**교육일지**

**교육 제목 : 딥러닝**

**교육 장소 : YGL C6 강의실**

**교육 일시 : 2021/10/25**

**Gradient Tape basic**

**test1 = [1, 2, 3]**

**test2 = [10, 20, 30]**

**t1 = tf.Variable(test1, dtype=tf.float32)**

**t2 = tf.Variable(test2, dtype=tf.float32)**

**with tf.GradientTape() as tape:**

**t3 = t1 \* t2**

**gradient = tape.gradient(t3, [t1, t2])**

**print(gradient[0])**

**print(gradient[1])**

**Gradient Tape이용한 linear regression**

**## data 선언**

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

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

**plt.plot(x\_data, y\_data, 'o')**

**## 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 = 0.001**

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

**w = np.array(W)**

**w = w.reshape(1)**

**B = np.array(b)**

**B = B.reshape(1)**

**plt.plot(x\_data, y\_data, 'o')**

**plt.plot([0,4], [b, (w\*4 +B)], 'r-')c**

**plt.plot(x\_data, y\_data, 'o')**

**plt.plot([0,4], [b, (w\*4 +B)], 'r-')**

**for epoch in range(2000):**

**total\_error = 0**

**for x, y in zip(x\_data, y\_data):**

**with tf.GradientTape() as tape:**

**y\_hat = W \* x + b**

**error = (y\_hat - y) \*\*2**

**gradients = tape.gradient(error, [W, b]) # 별명.gradient(타겟값, [업데이터시킬 파라미터, ...])**

**W = tf.Variable(W - lr \* gradients[0])**

**b = tf.Variable(b - lr \* gradients[1])**

**w\_trace.append(W.numpy())**

**b\_trace.append(b.numpy())**

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

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

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

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

**# 학습이 끝난 후 W와 b로 예측**

**print(history)**

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

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

**print("input 3", tf.add(tf.matmul([[3.]], W), b))**

**print("input 4", tf.add(tf.matmul([[4.]], W), b))**

**# loss function**

**plt.plot(history[:,0])**

**plt.title('loss function')**

**plt.xlabel('epoch')**

**plt.ylabel('loss')**

**multi regression**

**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**

**loss = tf.keras.losses.mse # mean squre error**

**optimizer = tf.keras.optimizers.SGD(lr=0.01)**

**metrics = tf.keras.metrics.mae # mean absolute error ㅣ예측값 - 정답ㅣ**

**# 모델 컴파일하기**

**model.compile(loss=loss, optimizer=optimizer, metrics=[metrics])**

**# 모델 동작하기**

**model.fit(x\_data, y\_data, epochs=20, batch\_size=2)**

**# 결과를 출력**

**print(model.weights)**

**print("test data [5., 5., 5.] 예측 값 :", model.predict(test\_data))**