|  |  |
| --- | --- |
| 교육 제목 | **딥러닝 MNIST dataset** |
| 교육 일시 | 2021년 12월 6일 월요일 |
| 교육 장소 | 영우글로벌러닝 2층 |
| **교육 내용** | |
| 오전 | 1. 이미지 인식의 꽃, CNN 익히기    1. 데이터 전처리       * #데이터 불러오기       * from keras.datasets import mnist       * (X\_train, Y\_class\_train), (X\_test, Y\_class\_test) = mnist.load\_data()       * #데이터 확인하기       * import matplotlib.pyplot as plt       * plt.imshow(X\_train[0], cmap='Greys')       * plt.show()    2. 28 × 28 = 784개의 속성을 이용해 0~9까지 10개 클래스 중 하나를 맞히는 문제가 됩니다.    3. 주어진 가로 28, 세로 28의 2차원 배열을 784개의 1차원 배열로 바꿔 주어야 합니다. 이를 위해 reshape() 함수를 사용합니다.    4. reshape(총 샘플 수, 1차원 속성의 수) 형식으로 지정합니다. 총 샘플 수는 앞서 사용한 X\_train.shape[0]을 이용하고, 1차원 속성의 수는 이미 살펴본 대로 784개입니다.    5. X\_train = X\_train.reshape(X\_train.shape[0], 784)    6. 딥러닝의 분류 문제를 해결하려면 원-핫 인코딩 방식을 적용 한다    7. 즉, 0~9까지의 정수형 값을 갖는 현재 형태에서 0 또는 1로만 이루어진 벡터로 값을 수정해야 합니다.    8. 지금 우리가 열어본 이미지의 class는 [5]였습니다. 이를 [0,0,0,0,0,1,0,0,0,0]로 바꿔야 합니다. 이를 가능하게 해 주는 함수가 바로 np\_utils.to\_categorical() 함수입니다. to\_categorical(클래스, 클래스의 개수)의 형식으로 지정합니다.    9. Y\_train = np\_utils.to\_categorical(Y\_class\_train,10)    10. Y\_test = np\_utils.to\_categorical(Y\_class\_test,10)    11. 딥러닝 기본 프레임 만들기   from keras.datasets import mnist  from keras.utils import np\_utils    import numpy  import sys  import tensorflow as tf    # seed 값 설정  seed = 0  numpy.random.seed(seed)  tf.random.set\_seed(3)    # MNIST 데이터셋 불러오기  (X\_train, Y\_class\_train), (X\_test, Y\_class\_test) = mnist.load\_data()    print("학습셋 이미지 수 : %d 개" % (X\_train.shape[0]))  print("테스트셋 이미지 수 : %d 개" % (X\_test.shape[0]))    # 그래프로 확인  import matplotlib.pyplot as plt  plt.imshow(X\_train[0], cmap='Greys')  plt.show()    # 코드로 확인  for x in X\_train[0]:  for i in x:  sys.stdout.write('%d\t' % i)  sys.stdout.write('\n')    # 차원 변환 과정  X\_train = X\_train.reshape(X\_train.shape[0], 784)  X\_train = X\_train.astype('float64')  X\_train = X\_train / 255    X\_test = X\_test.reshape(X\_test.shape[0], 784).astype('float64') / 255    # 클래스 값 확인  print("class : %d " % (Y\_class\_train[0]))    # 바이너리화 과정  Y\_train = np\_utils.to\_categorical(Y\_class\_train, 10)  Y\_test = np\_utils.to\_categorical(Y\_class\_test, 10)    print(Y\_train[0]) |
| 오후 | 1. 더 깊은 딥러닝   # 모델 프레임 설정  model = Sequential()  model.add(Dense(512, input\_dim=784, activation='relu'))  model.add(Dense(10, activation='softmax'))    # 모델 실행 환경 설정  model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])    # 모델 최적화 설정  MODEL\_DIR = './model/'  if not os.path.exists(MODEL\_DIR):  os.mkdir(MODEL\_DIR)    modelpath="./model/{epoch:02d}-{val\_loss:.4f}.hdf5"  checkpointer = ModelCheckpoint(filepath=modelpath, monitor='val\_loss', verbose=1, save\_best\_only=True)  early\_stopping\_callback = EarlyStopping(monitor='val\_loss', patience=10)    # 모델의 실행  history = model.fit(X\_train, Y\_train, validation\_data=(X\_test, Y\_test), epochs=30, batch\_size=200, verbose=0, callbacks=[early\_stopping\_callback,checkpointer])    # 테스트 정확도 출력  print("\n Test Accuracy: %.4f" % (model.evaluate(X\_test, Y\_test)[1]))    # 테스트셋의 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()   1. 컨볼루션 신경망(CNN) 2. 맥스 풀링 3. 컨볼루션 신경망 실행하기 4. 딥러닝을 이용한 자연어 처리    1. 텍스트의 토큰화    2. 단어의 원-핫 인코딩    3. 단어 임베딩    4. 텍스트를 읽고 긍정, 부정 예측하기딥러닝 필요 라이브러리 |