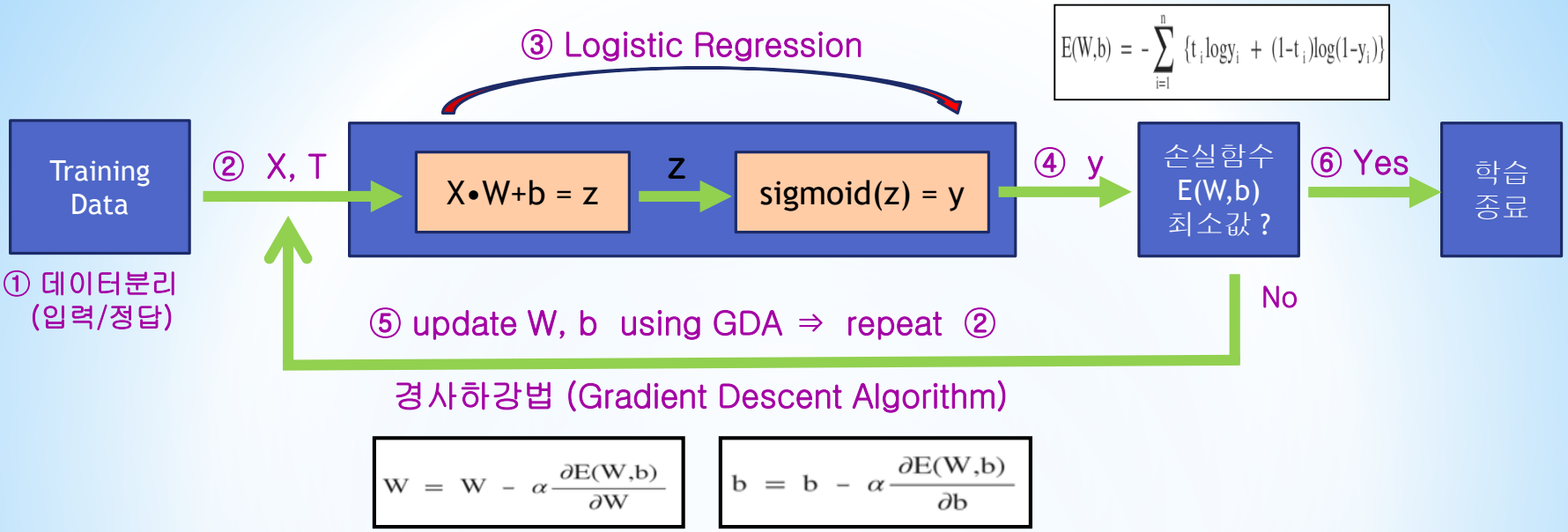
머신러닝/딥러닝을 위한

TensorFlow 기초 (III)

– Logistic Regression (loss • optimizer • accuracy) –

Review – Logistic Regression 동작원리



Training Data
(diabetes.csv)

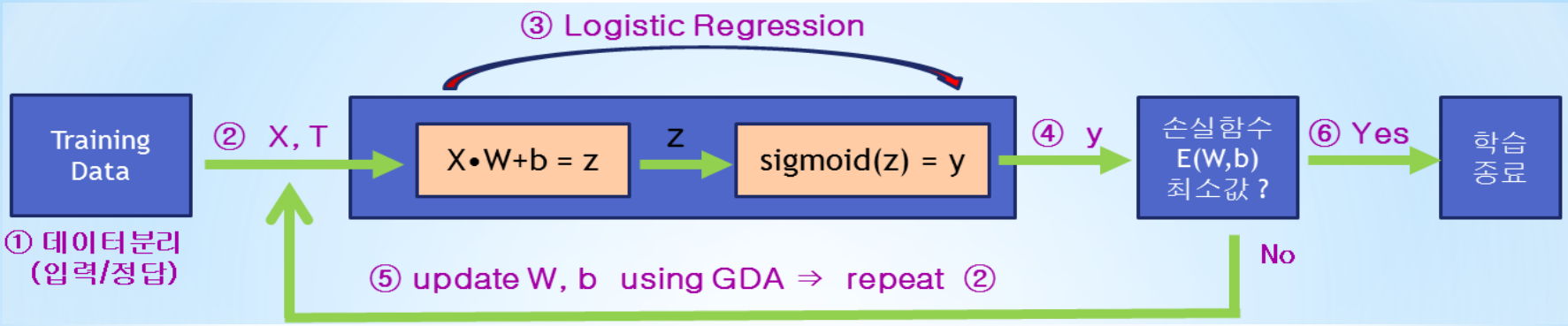
shape
(759 X 9)

9

759

-0.29412	0.487437	0.180328	-0.29293	0	0.00149	-0.53117	-0.03333	0
-0.88235	-0.14573	0.081967	-0.41414	0	-0.20715	-0.76687	-0.66667	1
-0.05882	0.839196	0.04918	0	0	-0.30551	-0.49274	-0.63333	0
-0.88235	-0.10553	0.081967	-0.53535	-0.77778	-0.16244	-0.924	0	1
0	0.376884	-0.34426	-0.29293	-0.60284	0.28465	0.887276	-0.6	0
.....								
-0.88235	0.899497	-0.01639	-0.53535	1	-0.10283	-0.72673	0.266667	0
-0.17647	0.005025	0	0	0	-0.10581	-0.65329	-0.63333	0
0	0.18593	0.377049	-0.05051	-0.45627	0.365127	-0.59607	-0.66667	0
-0.17647	0.075377	0.213115	0	0	-0.11774	-0.8497	-0.66667	0

TensorFlow – 노드 / 연산 정의 (I)



np.loadtxt(…) 이용하여 diabetes.csv 파일로부터 759X9 데이터를 읽어 들인 후, 슬라이스 기능을 이용하여 모든 행에 대해 1열~8열까지는 입력데이터(x_data)로 분리하고, 마지막 열인 9열 데이터는 정답데이터(t_data)로 분리함

①

```
import tensorflow as tf
import numpy as np

loaded_data = np.loadtxt('./diabetes.csv', delimiter=',')

x_data = loaded_data[ :, 0:-1]
t_data = loaded_data[ :, [-1]]

print("loaded_data = ", loaded_data.shape)
print("x_data = ", x_data.shape, ", t_data = ", t_data.shape)
```

```
loaded_data =  (759, 9)
x_data =  (759, 8) , t_data =  (759, 1)
```

세션을 통해 feed_dict으로 데이터를 넣어주기 위해서 입력데이터 노드 X, 정답데이터 노드 T를 정의함. 입력데이터가 8개(1열~8열) 이므로 가중치 노드 W 는 행렬곱을 위해 8X1로 정의하고, 바이어스 노드 b는 1개로 정의함

②

```
X = tf.placeholder(tf.float32, [None, 8])
T = tf.placeholder(tf.float32, [None, 1])

W = tf.Variable(tf.random_normal([8, 1]))
b = tf.Variable(tf.random_normal([1]))
```

현재 759X8 이나, None 지정하면 향후 1500X8, 168X8 등으로 확장 가능

현재의 X, W, b 를 바탕으로 선형회귀 값 z 와 sigmoid 값 y 계산 후에, y 와 정답 T 를 이용하여 손실함수 loss 를 Cross-Entropy 정의함

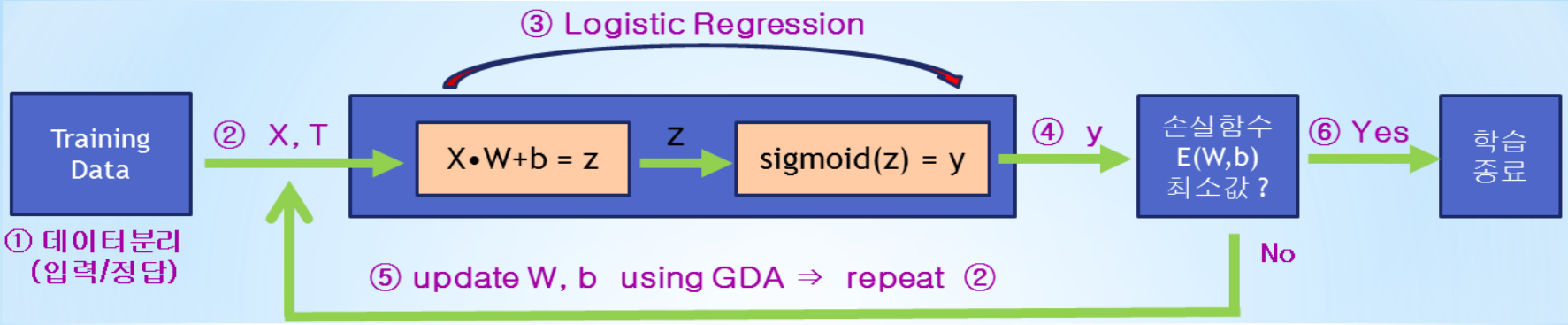
③, ④

```
z = tf.matmul(X, W) + b   # 선형회귀 값 z

y = tf.sigmoid(z)     # 시그모이드로 계산 값

# 손실함수는 Cross-Entropy
loss = -tf.reduce_mean( T*tf.log(y) + (1-T)*tf.log(1-y) )
```

TensorFlow – 노드 / 연산 정의 (II)



가중치 W, 바이어스 b 를 최적화 하기 위해 경사하강법 (Gradient Descent Algorithm)을 적용한 optimizer 정의. 이처럼 TensorFlow에서 경사하강법이 적용된 optimizer, 즉 GradientDescentOptimizer는 내부적으로 오차역전파 등을 이용하여 미분을 수행하면서 손실함수 loss를 최소화 하고, 최종적으로는 W, b 를 최적화 시킴

⑤

```
learning_rate = 0.01    # 학습율

optimizer = tf.train.GradientDescentOptimizer(learning_rate)

train = optimizer.minimize(loss)
```

경사하강법 알고리즘 적용되는 optimizer

손실함수가 최소값을 가지면, 학습을 종료하고 학습이 얼마나 정확하게 이루어 졌는지, 즉 정확도(accuracy)를 측정하여야 함.

먼저, 계산 값 y > 0.5 면 True를 리턴하고 그렇지 않으면 False 를 리턴함. 즉 759 개의 True 와 False 에 대해서 숫자 1 과 0 으로 타입변환을 위해 tf.cast 사용하여 예측 값을 나타내는 predicted 노드를 정의함,

그리고 accuracy 노드는 1 또는 0 값을 갖는 759 개의 데이터에 대해 평균을 구하는 연산(tf.reduce_mean)을 정의함 으로서 정확도를 계산 할 수 있음

⑥

```
predicted = tf.cast(y > 0.5, dtype=tf.float32)

accuracy = tf.reduce_mean(tf.cast(tf.equal(predicted, T), dtype=tf.float32))
```

시그모이드 값 y 형상(shape) 은 (759X8) • (8X1) = 759X1 임. 즉 y > 0.5 라는 것은 759 개의 모든 데이터에 대해 y > 0.5 비교하여 총 759 개의 True 또는 False 리턴함

데이터의 평균 계산

predicted 와 T 같으면 True , 아니면 False를 리턴하므로 tf.cast 를 이용하여 1 또는 0으로 변환해서 총 759 개의 1 또는 0 을 가짐.

TensorFlow – 노드 / 연산 실행

```
with  tf.Session()  as sess:

    sess.run(tf.global_variables_initializer())  # 변수 노드(tf.Variable) 초기화

    for step in range(20001):

        loss_val, _ = sess.run([loss, train], feed_dict={X: x_data, T: t_data})


        if step % 500 == 0:
            print("step = ", step, ", loss_val = ", loss_val)

    # Accuracy 확인
    y_val, predicted_val, accuracy_val = sess.run([y, predicted, accuracy], feed_dict={X: x_data, T: t_data})


    print("₩ny_val.shape = ", y_val.shape, ", predicted_val = ", predicted_val.shape)
    print("₩nAccuracy = ", accuracy_val)
```

feed_dict 으로 입력되는 데이터를 이용해서 수행되는 연산은 loss, train

feed_dict 으로 입력되는 데이터를 이용해서 수행되는 연산은 y, predicted, accuracy

```
step =  0 , loss_val =  0.9388
step =  500 , loss_val =  0.7686692
step =  1000 , loss_val =  0.6987655
step =  1500 , loss_val =  0.6472449
step =  2000 , loss_val =  0.6091539
step =  2500 , loss_val =  0.58087623
step =  3000 , loss_val =  0.5596594
```

.....

```
step =  17000 , loss_val =  0.4741609
step =  17500 , loss_val =  0.47395033
step =  18000 , loss_val =  0.47376096
step =  18500 , loss_val =  0.47359022
step =  19000 , loss_val =  0.4734362
step =  19500 , loss_val =  0.47329685
step =  20000 , loss_val =  0.47317058
```

```
y_val.shape =  (759, 1) , predicted_val =  (759, 1)

Accuracy =  0.770751
```