2-3. MLP

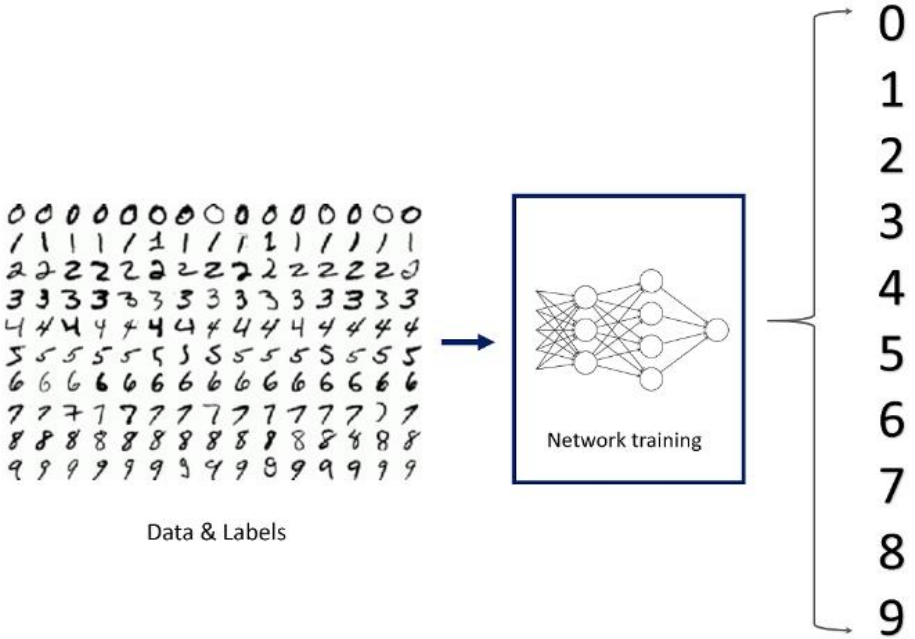
Multi-layer perceptrons

AILAB

Hanyang Univ.

오늘 실습 내용

- **Multi Layer Perceptron to slove XOR in classification**
- **MNIST classifier using MLP**



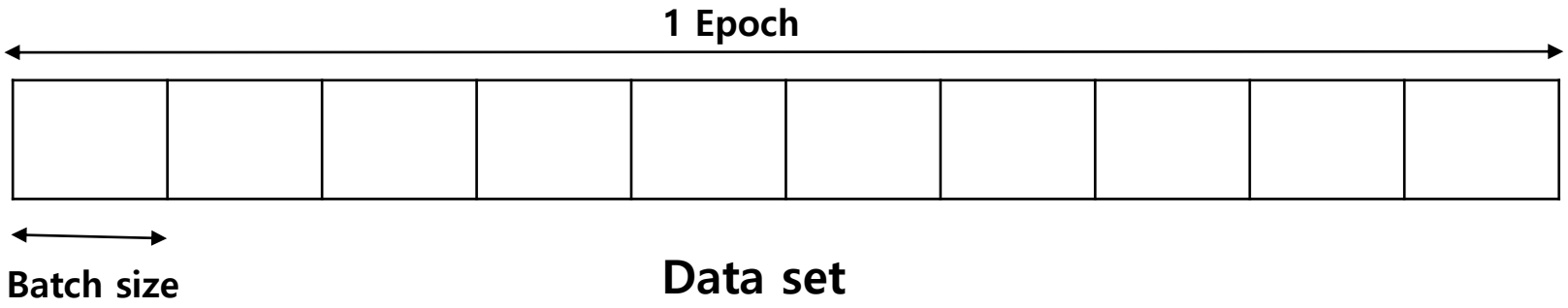
MNIST Dataset and Number Classification [1]

신경망모델 학습 프로세스

- 데이터 processing
- model 디자인
  - layer 종류, 개수 및 뉴런 개수 설정
  - 각 layer 마다의 activation function 설정
- Loss function 설정
- Optimizer 설정
- 학습

모델학습 관련 개념

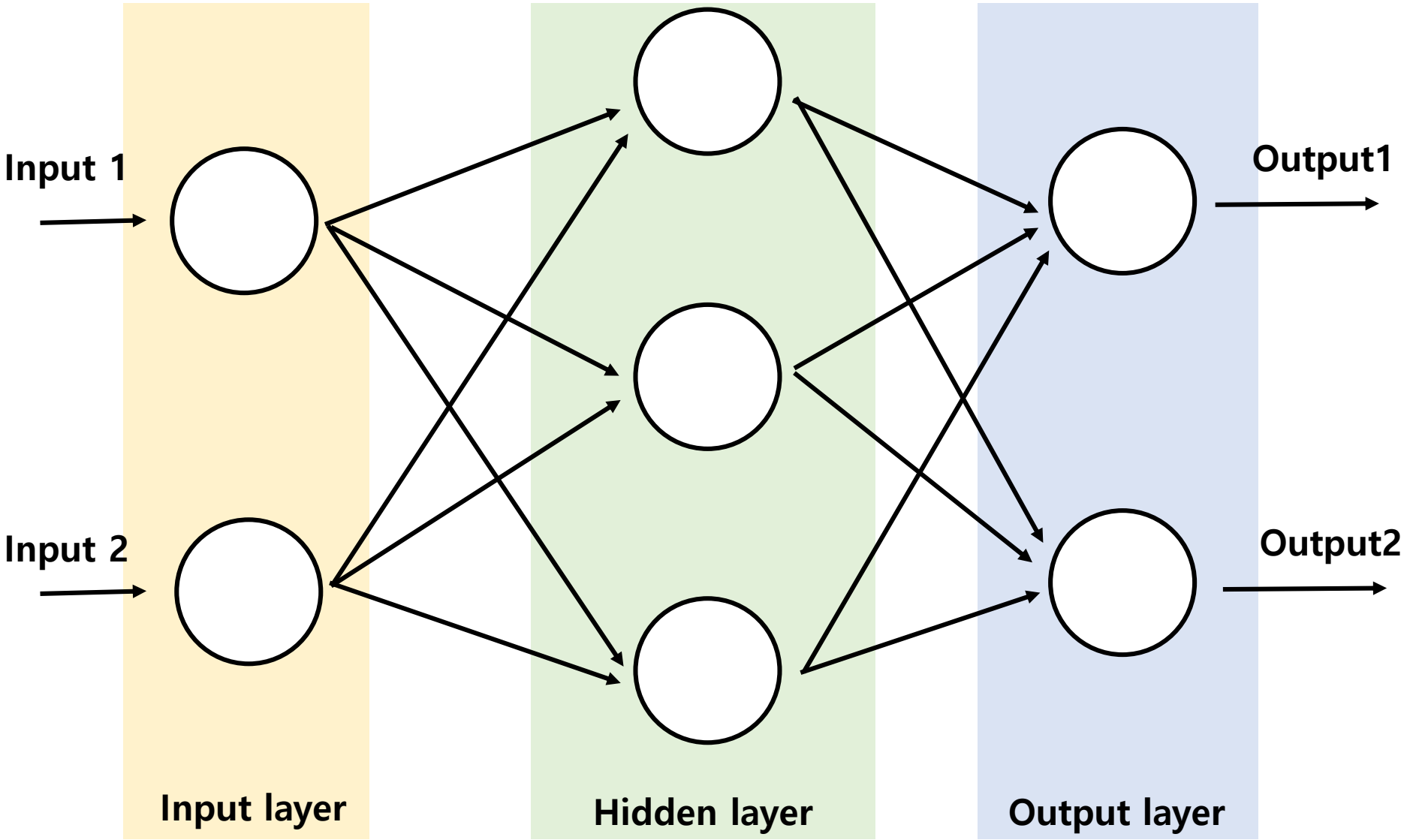
- **Epoch : 전체 Sample 데이터를 학습하는것**
- **Step : 1 step당 weight와 Bias를 1회씩 업데이트 하게됨**
- **Batch Size : 1 Step에서 사용한 데이터의 수**
- **Learning rate : 경사 하강법에서 학습 단계별로 움직이는 학습 속도**
- Ex) Batch Size 가 100, Step이 10이면 약 1000개의 데이터를 이용



학습 관련 실습

```
18  cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(labels=Y, logits=model))
19
20  opt = tf.train.GradientDescentOptimizer(learning_rate=0.1)
21  train_op = opt.minimize(cost)
22
23  init = tf.global_variables_initializer()
24
25  with tf.Session() as sess:
26      sess.run(init)
27      for step in range(8000):
28          sess.run(train_op, feed_dict={X:x, Y:y})
29          if step % 5 == 0:
30              print(step, sess.run(cost, feed_dict={X:x, Y:y}))
```

Multi Layer Perceptron 구성



Output Layer 구성

```
# weight, bias 초기화

W2 = tf.Variable(tf.random_uniform([3, 2], -1., 1.))

b2= tf.Variable(tf.random_uniforml([2], -1., 1.))

# 그 전 레이어의 아웃풋 L1와 가중치행렬 W2을 행렬 곱한 후 b2를 더함

model= tf.matmul(L1, W2) + b2

# softmax 사용

output_softmax = tf.nn.softmax(model)

# argmax 사용

output_argmax = tf.argmax(model, 1)
```

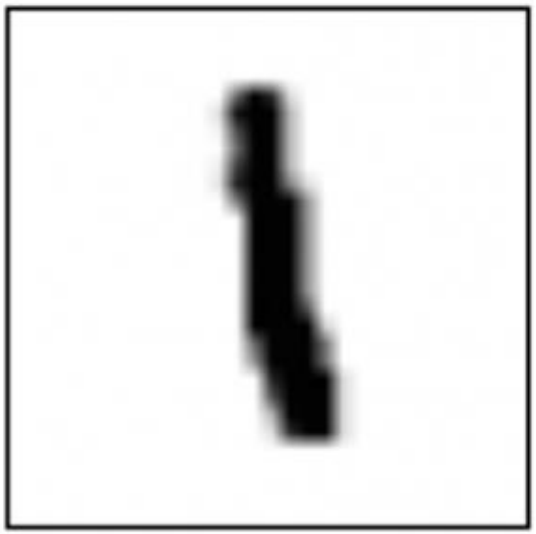
Multi Layer Perceptron 실습

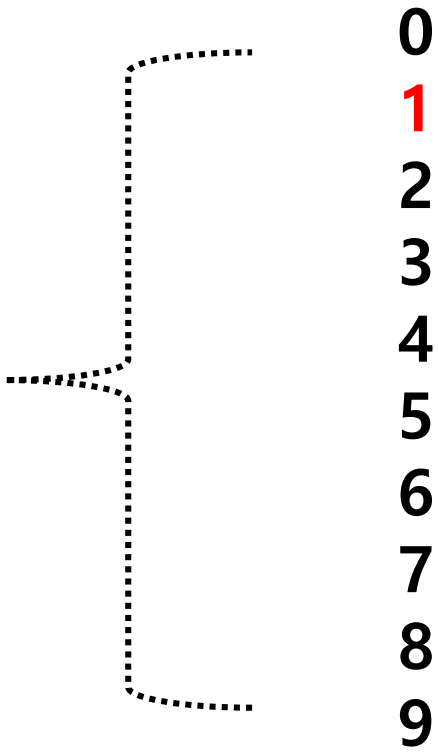
```python
import tensorflow as tf
import numpy as np

x = [[0, 0], [0, 1], [1, 0], [1, 1]]
y = [[1, 0], [0, 1], [0, 1], [1, 0]]

X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

W1 = tf.Variable(tf.random_uniform([2, 3], -1., 1.))
b1 = tf.Variable(tf.random_uniform([3], -1., 1.))
L1 = tf.sigmoid(tf.matmul(X, W1) + b1)

W2 = tf.Variable(tf.random_uniform([3, 2], -1., 1.))
b2 = tf.Variable(tf.random_uniform([2], -1., 1.))
model = tf.matmul(L1, W2) + b2
output_softmax = tf.nn.softmax(model)
output_argmax = tf.argmax(model, 1)

cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(labels=Y, logits=model))

opt = tf.train.GradientDescentOptimizer(learning_rate=0.1)
train_op = opt.minimize(cost)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for step in range(8000):
        _, total_cost = sess.run([train_op, cost], feed_dict={X:x, Y:y})

        if step % 5 == 0:
            print(step, total_cost)

    print("predict: ", sess.run(model, feed_dict={X:x}))
    print("predict with softmax: ", sess.run(output_softmax, feed_dict={X:x}))
    print("predict with argmax: ", sess.run(output_argmax, feed_dict={X:x}))
```

학습결과 확인

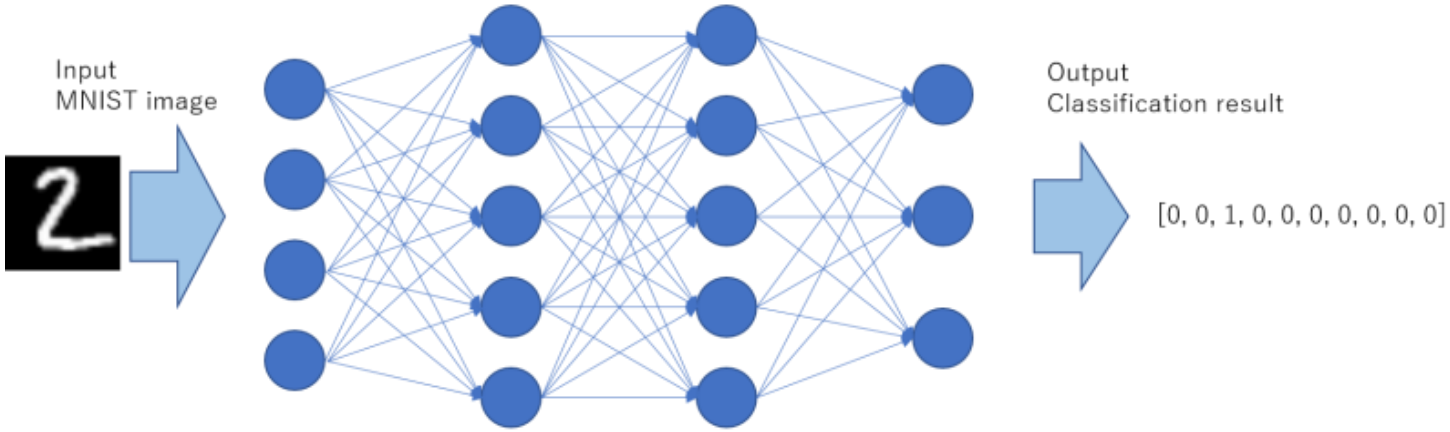
```
# 모델의 예측 값과 실제 레이블 값(Y)를 비교
# ex) tf.argmax([[1, 2, 1, 5, 3], [2, 3, 6, 1, 0]], 1) -> [3, 2]
# ex) tf.equal([1, 2, 4, 5], [1, 2, 3, 5]) -> [True, True, False, True]
is_correct = tf.equal(tf.argmax(model, 1), tf.argmax(Y, 1))

# accuracy 계산
# ex) tf.cast([True, True, False, True], tf.float32) -> [1., 1., 0., 1.]
# ex) tf.reduce_mean([1., 1., 0., 1.]) -> 0.75
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))
```

MNIST classifier using MLP





신경망모델의 구성


Input
MNIST image
Output
Classification result
[0, 0, 1, 0, 0, 0, 0, 0, 0, 0]

과제

- MLP를 이용하여 MNIST Classifier model 만들기
  - Skeleton code 이용
  - **Layers(input, hidden, output) 디자인 하기**
  - **Activation function 사용하기**
  - Loss function 설정하기
  - Optimizer 사용하기
  - 90% 이상의 정확성 보이기

과제 - Skeleton Code(1/2)

```python
import tensorflow as tf

from tensorflow.examples.tutorials.mnist import input_data

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

X = tf.placeholder(tf.float32, [None, 784])
Y = tf.placeholder(tf.float32, [None, 10])

```

…

Assignment) Design the Layers

…

과제 - Skeleton Code(2/2)

```
24
25  cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits=model, labels=Y))
26  optimizer = tf.train.GradientDescentOptimizer(0.001).minimize(cost)
27
28  init = tf.global_variables_initializer()
29  sess = tf.Session()
30  sess.run(init)
31
32  batch_size = 100
33  total_batch = int(mnist.train.num_examples / batch_size)
34
35  for epoch in range(15):
36      total_cost = 0
37
38      for i in range(total_batch):
39          batch_xs, batch_ys = mnist.train.next_batch(batch_size)
40
41          _, cost_val = sess.run([optimizer, cost], feed_dict={X: batch_xs, Y:batch_ys})
42          total_cost += cost_val
43
44      print('Epoch :', '%04d' % (epoch + 1),
45          'Avg. cost =', '{:.3f}'.format(total_cost / total_batch))
46
47
48  is_correct = tf.equal(tf.arg_max(model, 1), tf.math.argmax(Y, 1))
49  accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))
50  print('정확도', sess.run(accuracy, feed_dict={X: mnist.test.images, Y: mnist.test.labels}))
51
```

코드설명 - 데이터 준비

```
import tensorflow as tf

# mnist data 다운로드

from tensorflow.examples.tutorials.mnist import input_data

# one-hot vector 입력하기

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

코드설명 - Placeholders 설정

```
# 텐서 X는 MNIST 이미지를 784개의 실수 벡터로 저장하는데 사용됨,
# None 은 아직 정해지지 않은, 사용될 이미지의 배치 사이즈
X = tf.placeholder(tf.float32, [None, 784])
# 결과는 one_hot 방식.
Y = tf.placeholder(tf.float32, [None, 10])
```

코드설명 - 신경망 모델 학습(1/2)

```
# 세션 시작 전에 모든 변수를 초기화
init = tf.global_variables_initializer()
# 그래프 생성 및 초기화 실행
sess = tf.Session()
sess.run(init)
# 학습을 1회 할 때마다 100개의 데이터 사용
batch_size = 100
# 총 트레이닝에 사용되는 example은 55,000개 / batch_size 는 100
total_batch = int(mnist.train.num_examples / batch_size)
```

코드설명 - 신경망 모델 학습(2/2)

```
for epoch in range(15):    # 전체데이터를 총 15번 학습
    total_cost = 0
    for i in range(total_batch):    # total_batch = 550
        # 학습 데이터셋에서 무작위로 샘플링한 100개의 데이터로 구성된 'batch'를 가져옴
        batch_xs, batch_ys = mnist.train.next_batch(batch_size)
        # cost_val에 batch의 학습 loss 저장
        _, cost_val = sess.run([optimizer, cost], feed_dict={X: batch_xs, Y: batch_ys})
        # 전체 데이터의 loss를 알기 위해 각 batch의 학습 loss를 total_cost에 더해줌
        total_cost += cost_val

    # 1epoch 마다 정해진 format에 맞춰서 Average cost 출력
    print('Epoch:', '%04d' % (epoch + 1), 'Avg. cost =', '{:.3f}'.format(total_cost / total_batch))

print('최적화 완료')
```

코드설명 - 학습결과 확인

```
# 모델의 예측 값과 실제 레이블 값(Y)를 비교
# ex) tf.argmax([[1, 2, 1, 5, 3], [2, 3, 6, 1, 0]], 1) -> [3, 2]
# ex) tf.equal([1, 2, 4, 5], [1, 2, 3, 5]) -> [True, True, False, True]
is_correct = tf.equal(tf.argmax(model, 1), tf.math.argmax(Y, 1))

# accuracy 계산
# ex) tf.cast([True, True, False, True], tf.float32) -> [1., 1., 0., 1.]
# ex) tf.reduce_mean([1., 1., 0., 1.]) -> 0.75
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))

print('정확도:', sess.run(accuracy, feed_dict={X: mnist.test.images, Y: mnist.test.labels}))
```

과제 - 결과


Anaconda Prompt
Epoch : 0001 Avg. cost = 0.480
Epoch : 0002 Avg. cost = 0.183
Epoch : 0003 Avg. cost = 0.119
Epoch : 0004 Avg. cost = 0.088
Epoch : 0005 Avg. cost = 0.067
Epoch : 0006 Avg. cost = 0.052
Epoch : 0007 Avg. cost = 0.043
Epoch : 0008 Avg. cost = 0.033
Epoch : 0009 Avg. cost = 0.028
Epoch : 0010 Avg. cost = 0.021
Epoch : 0011 Avg. cost = 0.018
Epoch : 0012 Avg. cost = 0.015
Epoch : 0013 Avg. cost = 0.017
Epoch : 0014 Avg. cost = 0.011
Epoch : 0015 Avg. cost = 0.010
WARNING:tensorflow:From tensor_test.py:49: arg_max (from tensorflow.python.ops.gen_math_ops) is deprecated and will be r
emoved in a future version.
Instructions for updating:
Use `argmax` instead
정확도 0.975

과제

- 소스와 결과 캡쳐 GitLab에 제출
- 과제 기한 : **다음주 수요일 23:59** 까지
- 수업시간에 한 경우 바로 검사받고 **GitLab**에 제출
- GitLab 관련 사용법은 첨부 파일 확인