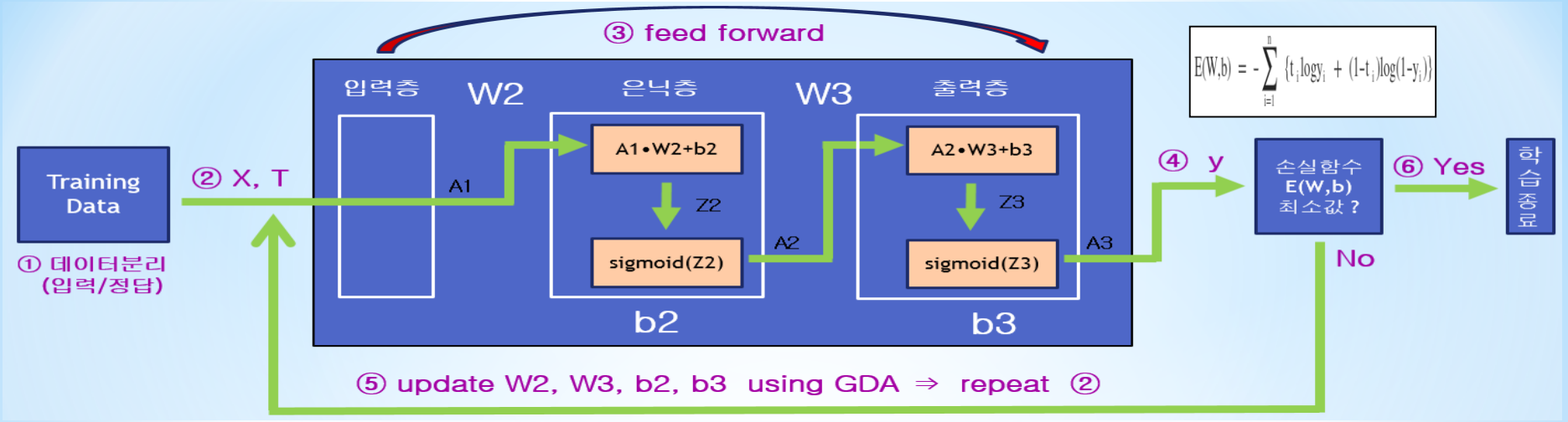
머신러닝/딥러닝을 위한

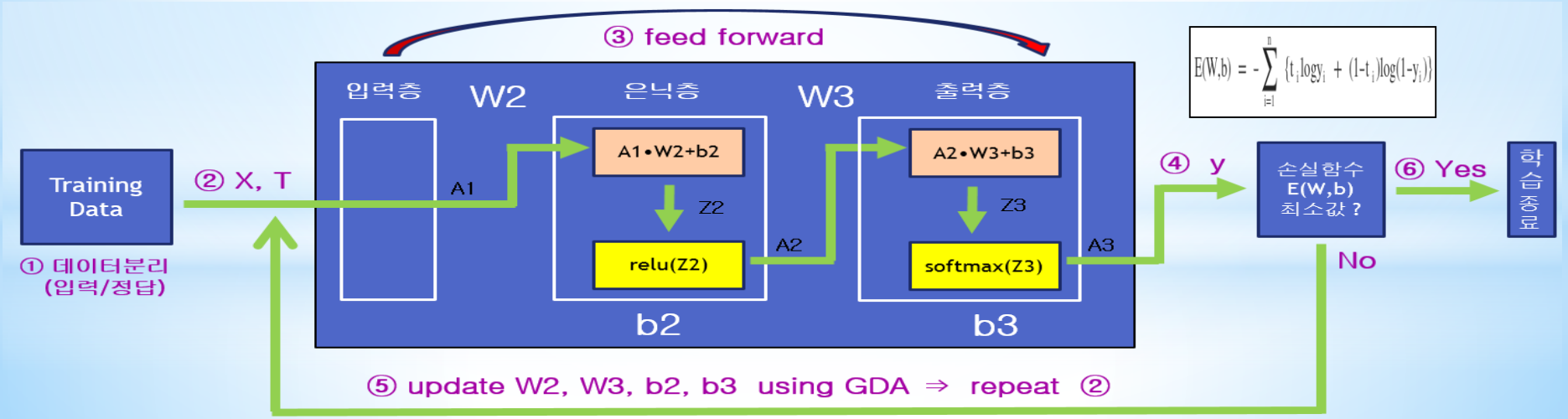
TensorFlow 기초 (IV)

- Neural Network (MNIST example) -

Review – Neural Network 동작원리

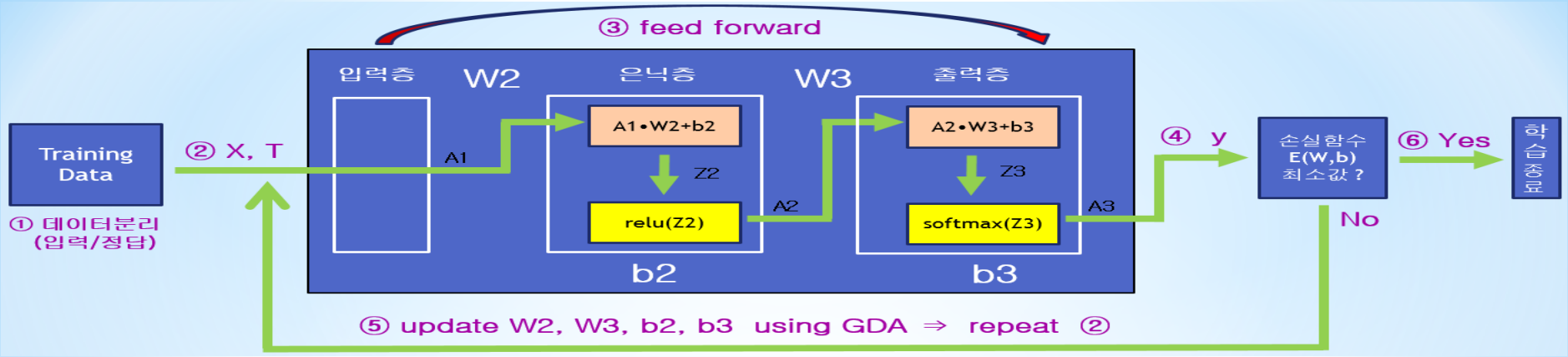


변경 사항



Review – relu • one-hot encoding • softmax

TensorFlow – 노드 / 연산 정의 (I)



read_data_sets() 를 통해 객체형태인 mnist로 받아오고 입력데이터와 정답데이터는 MNIST_data/ 디렉토리에 저장됨.

one_hot=True 옵션을 통해 정답데이터는 one-hot encoding 형태로 저장됨

mnist 객체는 train, test, validation 3개의 데이터 셋으로 구성되어 있으며. num_examples 값을 통해 데이터의 개수 확인 가능함

①

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
import numpy as np
from datetime import datetime


mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

print("")
print("train.num = ", mnist.train.num_examples,
      ", test.num = ", mnist.test.num_examples,
      ", validation.num = ", mnist.validation.num_examples)
```

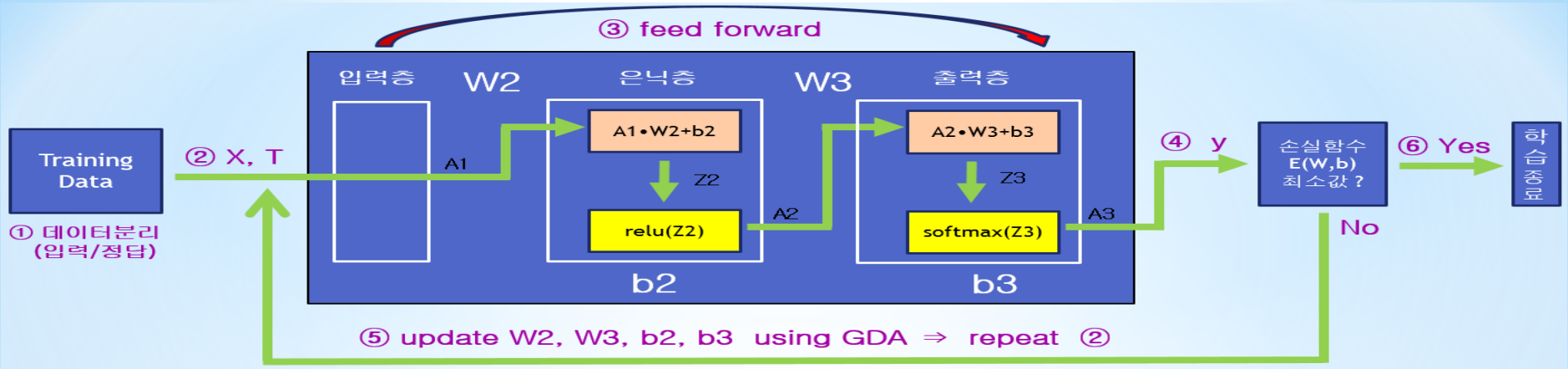
[확인사항]

[1] type(mnist) ? type(mnist.train.images) ? type(mnist.train.labels) ?

[2] mnist.train.images.shape ? np.shape(mnist.train.images) ?

[3] train 데이터의 정규화 여부 label의 one-hot encoding 여부

TensorFlow – 노드 / 연산 정의 (II)



입력노드 784개, 은닉노드 100개, 출력노드 10개, 학습율, 반복횟수(epochs), 한번에 입력으로 주어지는 데이터 개수인 배치 사이즈 등 설정

②

```
learning_rate = 0.1   # 학습율
epochs = 100          # 반복횟수
batch_size = 100      # 한번에 입력으로 주어지는 MNIST 개수

input_nodes = 784     # 입력노드 개수
hidden_nodes = 100    # 은닉노드 개수
output_nodes = 10     # 출력노드 개수
```

[1] 입력과 출력을 위한 노드 정의(X, T),

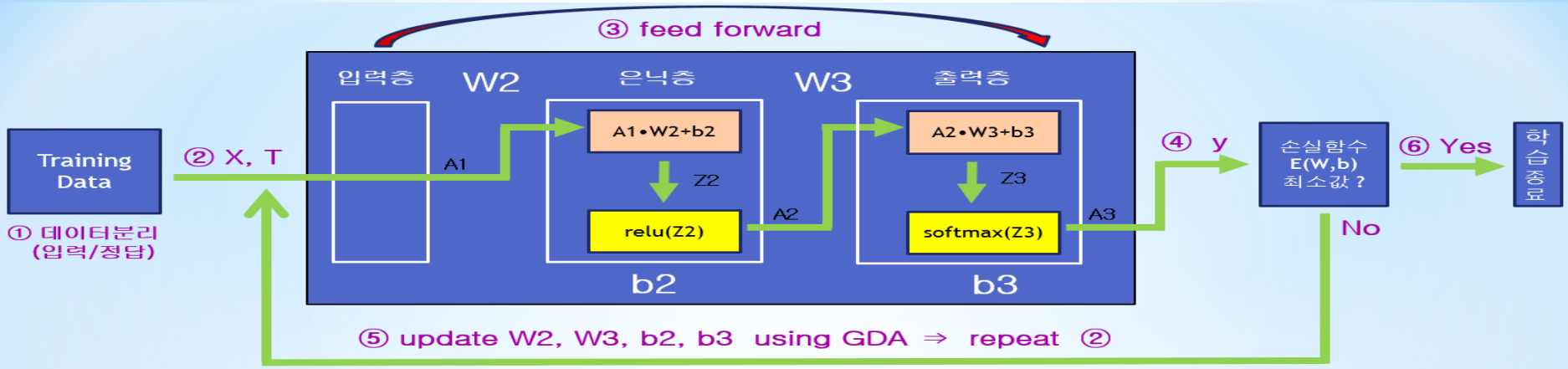
[2] 가중치와 바이어스 정의(W2, W3, b2, b3)

feed forward 수행. 은닉층 출력 값 A2는 sigmoid가 아닌 relu 사용하며, 출력층에서의 선형회귀 값(logits) Z3 를 softmax 함수의 입력으로 넣어주어 출력층의 출력 값 (y=A3)을 계산함

③

```
Z2 = tf.matmul(X, W2) + b2    # 선형회귀 선형회귀 값 Z2
A2 = tf.nn.relu(Z2)           # 은닉층 출력 값 A2, sigmoid 대신 relu 사용

# 출력층 선형회귀  값 Z3, 즉 softmax 에 들어가는 입력 값
Z3 = logits = tf.matmul(A2, W3) + b3

y = A3 = tf.nn.softmax(Z3)
```

TensorFlow – 노드 / 연산 정의 (III)



출력층 선형회귀 값(logits) Z3와 정답 T 를 이용하여 손실 함수 loss 정의. 여기서 한번에 입력되는 배치사이즈가 100 이므로 Z3와 T shape 은 (100X10) 이며, tf.nn.softmax_cross_entropy_with_logits_v2(…) 에 의해 100 개의 데이터에 대해 각각의 소프트맥스가 계산된 후에 정답과의 비교를 통해 크로스 엔트로피 손실 함수 값이 계산되고, tf.reduce_mean(…) 에 의해서 100개의 손실함수 값의 평균이 계산됨

④, ⑤

```
loss = tf.reduce_mean( tf.nn.softmax_cross_entropy_with_logits_v2(logits=Z3, labels=T) )
optimizer = tf.train.GradientDescentOptimizer(learning_rate)

train = optimizer.minimize(loss)
```

출력층 선형회귀 값(logits) Z3와 정답 T 를 이용하여 손실함수 크로스 엔트로피(cross0entropy) 계산

one-hot encoding 에 의해서 출력 층 계산 값 A3와 정답 T 는 (batch_size X 10) shape을 가지는 행렬임. 따라서 argmax 의 두번째 인자에 1을 주어 행 단위로 A3와 T를 비교함. 위 예에서는 batch_size 가 100 이므로 (100X10) 행렬에 대해서 행 단위로 비교하고 있음

⑥

```
# batch_size X 10 데이터에 대해 argmax를 통해 행단위로 비교함
predicted_val = tf.equal( tf.argmax(A3, 1), tf.argmax(T, 1) )

# batch_size X 10 의 True, False 를 1 또는 0 으로 변환
accuracy = tf.reduce_mean(tf.cast(predicted_val, dtype=tf.float32))
```

출력층의 계산 값 A3와 정답 T 에 대해, 행 기준으로 값을 비교함

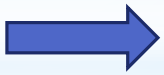
TensorFlow – 노드 / 연산 실행

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

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

    for i in range(epochs):     # 100 번 반복수행

        total_batch = int(mnist.train.num_examples / batch_size)  # 55,000 / 100

        for step in range(total_batch):

            batch_x_data, batch_t_data = mnist.train.next_batch(batch_size)

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

            if step % 100 == 0:
                print("step = ", step, ", loss_val = ", loss_val)

    # Accuracy 확인
    test_x_data = mnist.test.images     # 10000 X 784
    test_t_data = mnist.test.labels     # 10000 X 10

    accuracy_val = sess.run(accuracy, feed_dict={X: test_x_data, T: test_t_data})

    print("₩nAccuracy = ", accuracy_val)
```

```
step =  0 , loss_val =  98.7612
step =  100 , loss_val =  4.2225432
step =  200 , loss_val =  3.0104418
step =  300 , loss_val =  4.0216117
step =  400 , loss_val =  1.4966797
step =  500 , loss_val =  3.3441987
step =  0 , loss_val =  2.2555325
step =  100 , loss_val =  0.7794918
```

.....

```
step =  500 , loss_val =  0.022581069
step =  0 , loss_val =  0.14176598
step =  100 , loss_val =  0.05623051
step =  200 , loss_val =  0.025527965
step =  300 , loss_val =  0.10016465
step =  400 , loss_val =  0.079585046
step =  500 , loss_val =  0.08565387
```



Accuracy = 0.9554

MNIST와 같은 이미지데이터에 대한 인식정확도를 99% 이상으로 높이기 위해서는 신경망 아키텍처를 CNN (Convolutional Neural Network) 으로 전환하는것이 필요함

[확인사항]

[1] accuracy 확인 후, 다음과 같은 index_label_false_list 를 만들고

index_label_false_list = [[index 1, label 1, false_prediction 1],
[index 2, label 2, false_prediction 2],
[……],
[……]]

[2] 오답에 대해서 다음과 같이,
오답(false_prediction)을 확인하고,
plt.savefig(…)를 이용하여 파일로 저장하시오