|  |  |
| --- | --- |
| 교육 제목 | **딥러닝 연습** |
| 교육 일시 | 2021년 12월 3일 금요일 |
| 교육 장소 | 영우글로벌러닝 2층 |
| **교육 내용** | |
| 오전 | 1. 딥러닝 필요 라이브러리    1. from keras.datasets import mnist    2. from keras.utils import np\_utils    3. from keras.models import Sequential    4. from keras.layers import Dense    5. from keras.callbacks import ModelCheckpoint,EarlyStopping    6. import matplotlib.pyplot as plt    7. import numpy    8. import os    9. import tensorflow as tf 2. mnist 에서 데이터를 load 합니다    1. (X\_train, Y\_class\_train), (X\_test, Y\_class\_test) = mnist.load\_data() 3. train 데이터를 70%만 가져옵니다 4. 데이터 전처리    1. 다차원을 2차원으로 변경       1. X\_train = X\_train.reshape(X\_train.shape[0], 784)       2. # X\_train = X\_train.reshape(X\_train.shape[0], X\_train.shape[1] \* X\_train.shape[2])       3. X\_train = X\_train.astype('float64')       4. X\_train = X\_train / 255 #정규화 진행 숫자    2. 0 ~ 1 사이의 값으로 정규화       1. X\_train = X\_train.reshape(X\_train.shape[0], 784).astype('float32') / 255       2. X\_test = X\_test.reshape(X\_test.shape[0], 784).astype('float32') / 255    3. 원핫 인코딩 (Y의 값을 class라고 함) |
| 오후 | 1. 모델 설정    1. 레이어를 2개층으로 모델 설정       1. # 모델 프레임 설정       2. model = Sequential()       3. model.add(Dense(512, input\_dim=784, activation='relu'))       4. model.add(Dense(10, activation='softmax'))    2. # 모델 실행 환경 설정       1. model.compile(loss='categorical\_crossentropy',       2. optimizer='adam',       3. metrics=['accuracy']) 2. 모델 최적화    1. # 모델 최적화 설정       1. MODEL\_DIR =       2. '/content/drive/MyDrive/Colab\_Notebooks/dataset/mnist\_model/'       3. if not os.path.exists(MODEL\_DIR):       4. os.mkdir(MODEL\_DIR)       5. modelpath='/content/drive/MyDrive/Colab\_Notebooks/dataset/       6. mnist\_model/{epoch:02d}-{val\_loss:.4f}.hdf5'       7. checkpointer = ModelCheckpoint(filepath=modelpath,       8. monitor='val\_loss', verbose=1, save\_best\_only=True)       9. early\_stopping\_callback = EarlyStopping(monitor='val\_loss', patience=10)    2. # 모델의 실행 3. Epoch 실행할 때마다 모델 저장 4. 디렉토리 생성 후 저장 epoch 회수 –에러값.hdf5를 파일명으로 5. 더이상 값의 변화가 없는 epoch 의 수는 30 6. 모델의 실행 결과를 그래프로 표현하기 위해 변수에 저장    1. # 그래프를 그려보기 위해서    2. history = model.fit(X\_train, Y\_train,    3. validation\_data=(X\_test, Y\_test), epochs=30, batch\_size=200,    4. verbose=0, callbacks=[early\_stopping\_callback,checkpointer]) 7. 저장된 변수의 학습셋의 정확도와 테스트셋의 오차를 화면에 표시    1. # 테스트 셋의 오차       1. y\_vloss = history.history['val\_loss']    2. # 학습셋의 오차       1. y\_loss = history.history['loss']    3. # 그래프로 표현       1. x\_len = numpy.arange(len(y\_loss))       2. plt.plot(x\_len, y\_vloss, marker='.', c="red", label='Testset\_loss')       3. plt.plot(x\_len, y\_loss, marker='.', c="blue", label='Trainset\_loss')    4. # 그래프에 그리드를 주고 레이블을 표시       1. plt.legend(loc='upper right')    5. # plt.axis([0, 20, 0, 0.35])       1. plt.grid()       2. plt.xlabel('epoch')       3. plt.ylabel('loss')       4. plt.show() |