교육 제목	데이터 기반 인공지능 시스템 엔지니어 양성 과정
교육 일시	2021년 10월 25일
교육 장소	YGL C-6 학과장 & 자택(디스코드 이용한 온라인)
교육 내용	
오전	딥러닝 1. Gradient Tape basic ○ tape.gradient(타겟, [미분대상, , ,]) 함수를 이용하여 미분할 수 있다. ○ tf.costant() 함수는 상수를 만듦으로 미분할 수 없다. 2. Gradient Tape 이용한 linear regression 3. Multi regression 4. Binary classification / Logistic Regression - sigmoid 활용

```
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])

    W = tf.Variable(W - lr * gradients[0])
    b = tf.Variable(b - lr * gradients[1])
```

```
## tf.keras를 활용한 perceptron 모델 구현.
model = tf.keras.Sequential()   ## 모델 만들기 위해 sequential 매서드
model.add(tf.keras.layers.Dense(1, input_dim=3)) # 선언된 모델에 ac
model.summary() ## 설계한 모델 프린트
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 4
=================================================================
Total params: 4
Trainable params: 4
Non-trainable params: 0
_________________________________________________________________
```

```
loss = tf.keras.losses.mse # mean squre error
optimizer = tf.keras.optimizers.SGD(lr=0.01)
metrics = tf.keras.metrics.mae # mean absolute error |예측값 - 정답

# 모델 컴파일하기
model.compile(loss=loss, optimizer=optimizer, metrics=[metrics])

# 모델 동작하기
model.fit(x_data, y_data, epochs=20, batch_size=2)
```

```
## tf.keras를 활용한 perceptron 모델 구현.
model = tf.keras.Sequential() ## 모델 선언
model.add(tf.keras.layers.Dense(1, input_dim=1, activation='sigmoid'))
model.summary()
```

```
loss = tf.keras.losses.mse # mean squre error
optimizer = tf.keras.optimizers.SGD(lr=0.01) 
metrics = tf.keras.metrics.binary_accuracy # 
```

오후

딥러닝 계속

5. XOR Problem

```
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()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 4)                 12
_________________________________________________________________
dense_5 (Dense)              (None, 5)                 25
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 6
=================================================================
Total params: 43
Trainable params: 43
Non-trainable params: 0
```

```
optimizer=tf.keras.optimizers.SGD(learning_rate=0.5) ### 경사 하강법으
loss=tf.keras.losses.binary_crossentropy  ## 예측값 과 정답의 오차값 정의
metrics=tf.keras.metrics.binary_accuracy ### 학습하면서 평가할 메트릭스 선

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

6. House Price of area prediction

```
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()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 64)                384
_________________________________________________________________
dense_8 (Dense)              (None, 128)               8320
_________________________________________________________________
dense_9 (Dense)              (None, 256)               33024
_________________________________________________________________
dense_10 (Dense)             (None, 1)                 257
=================================================================
Total params: 41,985
Trainable params: 41,985
Non-trainable params: 0
```

```
optimizer=tf.keras.optimizers.SGD(learning_rate=0.04) ### 경사 하강법으
loss=tf.keras.losses.mean_squared_error  ## 예측값 과 정답의 오차값 정의.
metrics=tf.keras.metrics.RootMeanSquaredError() ### 학습하면서 평가할 메!

model.compile(loss =loss, optimizer= optimizer, metrics=['metrics']
```

*** 딥러닝 적용 순서

1. **Data 생성 or DATA 읽기**
2. **DATA의 변수에 따른 차원 확인**
3. **PERCEPTRON의 입력 차원 맞추고 Layer 완성하기**
4. **PERCEPTRON 생성 (이 때, Activation 주의!!)**
5. **PERCEPTRON LOSS 함수 지정**
6. **경사하강법 (LOSS 함수의 MIN을 찾기 위한 최적화 방법 선택)**
7. **Metric 설정하고 나서, Complie! (Task에 따라서 Metric 선택에 주의)**
8. **학습완료에 따른 결과 보기**

*** 뉴럴네트워크(딥러닝)의 구조