|  |  |
| --- | --- |
| 교육제목 | 데이터 기반 인공지능 시스템 엔지니어 양성 과정 |
| 교육일시 | 211024 |
| 교육장소 | 자택(디스코드 사용 온라인 학습) |
| **교육내용** | |
| 1. Single neuron model : perception linear regression(선형회귀)   캡처.JPG  import tensorflow as tf  ## data 선언  x\_data = [[2.,0.,7.], [6.,4.,2.], [5.,2.,4.],[8.,4.,1]]  y\_data = [[75], [95], [91], [97]]  test\_data=[[5.,5.,5.]]  print(len(x\_data),len(x\_data[1])) # 행크기 , 열크기  ## tf.keras를 활용한 perceptron 모델 구현.  model = tf.keras.Sequential() ## 모델 만들기 위해 sequential 매서드를 선언. 이를 통해 모델을 만들 수 있다.  model.add(tf.keras.layers.Dense(1, input\_dim=3)) # 선언된 모델에 add를 통해 쌓아감. , 현재는 입력 변수 갯수 3, perceptron 1개.  model.summary() ## 설계한 모델 프린트  # 모델 loss, 학습 방법 결정하기  optimizer= tf.keras.optimizers.SGD(lr=0.01) ### 경사 하강법으로 global min 에 찾아가는 최적화 방법 선언.  # lr(러닝레이트)는 여기에서 지정  loss=tf.keras.losses.mse ## 예측값 과 정답의 오차값 정의. mse는 mean squre error로 (예측값 - 정답)^2 를 의미  metrics= tf.keras.metrics.mae ### 학습하면서 평가할 메트릭스 선언 mse는 mean\_absolute\_error |예측값 - 정답| 를 의미  # 모델 컴파일하기  model.compile(loss=loss, optimizer=optimizer, metrics=[metrics])  # 모델 동작하기  model.fit(x\_data, y\_data, epochs=20, batch\_size=4)  # 결과를 출력합니다.  print(model.weights)  print(" test data [5.,5.,5.] 예측 값 : ", model.predict(test\_data))   1. Classification(분류)   캡처.JPG  캡처.JPG  import tensorflow as tf  import keras  ## data 선언  x\_data = [[0.,0.], [0.,1.], [1.,0.],[1.,1.]]  y\_data = [[0.], [1.], [1.], [1.]]  test\_data=[[0.3, 0.3]]  print(len(x\_data),len(x\_data[1]))  ## tf.keras를 활용한 perceptron 모델 구현.  model = tf.keras.Sequential()  model.add(tf.keras.layers.Dense(1,input\_dim=2)) # 0,1 둘중하나나오니 1, 인풋은 2.  model.summary() ## 설계한 모델 프린트  # 모델 loss, 학습 방법 결정하기  optimizer= tf.keras.optimizers.SGD() ### 경사 하강법으로 global min 에 찾아가는 최적화 방법 선언.  loss= tf.keras.losses.mse ## 예측값 과 정답의 오차값 정의. mse는 mean squre error로 (예측값 - 정답)^2 를 의미  metrics= tf.keras.metrics.mae ### 학습하면서 평가할 메트릭스 선언  # 모델 컴파일하기  model.compile(loss=loss, optimizer=optimizer, metrics=[metrics])  # 모델 동작하기  model.fit(x\_data , y\_data , epochs=20 , batch\_size= 1 )  # 결과를 출력합니다.  print(model.weights)  print(" test data [0.3, 0.3] 예측 값 : ", model.predict(test\_data))  if model.predict(test\_data)>0.5:  print(" 합격 " )  else:  print(" 불합격 ")   1. Logistic Regression   캡처.JPG  캡처.JPG   * Curve로 fitting한다면?   ## data 선언  x\_data = [[5.], [30.], [95.], [100.], [265.], [270.], [290.], [300.],[365.]]  y\_data = [[0.], [0.], [0.], [0.], [1.], [1.], [1.], [1.], [1.]]  test\_data= [[7.]]  test\_data2= [[80.]]  test\_data3= [[110.]]  test\_data4= [[180.]]  test\_data5= [[320.]]  ## tf.keras를 활용한 perceptron 모델 구현.  model = tf.keras.Sequential() ## 모델 선언  model.add(tf.keras.layers.Dense(1, input\_dim=1, activation='sigmoid'))  loss = tf.keras.losses.mse # mean squre error ### 경사 하강법으로 global min 에 찾아가는 최적화 방법 선언.  optimizer = tf.keras.optimizers.SGD(lr=0.01) ## 예측값 과 정답의 오차값 정의. mse는 mean squre error로 (예측값 - 정답)^2 를 의미  metrics = tf.keras.metrics.binary\_accuracy # mean absolute error ㅣ예측값 - 정답ㅣ  # 모델 컴파일하기  model.compile(loss=loss, optimizer=optimizer, metrics=[metrics])  # 모델 동작하기  model.fit(x\_data, y\_data, epochs=100, batch\_size=2) # 총데이터가 9개니까 2개씩 나누고 남은 1개까지해서 5묶음.   1. XOR Problem   캡처.JPG  캡처.JPG  캡처.JPG  캡처.JPG  캡처.JPG  x\_data = [[0.,0.], [0.,1.], [1., 0.], [1.,1.]]  y\_data = [[0.], [1.], [1.],[0.]]  test\_data = [[0.5, 0.5]]  model = tf.keras.Sequential()  model.add(tf.keras.layers.Dense(4, input\_dim=2, activation='sigmoid'))  model.add(tf.keras.layers.Dense(5, activation='sigmoid'))  model.add(tf.keras.layers.Dense(1, activation='sigmoid'))  model.summary()  optimizer=tf.keras.optimizers.SGD(learning\_rate=0.5) ### 경사 하강법으로 global min 에 찾아가는 최적화 방법 선언.  loss=tf.keras.losses.binary\_crossentropy ## 예측값 과 정답의 오차값 정의.  metrics=tf.keras.metrics.binary\_accuracy ### 학습하면서 평가할 메트릭스 선언언  model.compile(loss =loss, optimizer= optimizer, metrics=[metrics])  hist = model.fit(x\_data, y\_data, epochs=2000, batch\_size=4)  plt.plot(hist.history['loss'])  print(" test data [0.5 0.5] 예측 값 : ", model.predict(test\_data))   1. house price of area prediction   from google.colab import files  file\_uploaded = files.upload()  import matplotlib.pyplot as plt  import pandas as pd  import tensorflow as tf  df = pd.read\_csv("./house\_price\_of\_unit\_area.csv")  print(df.info())  x\_data = df.copy()  tf.random.set\_seed(777) # random seed를 고정해줘서 돌릴때마다 랜덤값을 고정  y\_data = x\_data.pop("house price of unit area")  print(y\_data)  print(x\_data)  model = tf.keras.Sequential()  model.add(tf.keras.layers.Dense(64, input\_dim=5, activation='sigmoid'))  model.add(tf.keras.layers.Dense(128, activation='sigmoid'))  model.add(tf.keras.layers.Dense(256, activation='sigmoid'))  model.add(tf.keras.layers.Dense(1))  model.summary()  optimizer=tf.keras.optimizers.SGD(learning\_rate=0.04) ### 경사 하강법으로 global min 에 찾아가는 최적화 방법 선언.  loss=tf.keras.losses.mean\_squared\_error ## 예측값 과 정답의 오차값 정의.  metrics=tf.keras.metrics.RootMeanSquaredError() ### 학습하면서 평가할 메트릭스 선언언  model.compile(loss =loss, optimizer= optimizer, metrics=['metrics']  result = model.fit(x\_data, y\_data, epochs=100, batch\_size=100)  print(result.history.keys())  loss = result.history['loss']  # loss 그래프  epochs = range(1, len(loss)+1)  plt.subplot(211)  plt.plot(epochs, loss, 'b-', label='Training loss')  plt.title('Training loss')  plt.xlabel('Epochs')  plt.ylabel('Loss')  plt.legend()  mae = result.history['root\_mean\_squared\_error']  epochs = range(1, len(mae)+1)  # mean absolute error 그래프화  plt.subplot(212)  plt.plot(epochs, mae, 'r-', label='Training mae')  plt.title('Training rmse')  plt.xlabel('Epochs')  plt.ylabel('rmse')  plt.legend()  # x\_data -> model -> predict값 - y\_data값을 추출  print(model.evaluate(x\_data, y\_data))  print("\n Test rmse : %.4f" % (model.evaluate(x\_data, y\_data)[1])) | |