.

Convolutional Neural Networks (CNNs)

- ***Evolution:*** Built on ANNs, specialized for spatial data recognition.
- ***Strengths:*** Exceptional at image processing tasks like recognition and classification.
- ***Key Feature:*** Utilizes convolutional layers to capture spatial hierarchies in data.

Recurrent Neural Networks (RNNs)

- ***Evolution:*** Another ANN extension, designed for sequential data processing.
- ***Strengths:*** Ideal for time series analysis, natural language processing, and anything involving sequence and context.
- ***Key Feature:*** Can remember past information and use it to influence current processing (memory).

The Relationship

- ***CNNs:*** Tailored for data with spatial relationships and patterns.
- ***RNNs:*** Suited for data with temporal sequences and time-dependent patterns.

** All stem from the fundamental ANN structure.*

Neural Network

Convolutional Neural Network [CNN]

Overview

- Specialized for processing grid-like data (e.g., images).
- Exceptional in identifying patterns in spatial data.

Key Components

- ***Convolutional Layers:*** Extract features by applying filters to the input.
- ***Activation Functions:*** Introduce non-linearity (typically ReLU).
- ***Pooling Layers:*** Reduce dimensionality and highlight key features.
- ***Fully Connected Layers:*** Make final predictions based on learned features.

How It Works

- Layers progressively learn higher-level features (edges, textures, complex objects).
- Robust to variations in object positions within the image.

Applications

- Image and video recognition.
- Medical image analysis.
- Self-driving car technology.

Neural Network

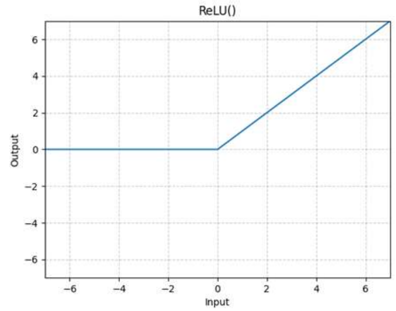
Convolutional Neural Network [CNN]

1. Convolutional Layers

Function: Apply various filters (kernels) to the input image to extract features. Each filter scans a small region of the image, calculating the dot product of the filter and the pixel values of that region.

Result: This process creates a feature map that captures essential information (e.g., edges, textures) from the image.

2. Activation Functions



Common Example: ReLU (Rectified Linear Unit)

Function: Introduce non-linearity to the network. ReLU converts all negative values to zero, enabling the neural network to solve non-linear problems.

3. Pooling Layers

Type: Max Pooling is most commonly used.

Function: Reduce the size of the feature map (downsampling) and emphasize important features.

4. Fully Connected Layers

Function: Make the final classification or prediction based on the extracted features.

Flattening Process: Positioned at the end of the CNN, it flattens the output of the previous layers into a one-dimensional vector and then generates outputs for classification or other tasks.

These layers in a CNN progressively learn and extract low to high-level features from each image, enabling the network to perform complex visual tasks such as image classification, object detection, and image segmentation.

Neural Network

Recurrent Neural Network [RNN]

RNNs are tailored for data where order and context over time are crucial, such as **text or time-series data.**

Working Principle:

Sequence Processing: RNN processes data sequentially. Each timestep's input depends on the output from the previous timestep.

Internal Memory: RNN retains previous information and combines it with the current input to produce output. This feature allows it to consider the temporal continuity of data.

Learning through Backpropagation: Like other neural networks, RNNs use backpropagation for learning. However, they employ a method known as 'backpropagation through time' for this purpose.

Key Challenge:

Long-Term Dependencies: Standard RNNs struggle to learn dependencies over long sequences. This issue is addressed in advanced variants like LSTM (Long Short-Term Memory) or GRU (Gated Recurrent Units).

Applications:

Natural Language Processing: Text generation, machine translation, speech recognition, etc.

Time Series Prediction: Stock price forecasting, weather prediction, etc.

Other Sequential Data Processing: Learning temporal features in video data, music generation, etc.

Neural Network

- **ANN (Artificial Neural Network)**
  - **Working Mechanism:** Utilizes a Multilayer Perceptron (MLP) structure, receiving data in the input layer, processing it through hidden layers, and outputting it in the output layer.
  - **Data Processing:** Each neuron processes signals using weights and activation functions and passes them to the next layer. ANNs can perform general pattern recognition on fixed input sizes.
  - **Main Applications:** Basic classification and regression problems, simple image and text data processing, etc.
- **CNN (Convolutional Neural Network)**
  - **Working Mechanism:** Designed to process spatial data like images. Extracts features from images through convolutional and pooling layers.
  - **Data Processing:** CNNs learn local patterns in images (e.g., edges, textures) and can recognize these patterns at various positions, achieved through spatial hierarchical structures.
  - **Main Applications:** Image and video recognition, medical image analysis, etc.
- **RNN (Recurrent Neural Network)**
  - **Working Mechanism:** Designed for sequential data processing (e.g., text, time-series data). It processes current inputs in conjunction with the previous outputs, considering temporal continuity.
  - **Data Processing:** RNNs 'remember' previous information and learn patterns over time, which is possible due to sequential processing of each element in a sequence.
  - **Main Applications:** Natural language processing (text generation, machine translation), time-series data analysis, etc.

Neural Network

- **ANN (Artificial Neural Network)**
  - **Working Mechanism:** Utilizes a Multilayer Perceptron (MLP) structure, receiving data in the input layer, processing it through hidden layers, and outputting it in the output layer.
  - **Data Processing:** Each neuron processes signals using weights and activation functions and passes them to the next layer. ANNs can perform general pattern recognition on fixed input sizes.
  - **Main Applications:** Basic classification and regression problems, simple image and text data processing, etc.

- **Key Differences**

ANN: General pattern recognition for fixed input sizes.
CNN: Learning and recognizing local features in spatial data.
RNN: Sequential processing and pattern recognition in data with temporal continuity.

  - **Data Processing:** CNNs learn local patterns in images (e.g., edges, textures) and can recognize these patterns at various positions, achieved through spatial hierarchical structures.
  - **Main Applications:** Image and video recognition, medical image analysis, etc.

- **RNN (Recurrent Neural Network)**
  - **Working Mechanism:** Designed for sequential data processing (e.g., text, time-series data). It processes current inputs in conjunction with the previous outputs, considering temporal continuity.
  - **Data Processing:** RNNs 'remember' previous information and learn patterns over time, which is possible due to sequential processing of each element in a sequence.
  - **Main Applications:** Natural language processing (text generation, machine translation), time-series data analysis, etc.

Neural Network

Building a simple classification model to categorize digits using the MNIST handwritten dataset [ANN]

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

# Load Dataset
(train_images, train_labels), (test_images, test_labels) =
mnist.load_data()

# Data Preprocessing
train_images = train_images / 255.0
test_images = test_images / 255.0

# Model composition
model = Sequential([
    Flatten(input_shape=(28, 28)),  # Input layer
    Dense(128, activation='relu'),  # Hidden layer
    Dense(10, activation='softmax') # Output layer
])

# Model compile
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Model Training
model.fit(train_images, train_labels, epochs=5)

# Model Evaluation
test_loss, test_accuracy = model.evaluate(test_images, test_labels)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')
```

Neural Network

Building a simple classification model to categorize digits using the MNIST handwritten dataset [ANN]

```
test_loss, test_accuracy = model.evaluate(test_images, test_labels)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 10s 1us/step
Epoch 1/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.2567 - accuracy: 0.9260
Epoch 2/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.1133 - accuracy: 0.9665
Epoch 3/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.0769 - accuracy: 0.9765
Epoch 4/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.0571 - accuracy: 0.9820
Epoch 5/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.0433 - accuracy: 0.9867
313/313 [==============================] - 0s 1ms/step - loss: 0.0774 - accuracy: 0.9748
Test Loss: 0.07743441313505173, Test Accuracy: 0.9747999906539917
```

Test Loss: Indicates how well the model performs on the test data, represented by a loss value. A lower loss value suggests that the model has fit well to the data.

Test Accuracy: Represents how accurately the model classifies the test data. Higher accuracy indicates that the model is making accurate predictions on the test dataset

Neural Network

Epoch

```
test_loss, test_accuracy = mode
print(f'Test Loss: {test_loss},

Downloading data from https://s
11490434/11490434 [============
Epoch 1/5
1875/1875 [====================
Epoch 2/5
1875/1875 [====================
Epoch 3/5
1875/1875 [====================
Epoch 4/5
1875/1875 [====================
Epoch 5/5
1875/1875 [====================
313/313 [======================
Test Loss: 0.07743441313505173,
```

An epoch in machine learning and deep learning refers to a single cycle through the entire training dataset. Simply put, an epoch represents one complete pass of the training data through the model.

Importance of Epochs

Learning Process: During each epoch, the model sees every sample in the dataset once, updating its weights based on these samples.

Performance Improvement: As the model goes through multiple epochs, it gradually reduces errors and improves its performance.

Overfitting Monitoring: Setting the right number of epochs is crucial as too many epochs can lead to overfitting, where the model performs well on the training data but poorly on new, unseen data.

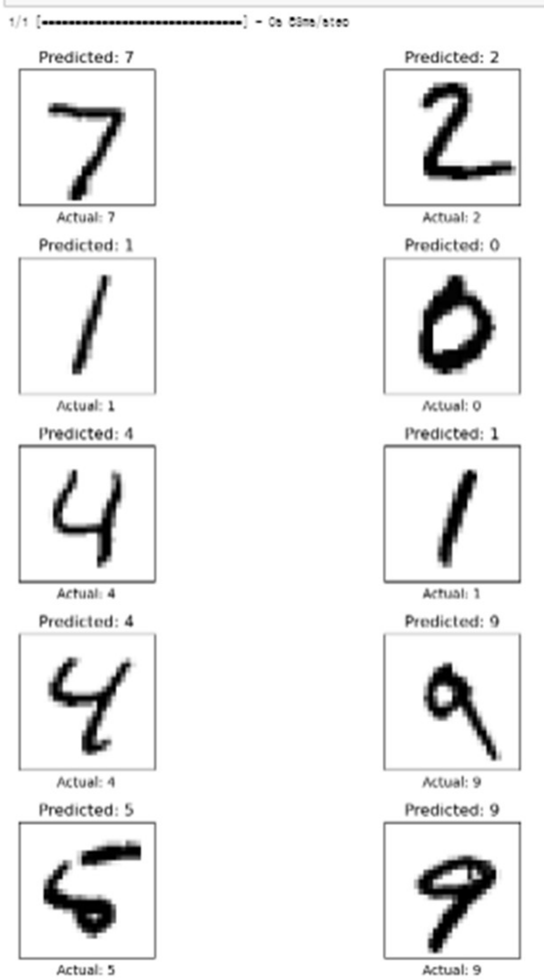
Neural Network

```python
test_loss, test_accuracy = model.evaluate(test_images, test_labels)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 10s 1us/step
Epoch 1/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.2567 - accuracy: 0.9260
Epoch 2/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.1133 - accuracy: 0.9665
Epoch 3/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.0769 - accuracy: 0.9765
Epoch 4/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.0571 - accuracy: 0.9820
Epoch 5/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.0433 - accuracy: 0.9867
313/313 [==============================] - 0s 1ms/step - loss: 0.0774 - accuracy: 0.9748
Test Loss: 0.07743441313505173, Test Accuracy: 0.9747999906539917
```

Neural Network

```
import matplotlib.pyplot as plt
import numpy as np

# Selecting images from test dataset
test_images_subset = test_images[:10]
test_labels_subset = test_labels[:10]

# Prediction as using the model
predictions = model.predict(test_images_subset)

# Visualization of the result
plt.figure(figsize=(10, 10))
for i in range(10):
    plt.subplot(5, 2, i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(test_images_subset[i], cmap=plt.cm.binary)
    plt.xlabel(f"Actual: {test_labels_subset[i]}")
    plt.title(f"Predicted: {np.argmax(predictions[i])}")
plt.tight_layout()
plt.show()
```



Neural Network

Configure the CNN model & training for classification Fashion MNIST data

```
import tensorflow as tf
from tensorflow.keras.layers import Conv2D, MaxPooling2D,
Flatten, Dense
from tensorflow.keras.models import Sequential

# Load the Fashion MNIST dataset
fashion_mnist = tf.keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) =
fashion_mnist.load_data()

# Data preprodessing: Normalize the images to be between 0
and 1
train_images = train_images / 255.0
test_images = test_images / 255.0

# Expand the dimensions of the image data (to make it
compatible with CNNs, which require 4D input)
train_images = train_images.reshape(train_images.shape[0],
28, 28, 1)
test_images = test_images.reshape(test_images.shape[0], 28,
28, 1)
```

```
# Configure the CNN model
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(28, 28, 1)),  # Convolutional layer
    MaxPooling2D(2, 2),  # Pooling layer
    Conv2D(64, (3,3), activation='relu'),  # Another convolutional layer
    MaxPooling2D(2,2),  # Another pooling layer
    Flatten(),  # Flattening layer
    Dense(128, activation='relu'),  # Fully connected layer
    Dense(10, activation='softmax')  # Output layer
])

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
model.fit(train_images, train_labels, epochs=10)

# Evaluate the model
test_loss, test_accuracy = model.evaluate(test_images, test_labels)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')
```

Neural Network

Configure the CNN model & training for classification Fashion MNIST data

```python
# Evaluate the model
test_loss, test_accuracy = model.evaluate(test_images, test_labels)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')
```

```
Epoch 1/10
1875/1875 [==============================] - 22s 11ms/step - loss: 0.4434 - accuracy: 0.8401
Epoch 2/10
1875/1875 [==============================] - 21s 11ms/step - loss: 0.3052 - accuracy: 0.8882
Epoch 3/10
1875/1875 [==============================] - 21s 11ms/step - loss: 0.2597 - accuracy: 0.9039
Epoch 4/10
1875/1875 [==============================] - 22s 12ms/step - loss: 0.2242 - accuracy: 0.9162
Epoch 5/10
1875/1875 [==============================] - 22s 12ms/step - loss: 0.1970 - accuracy: 0.9254
Epoch 6/10
1875/1875 [==============================] - 22s 12ms/step - loss: 0.1754 - accuracy: 0.9340
Epoch 7/10
1875/1875 [==============================] - 22s 12ms/step - loss: 0.1551 - accuracy: 0.9425
Epoch 8/10
1875/1875 [==============================] - 22s 12ms/step - loss: 0.1378 - accuracy: 0.9484
Epoch 9/10
1875/1875 [==============================] - 23s 12ms/step - loss: 0.1220 - accuracy: 0.9536
Epoch 10/10
1875/1875 [==============================] - 22s 12ms/step - loss: 0.1087 - accuracy: 0.9593
313/313 [==============================] - 1s 4ms/step - loss: 0.2835 - accuracy: 0.9195
Test Loss: 0.28352105617523193, Test Accuracy: 0.9194999933242798
```

Neural Network

Configure the CNN model & training for classification Fashion MNIST data

```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D,
MaxPooling2D

# Generate predictions for the first 25 images in the test dataset
predictions = model.predict(test_images[:25])
```

```
# Visualize these predictions along with the actual labels
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(test_images[i].reshape(28, 28), cmap=plt.cm.binary)
    predicted_label = np.argmax(predictions[i])
    true_label = test_labels[i]
    if predicted_label == true_label:
        color = 'blue'
    else:
        color = 'red'
    plt.xlabel("{} ({})".format(class_names[predicted_label],
                                class_names[true_label]),
                                color=color)
plt.show()
```

Neural Network

Neural Network

Build & train a simple Recurrent Neural Network (RNN) using TensorFlow's Keras API on the IMDB movie review sentiment classification task

```
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense

# Load dataset IMDB
# Load words of num_words
num_words = 10000
max_review_length = 500

(train_data, train_labels), (test_data, test_labels) =
imdb.load_data(num_words=num_words)

# Data Preprocessing: padding or cutting Review length to
max_review_length
train_data = sequence.pad_sequences(train_data,
maxlen=max_review_length)
test_data = sequence.pad_sequences(test_data,
maxlen=max_review_length)

# Build RNN Model
model = Sequential()
model.add(Embedding(num_words, 32))
model.add(SimpleRNN(32))  # composition 32 neurons for RNN layer
model.add(Dense(1, activation='sigmoid'))

# Compile model
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])

# Training Model
history = model.fit(train_data, train_labels, epochs=10, batch_size=128,
validation_split=0.2)

# Summary of the result
history_dict = history.history
print(history_dict.keys())

# Model Evaluation
test_loss, test_accuracy = model.evaluate(test_data, test_labels)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')
```

Neural Network

Build & train a simple Recurrent Neural Network (RNN) using TensorFlow's Keras API on the IMDB movie review sentiment classification task

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 [==============================] - 21s 1us/step
Epoch 1/10
157/157 [==============================] - 13s 75ms/step - loss: 0.6307 - acc: 0.6302 - val_loss: 0.5586 - val_acc: 0.7190
Epoch 2/10
157/157 [==============================] - 11s 73ms/step - loss: 0.4043 - acc: 0.8249 - val_loss: 0.4168 - val_acc: 0.8108
Epoch 3/10
157/157 [==============================] - 11s 72ms/step - loss: 0.3311 - acc: 0.8673 - val_loss: 0.3810 - val_acc: 0.8380
Epoch 4/10
157/157 [==============================] - 11s 73ms/step - loss: 0.2661 - acc: 0.8972 - val_loss: 0.3957 - val_acc: 0.8234
Epoch 5/10
157/157 [==============================] - 11s 73ms/step - loss: 0.2096 - acc: 0.9220 - val_loss: 0.4044 - val_acc: 0.8584
Epoch 6/10
157/157 [==============================] - 12s 75ms/step - loss: 0.1730 - acc: 0.9362 - val_loss: 0.4525 - val_acc: 0.8272
Epoch 7/10
157/157 [==============================] - 13s 82ms/step - loss: 0.1359 - acc: 0.9514 - val_loss: 0.4632 - val_acc: 0.8058
Epoch 8/10
157/157 [==============================] - 12s 79ms/step - loss: 0.1008 - acc: 0.9649 - val_loss: 0.4428 - val_acc: 0.8622
Epoch 9/10
157/157 [==============================] - 13s 86ms/step - loss: 0.0814 - acc: 0.9740 - val_loss: 0.5024 - val_acc: 0.8326
Epoch 10/10
157/157 [==============================] - 14s 87ms/step - loss: 0.0533 - acc: 0.9836 - val_loss: 0.5168 - val_acc: 0.8486
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
782/782 [==============================] - 12s 15ms/step - loss: 0.5240 - acc: 0.8472
Test Loss: 0.5240156054496765, Test Accuracy: 0.8471999764442444
```

Neural Network

Build & train a simple Recurrent Neural Network (RNN) using TensorFlow's Keras API on the IMDB movie review sentiment classification task

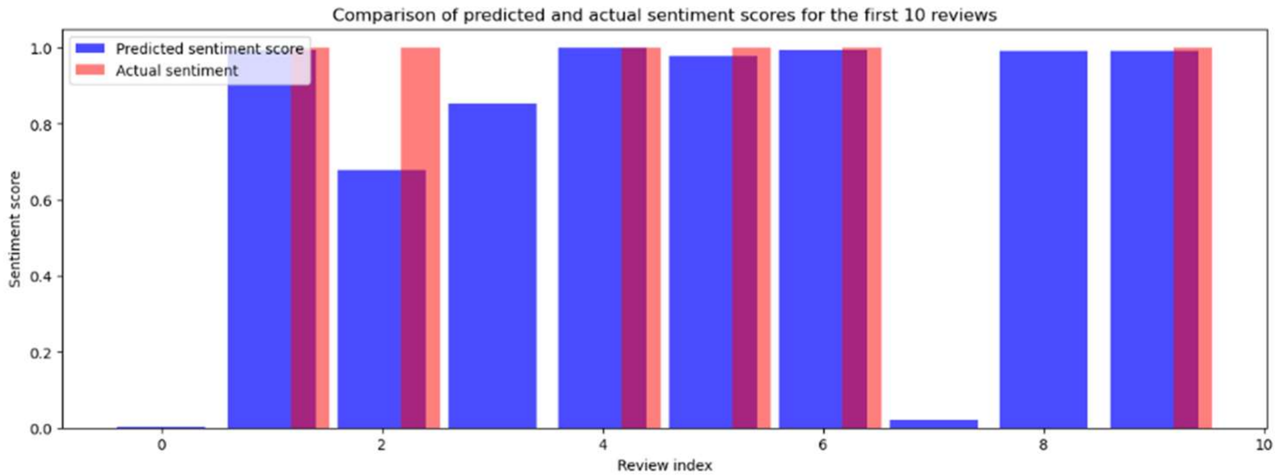
```
import matplotlib.pyplot as plt

# Load predicted values of the model
predictions = model.predict(test_data)

# Comparing actual label and predicted result for 10 reviews
for i in range(10):
    print(f'Review: {i+1}')
    print(f'Predicted sentiment: {"Positive" if predictions[i] > 0.5 else "Negative"}')
    print(f'Actual sentiment: {"Positive" if test_labels[i] == 1 else "Negative"}')
    print('-----------------------------------')

# Visualization that actual and Predicted label as the bar chart
plt.figure(figsize=(15, 5))
plt.bar(range(10), predictions[:10, 0], color='blue', alpha=0.7, label='Predicted sentiment score')
plt.bar(np.array(range(10)) + 0.35, test_labels[:10], color='red', alpha=0.5, width=0.35, label='Actual sentiment')
plt.xlabel('Review index')
plt.ylabel('Sentiment score')
plt.title('Comparison of predicted and actual sentiment scores for the first 10 reviews')
plt.legend()
plt.show()
```

Neural Network

Build & train a simple Recurrent Neural Network (RNN) using TensorFlow's Keras API on the IMDB movie review sentiment classification task

```
782/782 [==============================] - 11s 14ms/step
Review: 1
Predicted sentiment: Negative
Actual sentiment: Negative
-----------------------------------
Review: 2
Predicted sentiment: Positive
Actual sentiment: Positive
-----------------------------------
Review: 3
Predicted sentiment: Positive
Actual sentiment: Positive
-----------------------------------
Review: 4
Predicted sentiment: Positive
Actual sentiment: Negative
-----------------------------------
Review: 5
Predicted sentiment: Positive
Actual sentiment: Positive
-----------------------------------
Review: 6
Predicted sentiment: Positive
Actual sentiment: Positive
-----------------------------------
Review: 7
Predicted sentiment: Positive
Actual sentiment: Positive
-----------------------------------
Review: 8
Predicted sentiment: Negative
Actual sentiment: Negative
-----------------------------------
Review: 9
Predicted sentiment: Positive
Actual sentiment: Negative
-----------------------------------
```

```
-----------------------------------
Review: 10
Predicted sentiment: Positive
Actual sentiment: Positive
-----------------------------------
```



Final Exam

Exam Day: 12/18 (Next week)

Time: 4:30 pm ~ 6:00 pm (If you finish earlier, you are allowed to leave)

Exam Method: Paper.

You are allowed one cheat sheet (A4 size)
however, it must be handwritten.
Printed cheat sheets are not permitted!

The exam will consist of **25 multiple-choice** questions and **5 short-answer** questions.

Final Exam

What is the range of the output value of a logistic regression model?

a. 0 or 1

b. -∞ to +∞

c. 0 to 1

d. Any integer value

What does the Least Squares Method aim to minimize?

a. The sum of the squared errors between observed and predicted values

b. The maximum error between observed and predicted values

c. The sum of the absolute errors between observed and predicted values

d. The sum of the observed values

Final Exam

In which area is the Naïve Bayes classification algorithm primarily used?

a. Image recognition

b. Text classification and spam filtering

c. Speech recognition

d. Game theory

Review

What did you study in last week?

Text Mining

TF, DF, TF-IDF

These are key concepts used to quantify document content in text mining and information retrieval

TF (Term Frequency)

$$TF(t,d) = \frac{Number\ of\ times\ term\ t\ appears\ in\ document\ d}{Total\ number\ of\ terms\ in\ document\ d}$$

DF (Document Frequency)

$$DF(t) = Number\ of\ documents\ containing\ term\ t$$

TF-IDF(Term Frequency-Inverse Document Frequency)

$$TF - IDF(t,d) = TF(t,d) \times \log\left(\frac{Total\ number\ of\ documents}{DF(t)}\right)$$

Final Exam

Examples

There is 3 documents that was removed stop words. Let’s find TF-IDF(“cat”, Document 1). [*There's no need to calculate log(), please write down the solution process.*]

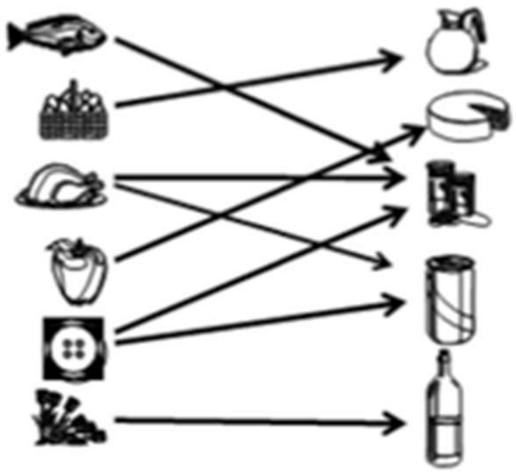
- Document 1: “cat sat mat“
- Document 2: "dog sat log"
- Document 3: "cat sat dog"

Review

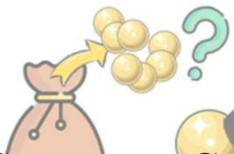
Sentiment Analysis

Sentiment Analysis is a process in text mining that aims to determine the sentiment expressed in a piece of text

Association Rule Analysis



A data analysis technique that identifies relationships between different items or variables.

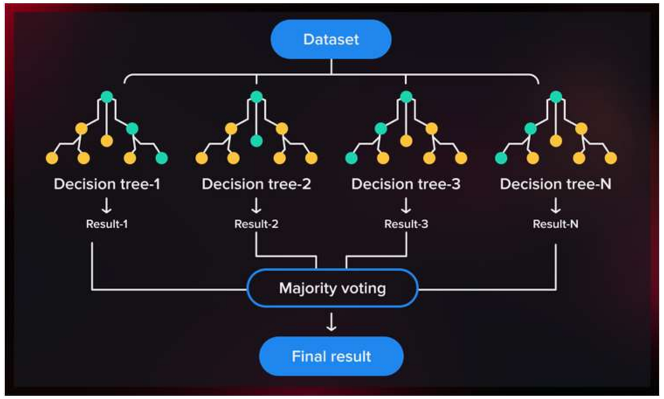


Support

Confidence

Lift

Random Forest



An ensemble machine learning method that combines multiple decision trees for more accurate predictions and classifications.

Ensemble

Bootstrapping

THANK YOU

Neural Network

- **RNN**
- **CNN**

Review