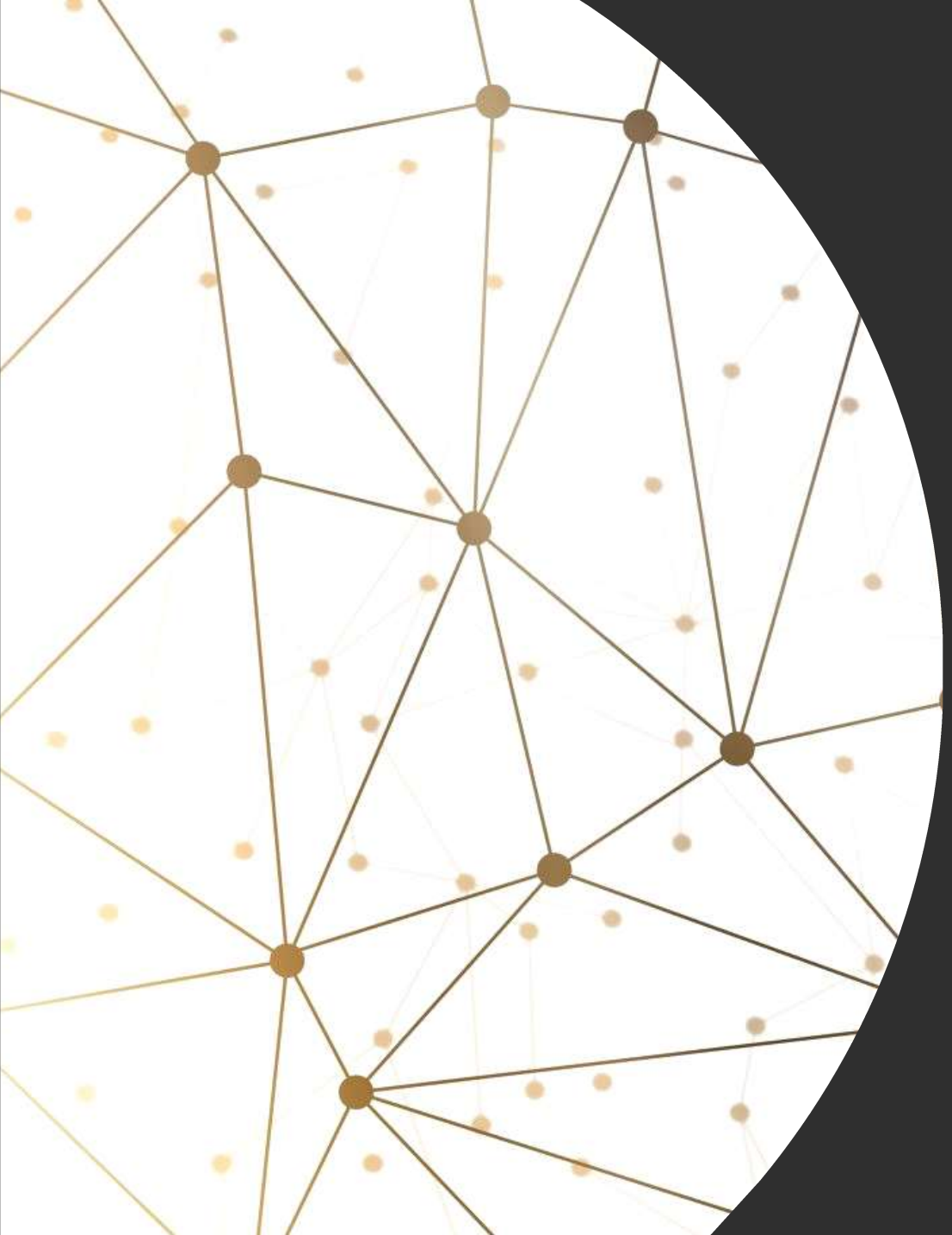
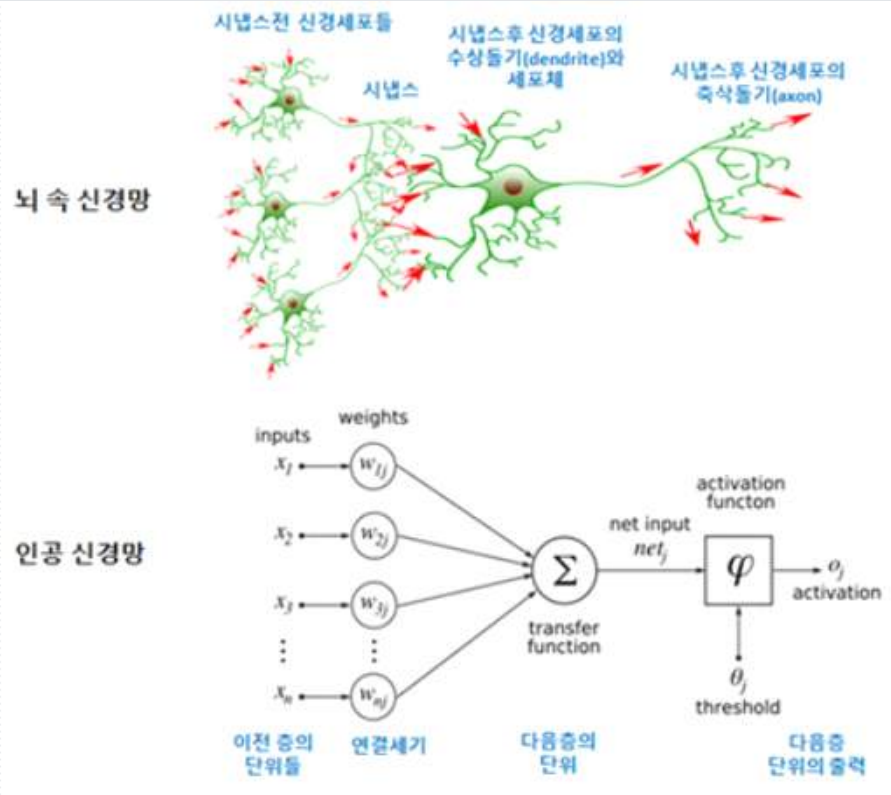


GDGoC-SCHU Week4_task2

24-25 [AI] 황효동

인공신경망이란?



- 생물학적 신경망에서 영감을 받아 설계된 컴퓨팅 시스템
- 주로 인간 뇌의 신경 세포(뉴런)가 정보를 처리하고 전달하는 방식을 수학적으로 모델링하여 구현한 것이며, 이는 기계학습과 딥러닝의 핵심 기술로 사용하게 된다.

인공신경망 종류

- **퍼셉트론**
  - 인공 신경망의 가장 기본적인 형태로, 단층 신경망을 기반으로 한다.

- **다층 퍼셉트론 (MLP)**
  - 다층 퍼셉트론은 퍼셉트론을 여러 층으로 확장한 형태의 신경망

- **역전파 알고리즘**
  - 다층 퍼셉트론을 학습시키기 위한 기본 알고리즘

신경망 설계의 주요 요소

- 은닉층의 수
- 은닉층의 뉴런 수
- 학습률
- 배치 크기

케라스란?

- 텐서플로(TensorFlow)의 고수준 딥러닝 API, 신경망 설계를 간단하게 도와주는 도구

주요 특징

- 모듈화
- 확장성
- 유연성

과제

데이터셋_분할

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

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

- train_images, train_labels:

학습용 데이터와 정답 레이블

- test_images, test_labels:

테스트용 데이터와 정답 레이블

과제

데이터셋_정규화

```
train_images = train_images / 255.0
test_images = test_images / 255.0

train_labels = to_categorical(train_labels, 10)
test_labels = to_categorical(test_labels, 10)
```

- train_images, test_images 255로 나누어 데이터를 0~1 범위로 정규화

- train_labels, test_labels

One-hot 인코딩:

숫자 레이블(0~9)을 one-hot 벡터로 변환

***One-hot 인코딩:** 범주형 데이터를 숫자로 변환하는 방법
각 카테고리를 고유한 벡터로 표현하며, 모델이 데이터를 이해하고 처리할 수 있도록 하기 위해 사용

과제

모델 구조

```
model = Sequential([
    Flatten(input_shape=(28, 28)),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
```

- Flatten:입력 이미지를 1D벡터로 펼침

- Dense:

은닉층: 128개의 뉴런, ReLU 활성화

출력층: 10개의 노드와 softmax 활성화

모델 컴파일

```
model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

- **옵티마이저:** adam
- **손실 함수:** categorical_crossentropy
- **평가지표:** accuracy

훈련

```
model.fit(
    train_images, train_labels,
    epochs=5,
    batch_size=32,
)
```

```
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 780us/step - accuracy: 0.8750 - loss: 0.4379
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 814us/step - accuracy: 0.9633 - loss: 0.1252
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 749us/step - accuracy: 0.9757 - loss: 0.0815
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 773us/step - accuracy: 0.9831 - loss: 0.0561
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 815us/step - accuracy: 0.9862 - loss: 0.0455
<keras.src.callbacks.history.History at 0x2b07291e000>
```

- fit: 모델 학습
- epochs:

  데이터셋을 5번 반복 학습
- Batch_size:

  데이터를 32개씩 나누어 학습

성능 평가

```
test_loss, test_accuracy = model.evaluate(test_images, test_labels)
print(f"\nTest Accuracy: {test_accuracy:.4f}")
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 602us/step - accuracy: 0.9741 - loss: 0.0853

Test Accuracy: 0.9772
```

- evaluate: 테스트 데이터로 모델 성능 평가
  정확도와 손실 값을 반환

성능 평가

```
import numpy as np
predictions = model.predict(test_images)
predicted_label = np.argmax(predictions[0])
actual_label = np.argmax(test_labels[0])
print(f"Predicted Label: {predicted_label}, Actual Label: {actual_label}")
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 447us/step
Predicted Label: 7, Actual Label: 7
```

- predict: 모델이 입력 데이터에 대해 예측한 결과 반환
- np.argmax: 확률이 가장 높은 인덱스 반환