**교육일지**

**교육 제목 : 딥러닝**

**교육 장소 : YGL C6 강의실**

**교육 일시 : 2021/10/21**

**import tensorflow as tf**

**print(tf.\_\_version\_\_) #버전 체크**

**## tensorflow로 (5\*2) - (3+2) 연산을 수행하시오**

**# 1. tf.costant로 상수 선언해서 수행하기**

**x1= tf.constant(5) x1= tf.Variable(5)**

**x2= tf.constant(3) x2= tf.Variable(3)**

**x3= tf.constant(2) x3= tf.Variable(2)**

**#연산**

**add\_result =tf.add(x2,x3) #**

**mul\_result =tf.multiply(x1,x3)**

**sub\_result =tf.subtract(mul\_result,add\_result)**

**print("add\_result = {}".format(add\_result))**

**print("mul\_result = {}".format(mul\_result))**

**print("sub\_result = {}".format(sub\_result))**

**# tf를 선언하고 for문을 이용하여 1부터 10까지 더해지는 코드를 작성하고**

**# 변수선언**

**x= tf.Variable(0)**

**#for**

**for i in range(1, 11):**

**x = x + i**

**print("value : {}".format(x))**

**print("total sum : {}".format(x))**

**#랜덤 변수 선언하기**

**#평균이 0이고, 분산이 파라미터의 정규분포로부터 값 구하기**

**x1 = tf.Variable(tf.random.normal((1,1), mean=0, stddev=1.0))**

**x2 = tf.Variable(tf.random.normal((1,1), mean=0, stddev=1.0))**

**x3 = tf.Variable(tf.random.normal((1,1), mean=0, stddev=1.0))**

**add\_result =tf.add(x2,x3)**

**mul\_result =tf.multiply(x1,x3)**

**sub\_result =tf.subtract(mul\_result,add\_result)**

**print("add\_result = {}".format(add\_result))**

**print("mul\_result = {}".format(mul\_result))**

**print("sub\_result = {}".format(sub\_result))**

**# 다차원 변수 선언하기**

**x1 = tf.Variable([[1,2],[3,4]])**

**print("x1 = {}".format(x1))**

**print("x1 = {}".format(x1.shape))**

**x1\_reshape = tf.reshape(x1, [1,4])**

**print("x1 = {}".format(x1\_reshape))**

**print("x1 = {}".format(x1\_reshape.shape))**

**## step2 tensorflow matrix operation**

**## matrix 선언 하기.**

**matrix\_A = tf.constant() # 작성**

**matrix\_B = tf.constant() # 작성**

**## matrix 연산하기.**

**result = tf.mat( ) # 작성**

**## 결과**

**print("result = {}".format(result))**

**## data 선언**

**x\_data =[[1.],[2.],[3.],[4.]]**

**y\_data =[[1.],[3.],[5.],[7.]]**

**print(x\_data)**

**print(y\_data)**

**## 평균 0, 분산 1의 파라미터의 정규분포로 부터 값을 가져옴.**

**# 학습을 통해 업데이트가 되어 변화되는 모델의 파라미터인 w,b를 의미한다.**

**W= tf.Variable(tf.random.normal((1,1), mean=0, stddev=1 ))**

**b= tf.Variable(tf.random.normal((1,1), mean=0, stddev=1 ))**

**print("W : ", W)**

**print("b : ", b)**

**for j in range(len(x\_data)):**

**## data \* weight 작성**

**WX = tf.matmul([x\_data[j]], W)**

**## bias add 작성**

**y\_hat = tf.add(WX,b)**

**## W와 b로 예측 하기**

**print("y\_data: , ",y\_data[j], "prediction : ", y\_hat)**

**## step4 tensor\_linear**

**import tensorflow as tf**

**## data 선언**

**x\_data =[[1.],[2.],[3.],[4.]]**

**y\_data =[[1.],[3.],[5.],[7.]]**

**## 평균 0, 분산 1의 파라미터의 정규분포로 부터 값을 가져옴.**

**# 학습을 통해 업데이트가 되어 변화되는 모델의 파라미터인 w,b를 의미한다.**

**W=tf.Variable(tf.random.normal((1,1),mean=0, stddev=1.0))**

**b=tf.Variable(tf.random.normal((1,1),mean=0, stddev=1.0))**

**lr=tf.constant(0.0001)**

**history = np.zeros([2000,3], 'float32')**

**for i in range(2000): ## 에폭**

**total\_error = 0**

**for j in range(len(x\_data)): ## 배치 1**

**## data \* weight**

**WX =tf.matmul([x\_data[j]], W)**

**## bias add**

**y\_hat = tf.add(WX, b)**

**## 정답인 Y와 출력값의 error 계산**

**error = tf.subtract(y\_data[j], y\_hat)# ture - prediction**

**## 경사하강법으로 W와 b 업데이트.**

**## 도함수 구하기**

**diff\_W = tf.matmul(error, [x\_data[j]]) #error \* x의합**

**diff\_b = error**

**## 업데이트할 만큼 러닝레이트 곱**

**diff\_W = tf.multiply(lr,diff\_W)**

**diff\_b = tf.multiply(lr,diff\_b) # lr \* error**

**## w, b 업데이트**

**W = tf.add(W, diff\_W) # w + delta w + lr \* x \* (error)**

**b = tf.add(b, diff\_b) # b + delta b + lr \* (error)**

**#######**

**## 토탈 에러.**

**visual\_error = tf.square(error)**

**total\_error = total\_error + visual\_error**

**## 모든 데이터에 따른 error 값**

**print("epoch: ", i, "error : ", total\_error/len(x\_data))**

**history[i,:] = [(total\_error/len(x\_data))[0], W[0], b[0]]**