|  |  |
| --- | --- |
| 교육 제목 |  |
| 교육 일시 | 2021. 10. 27 |
| 교육 장소 | 영우글로벌러닝 |
| **교육 내용** | |
| 오전 | # 필요한 라이브러리를 불러옵니다.  import pandas as pd  import tensorflow as tf  from sklearn.model\_selection import train\_test\_split  import matplotlib.pyplot as plt  # csv read  data = pd.read\_csv('./ThoraricSurgery.csv')  data.head()  data = data.values  x = data[:,0:17]  y = data[:,17]  type(x)  # 전체 데이터에서 학습데이터와 테스트 데이터 구분  x\_train1, x\_test, y\_train1, y\_test = train\_test\_split(x, y, test\_size=0.2)  x\_train, x\_valid, y\_train, y\_valid = train\_test\_split(x\_train1, y\_train1, test\_size=0.3)  input\_layer = tf.keras.layers.Input(shape=(17,))  x = tf.keras.layers.Dense(10, activation='sigmoid', kernel\_initializer=tf.keras.initializers.he\_normal())(input\_layer) # he 초기화 방법  x = tf.keras.layers.Dense(10, activation='sigmoid', kernel\_initializer=tf.keras.initializers.glorot\_uniform)(x) # xavier초기화 방법  output\_layer = tf.keras.layers.Dense(1, activation='sigmoid')(x)  model = tf.keras.models.Model(inputs=[input\_layer], outputs=[output\_layer])  model.summary()  loss = tf.keras.losses.binary\_crossentropy  optimizer = tf.keras.optimizers.SGD(learning\_rate=0.4)  metrics = tf.keras.metrics.binary\_accuracy  model.compile(loss=loss, optimizer=optimizer, metrics=[metrics])  history = model.fit(x\_train, y\_train, epochs=100, batch\_size=10, validation\_data=(x\_valid, y\_valid))  print(history.history.keys())  ## history에서 loss와 val\_loss의 key를 가지는 값들만 추출  loss = history.history['loss']  val\_loss = history.history['val\_loss']  ### loss와 val\_loss를 그래프화  epochs = range(1, len(loss) + 1)  plt.subplot(211) ## 2x1 개의 그래프 중에 1번째  plt.plot(epochs, loss, 'bo', label='Training loss')  plt.plot(epochs, val\_loss, 'b', label='Validation loss')  plt.title('Training and validation loss')  plt.xlabel('Epochs')  plt.ylabel('Loss')  plt.legend()  ### history에서 binary\_accuracy와 val\_binary\_accuracy key를 가지는 값들만 추출  rmse = history.history['binary\_accuracy']  val\_rmse = history.history['val\_binary\_accuracy']  epochs = range(1, len(rmse) + 1)  ### binary\_accuracy와 val\_binary\_accuracy key를 그래프화  plt.subplot(212) ## 2x1 개의 그래프 중에 2번째  plt.plot(epochs, rmse, 'ro', label='Training binary\_accuracy')  plt.plot(epochs, val\_rmse, 'r', label='Validation binary\_accuracy')  plt.title('Training and validation binary\_accuracy')  plt.xlabel('Epochs')  plt.ylabel('binary\_accuracy')  plt.legend()  # 결과를 출력합니다.  print("-----")  print(model.evaluate(x\_test, y\_test))  print("\n Accuracy: %.4f" % (model.evaluate(x, y)[1]))  plt.show()  --------------------------------------------------------------------------------  import tensorflow as tf  import matplotlib.pyplot as plt  import numpy  import os  (x\_train, y\_train), (x\_test, y\_test)=tf.keras.datasets.mnist.load\_data()  print(len(x\_train))  print(len(y\_train))  print(x\_train.shape)  print(y\_train.shape)  plt.imshow(x\_train[500], cmap=plt.cm.binary)  print(y\_train[500])  index=74  plt.imshow(x\_train[index], cmap=plt.cm.binary)  plt.show()  print((index+1), "번째 이미지의 숫자는 바로 ", y\_train[index], "입니다.")  print(x\_train.shape)  print(x\_test.shape)  print(y\_train.shape)  print(y\_test.shape)  import numpy as np  print('최소값:', np.min(x\_train), '최대값:', np.max(x\_train))  # 정규화  x\_train\_norm = x\_train / 255.0  x\_test\_norm = x\_test / 255.0  print('최소값:', np.min(x\_train\_norm), '최대값:', np.max(x\_test\_norm))  # reshape 28\*28 => 784, 1 변환  x\_train=x\_train\_norm.reshape(x\_train.shape[0], 784).astype('float32')  x\_test=x\_test\_norm.reshape(x\_test.shape[0], 784).astype('float32')  y\_train=tf.keras.utils.to\_categorical(y\_train, 10)  y\_test=tf.keras.utils.to\_categorical(y\_test, 10)  print(y\_train[0])  from tensorflow import keras  model=keras.models.Sequential()  model.add(keras.layers.Dense(512, activation='relu', input\_shape=(784,)))  model.add(keras.layers.Dense(256, activation='relu'))  model.add(keras.layers.Dense(10, activation='softmax'))  model.summary()  print(len(model.layers))  model.compile(optimizer='adam',  loss='categorical\_crossentropy',  metrics=['accuracy'])  history=model.fit(x\_train, y\_train, validation\_split=0.1, epochs=10, batch\_size=200, verbose=1)  test\_loss, test\_accuracy = model.evaluate(x\_test, y\_test, verbose=2)  print('test loss : {}'.format(test\_loss))  print('test\_accuracy: {}'.format(test\_accuracy))  # 예측  predicted\_result = model.predict(x\_test)  predicted\_labels = np.argmax(predicted\_result,axis=1)  idx = 0  print('model.predict() 결과 :',predicted\_result[idx])  print('model이 추론한 가장 가능성이 높은 결과 : ',predicted\_labels[idx])  print('실제 데이터의 라벨 :',y\_test[idx])  x\_test.shape  x\_test\_result = x\_test.reshape(x\_test.shape[0], 28, 28)  x\_test\_result.shape  plt.imshow(x\_test\_result[idx], cmap=plt.cm.binary)  plt.show()  # 테스트 셋의 오차  y\_vloss = history.history['val\_loss']  # 학습셋의 오차  y\_loss = history.history['loss']  # 그래프로 표현  x\_len = numpy.arange(len(y\_loss))  plt.plot(x\_len, y\_vloss, marker='.', c="red", label='Testset\_loss')  plt.plot(x\_len, y\_loss, marker='.', c="blue", label='Trainset\_loss')  # 그래프에 그리드를 주고 레이블을 표시  plt.legend(loc='upper right')  # plt.axis([0, 20, 0, 0.35])  plt.grid()  plt.xlabel('epoch')  plt.ylabel('loss')  plt.show() |
| 오후 | ## 구글 드라이브 마운트  from google.colab import drive  drive.mount('/content/drive', force\_remount=True)  ## 학습데이터.zip 업로드  from google.colab import files  file\_uploaded = files.upload()  !unzip "rock\_scissor\_paper.zip" -d"./rock\_scissor\_paper" # 안에 있는 폴더 나옴  !unzip "scissor.zip" -d "./scissor"  !unzip "paper.zip" -d "./paper"  !unzip "rock.zip" -d "./rock"  from PIL import Image  import os, glob  def resize\_images(img\_path):  images=glob.glob(img\_path + "/\*.jpg")    print(len(images), " images to be resized.")  # 파일마다 모두 28x28 사이즈로 바꾸어 저장합니다.  target\_size=(28,28)  for img in images:  old\_img=Image.open(img)  new\_img=old\_img.resize(target\_size,Image.ANTIALIAS)  new\_img.save(img, "JPEG")    print(len(images), " images resized.")  # 가위 이미지가 저장된 디렉토리 아래의 모든 jpg 파일을 읽어들여서  image\_dir\_path = "./scissor/"  resize\_images(image\_dir\_path)  print("가위 이미지 resize 완료!")  # 바위 이미지가 저장된 디렉토리 아래의 모든 jpg 파일을 읽어들여서  image\_dir\_path = "./rock/"  resize\_images(image\_dir\_path)  print("바위 이미지 resize 완료!")  # 보 이미지가 저장된 디렉토리 아래의 모든 jpg 파일을 읽어들여서  image\_dir\_path = "./paper/"  resize\_images(image\_dir\_path)  print("보 이미지 resize 완료!")  import numpy as np  def load\_data(img\_path, number\_of\_data=300): # 가위바위보 이미지 개수 총합에 주의하세요.  # 가위 : 0, 바위 : 1, 보 : 2  img\_size=28  color=3  #이미지 데이터와 라벨(가위 : 0, 바위 : 1, 보 : 2) 데이터를 담을 행렬(matrix) 영역을 생성합니다.  imgs=np.zeros(number\_of\_data\*img\_size\*img\_size\*color,dtype=np.int32).reshape(number\_of\_data,img\_size,img\_size,color)  labels=np.zeros(number\_of\_data,dtype=np.int32)  idx=0  for file in glob.iglob(img\_path+'/scissor/\*.jpg'):  img = np.array(Image.open(file),dtype=np.int32)  imgs[idx,:,:,:]=img # 데이터 영역에 이미지 행렬을 복사  labels[idx]=0 # 가위 : 0  idx=idx+1  for file in glob.iglob(img\_path+'/rock/\*.jpg'):  img = np.array(Image.open(file),dtype=np.int32)  imgs[idx,:,:,:]=img # 데이터 영역에 이미지 행렬을 복사  labels[idx]=1 # 바위 : 1  idx=idx+1    for file in glob.iglob(img\_path+'/paper/\*.jpg'):  img = np.array(Image.open(file),dtype=np.int32)  imgs[idx,:,:,:]=img # 데이터 영역에 이미지 행렬을 복사  labels[idx]=2 # 보 : 2  idx=idx+1    print("학습데이터(x\_train)의 이미지 개수는", idx,"입니다.")  return imgs, labels  image\_dir\_path = "/content/drive/MyDrive/Colab Notebooks/dataset/rock-scissers-papper/"  (x\_train, y\_train)=load\_data(image\_dir\_path)  x\_train\_norm = x\_train/255.0 # 입력은 0~1 사이의 값으로 정규화  print("x\_train shape: {}".format(x\_train.shape))  print("y\_train shape: {}".format(y\_train.shape))  import matplotlib.pyplot as plt  plt.imshow(x\_train[0])  print('라벨: ', y\_train[0]) |