5월12일 수업중코드

20174119 한지훈

import numpy as np

import tensorflow as tf

# 데이터를 학습 데이터와 테스트 데이터로 나눈다.

(x\_train, y\_train), (x\_test, y\_test) = tf.keras.datasets.mnist.load\_data()

data\_size = x\_train.shape[0]

batch\_size = 12 # 배치 크기

selected = np.random.choice(data\_size, batch\_size)

print(selected)

x\_batch = x\_train[selected]

y\_batch = y\_train[selected]

[11866 25726 6295 39697 11062 1367 39355 37580 41845 57300 22893 12961]

import numpy as np

# 시그모이드 함수

def actf(x):

return 1/(1+np.exp(-x))

# 시그모이드 함수의 미분치

def actf\_deriv(x):

return x\*(1-x)

# 입력유닛의 개수, 은닉유닛의 개수, 출력유닛의 개수

inputs, hiddens, outputs = 2, 2, 1

learning\_rate = 0.2

# 훈련 입력과 출력

X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])

T = np.array([[0], [1], [1], [0]])

# 가중치를 –1.0에서 1.0 사이의 난수로 초기화한다.

W1 = 2\*np.random.random((inputs, hiddens))-1

W2 = 2\*np.random.random((hiddens, outputs))-1

B1 = np.zeros(hiddens)

B2 = np.zeros(outputs)

def predict(x):

layer0 = x # 입력을 layer0에 대입한다.

Z1 = np.dot(layer0, W1)+B1 # 행렬의 곱을 계산한다.

layer1 = actf(Z1) # 활성화 함수를 적용한다.

Z2 = np.dot(layer1, W2)+B2 # 행렬의 곱을 계산한다.

layer2 = actf(Z2) # 활성화 함수를 적용한다.

return layer0, layer1, layer2

# 역방향 전파 계산

def fit():

global W1, W2, B1, B2

for i in range(60000):

layer0, layer1, layer2 = predict(X)

layer2\_error = layer2-T

layer2\_delta = layer2\_error\*actf\_deriv(layer2)

layer1\_error = np.dot(layer2\_delta, W2.T)

layer1\_delta = layer1\_error\*actf\_deriv(layer1)

W2 += -learning\_rate\*np.dot(layer1.T, layer2\_delta)/4.0

W1 += -learning\_rate\*np.dot(layer0.T, layer1\_delta)/4.0

B2 += -learning\_rate\*np.sum(layer2\_delta, axis=0)/4.0

B1 += -learning\_rate\*np.sum(layer1\_delta, axis=0)/4.0

def test():

for x, y in zip(X, T):

x = np.reshape(x, (1, -1)) # 하나여도 2차원 형태이어야 한다.

layer0, layer1, layer2 = predict(x)

print(x, y, layer2)

fit()

test()

[[0 0]] [0] [[0.02695735]]

[[0 1]] [1] [[0.97679547]]

[[1 0]] [1] [[0.9767719]]

[[1 1]] [0] [[0.0240363]]

import matplotlib.pyplot as plt

import tensorflow as tf

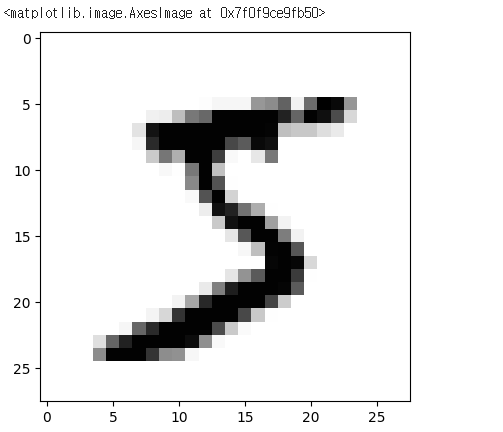
(train\_images, train\_labels), (test\_images, test\_labels)= tf.keras.datasets.mnist.load\_data()

train\_images.shape

train\_labels

test\_images.shape

plt.imshow(train\_images[0], cmap="Greys")



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'))

model.compile(optimizer='rmsprop',

loss='mse',

metrics=['accuracy'])

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

train\_labels = tf.keras.utils.to\_categorical(train\_labels)

test\_labels = tf.keras.utils.to\_categorical(test\_labels)

model.fit(train\_images, train\_labels, epochs=5, batch\_size=128)

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)

print('테스트 정확도:', test\_acc)

Epoch 1/5 469/469 [==============================] - 8s 14ms/step - loss: 0.0250 - accuracy: 0.8743 Epoch 2/5 469/469 [==============================] - 5s 11ms/step - loss: 0.0127 - accuracy: 0.9321

Epoch 3/5 469/469 [==============================] - 5s 10ms/step - loss: 0.0100 - accuracy: 0.9463 Epoch 4/5 469/469 [==============================] - 6s 13ms/step - loss: 0.0084 - accuracy: 0.9554

Epoch 5/5 469/469 [==============================] - 5s 10ms/step - loss: 0.0073 - accuracy: 0.9616 313/313 [==============================] - 1s 3ms/step - loss: 0.0071 - accuracy: 0.9643

테스트 정확도: 0.9642999768257141

history = model.fit(train\_images, train\_labels, epochs=5, batch\_size=128)

loss = history.history['loss']

acc = history.history['accuracy']

epochs = range(1, len(loss)+1)

plt.plot(epochs, loss, 'b', label='Training Loss')

plt.plot(epochs, acc, 'r', label='Accuracy')

plt.xlabel('epochs')

plt.ylabel('loss/acc')

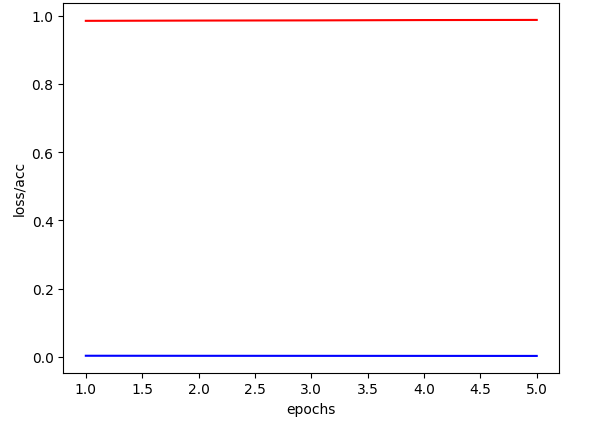
plt.show()

Epoch 1/5 469/469 [==============================] - 6s 12ms/step - loss: 0.0030 - accuracy: 0.9857

Epoch 2/5 469/469 [==============================] - 5s 11ms/step - loss: 0.0028 - accuracy: 0.9865

Epoch 3/5 469/469 [==============================] - 5s 10ms/step - loss: 0.0027 - accuracy: 0.9870 Epoch 4/5 469/469 [==============================] - 6s 12ms/step - loss: 0.0026 - accuracy: 0.9880

Epoch 5/5 469/469 [==============================] - 5s 10ms/step - loss: 0.0024 - accuracy: 0.9884



import cv2 as cv

image = cv.imread('test88.png', cv.IMREAD\_GRAYSCALE)

image = cv.resize(image, (28, 28))

image = image.astype('float32')

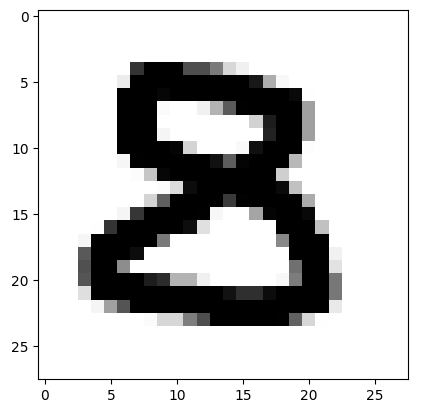
image = image.reshape(1, 784)

image = 255-image

image /= 255.0

plt.imshow(image.reshape(28, 28),cmap='Greys')

plt.show()



pred = model.predict(image.reshape(1, 784), batch\_size=1)

print("추정된 숫자=", pred.argmax())

1/1 [==============================] - 0s 23ms/step

추정된 숫자= 8

연습문제 8번

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.activations import hard\_sigmoid

import numpy as np

X=np.array([[0,0],[0,1],[1,0],[1,1]])

Y=np.array([[1],[0],[0],[1]])

model=Sequential()

model.add(Dense(1,input\_shape=(2,),activation=hard\_sigmoid))

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

model.fit(X,Y,epochs=3000,batch\_size=1,verbose=1)

4/4 [==============================] - 0s 4ms/step - loss: 0.2504 - accuracy: 0.5000  
Epoch 2991/3000  
4/4 [==============================] - 0s 4ms/step - loss: 0.2505 - accuracy: 0.5000  
Epoch 2992/3000  
4/4 [==============================] - 0s 4ms/step - loss: 0.2504 - accuracy: 0.5000  
Epoch 2993/3000  
4/4 [==============================] - 0s 4ms/step - loss: 0.2504 - accuracy: 0.5000  
Epoch 2994/3000  
4/4 [==============================] - 0s 4ms/step - loss: 0.2504 - accuracy: 0.5000  
Epoch 2995/3000  
4/4 [==============================] - 0s 4ms/step - loss: 0.2504 - accuracy: 0.5000  
Epoch 2996/3000  
4/4 [==============================] - 0s 4ms/step - loss: 0.2504 - accuracy: 0.5000  
Epoch 2997/3000  
4/4 [==============================] - 0s 4ms/step - loss: 0.2504 - accuracy: 0.5000  
Epoch 2998/3000  
4/4 [==============================] - 0s 4ms/step - loss: 0.2504 - accuracy: 0.5000  
Epoch 2999/3000  
4/4 [==============================] - 0s 4ms/step - loss: 0.2504 - accuracy: 0.5000  
Epoch 3000/3000  
4/4 [==============================] - 0s 4ms/step - loss: 0.2504 - accuracy: 0.5000

<keras.callbacks.History at 0x7f0f99fde8f0>

연습문제9번

import numpy as np

import tensorflow as tf

# 데이터를 학습 데이터와 테스트 데이터로 나눈다.

(train\_images, train\_labels), (test\_images, test\_labels)= tf.keras.datasets.mnist.load\_data()

model = tf.keras.models.Sequential()

model.add(tf.keras.layers.Dense(512, activation='relu', input\_shape=(784,)))

model.add(tf.keras.layers.Dense(256, activation='sigmoid'))

model.add(tf.keras.layers.Dense(10, activation='sigmoid'))

model.compile(optimizer='rmsprop',

loss='mse',

metrics=['accuracy'])

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

train\_labels = tf.keras.utils.to\_categorical(train\_labels)

test\_labels = tf.keras.utils.to\_categorical(test\_labels)

model.fit(train\_images, train\_labels, epochs=5, batch\_size=32)

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)

print('테스트 정확도:', test\_acc)

Epoch 1/5 1875/1875 [==============================] - 19s 10ms/step - loss: 0.0224 - accuracy: 0.8741

Epoch 2/5 1875/1875 [==============================] - 18s 10ms/step - loss: 0.0100 - accuracy: 0.9423

Epoch 3/5 1875/1875 [==============================] - 22s 12ms/step - loss: 0.0074 - accuracy: 0.9562

Epoch 4/5 1875/1875 [==============================] - 19s 10ms/step - loss: 0.0059 - accuracy: 0.9657 Epoch 5/5 1875/1875 [==============================] - 25s 14ms/step - loss: 0.0049 - accuracy: 0.9726 313/313 [==============================] - 1s 3ms/step - loss: 0.0050 - accuracy: 0.9703

테스트 정확도: 0.970300018787384

import numpy as np

import tensorflow as tf

# 데이터를 학습 데이터와 테스트 데이터로 나눈다.

(train\_images, train\_labels), (test\_images, test\_labels)= tf.keras.datasets.mnist.load\_data()

model = tf.keras.models.Sequential()

model.add(tf.keras.layers.Dense(512, activation='relu', input\_shape=(784,)))

model.add(tf.keras.layers.Dense(256, activation='sigmoid'))

model.add(tf.keras.layers.Dense(10, activation='sigmoid'))

model.compile(optimizer='adam',

loss='mse',

metrics=['accuracy'])

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

train\_labels = tf.keras.utils.to\_categorical(train\_labels)

test\_labels = tf.keras.utils.to\_categorical(test\_labels)

model.fit(train\_images, train\_labels, epochs=5, batch\_size=32)

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)

print('테스트 정확도:', test\_acc)

Epoch 1/5 1875/1875 [==============================] - 21s 11ms/step - loss: 0.0123 - accuracy: 0.9236

Epoch 2/5 1875/1875 [==============================] - 21s 11ms/step - loss: 0.0045 - accuracy: 0.9723

Epoch 3/5 1875/1875 [==============================] - 20s 11ms/step - loss: 0.0031 - accuracy: 0.9814

Epoch 4/5 1875/1875 [==============================] - 20s 11ms/step - loss: 0.0024 - accuracy: 0.9861

Epoch 5/5 1875/1875 [==============================] - 19s 10ms/step - loss: 0.0019 - accuracy: 0.9891 313/313 [==============================] - 2s 5ms/step - loss: 0.0030 - accuracy: 0.9815

테스트 정확도: 0.9815000295639038

import numpy as np

import tensorflow as tf

# 데이터를 학습 데이터와 테스트 데이터로 나눈다.

(train\_images, train\_labels), (test\_images, test\_labels)= tf.keras.datasets.mnist.load\_data()

model = tf.keras.models.Sequential()

model.add(tf.keras.layers.Dense(512, activation='relu', input\_shape=(784,)))

model.add(tf.keras.layers.Dense(256, activation='sigmoid'))

model.add(tf.keras.layers.Dense(10, activation='sigmoid'))

model.compile(optimizer='Adadelta',

loss='mse',

metrics=['accuracy'])

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

train\_labels = tf.keras.utils.to\_categorical(train\_labels)

test\_labels = tf.keras.utils.to\_categorical(test\_labels)

model.fit(train\_images, train\_labels, epochs=5, batch\_size=32)

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)

print('테스트 정확도:', test\_acc)

Epoch 1/5 1875/1875 [==============================] - 24s 12ms/step - loss: 0.2606 - accuracy: 0.0986

Epoch 2/5 1875/1875 [==============================] - 23s 12ms/step - loss: 0.1553 - accuracy: 0.0986

Epoch 3/5 1875/1875 [==============================] - 22s 12ms/step - loss: 0.1110 - accuracy: 0.1011

Epoch 4/5 1875/1875 [==============================] - 23s 12ms/step - loss: 0.0972 - accuracy: 0.1286

Epoch 5/5 1875/1875 [==============================] - 23s 12ms/step - loss: 0.0927 - accuracy: 0.1878 313/313 [==============================] - 1s 3ms/step - loss: 0.0915 - accuracy: 0.2124

테스트 정확도: 0.21240000426769257

import numpy as np

import tensorflow as tf

# 데이터를 학습 데이터와 테스트 데이터로 나눈다.

(train\_images, train\_labels), (test\_images, test\_labels)= tf.keras.datasets.mnist.load\_data()

model = tf.keras.models.Sequential()

model.add(tf.keras.layers.Dense(512, activation='relu', input\_shape=(784,)))

model.add(tf.keras.layers.Dense(256, activation='sigmoid'))

model.add(tf.keras.layers.Dense(10, activation='sigmoid'))

model.compile(optimizer='Adagrad',

loss='mse',

metrics=['accuracy'])

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

train\_labels = tf.keras.utils.to\_categorical(train\_labels)

test\_labels = tf.keras.utils.to\_categorical(test\_labels)

model.fit(train\_images, train\_labels, epochs=5, batch\_size=32)

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)

print('테스트 정확도:', test\_acc)

Epoch 1/5  
1875/1875 [==============================] - 17s 9ms/step - loss: 0.1117 - accuracy: 0.1133  
Epoch 2/5  
1875/1875 [==============================] - 16s 8ms/step - loss: 0.0903 - accuracy: 0.1686  
Epoch 3/5  
1875/1875 [==============================] - 16s 8ms/step - loss: 0.0894 - accuracy: 0.2322  
Epoch 4/5  
1875/1875 [==============================] - 17s 9ms/step - loss: 0.0891 - accuracy: 0.2910  
Epoch 5/5  
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0888 - accuracy: 0.3134  
313/313 [==============================] - 1s 3ms/step - loss: 0.0886 - accuracy: 0.3236  
테스트 정확도: 0.32359999418258667