머신러닝/딥러닝을 위한

TensorFlow 기초 (II)

– Linear Regression (loss • optimizer) –

Review – Linear Regression (multi-variable example)

data-01.csv

x_1	x_2	x_3	t
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142
53	46	55	101
69	74	77	149
47	56	60	115
87	79	90	175
79	70	88	164
69	70	73	141
70	65	74	141
93	95	91	184
79	80	73	152
70	73	78	148
93	89	96	192
78	75	68	147
81	90	93	183
88	92	86	177
78	83	77	159
82	86	90	177
86	82	89	175
78	83	85	175
76	83	71	149
96	93	95	192

① 데이터분리 (입력/정답)

③ Linear Regression

$$E(W,b) = \frac{1}{n}\sum_{i=1}^{n}[t_i-(Wx_i+b)]^2$$

② X, T → $X \cdot W + b = y$ → ④ y → 손실함수 E(W,b) 최소값 ? → Yes → 학습 종료

No → ⑤ update W, b using GDA ⇒ repeat ②

경사하강법 (Gradient Descent Algorithm)

$$W = W - \alpha\frac{\partial E(W,b)}{\partial W} \qquad b = b - \alpha\frac{\partial E(W,b)}{\partial b}$$

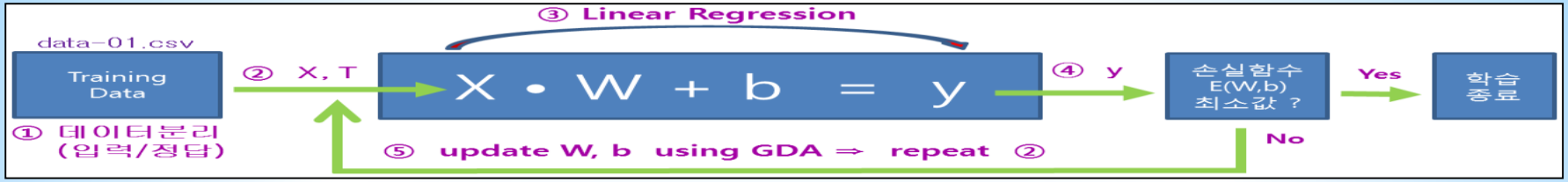
$$X \cdot W + b = y$$

$$\begin{pmatrix} 73 & 80 & 75 \\ 93 & 88 & 93 \\ & \cdots & \\ & \cdots & \\ 76 & 83 & 71 \\ 96 & 93 & 95 \end{pmatrix} \cdot \begin{pmatrix} W_1 \\ W_2 \\ W_3 \end{pmatrix} + b = y$$

입력데이터가 3개이므로 가중치도 3개 필요

(25 X 3) · (3 X 1) = (25 X 1)

TensorFlow – 노드 / 연산 정의 개요



	개념 및 수식	TensorFlow (노드/연산 정의)

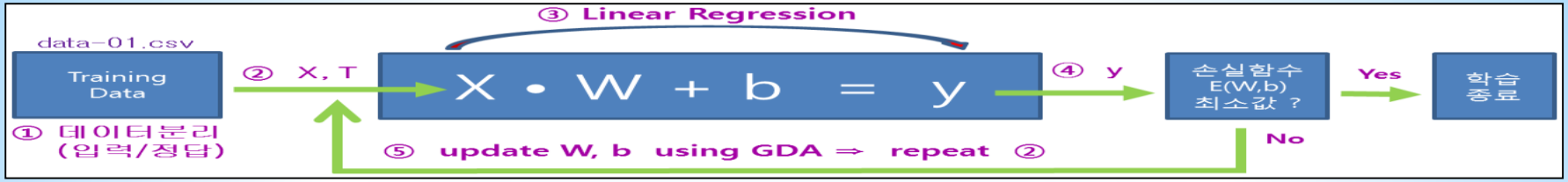
① 데이터분리	data-01.csv 파일로부터 (25X4) training data를 읽어 들인 후, 입력 데이터와 정답 데이터 분리	`import tensorflow as tf`
		`import numpy as np`
		
		`loaded_data = np.loadtxt('./data-01.csv', delimiter=',')`
		
		`x_data = loaded_data[ :, 0:-1]`
		`t_data = loaded_data[ :, [-1]]`

② 데이터 입력	가중치 W, 바이어스 b 정의 후, 입력데이터와 정답데이터 선형회귀 시스템에 입력	`W = tf.Variable(tf.random_normal([3, 1]))`
		`b = tf.Variable(tf.random_normal([1]))`
		
		`X = tf.placeholder(tf.float32, [None, 3])`
		`T = tf.placeholder(tf.float32, [None, 1])`

③ 선형 회귀	$y = X \bullet W + b$	`y = tf.matmul(X, W) + b`
④ 손실 함수 (MSE)	$E(W,b) = \frac{1}{n}\sum_{i=1}^{n}[t_i-(Wx_i+b)]^2$	`loss = tf.reduce_mean(tf.square(y - T))`

⑤ W, b 최적화 (경사하강법)	$W = W - \alpha\frac{\partial E(W,b)}{\partial W}$ $b = b - \alpha\frac{\partial E(W,b)}{\partial b}$	`optimizer = tf.train.GradientDescentOptimizer(learning_rate)`
		
		`train = optimizer.minimize(loss)`

TensorFlow – 노드 / 연산 구현



np.loadtxt(…) 이용하여 data-01.csv 파일로부터 25X4 데이터를 읽어 들인 후, 슬라이스 기능을 이용하여 모든 행에 대해 1열~3열까지는 입력데이터(x_data)로 분리하고, 마지막 열인 4열 데이터는 정답 데이터(t_data)로 분리함

①

```
import tensorflow as tf
import numpy as np

loaded_data = np.loadtxt('./data-01.csv', delimiter=',')

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

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

```
x_data.shape =  (25, 3)
t_data.shape =  (25, 1)
```

가중치 노드 W, 바이어스 노드 b 를 정의하고, feed_dict을 통해 데이터를 넣어주기 위해서 입력데이터 노드와 정답데이터 노드를 정의함.

②

```
W = tf.Variable(tf.random_normal([3, 1]))
b = tf.Variable(tf.random_normal([1]))

X = tf.placeholder(tf.float32, [None, 3])
T = tf.placeholder(tf.float32, [None, 1])
```

현재 25X3 이지만, 25X3 이 아닌 None을 지정하면 차후 50X3, 125X3 등으로 확장이 가능함

현재의 X, W, b 를 바탕으로 선형회귀 값 y 계산하고, y 와 정답 T 를 이용하여 손실함수 정의

③, ④

```
y = tf.matmul(X, W) + b  # 현재 X, W, b, 를 바탕으로 계산된 값

loss = tf.reduce_mean(tf.square(y - T))  # MSE 손실함수 정의
```

가중치 W, 바이어스 b 를 최적화 하기 위해서 경사하강법(Gradient Descent Algorithm)을 적용한 optimizer 정의함. 이처럼 TensorFlow는 다양한 optimizer를 이용하여 손실함수를 최소화 하고, 최종적으로 W, b 를 최적화 시킴

⑤

```
learning_rate = 1e-5    # 학습율

optimizer = tf.train.GradientDescentOptimizer(learning_rate)

train = optimizer.minimize(loss)
```

경사하강법 알고리즘 적용되는 optimizer

optimizer를 통한 손실함수 최소화

TensorFlow – 노드 / 연산 실행

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

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

    for step in range(8001):

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

        if step % 400 == 0:
            print("step = ", step, ", loss_val = ", loss_val)

    print("₩nPrediction is ", sess.run(y, feed_dict={X: [ [100, 98, 81] ]}))
```

```
step =  0 , loss_val =  763.14166
step =  400 , loss_val =  34.777767
step =  800 , loss_val =  26.467009
step =  1200 , loss_val =  20.603018
step =  1600 , loss_val =  16.460691
step =  2000 , loss_val =  13.530931
step =  2400 , loss_val =  11.455769
step =  2800 , loss_val =  9.983346
step =  3200 , loss_val =  8.936595
step =  3600 , loss_val =  8.190849
step =  4000 , loss_val =  7.658225
step =  4400 , loss_val =  7.276743
step =  4800 , loss_val =  7.00268
step =  5200 , loss_val =  6.805072
step =  5600 , loss_val =  6.6620564
step =  6000 , loss_val =  6.5580926
step =  6400 , loss_val =  6.4821715
step =  6800 , loss_val =  6.4264565
step =  7200 , loss_val =  6.3853254
step =  7600 , loss_val =  6.354803
step =  8000 , loss_val =  6.332011

Prediction is  [[178.82211]]
```

feed_dict을 통해 입력되는 데이터를 이용하여 수행되는 연산은 loss, y, train 임

이전 Linear Regression에서 예측했던 179.0 일치함