AI 프로그래밍

- 2024

9주차

- **5월 6일 수업 => 비 대면 동영상 수업**
- **출석 처리 기준**
  - 동영상 끝까지 시청
  - 다음 수업 전날 까지 과제 제출

- **5월 8일 (수요일)**
  - 성공 취업 선배 특강

오늘 수업

- 케라스 문법
- 딥러닝에 필요한 개념 들

Sequential model 을 생성

```
model=keras.Sequential()
```

Layer 추가

```
model.add(layers.Dense(units=2,activation='sigmoid',input_shape=(2,)))
model.add(layers.Dense(units=1,activation='sigmoid'))
```

학습시 option 들 결정

```
model.compile(optimizer=keras.optimizers.SGD(learning_rate=0.7),loss='mse')
model.fit(X,y,epochs=300,batch_size=1)
```

학습

```
model.summary()

print(model.predict(X))
```

예측

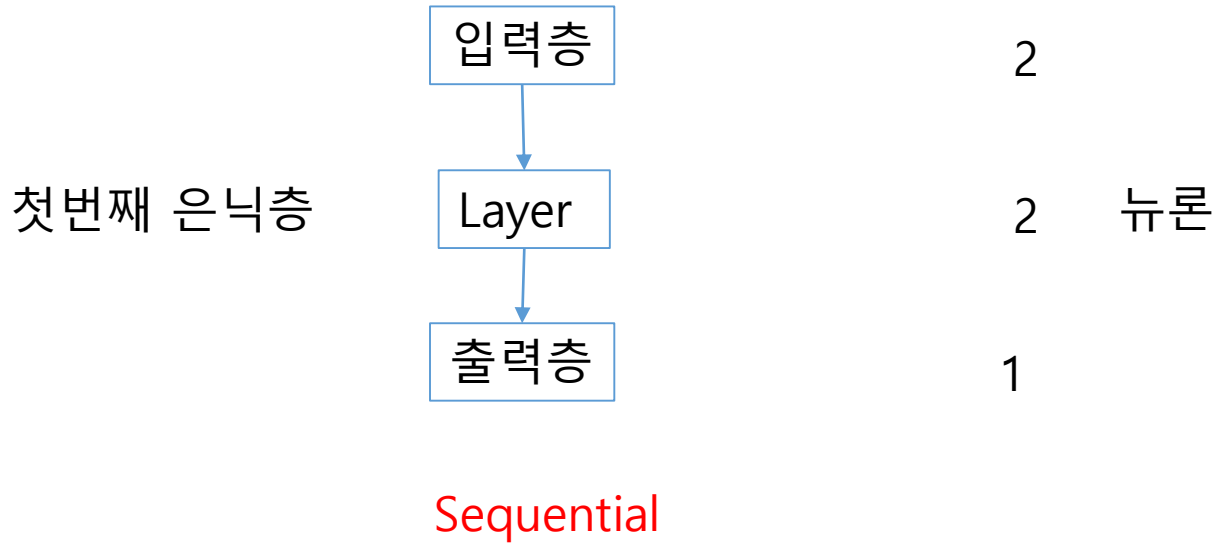
Sequential 모델

- compile(optimizer, loss=None, metrics=None): 훈련을 위해서 모델을 구성하는 메소드
- fit(x=None, y=None, batch_size=None, epochs=1, verbose=1): 훈련 메소드
- evaluate(x=None, y=None): 테스트 모드에서 모델의 손실 함수 값과 측정 항목 값을 반환
- predict(x, batch_size=None): 입력 샘플에 대한 예측값을 생성
- add(layer): 레이어를 모델에 추가한다.

케라스를 사용하는 방법

(1) Sequential 모델을 만들고 모델에 필요한 레이어를 1추가하는 방법이다.

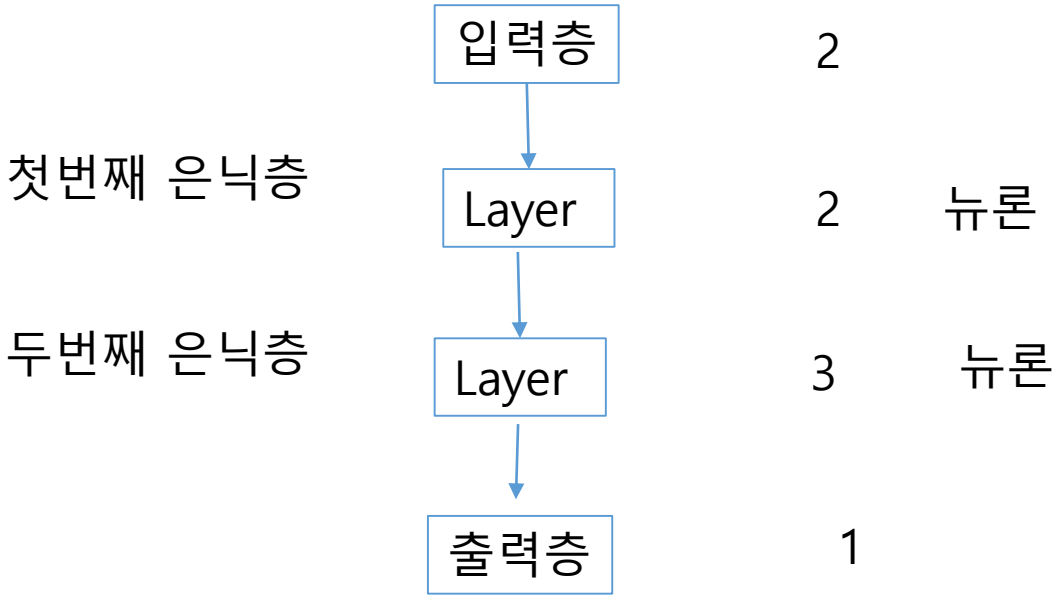
```
model = Sequential()

model.add(Dense(units=2, input_shape=(2,), activation='sigmoid'))
model.add(Dense(units=1,  activation='sigmoid'))
```



케라스를 사용하는 방법

(1) Sequential 모델을 만들고 모델에 필요한 레이어를 추가하는 방법이다.

```
model = Sequential()

model.add(Dense(units=2, input_shape=(2,), activation='sigmoid'))
 model.add(Dense(units=3, activation='sigmoid'))
model.add(Dense(units=1,  activation='sigmoid'))
```



케라스의 클래스들

- 모델: 하나의 신경망을 나타낸다.
- 레이어: 신경망에서 하나의 층이다.
- 입력 데이터: 텐서플로우 텐서 형식이다.
- 손실 함수: 신경망의 출력과 정답 레이블 간의 차이를 측정하는 함수이다.
- 옵티마이저: 학습을 수행하는 최적화 알고리즘이다. 학습률과 모멘텀을 동적으로 변경한다

레이어 클래스들

- **Input(shape, batch_size, name): 입력을 받아서 케라스 텐서를 생성하는 객체**
- **Dense(units, activation=None, use_bias=True, input_shape): 유닛들이 전부 연결된 레이어**

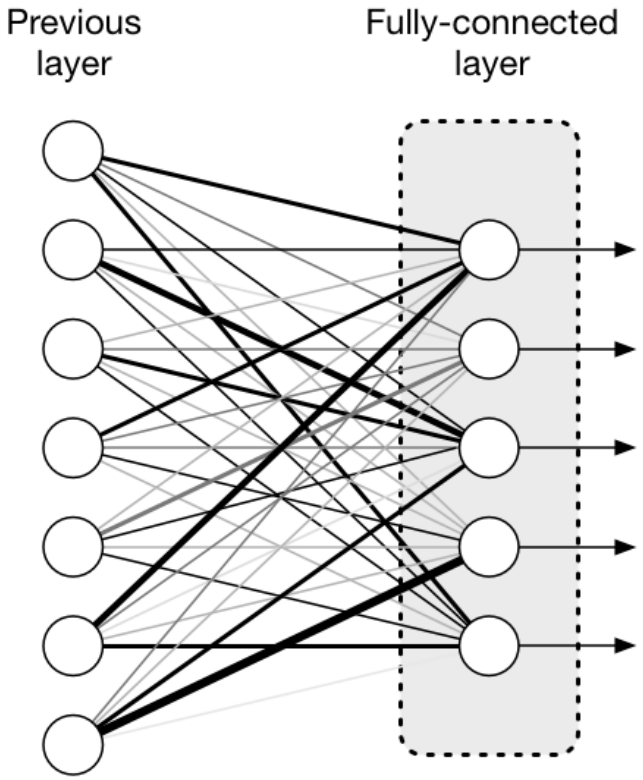
케라스 신경망 파라미터

인하공전 컴퓨터 정보공학과

tf.keras.layers.**Dense**

완전 연결 계층 : (Fully Connected Layer, Densely connected layer)

한 층 (layer)의 모든 뉴런이 그 다음 층의 모든 뉴런과 연결된 상태



https://keras.io/api/layers/core_layers/

케라스 신경망 파라미터 –loss (손실함수)

학습 하는 과정에서 오차를 측정 하는 방법

몸무게 예측 : 67, 53
몸무게 정담: 63, 50

((67-63)^2+(53-50)^2)/1

- **회귀 (regression)**
  - **MeanSquaredError**: 정답 레이블과 예측값 사이의 평균 제곱 오차를 계산한다.
  - 분류에 사용 되기도 함.

분류 예측 (강아지, 강아지 고양이)
정답 (강아지, 고양이, 고양이)

- **분류(classification)**
  - **BinaryCrossentropy**: 정답 레이블과 예측 레이블 간의 교차 엔트로피 손실을 계산한다 (예를 들어서 강아지, 강아지 아님).
  - **CategoricalCrossentropy**: 정답 레이블과 예측 레이블 간의 교차 엔트로피 손실을 계산한다(예를 들어서 강아지, 고양이, 호랑이). 정답 레이블은 원핫 인코딩으로 제공되어야 한다.
  - **SparseCategoricalCrossentropy**: 정답 레이블과 예측 레이블 간의 교차 엔트로피 손실을. 계산한다 (예를 들어서 강아지, 고양이, 호랑이). 정답 레이블은 정수로 제공되어야 한다.

참고자료 : 딥러닝 express

평균 제곱 오차 (Mean Squared Error: MSE)

$$\text{평균 제곱 오차(MSE)} = \frac{1}{n}\sum (\hat{y}_i - y_i)^2$$

$\hat{y}_i$: 예측값

y_i: 정답

y_i: 정답

$\hat{y}_i$: 예측값

공부한 시간	2	4	6	8
성적	80	84	92	94
예측 값	70	80	90	100

$$\text{MSE}=1/4\ (\ (70-80)^2+(80-84)^2+(90-92)^2+(100-94)^2\ )$$

$$= \tfrac{1}{4}(100+16+4+36)=1/4(156)= 39$$

케라스 신경망 파라미터 –loss (손실함수)

인하공전 컴퓨터 정보공학과

회귀 (regression)문제: 평균 제곱 계열
    Mean Squared Error, Mean Absolute Error 등
분류 (classification) 문제: 교차 엔트로피, 평균 제곱 계열도 가능
    Binary_crossentropy, categorical_crossentropy

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - x|$$

One hot encoding

인하공전 컴퓨터 정보공학과

- **3개의 class 강아지, 고양이, 호랑이 일 때**

한 개의 1 과 복수개의 0으로 이루어진 벡터로 표현

- **강아지 (0) => 1, 0 , 0**
- **고양이 (1) => 0, 1, 0**
- **호랑이 (2)=> 0, 0, 1**

참고자료 : 딥러닝 express

Loss function (손실 함수 실습)


```
from tensorflow import keras

y_true = [20, 30, 45, 50 , 53]
y_pred = [25, 26, 47, 52,  57 ]
mse=keras.losses.mean_squared_error
print(mse(y_true, y_pred).numpy())
```

13

```
from tensorflow import keras

y_true = [0, 1, 0, 0]
y_pred = [0.91, 0.1, 0.2, 0.2]
bce = keras.losses.BinaryCrossentropy()
print(bce(y_true, y_pred).numpy())
```

1.2892041

```
from tensorflow import keras

y_true = [[0, 1, 0], [0, 0, 1]]
y_pred = [[0.1, 0.9, 0], [0.1, 0.8, 0.1]]

cce = keras.losses.CategoricalCrossentropy()
print(cce(y_true, y_pred).numpy())
```

1.2039728

Loss function (손실 함수 실습)


```
from tensorflow import keras

y_true = [1, 2 ]
y_pred = [[0.1, 0.9, 0], [0.1, 0.1, 0.8]]

sce = keras.losses.SparseCategoricalCrossentropy()
print(cce(y_true, y_pred).numpy())
```

```
0.1642521
```

Loss function (손실 함수 실습)

- **회귀 (regression)**

1)

공부한 시간	2	4	6	8
성적	80	84	92	94
예측 값	70	80	90	100

mean_squared_error

- **이진 분류 (binary classification)**

BinaryCrossentropy

2)

강아지 사진 ?	사진 1	사진 2	사진3	사진4
정답	1	0	0	1
예측 값	0.7	0.2	0.3	0.9

3)

강아지 사진 ?	사진 1	사진 2	사진3	사진4
정답	1	0	0	1
예측 값	0.6	0.3	0.4	0.7

Loss function (손실 함수 실습)

- **다중 분류 (multiclass classification)**

-강아지, 고양이, 호랑이 사진 분류

- 강아지 : 1, 0, 0
- 고양이 : 0,1, 0
- 호랑이: 0, 0, 1

4)

사진 분류	사진 1	사진 2	사진3
정답	1,0,0	0,1,0	1,0,0
예측 값	0.6,0.3,0.1	0.3, 0.6, 0.1	0.7, 0.1, 0.2

5)

사진 분류	사진 1	사진 2	사진3
정답	1,0,0	0,1,0	1,0,0
예측 값	0.9,0.0,0.1	0.2, 0.7, 0.1	0.8, 0.1, 0.1

Loss function (손실 함수 실습)

- **다중 분류 (multiclass classification)**

-강아지, 고양이, 호랑이 사진 분류

- 강아지 : 1, 0, 0
- 고양이 : 0,1, 0
- 호랑이: 0, 0, 1

6)

사진 분류	사진 1	사진 2	사진3
정답	0	1	0
예측 값	0.6,0.3,0.1	0.3, 0.6, 0.1	0.7, 0.1, 0.2

7)

사진 분류	사진 1	사진 2	사진3
정답	0	1	0
예측 값	0.9,0.0,0.1	0.2, 0.7, 0.1	0.8, 0.1, 0.1

과제 1) 16,17p code를 이용하여 1)~7) 까지의 loss (오차) 값을 구하시오
결과값 입력

Sequential model 을 생성

```
model=keras.Sequential()
```

Layer 추가

```
model.add(layers.Dense(units=2,activation='sigmoid',input_shape=(2,)))
model.add(layers.Dense(units=1,activation='sigmoid'))
```

학습시 option 들 결정

```
model.compile(optimizer=keras.optimizers.SGD(learning_rate=0.7),loss='mse')
model.fit(X,y,epochs=300,batch_size=1)
```

학습

```
model.summary()

print(model.predict(X))
```

예측

고급 경사 하강법

경사 하강법 (**GD** : Gradient Descent) : 정확 하지만 한번 업데이트 할 때마다 전체 데이터를 미분 해야 함. 속도 문제

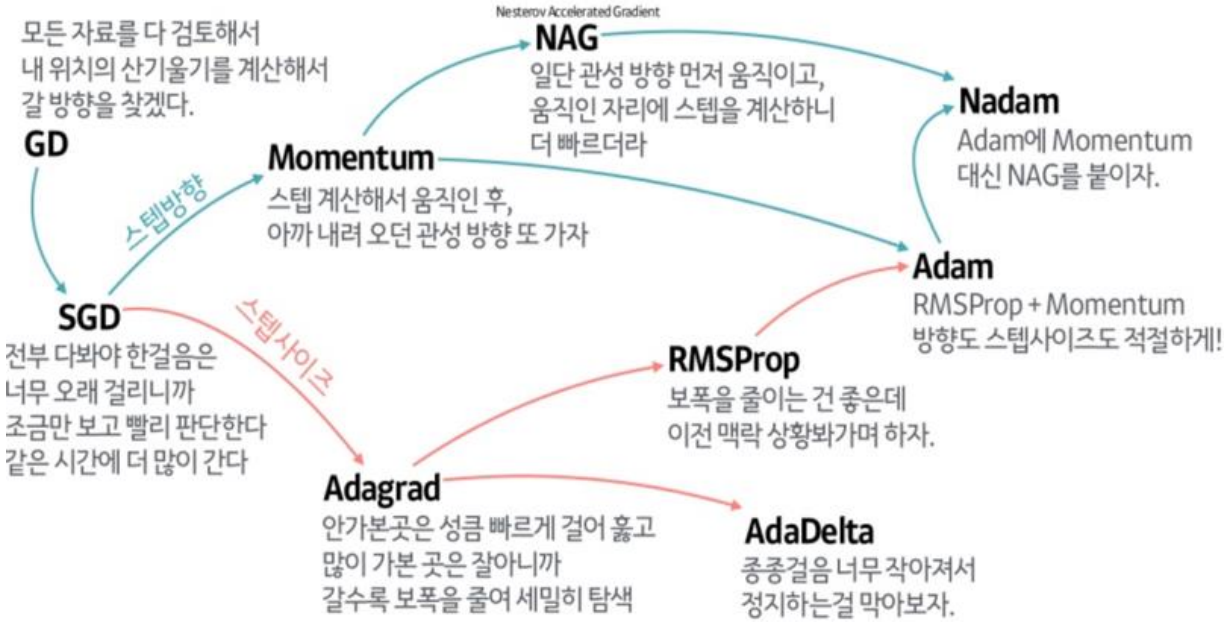
➡ 확률적 경사 하강법 (**SGD** : Stochastic Gradient Descent)
전체 데이터를 사용하지 않고 랜덤하게 추출한 일부 데이터를 사용
일부 데이터를 사용하기 때문에 진폭이 크고 불안정 할수 있음.
속도 개선 효과

고급 경사 하강법

인하공전 컴퓨터 정보공학과

- 네스테로프 모멘텀 (**NAG**) : 모멘텀 개선. 불필요한 계산을 줄임.
- 아다그라드 (**Adagrad**) : 가변 학습률 적용. 변수 업데이트가 너무 크면 학습률을 줄여줌.
- 알엠에스프롭 (**RMSprop**): Adagrad 개선. 학습률이 너무 작아지는것을 방지
- 아담 (**Adam**: Adaptive Moment Estimation)
  : RMSprop+ Momentum
  : **현재 가장 인기 있는 최적화 방법**.

주로 Adam을 사용



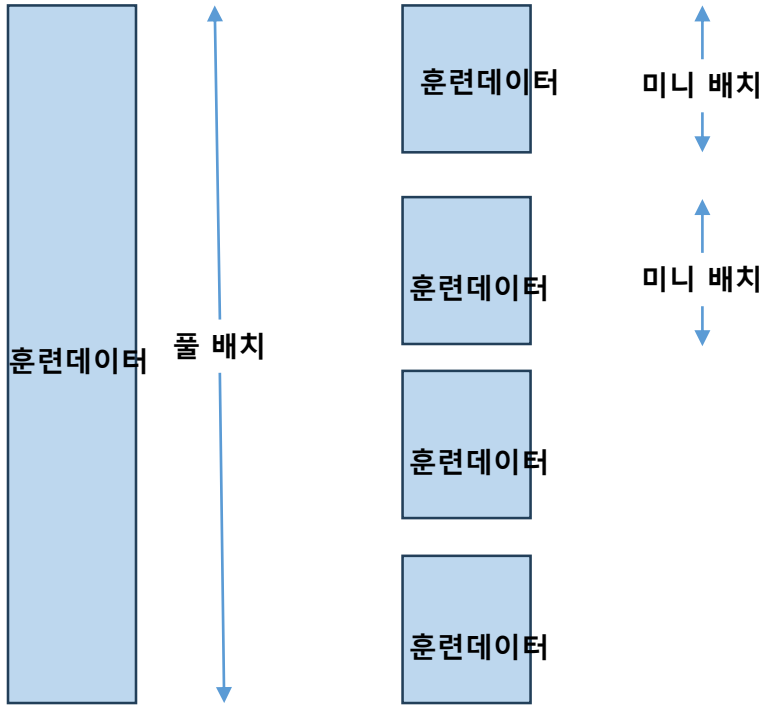
경사 하강법 비교 1

https://cs231n.github.io/assets/nn3/opt2.gif

경사 하강법 비교 2

https://cs231n.github.io/assets/nn3/opt1.gif

각 방법들의 비교



풀배치 : 모든 훈련데이터마다 기울기를 구한 후 평균을 계산 해서 최종 기울기를 구함.

참고자료 : 딥러닝 express

케라스 신경망 파라미터 –batch_size

- batch_size
  - 샘플을 몇 개 사용하여 가중치를 변경할지 단위
  - batch_size가 너무 크면 학습 속도가 느려지고,
  - 너무 작으면 학습과정이 불안정 할 수가 있음.
  - 변수 (w,b)가 업데이트 되는 단위.

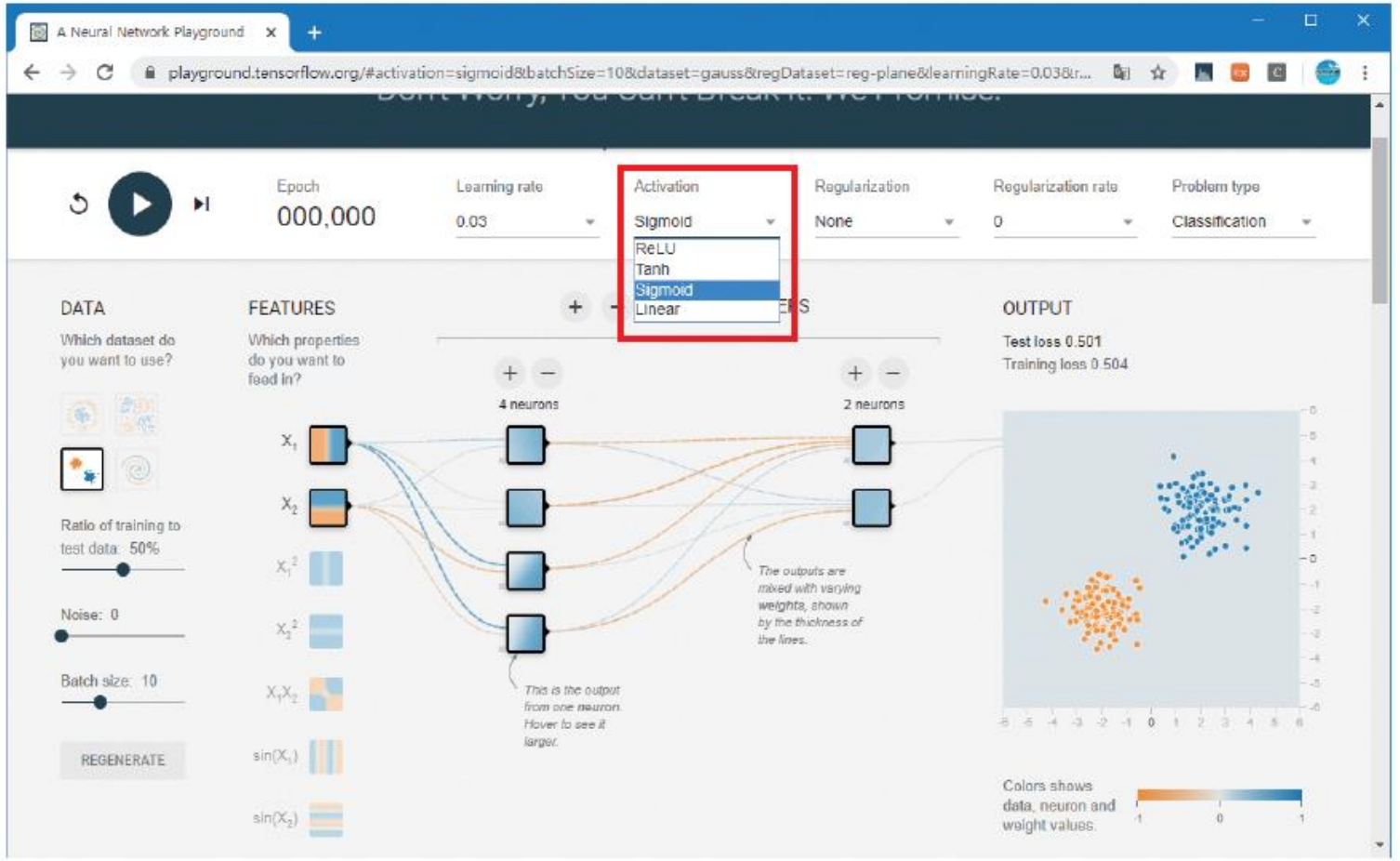
1 epoch는 전체 샘플이 처리되는 기준

전체 샘플이 이 300 개이고 batch size가 30이면 1 epoch동안 10번 가중치가 update됨.
전체 샘플이 이 300 개이고 batch size가 10이면 1 epoch동안 30번 가중치가 update됨.

배치크기 실습

https://playground.tensorflow.org/

Batch size를
1, 10, 30 으로 변경하고 학습



배치 사이즈 선택

- 메모리 사이즈
- 속도
- 학습 안정성 고려

Metric : 측정 항목

- **Accuracy: 정확도이다. 예측값이 정답 레이블과 같은 횟수를 계산한다.**
  - 100개 데이터 중에서 몇 개를 맞췄는지?
- **MeanSquaredError: MSE**

케라스의 입력 데이터

- 넘파이 배열:
- TensorFlow Dataset 객체: 크기가 커서, 메모리에 한 번에 적재될 수 없는 경우에 디스크 또는 분산 파일 시스템에서 스트리밍될 수 있다.
- 파이썬 제너레이터: 예를 들어서 keras.utils.Sequence 클래스는 하드 디스크에 위치한 파일을 읽어서 순차적으로 케라스 모델로 공급할 수 있다.

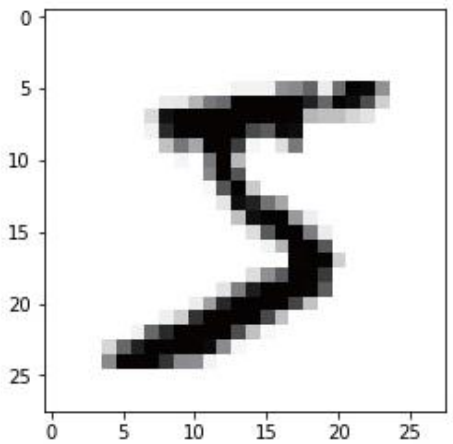
MNIST 분류

숫자 데이터 가져오기


```
import matplotlib.pyplot as plt
import tensorflow as tf

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

숫자 데이터 표시하기


```
>>> train_images.shape
(60000, 28, 28)

>>> train_label
array([5, 0, 4, ..., 5, 6, 8], dtype=uint8)

>>> test_images.shape
(10000, 28, 28)

>>> plt.imshow(train_images[0], cmap="Greys")
```



신경망 모델 구축하기


```
model = tf.keras.models.Sequential()

model.add(tf.keras.layers.Dense(512, activation='relu', input_shape=(784,)))
model.add(tf.keras.layers.Dense(10, activation='sigmoid'))
```

옵티마이저와 손실함수, 지표 등을 정의하는 컴파일 단계


```
model.compile(optimizer='rmsprop',
              loss='mse',
              metrics=['accuracy'])
```

데이터 전처리


```
train_images = train_images.reshape((60000, 784))
train_images = train_images.astype('float32') / 255.0

test_images = test_images.reshape((10000, 784))
test_images = test_images.astype('float32') / 255.0
```

정답 레이블 형태 변경(원핫 인코딩)


```
train_labels = tf.keras.utils.to_categorical(train_labels)
test_labels = tf.keras.utils.to_categorical(test_labels)
```

학습


```
model.fit(train_images, train_labels, epochs=5, batch_size=128)
```

테스트


```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('테스트 정확도:', test_acc)
```

```
313/313 [==============================] - 0s 892us/step - loss: 0.0039 -
accuracy: 0.9788
테스트 정확도: 0.9787999987602234
```

하이퍼 매개변수

- 학습률 은닉층을 몇 개로 할 것이며, 은닉층의 개수나 유닛,배치 크기

과제 2

mnist_keras.ipynb

1. 유닛 수 변경 test
(batch size=128, 은닉층 활성화 함수=Relu)

	유닛 수	Accuracy (정확도)
Test1	512	
test2	256	
test3	64	

2. 배치 크기 test (유닛 수 =512, 은닉층 활성화 함수=Relu)

	batch size	Accuracy (정확도)
Test1	512	
test2	256	
test3	64	

3. 은닉층 활성화 함수 test (batch size=128, 유닛 수 = 512)

	batch size	Accuracy (정확도)
Test1	512	
test2	256	
test3	64	

폐암 환자 생존율 실습

17개의 속성이 있을때 생존 여부를 예측 하는 모델 생성

폐암 환자 생존율 실습


```
# 파일 선택을 통해 예제 데이터를 내 컴퓨터에서 불러옵
니다.
from google.colab import files
uploaded = files.upload()
my_data = 'ThoracicSurgery.csv'

# 딥러닝을 구동하는 데 필요한 케라스 함수를 불러옵니
다.
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# 필요한 라이브러리를 불러옵니다.
import numpy as np
import tensorflow as tf
```

폐암 환자 생존율 실습


```
# 불러온 데이터를 적용합니다.
Data_set = np.loadtxt(my_data, delimiter=",")

# 환자의 기록과 수술 결과를 X와 Y로 구분하여 저장합니다.
X = Data_set[:,0:17]
Y = Data_set[:,17]

# 딥러닝 구조를 결정합니다(모델을 설정하고 실행하는 부분입니다).
model = Sequential()
model.add(Dense(30, input_dim=17, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# 딥러닝을 실행합니다.
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
model.fit(X, Y, epochs=100, batch_size=10)
```

범주형 데이터 처리

- 입력 데이터 에 카테고리(문자열) 를 가지는 데이터들이 있을 때 숫자로 바꿔야함.

```
for ix in train.index:
  if train.loc[ix, 'Sex']=="male":
          train.loc[ix, 'Sex']=1
  else:
          train.loc[ix, 'Sex']=0
```

정수 인코딩: 정수로 변환
원 핫 인코딩: 이진 벡터로 변환

정수 인코딩

- sklearn 라이브러리가 제공하는 Label Encoder 클래스를 사용

```
import numpy as np
X = np.array([['Korea', 44, 7200],
              ['Japan', 27, 4800],
              ['China', 30, 6100]])

from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
X[:, 0] = labelencoder.fit_transform(X[:, 0])
print(X)
```

```
[['2' '44' '7200']
['1' '27' '4800']
['0' '30' '6100']]
```

```
import numpy as np
X = np.array([['Korea', 44, 7200],
              ['Japan', 27, 4800],
              ['China', 30, 6100]])

from sklearn.preprocessing import OneHotEncoder
onehotencoder = OneHotEncoder()

# 원하는 열을 뽑아서 2차원 배열로 만들어서 전달하여야 한다.
XX = onehotencoder.fit_transform(X[:,0].reshape(-1,1)).toarray()
print(XX)

X = np.delete(X, [0], axis=1)              # 0번째 열 삭제
X = np.concatenate((XX, X), axis = 1)  # X와 XX를 붙인다.
print(X)
```

```
[[0. 0. 1.]
 [0. 1. 0.]
 [1. 0. 0.]]
[['0.0' '0.0' '1.0' '44' '7200']
 ['0.0' '1.0' '0.0' '27' '4800']
 ['1.0' '0.0' '0.0' '30' '6100']]
```

정수형 인코딩


```
import numpy as np
X = np.array(['Korea','Japan','China'])

from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
X = labelencoder.fit_transform(X)
print(X)
```

```
[2 1 0]
```

원 핫 인코딩


```
import numpy as np
X = np.array(['Korea','Japan','China'])

from sklearn.preprocessing import OneHotEncoder
onehotencoder = OneHotEncoder()
XX = onehotencoder.fit_transform(X.reshape(-1,1)).toarray()
print(XX)
```

```
[[0. 0. 1.]
 [0. 1. 0.]
 [1. 0. 0.]]
```

원-핫 인코딩

- 케라스의 to_categorical()

```
class_vector =[2, 6, 6, 1]

from tensorflow.keras.utils import to_categorical
output = to_categorical(class_vector, num_classes = 7, dtype ="int32")
print(output)
```

```
[[0 0 1 0 0 0 0]
 [0 0 0 0 0 0 1]
 [0 0 0 0 0 0 1]
 [0 1 0 0 0 0 0]]
```

Y_obj=

```
array(['Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-
setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-
setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-
setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-
setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-
setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-
setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-
setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-
setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-
setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-
setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-
setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-
setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor',
'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
'Iris-versicolor',
```

```
from sklearn.preprocessing import LabelEncoder

e = LabelEncoder()
e.fit(Y_obj)
Y = e.transform(Y_obj)
```

정수 인코딩

Y=

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2])
```

```
from tensorflow.keras.utils import np_utils

Y_encoded = tf.keras.utils.to_categorical(Y)
```

원 핫 인코딩

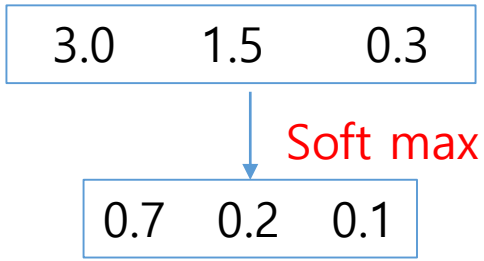
Y_encoded

```
array([[1., 0., 0.], [1., 0., 0.], [1., 0., 0.], [1.,
0., 0.], [1., 0., 0.], [1., 0., 0.], [1., 0., 0.],
[1., 0., 0.], [1., 0., 0.],
```

케라스 딥러닝 – 소프트 맥스

- 총합이 1인 형태로 변환

  그림 12-3에서와 같이 총합이 1인 형태로 바꿔서 계산해 주는 함수



출력

```
Model.add(Dense(10,input_dim=4, activate='relu'))
Model.add(Dense(3,activation='softmax')
```

1.0 0.0 0.0 (one hot label) 정답

과제 3)

- **정수형 인코딩, 원 핫 인코딩을 앞의 code를 이용하여**

구현 하시오.

code: category_exe.ipynb

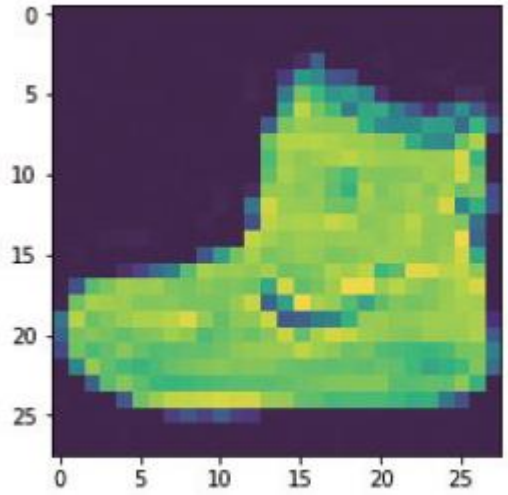
케라스 신경망 실습 – 패션 아이템 분류

- 이미지는 28x28 크기이고
- 픽셀 값은 0과 255 사이
- 레이블(label)은 0에서 9까지의

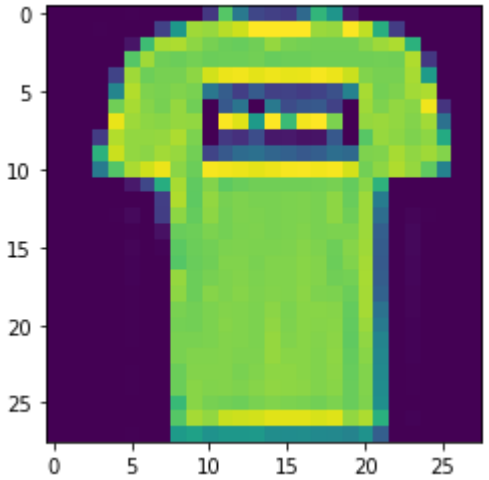
레이블	범주
0	T-shirt/top
1	trouser
2	pullover
3	dress
4	coat
5	sandal
6	shirt
7	sneaker
8	bag
9	Ankle boot

케라스 신경망 실습 – 패션 아이템 분류

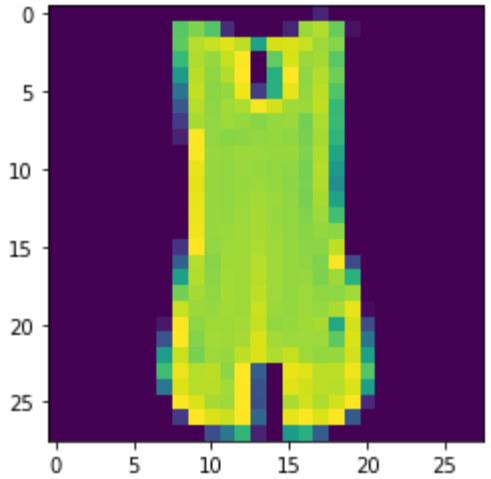
- **plt.imshow(train_images[0])**

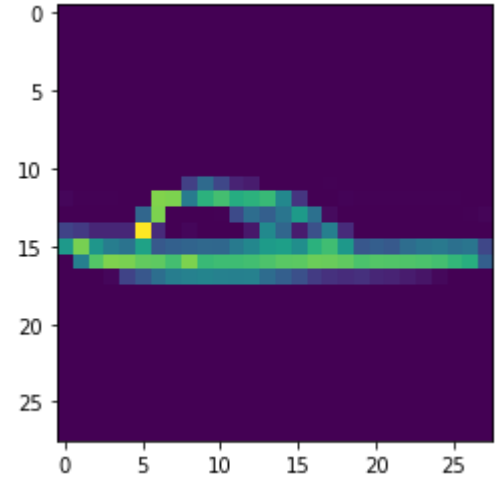


두번째



네번째





28x28 gray image

케라스 신경망 실습 – 패션 아이템 분류


```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras import datasets, layers, models

fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

plt.imshow(train_images[0])

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

케라스 신경망 실습 – 패션 아이템 분류


```
model = models.Sequential()
model.add(layers.Flatten(input_shape=(28, 28)))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

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

model.fit(train_images, train_labels, epochs=5)

test_loss, test_acc = model.evaluate(test_images, test_labels)
print('정확도:', test_acc)
```

케라스 신경망 실습 – 패션 아이템 분류

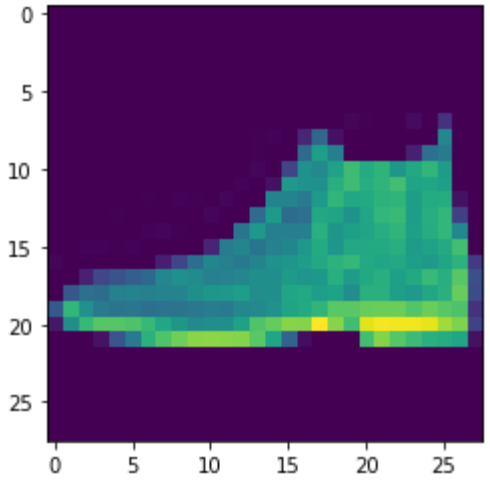

```
model = models.Sequential()
model.add(layers.Flatten(input_shape=(28, 28)))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

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

model.fit(train_images, train_labels, epochs=5)

test_loss, test_acc = model.evaluate(test_images, test_labels)
print('정확도:', test_acc)
```

```
10000/10000 [==============================] - 0s 32us/sample - loss: 0.3560 -
acc: 0.8701
정확도: 0.8701
```

케라스 신경망 실습 – 패션 아이템 분류


```
test_pred=model.predict(test_images)
plt.imshow(test_images[0])
print(np.round(test_pred[0],2))
```

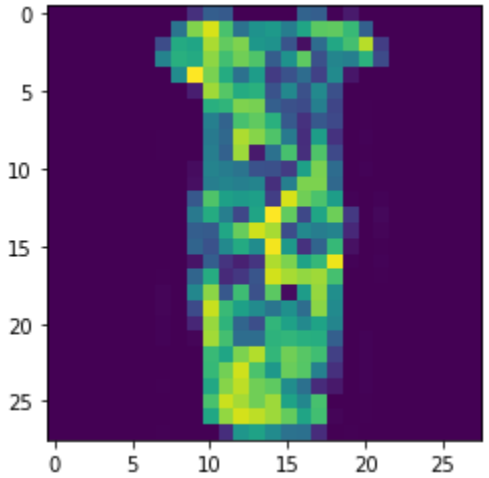
```
[0.  0.  0.  0.  0.  0.  0.  0.04  0.  0.96]
```



케라스 신경망 실습 – 패션 아이템 분류


```
plt.imshow(test_images[100])
print(np.round(test_pred[100],2))
```

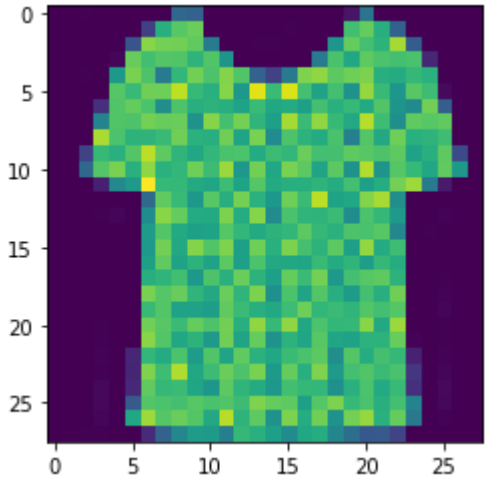
```
[0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
```



케라스 신경망 실습 – 패션 아이템 분류


```
plt.imshow(test_images[1000])
print(np.round(test_pred[1000],2))
```

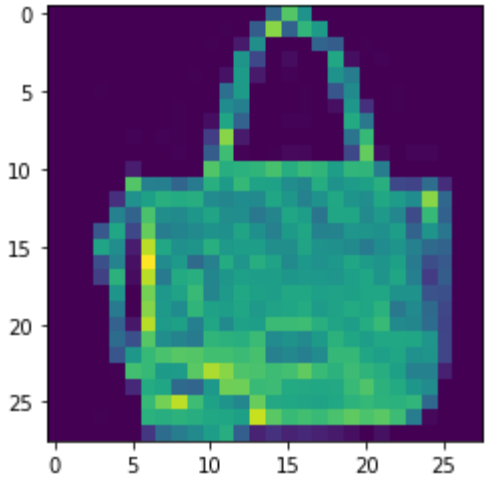
```
[0.43 0.   0.05 0.02 0.   0.   0.51 0.   0.   0.  ]
```



케라스 신경망 실습 – 패션 아이템 분류


```
plt.imshow(test_images[2000])
print(np.round(test_pred[2000],2))
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
```



케라스 신경망 실습 – 패션 아이템 분류

인하공전 컴퓨터 정보공학과

과제 4 fasion_org_exam.ipynb

모델을 다음과 같이 변경 하고 code (모델 생성 부분만)과 정확도를 제출 하시오.

1)

입력 층 : 변동 없음
은닉층 1 : 32 (relu)
출력 층 변동 없음.

2)

입력 층 : 변동 없음
은닉층 1 : 64 (relu)
은닉측 2 : 16 (relu)
출력 층 변동 없음.

3) 1050번째 test이미지의 사진을 붙이고 카테고리 예측 결과를 제출 하시오
4) 615번째 test 이미지와 사진을 붙이고 카테고리 예측 결과를 제출 하시오.

수고하셨습니다

jhmin@inhatc.ac.kr