00 목차

2018년 9월 27일 목요일 오후 6:14

01 텐서플로우

2018년 8월 13일 월요일 오후 4:34

Table of Contents

신경망 데이터 학습 순서

1. MNIST
2. cifar10
3. 개/고양이
4. 폐결절/정상폐
5. 이파리 사진

텐서플로우 란?

텐서플로우(TensorFlow)는 기계학습과 딥러닝을 위해 구글에서 만든 오픈소스 라이브러리이다

텐서플로우의 특징

1. 데이터 플로우 그래프를 통한 풍부한 표현력

cafe.daum.net/oracleoracle 참고할 것
데이터 플로우 그래프 (Data Flow Graph) 방식을 사용하였고 그래프를 시각화 하려면 '텐서보드' 를 사용하면 된다

2. 코드 수정없이 CPU/GPU 모드로 동작

한국시간 2016년 11월 29일에 TensorFlow v0.12.0 RC0 이 업데이트 되었고 2016년 11월 29일에 나온 버전의 핵심 변경사항은 Window 에서 GPU 버전의 텐서플로우를 지원한다는 것이다. 예전에는 우분투에서만 가능했었다 (GPU)

텐서플로우 용어 설명

1. 오퍼레이션 (Operation)

그래프 상의 노드는 오퍼레이션 (줄임말 op) 로 불린다.

오퍼레이션은 하나 이상의 텐서를 받을 수 있다.
오퍼레이션은 계산을 수행하고, 결과를 하나 이상의 텐서로 반환할 수 있다.

2. 텐서 (Tensor)

내부적으로 모든 데이터는 텐서를 통해 표현된다.
텐서는 일종의 다차원 배열인데 그래프 내의 오퍼레이션 간에 텐서가 전달된다.

3. 세션 (Session)

그래프를 실행하기 위해서는 세션 객체가 필요하다.
세션은 오퍼레이션의 실행환경을 캡슐화한 것이다.

4. 변수 (Variable)

변수는 그래프의 실행 시, 파라미터를 저장하고 갱신하는데 사용된다.
메모리 상에서 텐서를 저장하는 버퍼 역할을 한다.

텐서플로우 설치

```
pip install tensorflow
```

기본 실습

텐서플로우 기본 실습 1

```
import tensorflow as tf

sess = tf.Session()  # 그래프를 실행할 세션을 구성한다
hello = tf.constant('Hello, Tensorflow')

print(sess.run(hello))
print(str(sess.run(hello), encoding='utf-8'))
```

b'Hello, Tensorflow'
Hello, Tensorflow

설명)

파이썬 3버전은 문자열 유니코드가 기본이므로 str에서 encoding 처리를 해줘야 binary 타입을 유니코드 타입으로 반환한다

위에서 변수를 정의했지만, 실행이 정의한 시점에서 실행되는 것은 아니다.
Session 객체와 run 메소드를 사용할 때 계산이 되어 실행이 된다

텐서플로우 기본 실습 2

```
import tensorflow as tf
```

```
x = tf.constant(35, name='x')  # x 라는 상수값을 만들고 숫자 35를 지정
y = tf.Variable(x+5, name='y')  # y 라는 변수를 만들고 방정식 y = x + 5 로 정의함

model = tf.global_variables_initializer()  # 변수 초기화

with tf.Session() as sess:
    sess.run(model)
    print(sess.run(y))
```

40

텐서플로우 기본 실습 3 : 빌딩구조와 실행구조 (Session) 가 분리되어 있음을 이해하는 실습

```
import tensorflow as tf

x2 = tf.linspace(-1.0, 1.0, 10)  # -1 ~ 1 사이에 숫자 중에 10개를 랜덤으로 출력하겠다

# print(x2)  # Tensor("LinSpace_1:0", shape=(10,), dtype=float32)

g = tf.get_default_graph()
print([op.name for op in g.get_operations()])

print("")

sess = tf.Session()
print(sess.run(x2))
sess.close()
```

['LinSpace/start', 'LinSpace/stop', 'LinSpace/num', 'LinSpace', 'LinSpace_1/start', 'LinSpace_1/stop', 'LinSpace_1/num', 'LinSpace_1']

[-1. -0.7777778 -0.5555556 -0.3333333 -0.1111111 0.11111116
  0.33333337 0.5555556 0.7777778 1.]

문제 1) 아래의 코드를 실행하여 구성하는 부분과 실행하는 부분을 깔끔하게 정리하도록 한다

답)

```
import tensorflow as tf

################### 모델(그래프)을 구성하는 부분 ##################

hello = tf.constant('Hello, Tensorflow!')
a = tf.constant(10)
```

```
b = tf.constant(32)
c = tf.add(a, b)



################### 모델(그래프)을 실행하는 부분 ##################

sess = tf.Session()
print(str(sess.run(hello), encoding='utf-8'))
print(sess.run(c))

sess.close()
```

Hello, Tensorflow!
42

문제 2) 아래의 모델(그래프) 을 실행하시오 (with 절을 사용해서 수행하시오)

답)

```
import tensorflow as tf

################### 모델(그래프)을 구성하는 부분 ##################

a = tf.add(1, 2)
b = tf.multiply(a, 3)
c = tf.add(4, 5)
d = tf.multiply(c, 6)
e = tf.multiply(4, 6)
f = tf.div(c, 6)
g = tf.add(b, d)
h = tf.multiply(g, f)



################### 모델(그래프)을 실행하는 부분 ##################

with tf.Session() as sess:
    print(sess.run(h))
```

63

텐서플로우 실행구조

```
session 은 fetch와 feed 2가지 방법으로 처리

feed: placeholder 에 값을 넣어 실행하는 방법
fetch: 연산 결과를 fetch (가져오는) 방법
```

파이썬 기본문법과 텐서플로우 문법 비교

Case 1) 1 ~ 5 까지의 숫자를 출력하는 경우

파이썬

```
x = 0
for i in range(5):
    x = x + 1
    print(x)
```

텐서플로우

```
import tensorflow as tf

x = tf.Variable(0, name = 'x')
model = tf.global_variables_initializer()

with tf.Session() as sess:
    for i in range(5):
        sess.run(model)
        x = x + 1
        print(sess.run(x))
```

문제 3) 구구단 2단을 텐서로 출력하시오

보기)

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
```

파이썬

```
for i in range(1, 10):
    x = 2
    print(x, "x", i, "=", x*i)
```

텐서플로우

```
import tensorflow as tf

x = tf.Variable(2, name = 'x')
y = tf.Variable(1, name = 'y')
model = tf.global_variables_initializer()

with tf.Session() as sess:
    for i in range(1, 10):
        sess.run(model)
        z = tf.multiply(x, y)
        print(sess.run(x), "x", sess.run(y), "=", sess.run(z))
        y = y + 1
```

문제 4) 구구단 2~9단을 텐서로 출력하시오

보기)

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
```

```
4 x 5 = 20
4 x 6 = 24
```

파이썬

```
for j in range(2, 10):
    for i in range(1, 10):
        print(j, "x", i, "=", j*i)
```

텐서플로우

```
import tensorflow as tf

x=tf.Variable(0,name='x')
y=tf.Variable(0,name='y')
z=tf.multiply(x,y, name='z')

model=tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(model)
    for i in range(2,10):
        for j in range(1,10):
            print(i,'x',j,'=',sess.run(z, feed_dict={x:i, y:j}))
```

Tensorflow 의 실행에 feed 예시

```
Session 은 feed 일 경우는 반드시 feed_dict 으로 처리값을 할당해야 한다.
```

예시

```
import  tensorflow  as  tf

a = tf.placeholder("float")
b = tf.placeholder("float")

y = tf.multiply(a,b)
z = tf.add(y,y)

with  tf.Session() as sess:
    sess.run(tf.global_variables_initializer() )
    print ( sess.run( y, feed_dict={a:3, b:3} ) )
    print ( sess.run( z, feed_dict={a:4, b:4} ) )
```

Numpy 와 Tensorflow 의 차이

Numpy	TensorFlow
a = np.zeros((2,2)); b = np.ones((2,2))	a = tf.zeros((2,2)), b = tf.ones((2,2))
np.sum(b, axis=1)	tf.reduce_sum(a,reduction_indices=[1])
a.shape	a.get_shape()
np.reshape(a, (1,4))	tf.reshape(a, (1,4))
b * 5 + 1	b * 5 + 1
np.dot(a,b)	tf.matmul(a, b)
a[0,0], a[:,0], a[0,:]	a[0,0], a[:,0], a[0,:]

문제 5) zero 와 숫자 1 을 채워넣는 배열을 생성하는 아래의 numpy 문법을 tensor 로 구현하시오

numpy

```
import numpy as np

a = np.zeros((2,2))
b = np.ones((2,2))

print (a)
print (b)
```

tensorflow

```
import tensorflow as tf

a = tf.zeros((2,2))
b = tf.ones((2,2))

with tf.Session() as sess:
    print (sess.run(a))
    print (sess.run(b))
```

문제 6) 아래의 numpy 문법을 텐써 플로우로 구현하시오

numpy

```
import numpy as np

a = np.array([0,0,0,1,0,0,0,0,0,0])
```

```
print ( np.argmax(a, axis=0) )
```

tensorflow

```
import tensorflow as tf
import numpy as np

a = np.array([0,0,1,0,0,0,0,0,0])
b = tf.argmax(a, axis=0)

with tf.Session() as sess:
    print(sess.run(b))
```

문제 7) 아래의 numpy 문법을 텐써 플로우로 구현하시오

numpy

```
import tensorflow as tf
import numpy as np

a = np.array([[[1, 2, 3],
               [2, 1, 4],
               [5, 2, 1],
               [6, 3, 2]],
              [[5, 1, 3],
               [1, 3, 4],
               [4, 2, 6],
               [3, 9, 3]],
              [[4, 5, 6],
               [7, 4, 3],
               [2, 1, 5],
               [4, 3, 1]]])

np.sum(a, axis=0)
```

tensorflow

```
import tensorflow as tf
import numpy as np

a = np.array([[[1, 2, 3],
               [2, 1, 4],
               [5, 2, 1],
               [6, 3, 2]],
              [[5, 1, 3],
```

```
                [1, 3, 4],
                [4, 2, 6],
                [3, 9, 3]],
               [[4, 5, 6],
                [7, 4, 3],
                [2, 1, 5],
                [4, 3, 1]]])
b = tf.reduce_sum(a, reduction_indices=[0])

with tf.Session() as sess:
    print(sess.run(b))
```

```
[[10  8 12]
 [10  8 11]
 [11  5 12]
 [13 15  6]]
```

문제 8) 아래의 numpy 문법을 텐써 플로우로 구현하시오

numpy

```
import numpy as np
a = np.array([ i for i in range(144)])
b = a.reshape(12, 12)

print(b)
print(b.shape)
```

tensorflow

```
import tensorflow as tf
import numpy as np

a = np.array([ i for i in range(144)])
b = tf.reshape(a, (12,12))

with tf.Session() as sess:
    print(sess.run(b))
    print(b.get_shape())
```

```
[[  0   1   2   3   4   5   6   7   8   9  10  11]
 [ 12  13  14  15  16  17  18  19  20  21  22  23]
 [ 24  25  26  27  28  29  30  31  32  33  34  35]
 [ 36  37  38  39  40  41  42  43  44  45  46  47]
```

```
 [ 48  49  50  51  52  53  54  55  56  57  58  59]
 [ 60  61  62  63  64  65  66  67  68  69  70  71]
 [ 72  73  74  75  76  77  78  79  80  81  82  83]
 [ 84  85  86  87  88  89  90  91  92  93  94  95]
 [ 96  97  98  99 100 101 102 103 104 105 106 107]
 [108 109 110 111 112 113 114 115 116 117 118 119]
 [120 121 122 123 124 125 126 127 128 129 130 131]
 [132 133 134 135 136 137 138 139 140 141 142 143]]
(12, 12)
```

문제 9) (텐서플로우로 구현한 단층신경망 이해에 중요 문법) 아래의 numpy 배열을 열 단위 sum을 출력하시오

numpy

```
import numpy as np
import tensorflow as tf

x = np.arange(6).reshape(2, 3)
np.sum(x, axis=0)
```

tensorflow

```
import numpy as np
import tensorflow as tf

a = np.arange(6).reshape(2,3)
b = tf.reduce_sum(a, 0)

with tf.Session() as sess:
    print(sess.run(b))
```

```
[3 5 7]
```

문제 10) (텐서플로우로 구현한 단층신경망 이해에 중요 문법)
숫자 0으로 채워진 2 x 3 행렬을 만들고 숫자 1로 채워진 2 x 3 행렬을 만들고 두 행렬의 합을 구하시오

답)

```
import numpy as np
import tensorflow as tf

a = tf.zeros([2, 3])
```

```
b = tf.ones([2, 3])
result = tf.add(a, b)

with tf.Session() as sess:
#     print(sess.run(a))
#     print(sess.run(b))
    print(sess.run(result))
```

```
[[1. 1. 1.]
 [1. 1. 1.]]
```

문제 11) 숫자 2로 채워진 2 x 3 행렬을 x라는 변수로 만들고 숫자 3으로 채워진 3 x 2 행렬을 y로 만든 후에 x 행렬과 y 행렬의 내적을 구하시오

답)

```
import numpy as np
import tensorflow as tf

x = tf.placeholder('float', [2, 3])
y = tf.placeholder('float', [3, 2])
result = tf.matmul(x, y)

with tf.Session() as sess:
#     print(sess.run(x, feed_dict={x:[[2,2,2],[2,2,2]]}))
#     print(sess.run(y, feed_dict={y:[[3,3], [3,3], [3,3]]}))
    print(sess.run(result, feed_dict={x:[[2,2,2],[2,2,2]],
                                      y:[[3,3], [3,3], [3,3]]}))
```

복습

1. 텐서플로우를 사용했을 때의 이점?

a. 코드가 간결해진다 (모델 생성 부분, 모델 실행 부분)

모델 생성 부분 : 오퍼레이션, 변수

모델 실행 부분 : 세션

b. 신경망 구현에 필요한 모든 함수가 내장

c. 속도가 빠르다

d. GPU 를 사용할 수 있다

문제 12) (텐서플로우의 cast 함수의 이해) 아래의 배열의 True를 1로 변경하고 False를 0으로 변경하시오

보기)

```
import tensorflow as tf

correct_prediction = [ True, False , True  ,True  ,True  ,True  ,True,  True  ,True  ,True  ,True  ,True
  ,True  ,True  ,True, False , True  ,True, False , True  ,True  ,True  ,True  ,True
  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True
  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True,
  True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True
  ,True  ,True  ,True  ,True  ,True  ,True ,False , True  ,True  ,True  ,True  ,True
  ,True  ,True, False , True, False , True  ,True  ,True  ,True  ,True  ,True  ,True
  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True
 ,False , True  ,True  ,True ]
```

답)

```
with tf.Session() as sess:
    a = tf.cast(correct_prediction, "float")
    print(sess.run(a))
```

```
[1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1.
 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 0. 1. 1. 1.]
```

문제 13) 위의 출력된 결과에서 전체의 개수 중에 1이 몇 개나 되는지 즉, 정확도를 출력하시오

답)

```
with tf.Session() as sess:
    a = tf.cast(correct_prediction, "float")  # cast : 형 변환하는 함수
    b = tf.reduce_mean(a)
    print(sess.run(b))
```

0.93

취업을 위한 포트폴리오 목록 정리

1. MNIST (필기체) 데이터로 CNN 3층 신경망 구현
(정확도 : 훈련(99%), 테스트(99%))

2. cifa10 데이터로 CNN 4층 신경망 구현
(정확도 : 훈련(90%), 테스트(70%))

3. 개/고양이 이미지 데이터로 CNN 4층 신경망 구현
(정확도 : 훈련(?), 테스트(?))

4. 폐결절 사진/정상폐 사진 이미지 구분 CNN 4층 신경망 구현

5. 내가 수행한 포트폴리오

6. segmentation, object detection 등

---

02 MNIST 데이터로 단층 신경망 구현하기

2018년 8월 13일 월요일 오후 4:34

문제 14) 텐서플로우에 기본적으로 내장되어 있는 MNIST 데이터를 가져오시오

답)

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
batch_xs, batch_ys = mnist.train.next_batch(100)

print(batch_xs.shape)
print(batch_ys.shape)
```

(100, 784)
(100, 10)

문제 15) 위의 MNIST 데이터 중에 train 데이터의 라벨을 one hot 인코딩하지 말고 숫자로 100개의 라벨을 가져오시오

답)

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot=False)
batch_xs, batch_ys = mnist.train.next_batch(100)  # xs : train data, ys : label

batch_ys
```

```
array([5, 5, 6, 3, 8, 0, 5, 7, 6, 4, 2, 1, 4, 4, 8, 3, 0, 6, 2, 7, 9, 4,
       2, 4, 1, 3, 4, 1, 6, 8, 4, 7, 4, 3, 6, 7, 7, 8, 6, 8, 6, 4, 3, 4,
       8, 6, 8, 9, 8, 4, 7, 0, 5, 0, 4, 3, 5, 1, 2, 5, 2, 7, 5, 0, 1, 9,
       3, 6, 7, 7, 0, 3, 9, 7, 7, 1, 3, 1, 7, 8, 2, 0, 9, 1, 1, 9, 4, 7,
       1, 5, 0, 5, 1, 1, 4, 7, 9, 9, 1, 5], dtype=uint8)
```

문제 16) 이번에는 test data 와 test data 의 label 100개를 가져오는데 shape 만 출력해보시오

답)

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
batch_ts, batch_ty = mnist.test.next_batch(100)  # xs : test data, ys : label

print(batch_ts.shape)
print(batch_ty.shape)
```

```
(100, 784)
(100, 10)
```

문제 17) 숫자 2로 채워진 행렬 2 x 3 행렬을 텐서플로우로 출력하시오

답)

```
import tensorflow as tf
import numpy as np

a = tf.placeholder('float')
b = tf.fill((2,3), a)

with tf.Session() as sess:
    print(sess.run(b, feed_dict={a:2}))
```

```
[[2. 2. 2.]
 [2. 2. 2.]]
```

답 2)

```
import tensorflow as tf
import numpy as np

a = tf.placeholder('float', (2, 3))

with tf.Session() as sess:
    print(sess.run(a, feed_dict={ a : [[2,2,2], [2,2,2]]}))
```

문제 18) 위의 답2에서 행 부분을 None으로 쓰면 실행되는지 확인하고 의미를 확인하시오

답)

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

```
a = tf.placeholder('float', (None, 3))

with tf.Session() as sess:
    print(sess.run(a, feed_dict={ a : [[2,2,2], [2,2,2]]}))
```

설명)

```
앞의 행의 개수가 몇 개로 들어오던 관계 없다는 의미

아래의 식을 실행해보고 의미를 확인해라
```

예제)

```
import tensorflow as tf
import numpy as np

a = tf.placeholder('float', (None, 3))

with tf.Session() as sess:
    print(sess.run(a, feed_dict={ a : [[2,2,2], [2,2,2]]}))
    print(sess.run(a, feed_dict={ a : [[2,2,2], [2,2,2], [2,2,2]]}))
```

```
[[2. 2. 2.]
 [2. 2. 2.]]
[[2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]]
```

문제 19) MNIST 데이터 784 (28 x 28) 개의 맞게 x 변수를 선언하고 배치로 입력된 데이터의 개수는 몇 개든 상관없게 None 으로 변수를 만들고 MNIST 데이터를 x 변수에 100개를 담고 출력해보시오

답)

```
import tensorflow as tf
import numpy as np

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
batch_xs, batch_ys = mnist.train.next_batch(100)
# xs : train data, ys : label, Shuffle=True 로 디폴트설정되어 랜덤추출

x = tf.placeholder('float', (None, 784))

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

```
# sess = tf.Session() : 텐서플로우 그래프 연산을 시작하게끔 세션 객체 생성하는 것
    print(sess.run(x, feed_dict={ x : batch_xs }).shape)
```

(100, 784)

문제 20) 위의 코드를 수정해서 train data 100개 뿐만 아니라 label도 100개로 출력하게끔 코드를 수정하시오

답)

```
import tensorflow as tf
import numpy as np

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
batch_xs, batch_ys = mnist.train.next_batch(100)
# xs : train data, ys : label, Shuffle=True 로 디폴트설정되어 랜덤추출

x = tf.placeholder('float', (None, 784))
y = tf.placeholder('float', (None, 10))

# sess = tf.Session() : 텐서플로우 그래프 연산을 시작하게끔 세션 객체 생성하는 것
with tf.Session() as sess:
    print(sess.run(x, feed_dict={ x : batch_xs }).shape)
    print(sess.run(y, feed_dict={ y : batch_ys }).shape)
```

(100, 784)
(100, 10)

문제 21) (텐서플로우로 가중치를 랜덤으로 생성하는 방법) 2 x 3 행렬로 -1 에서 1 사이의 난수를 생성하는 변수를 W 로 생성하고 실행하시오

답)

```
import tensorflow as tf

W = tf.Variable(tf.random_uniform([2,3], -1, 1), name = 'W')
init = tf.global_variables_initializer()   # 왜 변수 초기화를 해야하지..?

with tf.Session() as sess:
    sess.run(init)
    print(sess.run(W))
```

[[-1. -1. -1.]

[-1. -1. -1.]]

문제 22) 위에서 배치 단위로 불러오는 입력 데이터 100 x 784 와 내적할 가중치 행렬 W를 784 x 50 으로 생성하시오

답)

```
import tensorflow as tf

W = tf.Variable(tf.random_uniform([784,50], -1, 1), name = 'W')
init = tf.global_variables_initializer()   # 왜 변수 초기화를 해야하지..?

with tf.Session() as sess:
    sess.run(init)
    print(sess.run(W).shape)
```

(784, 50)

문제 23) 위에서 만든 입력데이터 (100 x 784) 와 지금 만든 가중치 (785 x 50) 행렬과 내적을 한 결과를 출력하시오

답)

```
import tensorflow as tf
import numpy as np

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
batch_xs, batch_ys = mnist.train.next_batch(100)

x = tf.placeholder('float', (None, 784))
W = tf.Variable(tf.random_uniform([784,50], -1, 1), name = 'W')
result = tf.matmul(x, W)

init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)
    print(sess.run(result, feed_dict={x:batch_xs}).shape)
```

(100, 50)

문제 24) 1 x 50 으로 bias 를 생성하는데 변수를 b로 해서 생성하고 숫자를 다 1로 채우시오

답)

```
import tensorflow as tf
import numpy as np

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
batch_xs, batch_ys = mnist.train.next_batch(100)

x = tf.placeholder('float', (None, 784))
W = tf.Variable(tf.random_uniform([784,50], -1, 1), name = 'W')
b = tf.Variable(tf.ones([50]))
result = tf.matmul(x, W)

init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)
    print(sess.run(result, feed_dict={x:batch_xs}).shape)
    print(b.shape)
```

(100, 50)
(50,)

문제 25) 문제 23번에서 구한 두 행렬의 내적과 24번의 bias 의 합을 출력하시오

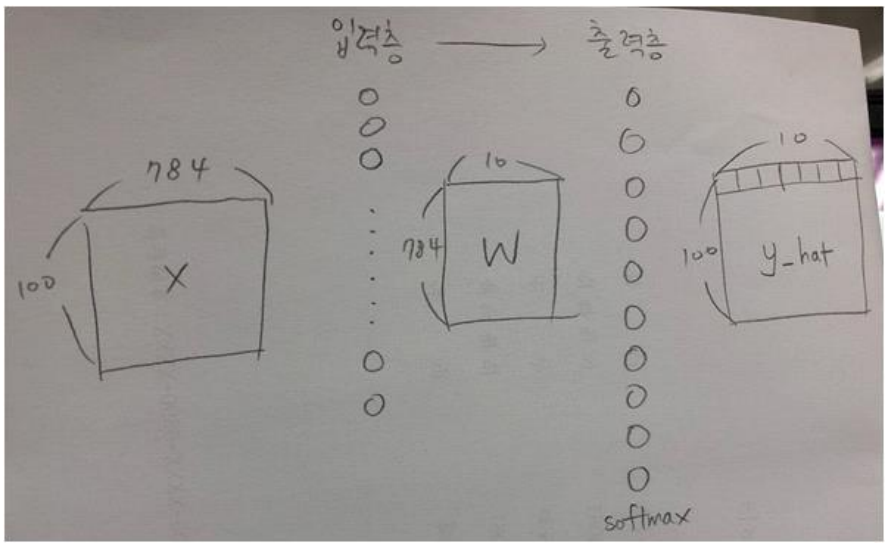
답)

```
import tensorflow as tf
import numpy as np

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
batch_xs, batch_ys = mnist.train.next_batch(100)

x = tf.placeholder('float', (None, 784))
W = tf.Variable(tf.random_uniform([784,50], -1, 1), name = 'W')
b = tf.Variable(tf.ones([50]))
result = tf.matmul(x, W) + b

init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)
    print(sess.run(result, feed_dict={x:batch_xs}).shape)
```

(100, 50)

문제 26) 문제 25번에서 구한 가중의 합의 결과인 result 값을 시그모이드 함수에 입력해서 출력한 결과를 출력하시오

보기)



답)

```
import tensorflow as tf
import numpy as np

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
batch_xs, batch_ys = mnist.train.next_batch(100)

x = tf.placeholder('float', (None, 784))
W = tf.Variable(tf.random_uniform([784,50], -1, 1), name = 'W')
b = tf.Variable(tf.ones([50]))
y = tf.matmul(x, W) + b
y_hat = tf.nn.sigmoid(y)

init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)
    print(sess.run(y_hat, feed_dict={x:batch_xs}))
```

문제 27) 위의 활성화 함수를 Relu 로 변경해 보시오

답)

```
import tensorflow as tf
import numpy as np

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
batch_xs, batch_ys = mnist.train.next_batch(100)

x = tf.placeholder('float', (None, 784))
W = tf.Variable(tf.random_uniform([784,50], -1, 1), name = 'W')
b = tf.Variable(tf.ones([50]))
y = tf.matmul(x, W) + b
y_hat = tf.nn.relu(y)

init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)
    print(sess.run(y_hat, feed_dict={x:batch_xs}))
```

문제 28) 위에서 출력한 y_hat 의 결과를 softmax 함수를 통과시킨 결과가 어떻게 되는지 출력하시오

답)

```
import tensorflow as tf
import numpy as np

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
batch_xs, batch_ys = mnist.train.next_batch(100)

x = tf.placeholder('float', (None, 784))
W = tf.Variable(tf.random_uniform([784,10], -1, 1), name = 'W')
b = tf.Variable(tf.ones([10]))
y = tf.matmul(x, W) + b
y_hat = tf.nn.softmax(y)

init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)
    print(sess.run(y_hat, feed_dict={x:batch_xs}).shape)
```

문제 29) argmax 함수를 이용해서 100 x 10 확률벡터들의 최대요소 인덱스 번호를 100개 출력하시오

답)

```
import tensorflow as tf
import numpy as np

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
batch_xs, batch_ys = mnist.train.next_batch(100)

x = tf.placeholder('float', (None, 784))
W = tf.Variable(tf.random_uniform([784,10], -1, 1), name = 'W')
b = tf.Variable(tf.ones([10]))
y = tf.matmul(x, W) + b
y_hat = tf.nn.softmax(y)
result = tf.argmax(y_hat, axis=1)  # axis=1 : 행 중에 최대 인덱스를 뽑아라

init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)
    print(sess.run(result, feed_dict={x:batch_xs}))
```

```
[8 0 6 2 7 2 0 2 9 2 5 8 8 2 9 6 8 6 5 9 6 2 6 2 6 6 2 2 8 9 7 6 8 5 7 6 9
 2 2 2 6 9 2 7 9 6 6 0 2 0 5 0 2 8 6 6 2 8 6 6 9 8 6 2 0 9 5 2 0 2 7 5 5 2
 8 2 2 8 7 8 5 9 6 9 2 7 6 7 2 6 2 5 7 6 6 2 8 2 7 7]
```

문제 30) 위의 코드에 라벨을 가져오는 코드를 추가해서 정확도를 출력하시오

답)

```
import tensorflow as tf
import numpy as np

mnist = input_data.read_data_sets("MNIST_data/", one_hot=False)
batch_xs, batch_ys = mnist.train.next_batch(100)

x = tf.placeholder('float', (None, 784))
y_label = batch_ys
W = tf.Variable(tf.random_uniform([784,10], -1, 1), name = 'W')
b = tf.Variable(tf.ones([10]))
y = tf.matmul(x, W) + b
y_hat = tf.nn.softmax(y)
y_result = tf.argmax(y_hat, axis=1)  # axis=1 : 행 중에 최대 인덱스를 뽑아라
correct_prediction = tf.equal(y_result, y_label)
```

```
cp = tf.cast(correct_prediction, 'float')
cp2 = tf.reduce_mean(cp)

init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)
    print(sess.run(cp2, feed_dict={x:batch_xs}))
```

0.08

텐서플로우로 구현하는 비용 함수

1. Mean Squared Error, MSE

```
loss = tf.square(y_predict, y_label)
```

2. Cross Entropy Error, CEE

```
loss = -tf.reduce_sum(y_onehot*tf.log(y_hat), axis=1)
```

문제 31) CEE 함수를 문제 30번 코드에 추가하시오

(내가 짰던 문제 30번에 추가하려 했으나 실패해서 선생님의 코드를 답에 적었다)

답)

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/",one_hot=True)

# 계층 생성해서 예측값 출력하는 코드
x = tf.placeholder("float",[None,784])
W = tf.Variable(tf.random_uniform([784,10],-1,1), name="W")
b = tf.Variable(tf.ones([10]))
y = tf.matmul(x,W) + b
y_hat = tf.nn.softmax( y )
y_predict = tf.argmax(y_hat, axis = 1  )



# 라벨을 원핫 인코딩해서 담을 변수를 생성
y_onehot = tf.placeholder("float",[None,10])

# 라벨을 숫자로 출력
y_label = tf.argmax(y_onehot, axis = 1)
```

```
# 손실 함수
loss = -tf.reduce_sum(y_onehot*tf.log(y_hat), axis=1)

# 정확도를 출력하기 위한 코드
correct_prediction = tf.equal(y_predict, y_label)
accuracy = tf.reduce_mean(tf.cast(correct_prediction,"float"))

# 변수 초기화
init = tf.global_variables_initializer()

# 세션 객체 생성
sess = tf.Session()
sess.run(init)

batch_xs, batch_ys = mnist.train.next_batch(100)

print (sess.run(y_predict, feed_dict={x:batch_xs}))
print (sess.run(y_label, feed_dict={y_onehot:batch_ys}))

print (sess.run(accuracy, feed_dict={x:batch_xs,y_onehot:batch_ys}))
```

경사하강법을 텐서플로우로 구현하는 방법

```
#########################################################
# 그외 옵티마이저
# optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
# optimizer = tf.train.AdagradOptimizer(learning_rate=0.01)
# optimizer = tf.train.MomentumOptimizer(learning_rate=0.01)
# optimizer = tf.train.AdamOptimizer(learning_rate=0.01)
#########################################################
```

SGD

```
미니배치만큼 랜덤으로 데이터를 추출해서 확률적으로 경사를 감소하여 Global Minima 로 찾아가는 방법
```

단점

```
Local Minima 에 빠질 수 있다
```

그래서 Adagrade 를 쓴다

Adagrade

```
러닝레이트가 학습되면서 자동 조절되는 경사하강법
```

Momentum

관성을 이용해서 Local Minima 에 빠지지 않게 하는 방법

Adam

Adagrade 의 장점 + Momentum 의 장점

문제 32) 문제 31번 코드에 SGD 경사하강법 코드를 적용해서 학습이 되게 구현하시오

답)

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/",one_hot=True)

# 계층 생성해서 예측값 출력하는 코드
x = tf.placeholder("float",[None,784])
W = tf.Variable(tf.random_uniform([784,10],-1,1), name="W")
b = tf.Variable(tf.ones([10]))
y = tf.matmul(x,W) + b
y_hat = tf.nn.softmax( y )
y_predict = tf.argmax(y_hat, axis = 1)


# 라벨을 원핫 인코딩해서 담을 변수를 생성
y_onehot = tf.placeholder("float",[None,10])


# 라벨을 숫자로 출력
y_label = tf.argmax(y_onehot, axis = 1)


# 손실 함수
loss = -tf.reduce_sum(y_onehot*tf.log(y_hat), axis=1)


########### 그외 옵티마이저 ####################
# SGD 경사하강법 구현
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
# # Adagrade 경사하강법 구현
# optimizer = tf.train.AdagradOptimizer(learning_rate=0.01)
# # Momentum 구현
# optimizer = tf.train.MomentumOptimizer(learning_rate=0.01)
# # Adam 구현
```

```
# optimizer = tf.train.AdamOptimizer(learning_rate=0.01)
#################################################


# 학습 오퍼레이션 정의
train = optimizer.minimize(loss)


# 정확도를 출력하기 위한 코드
correct_prediction = tf.equal(y_predict, y_label)
accuracy = tf.reduce_mean(tf.cast(correct_prediction,"float"))


# 변수 초기화
init = tf.global_variables_initializer()


# 세션 객체 생성
sess = tf.Session()
sess.run(init)

batch_xs, batch_ys = mnist.train.next_batch(100)

sess.run(train, feed_dict={x: batch_xs, y_onehot: batch_ys})

# 세션 닫기
sess.close()
```

문제 33) 문제 32번 코드의 정확도를 출력하시오

답)

```
# 세션 객체 생성
sess = tf.Session()
sess.run(init)

batch_xs, batch_ys = mnist.train.next_batch(100)

sess.run(train, feed_dict={x: batch_xs, y_onehot: batch_ys})
print(sess.run(accuracy, feed_dict={x: batch_xs, y_onehot: batch_ys}))

# 세션 닫기
sess.close()
```

문제 34) 문제 32번 코드 중 아래 보기의 3개의 코드를 for loop 문을 이용해서 1 에폭 돌게 구성하시오

보기)

```
batch_xs, batch_ys = mnist.train.next_batch(100)

sess.run(train, feed_dict={x: batch_xs, y_onehot: batch_ys})
print(sess.run(accuracy, feed_dict={x: batch_xs, y_onehot: batch_ys}))
```

답)

```
# 세션 객체 생성
sess = tf.Session()
sess.run(init)

for i in range(600):
    batch_xs, batch_ys = mnist.train.next_batch(100)

    sess.run(train, feed_dict={x: batch_xs, y_onehot: batch_ys})
print(sess.run(accuracy, feed_dict={x: batch_xs, y_onehot: batch_ys}))

# 세션 닫기
sess.close()
```

0.93

에폭을 늘려 여러 번 학습시키기

문제 35) 위의 코드를 수정해서 아래와 같이 결과가 출력되게 하시오

보기)

```
1 에폭 정확도 : 0.9
2 에폭 정확도 : 0.92
3 에폭 정확도 : 0.95
    :
    :
```

답)

```
# 세션 객체 생성
sess = tf.Session()
sess.run(init)

for i in range(1, 6001):
```

```
    batch_xs, batch_ys = mnist.train.next_batch(100)

    sess.run(train, feed_dict={x: batch_xs, y_onehot: batch_ys})

    if i % 600 == 0:
        print(int(i/600), '에폭:', sess.run(accuracy, feed_dict={x: batch_xs, y_onehot: batch_ys}))

# 세션 닫기
sess.close()
```

1 에폭: 0.92
2 에폭: 0.97
3 에폭: 0.96
4 에폭: 0.93
5 에폭: 0.93
6 에폭: 0.95
7 에폭: 0.93
8 에폭: 0.94
9 에폭: 0.97
10 에폭: 0.98

문제 36) 러닝 레이트를 0.05로 하고 학습시키고 정확도를 확인하시오

답)

```
############ 그외 옵티마이저 ####################
# SGD 경사하강법 구현
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.05)
# # Adagrade 경사하강법 구현
# optimizer = tf.train.AdagradOptimizer(learning_rate=0.01)
# # Momentum 구현
# optimizer = tf.train.MomentumOptimizer(learning_rate=0.01)
# # Adam 구현
# optimizer = tf.train.AdamOptimizer(learning_rate=0.01)
################################################
```

1 에폭: 0.08
2 에폭: 0.13
3 에폭: 0.09
4 에폭: 0.1
5 에폭: 0.1
6 에폭: 0.08
7 에폭: 0.13
8 에폭: 0.1

9 에폭: 0.2
10 에폭: 0.15

문제 35) 위의 코드를 수정해서 아래와 같이 결과가 출력되게 하시오

보기)

```
########################## 그외 옵티마이저 ##########################
# # SGD 경사하강법 구현
# optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.05)
# Adagrade 경사하강법 구현
optimizer = tf.train.AdagradOptimizer(learning_rate=0.01)
# # Momentum 구현
# optimizer = tf.train.MomentumOptimizer(learning_rate=0.01)
# # Adam 구현
# optimizer = tf.train.AdamOptimizer(learning_rate=0.01)
#####################################################################
```

1 에폭: 0.73
2 에폭: 0.83
3 에폭: 0.79
4 에폭: 0.76
5 에폭: 0.86
6 에폭: 0.87
7 에폭: 0.84
8 에폭: 0.9
9 에폭: 0.84
10 에폭: 0.85

복습

1. 텐서플로우를 사용했을 때의 이점?

a. 코드가 간결해진다 (모델 생성 부분, 모델 실행 부분)
모델 생성 부분 : 오퍼레이션, 변수
모델 실행 부분 : 세션
b. 신경망 구현에 필요한 모든 함수가 내장
c. 속도가 빠르다
d. GPU 를 사용할 수 있다

2. 텐서플로우로 단층신경망 구현
3. 텐서플로우로 다층신경망 구현
4. 텐서플로우로 CNN 구현
5. 개/고양이 사진 로드하고 학습하기
6. 정상 폐/폐결절 사진 로드하고 학습하기

텐서플로우로 Underfitting 막는 방법

1. 가중치 초기화

```
##############################################
# 그외 가중치 초기화 방법
# W = tf.Variable(tf.random_uniform([784,10], -1, 1)) # [784,10] 형상을 가진 -1~1 사이의 균등분포 어레이
# W = tf.get_variable(name="W", shape=[784, 10], initializer=tf.contrib.layers.xavier_initializer()) # xavier 초기값
# W = tf.get_variable(name='W', shape=[784, 10],
initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
# b = tf.Variable(tf.zeros([10]))
##############################################
```

텐서플로우에서 가중치 초기화할 때 주의사항

텐서플로우는 그래프를 메모리에 올려서 실행하게 되는데 파이참은 코드를 매번 실행할 때마다 메모리를 지워주는데 스파이더나 쥬피터는 대화형이라서 실행할 때마다 메모리에 그래프가 누적이 되어 에러가 난다.

따라서 맨 위에 tf.reset_default_graph() 명령어를 적어줘야 한다

문제 38) 문제 37번 코드에서 가중치 초기화 he 를 사용해서 정확도를 확인하시오

답)

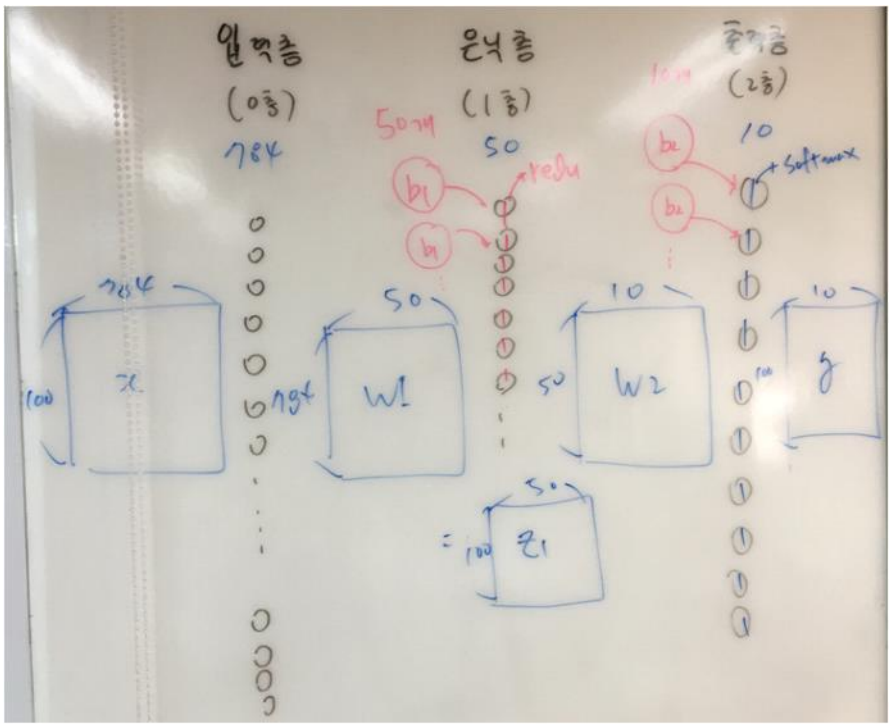
```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

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

tf.reset_default_graph()
# 계층 생성
x = tf.placeholder("float",[None,784])
W = tf.get_variable(name='W', shape=[784, 10],
initializer=tf.contrib.layers.variance_scaling_initializer())
b = tf.Variable(tf.ones([10]))
y = tf.matmul(x,W) + b
y_hat = tf.nn.softmax(y)
y_predict = tf.argmax(y_hat, axis = 1)

# 라벨을 저장하기 위한 변수 생성
y_onehot = tf.placeholder("float",[None,10])
```

```
y_label = tf.argmax(y_onehot, axis = 1)

# 정확도를 출력하기 위한 변수 생성
correct_prediction = tf.equal(y_predict, y_label)
acc = tf.reduce_mean(tf.cast(correct_prediction,"float"))

# 교차 엔트로피 오차 함수
loss = -tf.reduce_sum(y_onehot * tf.log(y_hat), axis = 1)

# SGD 경사 감소법
# optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.05)

# Adam 경사 감소법
optimizer = tf.train.AdamOptimizer(learning_rate=0.001)

# 학습 오퍼레이션 정의
Train = optimizer.minimize(loss)

# 변수 초기화
init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)
    for i in range(1, 6001):
        batch_xs, batch_ys = mnist.train.next_batch(100)
        sess.run(Train, feed_dict={x : batch_xs, y_onehot : batch_ys})
        if i % 600 == 0:
            print(int(i / 600),'epoch acc:', sess.run(acc, feed_dict={x:batch_xs, y_onehot: batch_ys}))
```

1 epoch acc: 0.9
2 epoch acc: 0.87
3 epoch acc: 0.94
4 epoch acc: 0.94
5 epoch acc: 0.95
6 epoch acc: 0.97
7 epoch acc: 0.92
8 epoch acc: 0.94
9 epoch acc: 0.89
10 epoch acc: 0.94

문제 39) 위의 단층 코드를 다층(2층) 으로 변경하시오

보기)



답)

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

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

# 은닉 1층
tf.reset_default_graph()  # 메모리에 그래프 누적되는 걸 막기 위해
x = tf.placeholder("float",[None,784])
W1 = tf.get_variable(name='W1', shape=[784, 50],
initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b1 = tf.Variable(tf.ones([50]))
y1 = tf.matmul(x,W1) + b1
y1_relu = tf.nn.relu(y1)

# 출력층 (2층)
W2 = tf.get_variable(name='W2', shape=[50, 10],
initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b2 = tf.Variable(tf.ones([10]))
y2 = tf.matmul(y1_relu,W2) + b2
y_hat = tf.nn.softmax(y2)
y_predict = tf.argmax(y_hat, axis = 1)

# 라벨을 저장하기 위한 변수 생성
y_onehot = tf.placeholder("float",[None,10])
y_label = tf.argmax(y_onehot, axis = 1)
```

```
# 정확도를 출력하기 위한 변수 생성
correct_prediction = tf.equal(y_predict, y_label)
acc = tf.reduce_mean(tf.cast(correct_prediction,"float"))

# 교차 엔트로피 오차 함수
loss = -tf.reduce_sum(y_onehot * tf.log(y_hat), axis = 1)

# SGD 경사 감소법
# optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.05)

# Adam 경사 감소법
optimizer = tf.train.AdamOptimizer(learning_rate=0.001)

# 학습 오퍼레이션 정의
Train = optimizer.minimize(loss)

# 변수 초기화
init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)
    for i in range(1, 6001):
        batch_xs, batch_ys = mnist.train.next_batch(100)
        sess.run(Train, feed_dict={x : batch_xs, y_onehot : batch_ys})
        if i % 600 == 0:
            print(int(i / 600),'epoch acc:', sess.run(acc, feed_dict={x:batch_xs, y_onehot: batch_ys}))
```

1 epoch acc: 0.91
2 epoch acc: 0.96
3 epoch acc: 0.98
4 epoch acc: 0.98
5 epoch acc: 0.98
6 epoch acc: 0.97
7 epoch acc: 0.96
8 epoch acc: 0.96
9 epoch acc: 0.99
10 epoch acc: 1.0

2. 배치 정규화

신경망 학습 시 가중치의 값의 데이터가 골고루 분산될 수 있도록 하는 것을 강제하는 장치

구현 코드

```
batch_z1 = tf.contrib.layers.batch_norm(z1, True)
```

문제 40) 문제 39번 다층 신경망 코드에 배치 정규화 코드를 추가하시오

답)

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

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

# 은닉 1층
tf.reset_default_graph()  # 메모리에 그래프 누적되는 걸 막기 위해
x = tf.placeholder("float",[None,784])
W1 = tf.get_variable(name='W1', shape=[784, 50],
initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b1 = tf.Variable(tf.ones([50]))
y1 = tf.matmul(x,W1) + b1
batch_z1 = tf.contrib.layers.batch_norm(y1, True)
y1_relu = tf.nn.relu(batch_z1)

# 출력층 (2층)
W2 = tf.get_variable(name='W2', shape=[50, 10],
initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b2 = tf.Variable(tf.ones([10]))
y2 = tf.matmul(y1_relu,W2) + b2
y_hat = tf.nn.softmax(y2)
y_predict = tf.argmax(y_hat, axis = 1)

# 라벨을 저장하기 위한 변수 생성
y_onehot = tf.placeholder("float",[None,10])
y_label = tf.argmax(y_onehot, axis = 1)

# 정확도를 출력하기 위한 변수 생성
correct_prediction = tf.equal(y_predict, y_label)
acc = tf.reduce_mean(tf.cast(correct_prediction,"float"))

# 교차 엔트로피 오차 함수
loss = -tf.reduce_sum(y_onehot * tf.log(y_hat), axis = 1)

# SGD 경사 감소법
# optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.05)

# Adam 경사 감소법
```

```
optimizer = tf.train.AdamOptimizer(learning_rate=0.001)

# 학습 오퍼레이션 정의
Train = optimizer.minimize(loss)

# 변수 초기화
init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)
    for i in range(1, 6001):
        batch_xs, batch_ys = mnist.train.next_batch(100)
        sess.run(Train, feed_dict={x : batch_xs, y_onehot : batch_ys})
        if i % 600 == 0:
            print(int(i / 600),'epoch acc:', sess.run(acc, feed_dict={x:batch_xs, y_onehot: batch_ys}))
```

1 epoch acc: 0.92
2 epoch acc: 0.96
3 epoch acc: 0.93
4 epoch acc: 0.97
5 epoch acc: 0.99
6 epoch acc: 0.98
7 epoch acc: 1.0
8 epoch acc: 0.96
9 epoch acc: 0.99
10 epoch acc: 0.98

훈련할때 만들었던 최적의 감마와 베타를 테스트 할 때 적용하는 코드

```
update_ops = tf.get_collection(tf.GraphKeys.UPDATE_OPS)
  with tf.control_dependencies(update_ops):
    train_op = optimizer.minimize(loss)
```

훈련할때 학습되면서 배치정규화의 최적의 감마와 베타를 생성한다

훈련하는 신경망에 테스트 하는 코드를 추가

문제 41) 지금 현재까지의 코드에는 신경망을 훈련만 시키는 코드였는데 테스트까지 진행해서 오버피팅이 발생했는지 확인할 수 있도록 에폭마다 훈련데이터의 정확도와 테스트데이터의 정확도를 같이 출력할 수 있도록 코드를 작성하시오

답)

```
위의 코드는 동일
```

```
# 변수 초기화
init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)
    for i in range(1, 6001):
        train_x, train_y = mnist.train.next_batch(100)
        test_x, test_y = mnist.train.next_batch(100)

        sess.run(Train, feed_dict={x : train_x, y_onehot : train_y})

        if i % 600 == 0:
            print(int(i / 600),'epoch train_acc:', sess.run(acc, feed_dict={x: train_x, y_onehot: train_y}))
            print(int(i / 600),'epoch test_acc:', sess.run(acc, feed_dict={x: test_x, y_onehot: test_y}))
```

```
1 epoch train_acc: 0.96
1 epoch test_acc: 0.94
2 epoch train_acc: 0.95
2 epoch test_acc: 0.96
3 epoch train_acc: 0.96
3 epoch test_acc: 0.95
4 epoch train_acc: 0.97
4 epoch test_acc: 0.97
5 epoch train_acc: 0.94
5 epoch test_acc: 0.95
6 epoch train_acc: 0.98
6 epoch test_acc: 0.97
7 epoch train_acc: 0.98
7 epoch test_acc: 0.99
8 epoch train_acc: 0.94
8 epoch test_acc: 0.98
9 epoch train_acc: 0.99
9 epoch test_acc: 0.98
10 epoch train_acc: 0.99
10 epoch test_acc: 0.99
```

문제 42) 위 코드를 시각화하시오

답)

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
import matplotlib.pyplot as plt
```

```
import numpy as np

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

# 은닉 1층
tf.reset_default_graph()  # 메모리에 그래프 누적되는 걸 막기 위해
x = tf.placeholder("float",[None,784])
W1 = tf.get_variable(name='W1', shape=[784, 50],
initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b1 = tf.Variable(tf.ones([50]))
y1 = tf.matmul(x,W1) + b1
batch_z1 = tf.contrib.layers.batch_norm(y1, True)
y1_relu = tf.nn.relu(batch_z1)

# 출력층 (2층)
W2 = tf.get_variable(name='W2', shape=[50, 10],
initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b2 = tf.Variable(tf.ones([10]))
y2 = tf.matmul(y1_relu,W2) + b2
y_hat = tf.nn.softmax(y2)
y_predict = tf.argmax(y_hat, axis = 1)

# 라벨을 저장하기 위한 변수 생성
y_onehot = tf.placeholder("float",[None,10])
y_label = tf.argmax(y_onehot, axis = 1)

# 정확도를 출력하기 위한 변수 생성
correct_prediction = tf.equal(y_predict, y_label)
acc = tf.reduce_mean(tf.cast(correct_prediction,"float"))

# 교차 엔트로피 오차 함수
loss = -tf.reduce_sum(y_onehot * tf.log(y_hat), axis = 1)

# SGD 경사 감소법
# optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.05)

# Adam 경사 감소법
optimizer = tf.train.AdamOptimizer(learning_rate=0.001)

# 학습 오퍼레이션 정의
Train = optimizer.minimize(loss)

# 변수 초기화
init = tf.global_variables_initializer()
```

```
train_loss_list = []
train_acc_list = []
test_acc_list = []

with tf.Session() as sess:
    sess.run(init)
    for i in range(1, 6001):
        train_x, train_y = mnist.train.next_batch(100)
        test_x, test_y = mnist.train.next_batch(100)

        sess.run(Train, feed_dict={x : train_x, y_onehot : train_y})

        if i % 600 == 0:
            print(int(i / 600),'epoch train_acc:', sess.run(acc, feed_dict={x: train_x, y_onehot: train_y}))
            print(int(i / 600),'epoch test_acc:', sess.run(acc, feed_dict={x: test_x, y_onehot: test_y}))
            print("========================================")
            train_acc = sess.run(acc, feed_dict={x: train_x, y_onehot: train_y})
            test_acc = sess.run(acc, feed_dict={x: test_x, y_onehot: test_y})
            train_acc_list.append(train_acc)
            test_acc_list.append(test_acc)


# 그래프 그리기
markers = {'train': 'o', 'test': 's'}
x = np.arange(len(train_acc_list))
plt.plot(x, train_acc_list, label='train acc')
plt.plot(x, test_acc_list, label='test acc', linestyle='--')
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.ylim(0, 1.0)
plt.legend(loc='lower right')
plt.show()
```

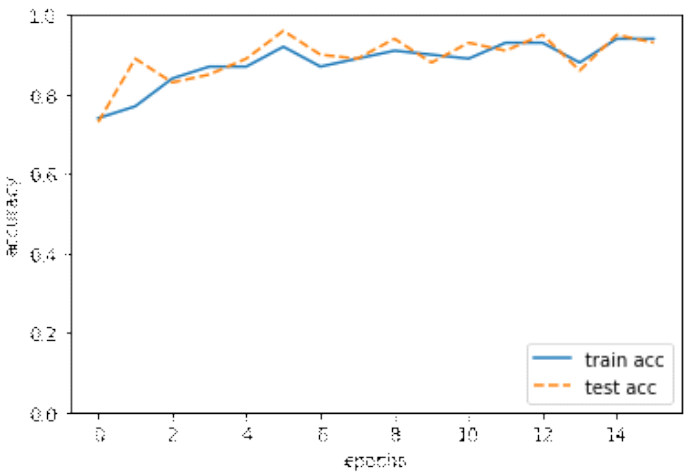
문제 43) 위의 코드가 오버피팅이 발생하지 않도록 dropout 을 적용하시오

답)

```
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.examples.tutorials.mnist import input_data

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

```
tf.reset_default_graph()
# 계층 생성

#입력층
#은닉 1층
x = tf.placeholder("float",[None,784])
W1 = tf.get_variable(name='W1', shape=[784, 100],
initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b1 = tf.Variable(tf.ones([100]))
y1 = tf.matmul(x,W1) + b1 # 내적
batch_y1=tf.contrib.layers.batch_norm(y1,True)
y1_relu = tf.nn.relu(y1) # 렐루 활성화 함수 사용

# drop out
keep_prob = tf.placeholder('float')
y1_drop=tf.nn.dropout(y1_relu,keep_prob=0.5)



# 출력 2층
W2 = tf.get_variable(name='W2', shape=[100, 10],
initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b2 = tf.Variable(tf.ones([10]))

y2 = tf.matmul(y1_drop,W2) + b2 # 내적
y_hat = tf.nn.softmax(y2)

#예측값 출력
y_predict = tf.argmax(y_hat, axis = 1)

# 라벨을 저장하기 위한 변수 생성
y_onehot = tf.placeholder("float",[None,10])
y_label = tf.argmax(y_onehot, axis = 1)

# 정확도를 출력하기 위한 변수 생성
correct_prediction = tf.equal(y_predict, y_label)
acc = tf.reduce_mean(tf.cast(correct_prediction,"float"))

# 교차 엔트로피 오차 함수
loss = -tf.reduce_sum(y_onehot * tf.log(y_hat), axis = 1)

# SGD 경사 감소법
# optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.05)

# Adam 경사 감소법
```

```
optimizer = tf.train.AdamOptimizer(learning_rate=0.0001)

# 학습 오퍼레이션 정의
Train = optimizer.minimize(loss)

# 변수 초기화
init = tf.global_variables_initializer()

train_acc_list=[]
test_acc_list=[]
with tf.Session() as sess:
    sess.run(init)
    for i in range(1, 10001):
        train_xs, train_ys = mnist.train.next_batch(100)
        test_xs, test_ys = mnist.test.next_batch(100)
        sess.run(Train, feed_dict={x : train_xs, y_onehot : train_ys})
        if i % 600 == 0:
            print(int(i / 600),'epoch train_acc :', sess.run(acc, feed_dict={x:train_xs, y_onehot: train_ys,
keep_prob:1.0 }))
            print(int(i / 600),'epoch test_acc:', sess.run(acc, feed_dict={x:test_xs, y_onehot: test_ys,
keep_prob:1.0 }))
            print("========================================")

            train_acc=sess.run(acc, feed_dict={x:train_xs, y_onehot: train_ys, keep_prob: 1.0})
            test_acc=sess.run(acc, feed_dict={x:test_xs, y_onehot: test_ys, keep_prob: 1.0})

            train_acc_list.append(train_acc)
            test_acc_list.append(test_acc)

# 그래프 그리기
markers = {'train': 'o', 'test': 's'}
x = np.arange(len(train_acc_list))
plt.plot(x, train_acc_list, label='train acc')
plt.plot(x, test_acc_list, label='test acc', linestyle='--')
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.ylim(0, 1.0)
plt.legend(loc='lower right')
plt.show()
```



문제 44) 배치정규화도 훈련 시에는 배치정규화를 켜고 테스트 시에는 배치정규화를 끄도록 코드를 구현하시오

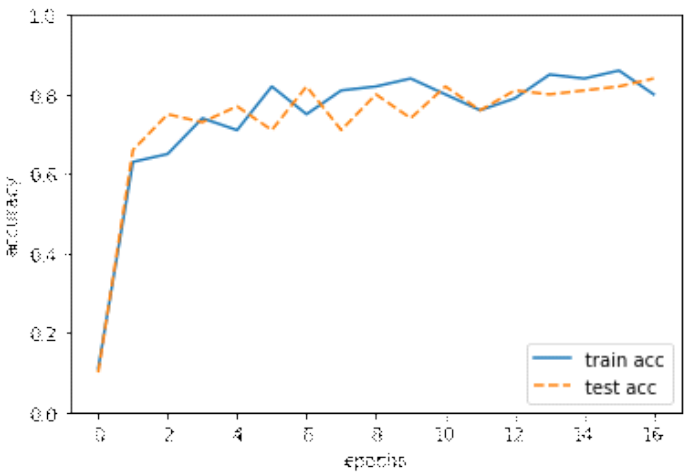
답)

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
import matplotlib.pyplot as plt
import numpy as np

mnist = input_data.read_data_sets('MNIST_data/', one_hot = True)
tf.reset_default_graph()

# 계층 생성

#입력층
x = tf.placeholder("float",[None,784])

# batchnorm switch
isTrain = tf.placeholder(tf.bool)

#은닉 1층
W1 = tf.get_variable(name='W1', shape=[784, 100],
initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b1 = tf.Variable(tf.ones([100]))
y1 = tf.matmul(x,W1) + b1 # 내적
z1 = tf.contrib.layers.batch_norm(y1, is_training=isTrain) # 배치 정규화
y1_relu = tf.nn.relu(z1) # 렐루 활성화 함수 사용

# 드롭아웃
keep_prob = tf.placeholder('float')
dropout = tf.nn.dropout(y1_relu, keep_prob)
```

```
# 출력 2층
W2 = tf.get_variable(name='W2', shape=[100, 10],
initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b2 = tf.Variable(tf.ones([10]))
y2 = tf.matmul(dropout,W2) + b2 # 내적
y_hat = tf.nn.softmax(y2)

#예측값 출력
y_predict = tf.argmax(y_hat, axis = 1)

# 라벨을 저장하기 위한 변수 생성
y_onehot = tf.placeholder("float",[None,10])
y_label = tf.argmax(y_onehot, axis = 1)

# 정확도를 출력하기 위한 변수 생성
correct_prediction = tf.equal(y_predict, y_label)
acc = tf.reduce_mean(tf.cast(correct_prediction,"float"))

# 교차 엔트로피 오차 함수
loss = -tf.reduce_sum(y_onehot * tf.log(y_hat), axis = 1)

# SGD 경사 감소법
# optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.05)

Train = tf.train.AdamOptimizer(learning_rate=0.001).minimize(loss)



# 변수 초기화
init = tf.global_variables_initializer()

train_acc_list = []
test_acc_list = []

with tf.Session() as sess:

    sess.run(init)

    for i in range(10000):

        train_xs, train_ys = mnist.train.next_batch(100) # 훈련 데이터
        test_xs, test_ys = mnist.test.next_batch(100) # 테스트 데이터

        sess.run(Train, feed_dict={x : train_xs, y_onehot : train_ys, keep_prob: 0.5 ,isTrain:True})
```

```
        if i % 600 == 0:

            train_acc = sess.run(acc, feed_dict={x: train_xs, y_onehot: train_ys, keep_prob: 1.0 ,
isTrain:False})
            test_acc = sess.run(acc, feed_dict={x: test_xs, y_onehot: test_ys, keep_prob: 1.0,
isTrain:False})

            print(i / 600 + 1, 'epoch train acc:', train_acc, ', test acc:', test_acc)

            train_acc_list.append(train_acc)  # 10000/600 개  16개 # 정확도가 점점 올라감
            test_acc_list.append(test_acc)  # 10000/600 개 16개 # 정확도가 점점 올라감



# 그래프 그리기
markers = {'train': 'o', 'test': 's'}
x = np.arange(len(train_acc_list))
plt.plot(x, train_acc_list, label='train acc')
plt.plot(x, test_acc_list, label='test acc', linestyle='--')
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.ylim(0, 1.0)
plt.legend(loc='lower right')
plt.show()
```

```
1.0 ecpo train acc: 0.11 , test acc: 0.1
2.0 ecpo train acc: 0.63 , test acc: 0.66
3.0 ecpo train acc: 0.65 , test acc: 0.75
4.0 ecpo train acc: 0.74 , test acc: 0.73
5.0 ecpo train acc: 0.71 , test acc: 0.77
6.0 ecpo train acc: 0.82 , test acc: 0.71
7.0 ecpo train acc: 0.75 , test acc: 0.82
8.0 ecpo train acc: 0.81 , test acc: 0.71
9.0 ecpo train acc: 0.82 , test acc: 0.8
10.0 ecpo train acc: 0.84 , test acc: 0.74
11.0 ecpo train acc: 0.8 , test acc: 0.82
12.0 ecpo train acc: 0.76 , test acc: 0.76
13.0 ecpo train acc: 0.79 , test acc: 0.81
14.0 ecpo train acc: 0.85 , test acc: 0.8
15.0 ecpo train acc: 0.84 , test acc: 0.81
16.0 ecpo train acc: 0.86 , test acc: 0.82
17.0 ecpo train acc: 0.8 , test acc: 0.84
```



문제 45) 훈련 시에 배치 정규화로 만들어진 최적의 베타와 감마를 계속 잘 유지시킬 수 있도록 위의 코드에 아래 코드를 추가하시오

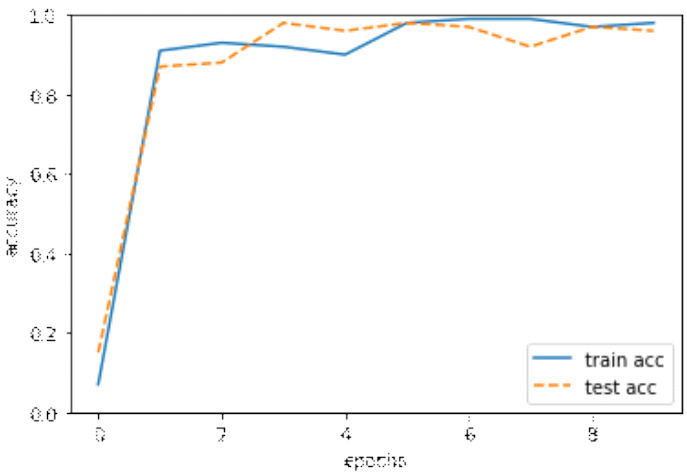
답)

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
import matplotlib.pyplot as plt
import numpy as np

mnist = input_data.read_data_sets('MNIST_data/', one_hot = True)
tf.reset_default_graph()

# 계층 생성

#입력층
x = tf.placeholder("float",[None,784])

# batchnorm switch
isTrain = tf.placeholder(tf.bool)

#은닉 1층
W1 = tf.get_variable(name='W1', shape=[784, 100],
initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b1 = tf.Variable(tf.ones([100]))
y1 = tf.matmul(x,W1) + b1 # 내적
z1 = tf.contrib.layers.batch_norm(y1, is_training=isTrain) # 배치 정규화
y1_relu = tf.nn.relu(z1) # 렐루 활성화 함수 사용

# 드롭아웃
keep_prob = tf.placeholder('float')
dropout = tf.nn.dropout(y1_relu, keep_prob)
```

```
# 출력 2층
W2 = tf.get_variable(name='W2', shape=[100, 10],
initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b2 = tf.Variable(tf.ones([10]))
y2 = tf.matmul(dropout,W2) + b2 # 내적
y_hat = tf.nn.softmax(y2)

#예측값 출력
y_predict = tf.argmax(y_hat, axis = 1)

# 라벨을 저장하기 위한 변수 생성
y_onehot = tf.placeholder("float",[None,10])
y_label = tf.argmax(y_onehot, axis = 1)

# 정확도를 출력하기 위한 변수 생성
correct_prediction = tf.equal(y_predict, y_label)
acc = tf.reduce_mean(tf.cast(correct_prediction,"float"))

# 교차 엔트로피 오차 함수
loss = -tf.reduce_sum(y_onehot * tf.log(y_hat), axis = 1)

# SGD 경사 감소법
# optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.05)

update_ops = tf.get_collection(tf.GraphKeys.UPDATE_OPS)
with tf.control_dependencies(update_ops):
    # Ensures that we execute the update_ops before performing the train_step
    Train = tf.train.AdamOptimizer(learning_rate=0.001).minimize(loss)

# 변수 초기화
init = tf.global_variables_initializer()

train_acc_list = []
test_acc_list = []

with tf.Session() as sess:

    sess.run(init)

    for i in range(10000):

        train_xs, train_ys = mnist.train.next_batch(100) # 훈련 데이터
        test_xs, test_ys = mnist.test.next_batch(100) # 테스트 데이터
```

```
        sess.run(Train, feed_dict={x : train_xs, y_onehot : train_ys, keep_prob: 0.5 ,isTrain:True})

        if i % 600 == 0:

            train_acc = sess.run(acc, feed_dict={x: train_xs, y_onehot: train_ys, keep_prob: 1.0 ,
isTrain:False})
            test_acc = sess.run(acc, feed_dict={x: test_xs, y_onehot: test_ys, keep_prob: 1.0,
isTrain:False})

            print(i / 600 + 1, 'epoch train acc:', train_acc, ', test acc:', test_acc)

            train_acc_list.append(train_acc)  # 10000/600 개  16개 # 정확도가 점점 올라감
            test_acc_list.append(test_acc)  # 10000/600 개 16개 # 정확도가 점점 올라감


# 그래프 그리기
markers = {'train': 'o', 'test': 's'}
x = np.arange(len(train_acc_list))
plt.plot(x, train_acc_list, label='train acc')
plt.plot(x, test_acc_list, label='test acc', linestyle='--')
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.ylim(0, 1.0)
plt.legend(loc='lower right')
plt.show()
```

```
1 epoch train acc: 0.07 , test acc: 0.15
2 epoch train acc: 0.91 , test acc: 0.87
3 epoch train acc: 0.93 , test acc: 0.88
4 epoch train acc: 0.92 , test acc: 0.98
5 epoch train acc: 0.9 , test acc: 0.96
6 epoch train acc: 0.98 , test acc: 0.98
7 epoch train acc: 0.99 , test acc: 0.97
8 epoch train acc: 0.99 , test acc: 0.92
9 epoch train acc: 0.97 , test acc: 0.97
10 epoch train acc: 0.98 , test acc: 0.96
```



MNIST 데이터 3층 신경망 코드

문제 46) 2층 신경망 → 3층 신경망으로 변경하고 정확도를 확인하시오

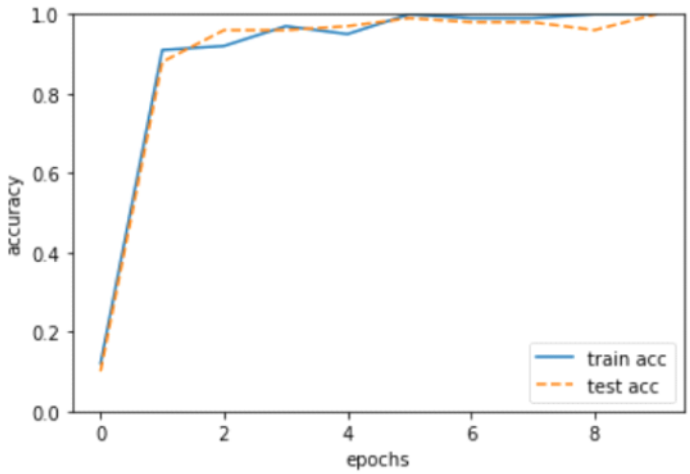
답)

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
import matplotlib.pyplot as plt
import numpy as np

mnist = input_data.read_data_sets('MNIST_data/', one_hot = True)
tf.reset_default_graph()

# 입력층
x = tf.placeholder("float",[None,784])


# batchnorm switch
isTrain = tf.placeholder(tf.bool)


# 은닉 1층
W1 = tf.get_variable(name='W1', shape=[784, 100],
initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b1 = tf.Variable(tf.ones([100]))
y1 = tf.matmul(x,W1) + b1 # 내적
z1 = tf.contrib.layers.batch_norm(y1, is_training=isTrain) # 배치 정규화
y1_relu = tf.nn.relu(z1) # 렐루 활성화 함수 사용


# 은닉 2층
W2 = tf.get_variable(name='W2', shape=[100, 100],
```

```
initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b2 = tf.Variable(tf.ones([100]))
y2 = tf.matmul(y1_relu,W2) + b2 # 내적
z2 = tf.contrib.layers.batch_norm(y2, is_training=isTrain) # 배치 정규화
y2_relu = tf.nn.relu(z2) # 렐루 활성화 함수 사용


# 드롭아웃
keep_prob = tf.placeholder('float')
dropout = tf.nn.dropout(y2_relu, keep_prob)


# 출력층
W3 = tf.get_variable(name='W3', shape=[100, 10],
initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b3 = tf.Variable(tf.ones([10]))
y3 = tf.matmul(dropout,W3) + b3 # 내적
y_hat = tf.nn.softmax(y3)


#예측값 출력
y_predict = tf.argmax(y_hat, axis = 1)


# 라벨을 저장하기 위한 변수 생성
y_onehot = tf.placeholder("float",[None,10])
y_label = tf.argmax(y_onehot, axis = 1)


# 정확도를 출력하기 위한 변수 생성
correct_prediction = tf.equal(y_predict, y_label)
acc = tf.reduce_mean(tf.cast(correct_prediction,"float"))


# 교차 엔트로피 오차 함수
loss = -tf.reduce_sum(y_onehot * tf.log(y_hat), axis = 1)


# SGD 경사 감소법
# optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.05)

update_ops = tf.get_collection(tf.GraphKeys.UPDATE_OPS)
with tf.control_dependencies(update_ops):
    # Ensures that we execute the update_ops before performing the train_step
```

```
    Train = tf.train.AdamOptimizer(learning_rate=0.001).minimize(loss)


# 변수 초기화
init = tf.global_variables_initializer()

train_acc_list = []
test_acc_list = []

with tf.Session() as sess:

    sess.run(init)

    for i in range(6000):

        train_xs, train_ys = mnist.train.next_batch(100) # 훈련 데이터
        test_xs, test_ys = mnist.test.next_batch(100) # 테스트 데이터

        sess.run(Train, feed_dict={x : train_xs, y_onehot : train_ys, keep_prob: 0.5 ,isTrain:True})

        if i % 600 == 0:

            train_acc = sess.run(acc, feed_dict={x: train_xs, y_onehot: train_ys, keep_prob: 1.0 ,
isTrain:False})
            test_acc = sess.run(acc, feed_dict={x: test_xs, y_onehot: test_ys, keep_prob: 1.0,
isTrain:False})

            print(int(i / 600)+1, 'epoch train acc:', train_acc, ', test acc:', test_acc)

            train_acc_list.append(train_acc)  # 10000/600 개  16개 # 정확도가 점점 올라감
            test_acc_list.append(test_acc)  # 10000/600 개 16개 # 정확도가 점점 올라감


# 그래프 그리기
markers = {'train': 'o', 'test': 's'}
x = np.arange(len(train_acc_list))
plt.plot(x, train_acc_list, label='train acc')
plt.plot(x, test_acc_list, label='test acc', linestyle='--')
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.ylim(0, 1.0)
plt.legend(loc='lower right')
plt.show()
```

```
1 epoch train acc: 0.12 , test acc: 0.1
2 epoch train acc: 0.91 , test acc: 0.88
3 epoch train acc: 0.92 , test acc: 0.96
4 epoch train acc: 0.97 , test acc: 0.96
5 epoch train acc: 0.95 , test acc: 0.97
6 epoch train acc: 1.0 , test acc: 0.99
7 epoch train acc: 0.99 , test acc: 0.98
8 epoch train acc: 0.99 , test acc: 0.98
9 epoch train acc: 1.0 , test acc: 0.96
10 epoch train acc: 1.0 , test acc: 1.0
```



---

03 MNIST 텐서플로우로 CNN 구현하기

2018년 8월 13일 월요일 오후 4:34

문제 47) 문제 46번 코드를 오전에 그린 CNN 코드로 구현하시오

답)

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
import matplotlib.pyplot as plt
import numpy as np


# MNIST 데이터 로드
mnist = input_data.read_data_sets('MNIST_data/', one_hot = True)


# 배치정규화 때 쥬피터에서 오류나는거 막는 방법
tf.reset_default_graph()


# 입력데이터를 저장할 변수를 선언 (CNN 이기 때문에 784로 풀지 않고 28 x 28 로 그대로 넣는다)
x = tf.placeholder(tf.float32, [None, 28, 28, 1])  # 흑백이니까 1


# 배치 정규화에서 사용한 변수
keep_prob = tf.placeholder(tf.float32)


# 은닉 1층 (Conv1 → Relu → Pooling)
W1 = tf.Variable(tf.random_normal([3, 3, 1, 32], stddev=0.01))  # 3 x 3 짜리 흑백(1) 32장 만든 것
L1 = tf.nn.conv2d(x, W1, strides=[1, 1, 1, 1], padding='SAME')
                    # 28 x 28 로 들어온 행렬을 feature map도 그대로 28 x 28 로 나오게 하는 옵션
L1 = tf.nn.relu(L1)
L1 = tf.nn.max_pool(L1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')  # 28을 14로

W2 = tf.Variable(tf.random_normal([3, 3, 32, 64], stddev=0.01))
L2 = tf.nn.conv2d(L1, W2, strides=[1, 1, 1, 1], padding='SAME')
L2 = tf.nn.relu(L2)
L2 = tf.nn.max_pool(L2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

W3 = tf.Variable(tf.random_normal([7 * 7 * 64, 256], stddev=0.01))
L3 = tf.reshape(L2, [-1, 7 * 7 * 64])
```

```
L3 = tf.matmul(L3, W3)
L3 = tf.nn.relu(L3)
L3 = tf.nn.dropout(L3, keep_prob)

W4 = tf.Variable(tf.random_normal([256, 10], stddev=0.01))
y2 = tf.matmul(L3,W4) # 내적
y_hat = tf.nn.softmax(y2)

#예측값 출력
y_predict = tf.argmax(y_hat, axis = 1)

# 라벨을 저장하기 위한 변수 생성
y_onehot = tf.placeholder("float",[None,10])
y_label = tf.argmax(y_onehot, axis = 1)

# 정확도를 출력하기 위한 변수 생성
correct_prediction = tf.equal(y_predict, y_label)
acc = tf.reduce_mean(tf.cast(correct_prediction,"float"))

# 교차 엔트로피 오차 함수
loss = -tf.reduce_sum(y_onehot * tf.log(y_hat), axis = 1)

# SGD 경사 감소법
# optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.05)

update_ops = tf.get_collection(tf.GraphKeys.UPDATE_OPS)
with tf.control_dependencies(update_ops):
    # Ensures that we execute the update_ops before performing the train_step
    Train = tf.train.AdamOptimizer(learning_rate=0.001).minimize(loss)



# 변수 초기화
init = tf.global_variables_initializer()

train_acc_list = []
test_acc_list = []

with tf.Session() as sess:

    sess.run(init)

    for i in range(10000):

        train_xs, train_ys = mnist.train.next_batch(100) # 훈련 데이터
```

```
        test_xs, test_ys = mnist.test.next_batch(100) # 테스트 데이터
        train_xs = train_xs.reshape(-1,28,28,1)
        test_xs = test_xs.reshape(-1,28,28,1)

        sess.run(Train, feed_dict={x : train_xs, y_onehot : train_ys, keep_prob: 0.9})

        if i % 600 == 0:

            train_acc = sess.run(acc, feed_dict={x: train_xs, y_onehot: train_ys, keep_prob: 1.0})
            test_acc = sess.run(acc, feed_dict={x: test_xs, y_onehot: test_ys, keep_prob: 1.0})

            print(i / 600 + 1, 'ecpo train acc:', train_acc, ', test acc:', test_acc)

            train_acc_list.append(train_acc)  # 10000/600 개  16개 # 정확도가 점점 올라감
            test_acc_list.append(test_acc)  # 10000/600 개 16개 # 정확도가 점점 올라감


# 그래프 그리기
markers = {'train': 'o', 'test': 's'}
x = np.arange(len(train_acc_list))
plt.plot(x, train_acc_list, label='train acc')
plt.plot(x, test_acc_list, label='test acc', linestyle='--')
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.ylim(0, 1.0)
plt.legend(loc='lower right')
plt.show()
```

---

04 cifar10 이미지 분류

2018년 8월 13일 월요일 오후 4:34

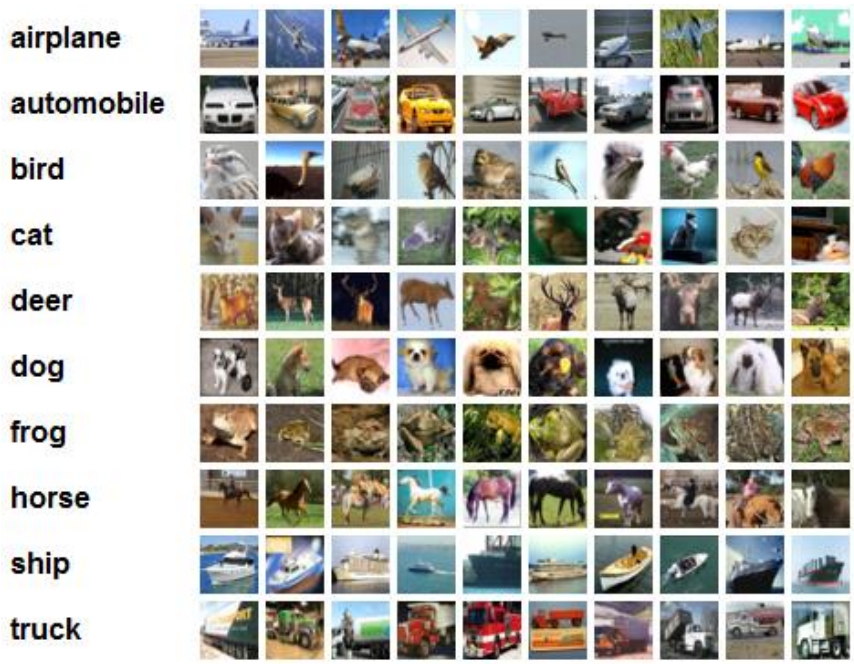
Table of Contents

cifar10 이란?

MNIST 데이터셋 다음에는 보통 CIFAR-10 데이터셋을 검증에 사용한다.
CIFAR-10 데이터셋은 아래와 같이 총 10개의 레이블로 이루어진 이미지 분류를 위한 데이셋으로

airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

각각의 레이블마다 32×32 크기 이미지인 50,000개의 training 데이터셋, 10,000개의 test 데이터셋이 존재하고, 결과적으로 총 60,000개의 32×32 크기의 이미지로 데이터셋이 구성되어 있다.



이미지 데이터를 신경망에 로드하기 위해 반드시 알아야 하는 내용

```
1. train data 이미지를 numpy 배열로 변환하는 방법
2. label data 를 numpy 배열로 변환하는 방법
3. 원핫 인코딩된 데이터에서 numpy 로 최대 인덱스 가져오는 방법
4. 배치 처리 하기 위해 next_batch 함수 만드는 방법
5. data 를 shuffle 하는 함수 생성
```

문제 48) C:/cifar10/test100 폴더를 만들고 test 이미지 100개를 이 폴더에 따로 복사하고 복사한 이미지를 아래와 같이 불러오는 함수를 생성하시오

답)

```
import os

# 특정 디렉토리의 파일 리스트 가져오는 함수
def image_load(path):
    file_list = os.listdir(path)
    return file_list

test_image = 'C:/cifar10/test100/'

print(image_load(test_image))
```

['1.png', '10.png', '100.png', '11.png', '12.png', '13.png', '14.png', '15.png', '16.png', '17.png', '18.png', '19.png', '2.png', '20.png', '21.png', '22.png', '23.png', '24.png', '25.png', '26.png', '27.png', '28.png', '29.png', '3.png', '30.png', '31.png', '32.png', '33.png', '34.png', '35.png', '36.png', '37.png', '38.png', '39.png', '4.png', '40.png', '41.png', '42.png', '43.png', '44.png', '45.png', '46.png', '47.png', '48.png', '49.png', '5.png', '50.png', '51.png', '52.png', '53.png', '54.png', '55.png', '56.png', '57.png', '58.png', '59.png', '6.png', '60.png', '61.png', '62.png', '63.png', '64.png', '65.png', '66.png', '67.png', '68.png', '69.png', '7.png', '70.png', '71.png', '72.png', '73.png', '74.png', '75.png', '76.png', '77.png', '78.png', '79.png', '8.png', '80.png', '81.png', '82.png', '83.png', '84.png', '85.png', '86.png', '87.png', '88.png', '89.png', '9.png', '90.png', '91.png', '92.png', '93.png', '94.png', '95.png', '96.png', '97.png', '98.png', '99.png']

문제 49) 위의 함수를 수정해서 아래와 같이 숫자만 출력되게 하시오

보기)

```
[1, 10, 100, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 3, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 4, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 5, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 6, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 7, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 8, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 9, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99]
```

답)

```
import os
```

```
import re

# 특정 디렉토리의 파일 리스트를 숫자만 남기고 모두 제거하고 가져오는 함수
def image_load(path):
    file_list = os.listdir(path)
    file_name = []
    for i in file_list:
        a = int(re.sub('[^0-9]', '', i))
        file_name.append(a)
    return file_name

test_image = 'C:/cifar10/test100/'
print(image_load(test_image))
```

문제 50) 위 문제의 결과를 정렬해서 출력되게 하시오

보기)

```
[1, 10, 100, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 3, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 4, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 5, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 6, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 7, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 8, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 9, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99]

            ↓

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100]
```

답)

```
import os
import re

# 특정 디렉토리의 파일 리스트를 숫자만 남기고 모두 제거하고 가져오는 함수
def image_load(path):
    file_list = os.listdir(path)
    file_name = []
    for i in file_list:
        a = int(re.sub('[^0-9]', '', i))
        file_name.append(a)
    file_name.sort()
    return file_name
```

```
test_image = 'C:/cifar10/test100/'
print(image_load(test_image))
```

문제 51) 문제 50번에서 나온 결과에 png를 붙여서 아래와 같이 결과가 출력되게 하시오

결과)

```
['1.png', '2.png', '3.png', '4.png', '5.png', '6.png', '7.png', '8.png', '9.png', '10.png', '11.png',
'12.png', '13.png', '14.png', '15.png', '16.png', '17.png', '18.png', '19.png', '20.png', '21.png',
'22.png', '23.png', '24.png', '25.png', '26.png', '27.png', '28.png', '29.png', '30.png', '31.png',
'32.png', '33.png', '34.png', '35.png', '36.png', '37.png', '38.png', '39.png', '40.png', '41.png',
'42.png', '43.png', '44.png', '45.png', '46.png', '47.png', '48.png', '49.png', '50.png', '51.png',
'52.png', '53.png', '54.png', '55.png', '56.png', '57.png', '58.png', '59.png', '60.png', '61.png',
'62.png', '63.png', '64.png', '65.png', '66.png', '67.png', '68.png', '69.png', '70.png', '71.png',
'72.png', '73.png', '74.png', '75.png', '76.png', '77.png', '78.png', '79.png', '80.png', '81.png',
'82.png', '83.png', '84.png', '85.png', '86.png', '87.png', '88.png', '89.png', '90.png', '91.png',
'92.png', '93.png', '94.png', '95.png', '96.png', '97.png', '98.png', '99.png', '100.png']
```

답)

```
import os
import re

def image_load(path):
    # 특정 경로의 디렉토리에 파일리스트 가져오기
    file_list = os.listdir(path)

    # 숫자만 남기고 모두 제거하기
    file_name = []
    for i in file_list:
        a = int(re.sub('[^0-9]', '', i))
        file_name.append(a)
    file_name.sort()

    # 'png' 붙이기
    result = []
    for i in file_name:
        result.append(str(i) + '.png')

    return result

test_image = 'C:/cifar10/test100/'
print(image_load(test_image))
```

문제 52) 이미지 이름 앞에 절대경로가 아래처럼 붙게 하시오

결과)

```
['C:/cifar10/test100/1.png', 'C:/cifar10/test100/2.png', 'C:/cifar10/test100/3.png',
'C:/cifar10/test100/4.png', 'C:/cifar10/test100/5.png', 'C:/cifar10/test100/6.png',
'C:/cifar10/test100/7.png', 'C:/cifar10/test100/8.png', 'C:/cifar10/test100/9.png',
'C:/cifar10/test100/10.png', 'C:/cifar10/test100/11.png', 'C:/cifar10/test100/12.png',
:
:
: ]
```

답)

```
import os
import re



def image_load(path):
    # 특정 경로의 디렉토리에 파일리스트 가져오기
    file_list = os.listdir(path)

    # 숫자만 남기고 모두 제거하기
    file_name = []
    for i in file_list:
        a = int(re.sub('[^0-9]', '', i))
        file_name.append(a)
    file_name.sort()

    # 'png' 붙이기
    result = []
    for i in file_name:
        result.append(str(i) + '.png')

    # 파일이름 앞에 절대경로 붙이기
    res = []
    for i in result:
        res.append(path + i)

    return res

test_image = 'C:/cifar10/test100/'
print(image_load(test_image))
```

문제 53) 위의 이미지들을 cv2.imread 함수를 이용해서 아래와 같이 숫자로 list 로 변환하시오

결과)

```
[array([[[ 49, 112, 158],
        [ 47, 111, 159],
        [ 51, 116, 165],
        ...,
        [ 36,  95, 137],
        [ 36,  91, 126],
        [ 33,  85, 116]],
        [[ 51, 112, 152],
        [ 40, 110, 151],
        [ 45, 114, 159],
        ...,
```

답)

```
import os
import re
import cv2
import numpy as np



def image_load(path):
    # 특정 경로의 디렉토리에 파일리스트 가져오기
    file_list = os.listdir(path)

    # 숫자만 남기고 모두 제거하기
    file_name = []
    for i in file_list:
        a = int(re.sub('[^0-9]', '', i))
        file_name.append(a)
    file_name.sort()

    # 'png' 붙이기
    result = []
    for i in file_name:
        result.append(str(i) + '.png')

    # 파일이름 앞에 절대경로 붙이기
    res = []
    for i in result:
        res.append(path + i)
```

```
    # RGB 행렬로 바꾸기
    image = []
    for i in res:
        img = cv2.imread(i)
        image.append(img)

    return image

test_image = 'C:/cifar10/test100/'
print(image_load(test_image))
```

1. train data 이미지를 numpy 배열로 변환하는 방법

문제 54) 위의 숫자 array를 numpy array로 변환하시오

답)

```
import os
import re
import cv2
import numpy as np


def image_load(path):
    # 특정 경로의 디렉토리에 파일리스트 가져오기
    file_list = os.listdir(path)

    # 숫자만 남기고 모두 제거하기
    file_name = []
    for i in file_list:
        a = int(re.sub('[^0-9]', '', i))
        file_name.append(a)
    file_name.sort()

    # 'png' 붙이기
    result = []
    for i in file_name:
        result.append(str(i) + '.png')

    # 파일이름 앞에 절대경로 붙이기
    res = []
    for i in result:
        res.append(path + i)
```

```
    # RGB 행렬로 바꾸기
    image = []
    for i in res:
        img = cv2.imread(i)
        image.append(img)

    return np.array(image)

test_image = 'C:/cifar10/test100/'
print(image_load(test_image))
```

문제 55) test 라벨을 로드하시오

답)

```
def label_load(path):
    file = open(path)
    labeldata = csv.reader(file)
    labellist = []

    for i in labeldata:
        labellist.append(i)

    return labellist

test_label = 'C:/cifar10/test_label.csv'
print(label_load(test_label))
```

문제 56) 위의 숫자 리스트를 numpy 배열로 변환하시오

답)

```
# 특정 경로의 csv 파일을 리스트 형태로 가져오기
def label_load(path):
    file = open(path)
    labeldata = csv.reader(file)
    labellist = []

    for i in labeldata:
        labellist.append(i)

    # numpy 배열로 변환
```

```
    label = np.array(labellist)

    return label

test_label = 'C:/cifar10/test_label.csv'
print(label_load(test_label))
```

2. label data 를 numpy 배열로 변환하는 방법

문제 57) 위의 결과가 문자가 아닌 숫자로 출력되게 하시오

답)

```
def label_load(path):
    file = open(path)
    labeldata = csv.reader(file)
    labellist = []

    for i in labeldata:
        labellist.append(i)

    # numpy 배열로 변환
    label = np.array(labellist)

    # 숫자형으로 바꾸기
    label = label.astype(int)

    return label


test_label = 'C:/cifar10/test_label.csv'
print(label_load(test_label))
```

3. 원핫 인코딩된 데이터에서 numpy 로 최대 인덱스 가져오는 방법

문제 58) 아래의 결과를 출력해보시오

결과)

```
[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
```

답)

```
import numpy as np
print(np.eye(10)[4])
```

문제 59) 문제 57번에서 가져온 숫자 리스트를 가지고 아래와 같이 one hot encoding 된 결과를 출력하시오

결과)

```
[[[0. 0. 0. ... 0. 0. 0.]]
 [[0. 0. 0. ... 0. 1. 0.]]
 [[0. 0. 0. ... 0. 1. 0.]]
 ...
```

답)

```
# 특정 경로의 csv 파일을 리스트 형태로 가져오기
def label_load(path):
    file = open(path)
    labeldata = csv.reader(file)
    labellist = []

    for i in labeldata:
        labellist.append(i)

    # numpy 배열로 변환
    label = np.array(labellist)

    # 숫자형으로 바꾸기
    label = label.astype(int)

    # 라벨 원핫인코딩 하기
    label = np.eye(10)[label]

    return label

test_label = 'C:/cifar10/test_label.csv'
print(label_load(test_label))
```

label.shape
(10000, 1, 10)
1행 10열 짜리가 10000개

문제 60) 위의 차원은 3차원인데 우리는 2차원으로 줄여야 한다. 왜냐 CNN 코드에서 라벨이 입력될 때는 아래처럼 2차원이기 때문이다. 그래서 차원을 줄이는 함수를 테스트하시오

답)

```
import numpy as np

x = np.array([[[0], [1], [2]]])
print(x) # (1,3,1)
print(np.squeeze(x)) # (3,)
print(np.squeeze(x, axis=0)) #(3,1)
print(np.squeeze(x, axis=2)) # (1,3)
```

결과)

```
[[[0]
  [1]
  [2]]]
[0 1 2]
[[0]
 [1]
 [2]]
[[0 1 2]]
```

문제 61) 라벨의 차원을 3차원에서 2차원으로 줄이시오

답)

```
# 특정 경로의 csv 파일을 리스트 형태로 가져오기
def label_load(path):
    file = open(path)
    labeldata = csv.reader(file)
    labellist = []

    for i in labeldata:
        labellist.append(i)

    # numpy 배열로 변환
    label = np.array(labellist)

    # 숫자형으로 바꾸기
    label = label.astype(int)

    # 라벨 원핫인코딩 하기
    label = np.eye(10)[label]

    # 라벨의 차원을 (10000, 1, 10)
    label = np.squeeze(label, axis=1)
```

```
    return label

test_label = 'C:/cifar10/test_label.csv'
print(label_load(test_label))
```

문제 62) 오전에 만들었던 2가지 함수 (image_load, label_load) 를 loader2.py 라는 파이썬 코드에 저장하고 아래와 같이 loader2.py 를 import 한 후에 cifar10 전체 데이터를 로드하는 코드를 구현하시오

답)

```
import loader2
import time

train_image = 'C:/cifar10/train/'
train_label = 'C:/cifar10/train_label.csv'
test_image = 'C:/cifar10/test/'
test_label = 'C:/cifar10/test_label.csv'

print("LOADING DATA")
start = time.time()

# train 데이터와 라벨 가져오기
trainX = loader2.image_load(train_image)
print(trainX.shape) # (50000, 32, 32,3)
trainY = loader2.label_load(train_label)
print(trainY.shape) # (50000, 10)

# test 데이터와 라벨 가져오기
testX = loader2.image_load(test_image)
print(testX.shape) # (10000,32, 32, 3)
testY = loader2.label_load(test_label)
print(testY.shape) # (10000, 10)

end = time.time()
print('image load time: %.2f' % float(end - start))
```

```
LOADING DATA
(10000, 32, 32, 3)
(50000, 10)
(10000, 32, 32, 3)
(10000, 10)
```

image load time: 11.29

4. 배치 처리 하기 위해 next_batch 함수 만드는 방법

문제 63) test100 폴더 밑에 100개의 데이터 중 10개 만 출력하시오

답)

```
import loader2
import time

train_image = 'C:/cifar10/test/'
train_label = 'C:/cifar10/train_label.csv'
test_image = 'C:/cifar10/test/'
test_label = 'C:/cifar10/test_label.csv'

print("LOADING DATA")
start = time.time()

trainX = loader2.image_load(test_image)
print(trainX[0:10])

end = time.time()
print('image load time: %.2f' % float(end - start))
```

문제 63) test100 폴더 밑에 100개의 데이터 중 10개 만 출력하시오

답)

```
import loader2
import time

train_image = 'C:/cifar10/test/'
train_label = 'C:/cifar10/train_label.csv'
test_image = 'C:/cifar10/test/'
test_label = 'C:/cifar10/test_label.csv'

print("LOADING DATA")
start = time.time()

trainX = loader2.image_load(test_image)
print(trainX[0:10])

end = time.time()
```

```
print('image load time: %.2f' % float(end - start))
```

문제 64) next_batch 함수를 만들어서 아래와 같이 데이터를 입력하고 함수를 실행하면 trainX 에서 100개의 데이터(numpy 배열)을 가져오게 하시오

보기)

```
import  loader2
import time

train_image = 'C:/cifar10/test/'
train_label = 'C:/cifar10/train_label.csv'
test_image = 'C:/cifar10/test/'
test_label = 'C:/cifar10/test_label.csv'

print("LOADING DATA")

testX = loader2.image_load(test_image)
print(next_batch(testX,testY,0,100))
```

답)

```
def next_batch(data,idx,batch_size):
    data_list=data[idx*batch_size: idx*batch_size + batch_size]

    return  data_list
```

문제 65) 이번에는 라벨도 배치 사이즈 만큼 같이 출력될 수 있도록 next_batch 함수에 코드를 추가해서 아래와 같이 출력되게 하시오

보기)

```
import loader2
import time

train_image = 'C:/cifar10/test/'
train_label = 'C:/cifar10/train_label.csv'
test_image = 'C:/cifar10/test/'
test_label = 'C:/cifar10/test_label.csv'

print("LOADING DATA")

testX = loader2.image_load(test_image)
testY = loader2.label_load(test_label)
```

```
print(next_batch(testX, testY, 0, 100))
```

답)

```
def next_batch(data, label, idx, batch_size):
    data_list = data[idx*batch_size : idx*batch_size + batch_size]
    label_list = label[idx * batch_size : idx * batch_size + batch_size]

    return  data_list, label_list
```

5. data 를 shuffle 하는 함수 생성

문제 66) 아래의 코드를 실행해 보시오

답)

```
import random
import numpy as np

np.arange(10)
```

array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

문제 67) 위의 숫자 10개가 랜덤으로 섞여서 출력되게 하시오 (random.shuffle 사용)

답)

```
import random
import numpy as np

data = np.arange(10)
random.shuffle(data)

data
```

array([9, 7, 2, 6, 0, 5, 3, 4, 8, 1])

문제 68) 위의 코드를 이용해서 shuffle_batch 함수를 만들어서 입력된 데이터가 shuffle 되게 하시오

답)

```
import loader2
import time
```

```
import random

train_image = 'C:/cifar10/test/'
train_label = 'C:/cifar10/train_label.csv'
test_image = 'C:/cifar10/test/'
test_label = 'C:/cifar10/test_label.csv'

print("LOADING DATA")
start = time.time()

testX = loader2.image_load(test_image)
testY = loader2.label_load(test_label)



def shuffle_batch(data_list, label):
    x = np.arange(len(data_list))
    random.shuffle(x)
    data_list2 = data_list[x]
    label2 = label[x]

    return data_list2, label2



shuffle_batch(testX[:100], testY[:100])

end = time.time()
print('image load time: %.2f' % float(end - start))
```

```
LOADING DATA
image load time: 5.26
```

문제 69) 위에서 만든 next_batch 함수와 shuffle_batch 함수를 loader2.py 에 추가하시오

답)

```
네
```

문제 70) 기존 MNIST 데이터를 CNN 신경망에 놓는 코드에 MNIST 대신에 cifar10 데이터를 이용해서 학습시키고 정확도 그래프를 볼 수 있도록 코드를 완성시키시오

답)

```
import tensorflow as tf
```

```
from tensorflow.examples.tutorials.mnist import input_data
import matplotlib.pyplot as plt
import numpy as np
import loader2



# cifar10 로드하기
train_image = 'C:/cifar10/test/'
train_label = 'C:/cifar10/train_label.csv'
test_image = 'C:/cifar10/test/'
test_label = 'C:/cifar10/test_label.csv'

print("LOADING DATA")

trainX = loader2.image_load(train_image)
trainY = loader2.label_load(train_label)
testX = loader2.image_load(test_image)
testY = loader2.label_load(test_label)

print("LOAD COMPLETED")



# 배치정규화 때 쥬피터에서 오류나는거 막는 방법
tf.reset_default_graph()



# 입력데이터를 저장할 변수를 선언 (CNN 이기 때문에 784로 풀지 않고 28 x 28 로 그대로 넣는다)
x = tf.placeholder(tf.float32, [None, 32, 32, 3])  # mnist는 흑백이라 0, cifar10은 컬러니까 1



# 배치 정규화에서 사용한 변수
keep_prob = tf.placeholder(tf.float32)



# 은닉 1층 (Conv1 → Relu → Pooling)
W1 = tf.Variable(tf.random_normal([3, 3, 3, 32], stddev=0.01))  # 3 x 3 짜리 흑백(1) 32장 만든 것
L1 = tf.nn.conv2d(x, W1, strides=[1, 1, 1, 1], padding='SAME')
                    # 28 x 28 로 들어온 행렬을 feature map도 그대로 28 x 28 로 나오게 하는 옵션
L1 = tf.nn.relu(L1)
L1 = tf.nn.max_pool(L1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')  # 28을 14로

W2 = tf.Variable(tf.random_normal([3, 3, 32, 64], stddev=0.01))
```

```
L2 = tf.nn.conv2d(L1, W2, strides=[1, 1, 1, 1], padding='SAME')
L2 = tf.nn.relu(L2)
L2 = tf.nn.max_pool(L2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

W3 = tf.Variable(tf.random_normal([8 * 8 * 64, 256], stddev=0.01))
L3 = tf.reshape(L2, [-1, 7 * 7 * 64])
L3 = tf.matmul(L3, W3)
L3 = tf.nn.relu(L3)
L3 = tf.nn.dropout(L3, keep_prob)

W4 = tf.Variable(tf.random_normal([256, 10], stddev=0.01))
y2 = tf.matmul(L3,W4) # 내적
y_hat = tf.nn.softmax(y2)


#예측값 출력
y_predict = tf.argmax(y_hat, axis = 1)


# 라벨을 저장하기 위한 변수 생성
y_onehot = tf.placeholder("float",[None,10])
y_label = tf.argmax(y_onehot, axis = 1)


# 정확도를 출력하기 위한 변수 생성
correct_prediction = tf.equal(y_predict, y_label)
acc = tf.reduce_mean(tf.cast(correct_prediction,"float"))


# 교차 엔트로피 오차 함수
loss = -tf.reduce_sum(y_onehot * tf.log(y_hat), axis = 1)


# SGD 경사 감소법
# optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.05)


update_ops = tf.get_collection(tf.GraphKeys.UPDATE_OPS)
with tf.control_dependencies(update_ops):
    # Ensures that we execute the update_ops before performing the train_step
    Train = tf.train.AdamOptimizer(learning_rate=0.001).minimize(loss)
```

```
# 변수 초기화
init = tf.global_variables_initializer()

train_acc_list = []
test_acc_list = []

with tf.Session() as sess:

    sess.run(init)

    for i in range(10000):

        trainX, trainY = loader2.shuffle_batch(trainX, trainY)
        testX, testY = loader2.shuffle_batch(testX, testY)

        train_xs, train_ys = loader2.next_batch(trainX, trainY, 0, 100) # 훈련 데이터
        test_xs, test_ys = loader2.next_batch(testX, testY, 0, 100) # 테스트 데이터


        sess.run(Train, feed_dict={x : train_xs, y_onehot : train_ys, keep_prob: 0.9})

        if i % 600 == 0:

            train_acc = sess.run(acc, feed_dict={x: train_xs, y_onehot: train_ys, keep_prob: 1.0})
            test_acc = sess.run(acc, feed_dict={x: test_xs, y_onehot: test_ys, keep_prob: 1.0})

            print(i / 600 + 1, 'ecpo train acc:', train_acc, ', test acc:', test_acc)

            train_acc_list.append(train_acc)  # 10000/600 개    16개 # 정확도가 점점 올라감
            test_acc_list.append(test_acc)  # 10000/600 개      16개 # 정확도가 점점 올라감



# 그래프 그리기
markers = {'train': 'o', 'test': 's'}
x = np.arange(len(train_acc_list))
plt.plot(x, train_acc_list, label='train acc')
plt.plot(x, test_acc_list, label='test acc', linestyle='--')
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.ylim(0, 1.0)
plt.legend(loc='lower right')
plt.show()
```

---

05 개, 고양이 이미지 분류

2018년 8월 13일 월요일 오후 4:34

문제 71) 개/고양이 사진이 있는 C:₩Users₩USER-PC₩Python_ROK₩catdog 폴더에 개사진 100장과 고양이 사진 100장을 넣고 아래와 같이 불러오는 함수를 생성하시오

답)

```
import os

def image_load(path):
    # 특정 경로의 디렉토리에 파일리스트 가져오기
    file_list = os.listdir(path)
    return file_list



# 이미지 불러오기
test_images = 'C:/Users/kwang/iCloudDrive/Python/Python_data/catdog'

print(image_load(test_images))
```

문제 72) 위의 결과에서 아래와 같이 숫자만 출력되게 하시오

답)

```
import os
import re

def image_load(path):
    # 특정 경로의 디렉토리에 파일리스트 가져오기
    file_list = os.listdir(path)

    file_name = []
    for i in file_list:
        file_name.append(int(re.sub('[^0-9]', '', i)))
    file_name.sort()

    return file_name



test_image = 'C:/Users/kwang/iCloudDrive/Python/Python_data/catdog'

print(image_load(test_image))
```

문제 73) 아래와 같이 결과를 정렬해서 출력하시오

답)

```
이미 위에서 정렬까지 했음..
```

문제 74) darknamer.exe 프로그램을 이용해서 (1).jpeg 를 1.jpeg 로 일괄 변경하시오

답)

```
네
```

문제 75) 문제73번에서 나온 결과에 jpeg 를 붙여서 아래와 같이 결과가 출력되게 하시오

답)

```
import os
import re



def image_load(path):
    # 특정 경로의 디렉토리에 파일리스트 가져오기
    file_list = os.listdir(path)

    # 숫자만 남기고 모두 제거하기
    file_name = []
    for i in file_list:
        a = int(re.sub('[^0-9]', '', i))
        file_name.append(a)
    file_name.sort()

    file_res = []
    for i in file_name:
        file_res.append('%d.jpeg' %i)

    return file_res



# 이미지 불러오기
test_image = 'C:/Users/kwang/iCloudDrive/Python/Python_data/catdog'

print(image_load(test_image))
```

문제 76) 이미지 앞에 절대경로가 아래처럼 붙게 하시오

답)

```
import os
import re



def image_load(path):
    # 특정 경로의 디렉토리에 파일리스트 가져오기
    file_list = os.listdir(path)

    # 숫자만 남기고 모두 제거하기
    file_name = []
    for i in file_list:
        a = int(re.sub('[^0-9]', '', i))
        file_name.append(a)
    file_name.sort()

    file_res = []
    for i in file_name:
        file_res.append('%d.jpeg' %i)

    # 파일이름 앞에 절대경로 붙이기
    res = []
    for i in file_res:
        res.append(path + i)

    return res



# 이미지 불러오기
test_image = 'C:/Users/kwang/iCloudDrive/Python/Python_data/catdog/'

print(image_load(test_image))
```

```
['C:/Users/kwang/ CloudDrive/Python/Pythor_data/catdog/1.jpeg', 'C:/Users/kwang/
rg/iCloudDrive/Python/Python_data/catdog/3.jpeg', 'C:/Users/kwang/iCloudDrive/Py
e/Python/Python_data/catdog/5.jpeg', 'C:/Users/kwang/iCloudDrive/Python/Python_d
r_data/catdog/7.jpeg', 'C:/Users/kwang/iCloudDrive/Python/Python_data/catdog/9.j
9.jpeg', 'C:/Users/kwang/iCloudDrive/Python/Python_data/catdog/11.jpeg', 'C:/ Use
'C:/Users/kwang/iCloudDrive/Python/Python_data/catdog/12.jpeg', 'C:/Users/kwang/
ang/ CloudDrive/Python/Pythor_data/catdog/14.jpeg', 'C:/Users/kwang/iCloudDrive/
ive/Python/Python data/catdog/16.jpeg', 'C:/Users/kwang/iCloudDrive/Python/Pytho
ython_data/catdog/18.jpeg', 'C:/Users/kwang/iCloudDrive/Python/Python_data/catdo
atdog/20.jpeg', 'C:/Users/kwang/ CloudDrive/Python/Python_data/catdog/21.jpeg',
g', 'C:/Users/kwang/iCloudDrive/Python/Python_data/catdog/23.jpeg', 'C:/Users/kw
s/kwang/iCloudDrive/Python/Python_data/catdog/25.jpeg', 'C:/Users/kwang/iCloudDr
udDrive/Pythor/Python data/catdog/27.jpeg', 'C:/Users/kwang/iCloudDrive/Python/P
```

문제 76) 이미지 앞에 절대경로가 아래처럼 붙게 하시오

답)

```
import os
import re



def image_load(path):
    # 특정 경로의 디렉토리에 파일리스트 가져오기
    file_list = os.listdir(path)

    # 숫자만 남기고 모두 제거하기
    file_name = []
    for i in file_list:
        a = int(re.sub('[^0-9]', '', i))
        file_name.append(a)
    file_name.sort()

    file_res = []
    for i in file_name:
        file_res.append('%d.jpeg' %i)

    # 파일이름 앞에 절대경로 붙이기
    res = []
    for i in file_res:
        res.append(path + i)

    return res



# 이미지 불러오기
test_image = 'C:/Users/kwang/iCloudDrive/Python/Python_data/catdog/'
```

```
print(image_load(test_image))
```

문제 77) 위의 이미지들을 cv2.imread 함수를 이용해서 숫자 list 로 변환하시오

답)

```
import os
import re
import cv2


def image_load(path):
    # 특정 경로의 디렉토리에 파일리스트 가져오기
    file_list = os.listdir(path)

    # 숫자만 남기고 모두 제거하기
    file_name = []
    for i in file_list:
        a = int(re.sub('[^0-9]', '', i))
        file_name.append(a)
    file_name.sort()

    file_res = []
    for i in file_name:
        file_res.append('%d.jpeg' %i)

    # 파일이름 앞에 절대경로 붙이기
    res = []
    for i in file_res:
        res.append(path + i)

    # RGB 행렬로 바꾸기
    image = []
    for i in res:
        img = cv2.imread(i)
        image.append(img)

    return image


# 이미지 불러오기
test_image = 'C:/Users/kwang/iCloudDrive/Python/Python_data/catdog/'

print(image_load(test_image))
```

문제 78) 위의 숫자 배열을 numpy 배열로 바꿔서 출력하시오

답)

```
import os
import re
import cv2
import numpy as np


def image_load(path):
    # 특정 경로의 디렉토리에 파일리스트 가져오기
    file_list = os.listdir(path)

    # 숫자만 남기고 모두 제거하기
    file_name = []
    for i in file_list:
        a = int(re.sub('[^0-9]', '', i))
        file_name.append(a)
    file_name.sort()

    file_res = []
    for i in file_name:
        file_res.append('%d.jpeg' %i)

    # 파일이름 앞에 절대경로 붙이기
    res = []
    for i in file_res:
        res.append(path + i)

    # RGB 행렬로 바꾸기
    image = []
    for i in res:
        img = cv2.imread(i)
        image.append(img)

    # numpy 배열로 바꾸기
    image = np.array(image)

    return image
```

```
# 이미지 불러오기
test_image = 'C:/Users/kwang/iCloudDrive/Python/Python_data/catdog/'

print(image_load(test_image))
```

문제 79) cat_dog_label.csv 를 읽어와서 아래와 같이 결과가 출력되는 함수를 생성하시오

답)

```
# 특정 경로의 csv 파일을 리스트 형태로 가져오기
def  label_load(path):
    file = open(path)
    label_data = csv.reader(file)
    label_list = []

    for i in label_data:
        label_list.append(i)

    return  label_list



# 이미지 불러오기
test_image = 'C:/Users/kwang/iCloudDrive/Python/Python_data/catdog/'
# 라벨 불러오기
test_label = 'C:/Users/kwang/iCloudDrive/Python/Python_data/cat_dog_label.csv'

print(label_load(test_label))
```

문제 80) 위의 숫자 list 를 numpy 배열로 변환하시오

답)

```
# 특정 경로의 csv 파일을 리스트 형태로 가져오기
def  label_load(path):
    file = open(path)
    label_data = csv.reader(file)
    label_list = []

    for i in label_data:
        label_list.append(i)

    return np.array(label_list)
```

```
# 이미지 불러오기
test_image = 'C:/Users/kwang/iCloudDrive/Python/Python_data/catdog/'
# 라벨 불러오기
test_label = 'C:/Users/kwang/iCloudDrive/Python/Python_data/cat_dog_label.csv'

print(label_load(test_label))
```

개, 고양이 사진 분류 신경망을 만들기 위한 전처리 코드

```
1. image_load, label_load 함수 생성
2. label_load 함수가 one_hot_encoding 되게 생성
```

문제 81) label_load 함수가 아래와 같이 one hot encoding 된 결과로 출력되게끔 코드를 추가하시오

보기)

```
[[[0. 1.]]
 [[0. 1.]]
 [[0. 1.]]
 [[0. 1.]]
 [[0. 1.]]
 [[0. 1.]]
     :
     :
```

답)

```
# 특정 경로의 csv 파일을 리스트 형태로 가져오기
def  label_load(path):
    file = open(path)
    label_data = csv.reader(file)
    label_list = []

    for i in label_data:
        label_list.append(i)

    # numpy 배열로 변환
    label = np.array(label_list)

    # 숫자형으로 바꾸기
    label = label.astype(int)
```

```
    # 라벨 원핫인코딩 하기
    label = np.eye(2)[label]

    return label


# 이미지 불러오기
test_image = 'C:/Users/kwang/iCloudDrive/Python/Python_data/catdog/'
# 라벨 불러오기
test_label = 'C:/Users/kwang/iCloudDrive/Python/Python_data/cat_dog_label.csv'

print(label_load(test_label))
```

문제 82) 위의 코드에서 label 의 shape를 뽑아보면 3차원으로 나온다. 하지만 CNN 에 넣으려면 2차원으로 줄여야 한다. 따라서 차원을 줄여보시오

보기)

```
(200, 1, 2)   →   (200, 2)
```

답)

```
# 특정 경로의 csv 파일을 리스트 형태로 가져오기
def  label_load(path):
    file = open(path)
    label_data = csv.reader(file)
    label_list = []

    for i in label_data:
        label_list.append(i)

    # numpy 배열로 변환
    label = np.array(label_list)

    # 숫자형으로 바꾸기
    label = label.astype(int)

    # 라벨 원핫인코딩 하기
    label = np.eye(2)[label]

    # 라벨의 차원을 (200, 1, 2)
    label = np.squeeze(label, axis=1)

    return label
```

```
# 이미지 불러오기
test_image = 'C:/Users/kwang/iCloudDrive/Python/Python_data/catdog/'
# 라벨 불러오기
test_label = 'C:/Users/kwang/iCloudDrive/Python/Python_data/cat_dog_label.csv'

print(label_load(test_label).shape)
```

(200, 2)

설명)
2차원으로 줄여야 하는 이유

```
y = tf.placeholder('float', [None, 2])
이 때 200개를 배치size 만큼 가져와야 하기 때문에 2차원으로 줄여야 한다
```

개, 고양이 이미지를 배치 단위로 신경망에 입력하는 코드

```
개 100, 고양이 100 → shuffle → 10장 씩 배치로 입력
```

문제 83) cifar10 이미지 신경망 생성할 때 사용했던 next_batch 함수를 가지고 와서 10개 씩 배치되게 하시오

답)

```
import os
import re
import cv2
import numpy as np
import csv



# 특정 경로의 디렉토리에 파일리스트 가져오기
def image_load(path):
    file_list = os.listdir(path)

    # 숫자만 남기고 모두 제거하기
    file_name = []
    for i in file_list:
        a = int(re.sub('[^0-9]', '', i))
        file_name.append(a)
    file_name.sort()
```

```
    file_res = []
    for i in file_name:
        file_res.append('%d.jpeg' %i)

    # 파일이름 앞에 절대경로 붙이기
    res = []
    for i in file_res:
        res.append(path + i)

    # RGB 행렬로 바꾸기
    image = []
    for i in res:
        img = cv2.imread(i)
        image.append(img)

    # numpy 배열로 바꾸기
    image = np.array(image)

    return image




# 특정 경로의 csv 파일을 리스트 형태로 가져오기
def  label_load(path):
    file = open(path)
    label_data = csv.reader(file)
    label_list = []

    for i in label_data:
        label_list.append(i)

    # numpy 배열로 변환
    label = np.array(label_list)

    # 숫자형으로 바꾸기
    label = label.astype(int)

    # 라벨 원핫인코딩 하기
    label = np.eye(2)[label]

    # 라벨의 차원을 (200, 1, 2)
```

```
    label = np.squeeze(label, axis=1)

    return label




def next_batch(data, label, idx, batch_size):
    train_data = data[idx*batch_size : idx*batch_size + batch_size]
    label_list = label[idx * batch_size : idx * batch_size + batch_size]

    return  train_data, label_list




def shuffle_batch(data_list, label):
    x = np.arange(len(data_list))
    random.shuffle(x)
    data_list2 = data_list[x]
    label2 = label[x]

    return data_list2, label2




# 이미지 불러오기
test_image = 'C:/Users/kwang/iCloudDrive/Python/Python_data/catdog/'
# 라벨 불러오기
test_label = 'C:/Users/kwang/iCloudDrive/Python/Python_data/cat_dog_label.csv'


print("LOADING DATA")
trainX = image_load(test_image)
trainY = label_load(test_label)
print(next_batch(trainX, trainY, 0, 10))
```

문제 84) cifa10 에서 사용했던 shuffle_batch 함수를 생성하시오

답)

```
import os
import re
import cv2
import numpy as np
import csv
import random



# 특정 경로의 디렉토리에 파일리스트 가져오기
def image_load(path):
    file_list = os.listdir(path)

    # 숫자만 남기고 모두 제거하기
    file_name = []
    for i in file_list:
        a = int(re.sub('[^0-9]', '', i))
        file_name.append(a)
    file_name.sort()

    file_res = []
    for i in file_name:
        file_res.append('%d.jpeg' %i)

    # 파일이름 앞에 절대경로 붙이기
    res = []
    for i in file_res:
        res.append(path + i)

    # RGB 행렬로 바꾸기
    image = []
    for i in res:
        img = cv2.imread(i)
        image.append(img)

    # numpy 배열로 바꾸기
    image = np.array(image)

    return image
```

```
# 특정 경로의 csv 파일을 리스트 형태로 가져오기
def  label_load(path):
    file = open(path)
    label_data = csv.reader(file)
    label_list = []

    for i in label_data:
        label_list.append(i)

    # numpy 배열로 변환
    label = np.array(label_list)

    # 숫자형으로 바꾸기
    label = label.astype(int)

    # 라벨 원핫인코딩 하기
    label = np.eye(2)[label]

    # 라벨의 차원을 (200, 1, 2)
    label = np.squeeze(label, axis=1)

    return label




def next_batch(data, label, idx, batch_size):
    train_data = data[idx*batch_size : idx*batch_size + batch_size]
    label_list = label[idx * batch_size : idx * batch_size + batch_size]

    return  train_data, label_list




def shuffle_batch(data_list, label):
    x = np.arange(len(data_list))
    random.shuffle(x)
    data_list2 = data_list[x]
```

```
    label2 = label[x]

    return data_list2, label2




# 이미지 불러오기
test_image = 'C:/Users/kwang/iCloudDrive/Python/Python_data/catdog/'
# 라벨 불러오기
test_label = 'C:/Users/kwang/iCloudDrive/Python/Python_data/cat_dog_label.csv'



print("LOADING DATA")
trainX = image_load(test_image)
trainY = label_load(test_label)
trainX, trainY = shuffle_batch(trainX, trainY)
next_batch(trainX, trainY, 0, 10)
```

문제 85) 오전에 만든 함수 4개를 loader3.py 라는 모듈로 저장하시오

답)

```
from loader3 import *



# 이미지 불러오기
test_image = 'C:/Users/kwang/iCloudDrive/Python/Python_data/catdog/'
# 라벨 불러오기
test_label = 'C:/Users/kwang/iCloudDrive/Python/Python_data/cat_dog_label.csv'



print("LOADING DATA")
trainX = image_load(test_image)
trainY = label_load(test_label)
trainX, trainY = shuffle_batch(trainX, trainY)
next_batch(trainX, trainY, 0, 10)
```

문제 86) 기존의 cifar10 CNN 코드 + 배치정규화를 적용한 코드를 가져와서 개/고양이에 맞게 코드를 수정하시오

답)

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
import matplotlib.pyplot as plt
import loader3


train_image = 'C:/Users/kwang/iCloudDrive/Python/Python_data/catdog_train/'
train_label = 'C:/Users/kwang/iCloudDrive/Python/Python_data/cat_dog_label_train.csv'
test_image = 'C:/Users/kwang/iCloudDrive/Python/Python_data/catdog_test/'
test_label = 'C:/Users/kwang/iCloudDrive/Python/Python_data/cat_dog_label_test.csv'


# cifar10 데이터 로드
print("LOADING DATA")
trainX = loader3.image_load(train_image)
trainY = loader3.label_load(train_label)
testX = loader3.image_load(test_image)
testY = loader3.label_load(test_label)


tf.reset_default_graph()

x = tf.placeholder(tf.float32, [None, 128, 128, 3])
y = tf.placeholder(tf.float32, [None, 2])
keep_prob = tf.placeholder(tf.float32)


# batchnorm switch
isTrain = tf.placeholder(tf.bool)


W1 = tf.Variable(tf.random_normal([3, 3, 3, 32], stddev=0.01))
#W1 = tf.get_variable(name='W1', shape=[3,3,3,32],
initializer=tf.contrib.layers.variance_scaling_initializer())
L1 = tf.nn.conv2d(x, W1, strides=[1, 1, 1, 1], padding='SAME')
z1 = tf.contrib.layers.batch_norm(L1, is_training=isTrain) # 배치 정규화
L1 = tf.nn.relu(z1)
L1 = tf.nn.max_pool(L1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

print(W1.shape)
print(L1.shape)

W2 = tf.Variable(tf.random_normal([3, 3, 32, 64], stddev=0.01))
```

```
#W2 = tf.get_variable(name='W2', shape=[3,3,32,64],
initializer=tf.contrib.layers.variance_scaling_initializer())
L2 = tf.nn.conv2d(L1, W2, strides=[1, 1, 1, 1], padding='SAME')
z2 = tf.contrib.layers.batch_norm(L2, is_training=isTrain) # 배치 정규화
L2 = tf.nn.relu(z2)
L2 = tf.nn.max_pool(L2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

print(W2.shape)
print(L2.shape)

W3 = tf.Variable(tf.random_normal([32 * 32 * 64, 256], stddev=0.01))
#W3 = tf.get_variable(name='W3', shape=[32 * 32 * 64, 256],
initializer=tf.contrib.layers.variance_scaling_initializer())
L3 = tf.reshape(L2, [-1, 32 * 32 * 64])
L3 = tf.matmul(L3, W3)
z3 = tf.contrib.layers.batch_norm(L3, is_training=isTrain) # 배치 정규화
L3 = tf.nn.relu(z3)
L3 = tf.nn.dropout(L3, keep_prob)

print(W3.shape)
print(L3.shape)

W4 = tf.Variable(tf.random_normal([256, 2], stddev=0.01))
#W4 = tf.get_variable(name='W4', shape=[256, 2],
initializer=tf.contrib.layers.variance_scaling_initializer())
y2 = tf.matmul(L3, W4) # 내적
y_hat = tf.nn.softmax(y2)

print(W4.shape)
print(y2.shape)
print(y_hat.shape)

#예측값 출력
y_predict = tf.argmax(y_hat, axis = 1)

# 라벨을 저장하기 위한 변수 생성
y_onehot = tf.placeholder("float",[None, 2])
y_label = tf.argmax(y_onehot, axis = 1)

# 정확도를 출력하기 위한 변수 생성
correct_prediction = tf.equal(y_predict, y_label)
acc = tf.reduce_mean(tf.cast(correct_prediction,"float"))

# 교차 엔트로피 오차 함수
```

```
loss= tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=y3, labels=y_onehot))
# loss = -tf.reduce_sum(y_onehot * tf.log(y_hat), axis = 1)

# SGD 경사 감소법
# optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.05)

update_ops = tf.get_collection(tf.GraphKeys.UPDATE_OPS)
with tf.control_dependencies(update_ops):
    # Ensures that we execute the update_ops before performing the train_step
    Train = tf.train.AdamOptimizer(learning_rate=0.001).minimize(loss)



# 변수 초기화
init = tf.global_variables_initializer()

train_acc_list = []
test_acc_list = []

with tf.Session() as sess:
    sess.run(init)
    for i in range(10000):

        trainX, trainY = loader3.shuffle_batch(trainX, trainY)
        testX, testY = loader3.shuffle_batch(testX, testY)

        train_xs, train_ys = loader3.next_batch(trainX, trainY, 0, 40) # 훈련 데이터
        test_xs, test_ys = loader3.next_batch(testX, testY, 0, 40) # 테스트 데이터



        sess.run(Train, feed_dict={x: train_xs, y_onehot: train_ys, keep_prob: 0.5, isTrain: True})

        if i % 600 == 0:
            train_acc = sess.run(acc, feed_dict={x: train_xs, y_onehot: train_ys, keep_prob: 1.0,
isTrain:False})
            test_acc = sess.run(acc, feed_dict={x: test_xs, y_onehot: test_ys, keep_prob: 1.0,
isTrain:False})

            print(i/600 + 1, 'ecpo train acc:', train_acc, ', test acc:', test_acc)

            train_acc_list.append(train_acc)  # 10000/600 개  16개 # 정확도가 점점 올라감
            test_acc_list.append(test_acc)  # 10000/600 개 16개 # 정확도가 점점 올라감
```

```
# 그래프 그리기
markers = {'train': 'o', 'test': 's'}
x = np.arange(len(train_acc_list))
plt.plot(x, train_acc_list, label='train acc')
plt.plot(x, test_acc_list, label='test acc', linestyle='--')
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.ylim(0, 1.0)
plt.legend(loc='lower right')
plt.show()
```

---

06 폐결절 사진/정상폐 사진 이미지 분류

2018년 8월 13일 월요일 오후 4:34

과정

1. 이미지 데이터의 전처리 작업
2. loader4.py 를 생성
3. 폐결절 사진에 맞는 cnn 신경망 구현

문제 87) 폐사진을 로드해서 numpy 배열로 변환하는 loader4.py 를 loader3.py를 가지고 수정해서 만드시오

답)

수정하여 생성하였음

문제 88) (학원에 있는 GPU PC로 수행하였음) 1~6470장을 train 폴더에 넣고 나머지를 test 폴더에 lung_train_label.csv 와 lung_test_label.csv 를 각각 생성하시오

답)

```
import loader4

test_image = 'C:/Users/USER-PC/Python_ROK/lung_data/test_lung/'
test_label = 'C:/Users/USER-PC/Python_ROK/lung_data/lung_test_label.csv'

train_image = 'C:/Users/USER-PC/Python_ROK/lung_data/train_lung/'
train_label = 'C:/Users/USER-PC/Python_ROK/lung_data/lung_train_label.csv'



print("LOADING DATA")
# cifar10 데이터 로드

trainX = loader4.image_load(train_image)
trainY = loader4.label_load(train_label)
testX = loader4.image_load(test_image)
testY = loader4.label_load(test_label)

print(trainX.shape)
print(trainY.shape)
print(testX.shape)
print(testY.shape)
```

문제 89) (학원에 있는 GPU PC로 수행하였음) 폐사진 이미지의 사이즈를 128 x 128 로 바꾸시오

답)

```
PlastiliqImageResizerInstall.exe  프로그램 사용
```

문제 90) (학원에 있는 GPU PC로 수행하였음) CNN 구현

답)

```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import loader4



test_image = 'C:/Users/USER-PC/Python_ROK/lung_data/test_lung_resize/'
test_label = 'C:/Users/USER-PC/Python_ROK/lung_data/lung_test_label.csv'

train_image = 'C:/Users/USER-PC/Python_ROK/lung_data/train_lung_resize/'
train_label = 'C:/Users/USER-PC/Python_ROK/lung_data/lung_train_label.csv'



print("LOADING DATA")
# 폐사진 데이터 로드

trainX = loader4.image_load(train_image)
trainY = loader4.label_load(train_label)
testX = loader4.image_load(test_image)
testY = loader4.label_load(test_label)
print(trainY)
tf.reset_default_graph()

x = tf.placeholder(tf.float32, [None, 128, 128, 3])
y = tf.placeholder(tf.float32, [None, 2])
keep_prob = tf.placeholder(tf.float32)

# batchnorm switch
isTrain = tf.placeholder(tf.bool)

W1 = tf.Variable(tf.random_normal([3, 3, 3, 32], stddev=0.01))
#W1 = tf.get_variable(name='W1', shape=[3,3,3,32],
```

```
initializer=tf.contrib.layers.variance_scaling_initializer())
L1 = tf.nn.conv2d(x, W1, strides=[1, 1, 1, 1], padding='SAME')
z1 = tf.contrib.layers.batch_norm(L1, is_training=isTrain) # 배치 정규화
L1 = tf.nn.relu(z1)
L1 = tf.nn.max_pool(L1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

print(W1.shape)
print(L1.shape)

W2 = tf.Variable(tf.random_normal([3, 3, 32, 64], stddev=0.01))
#W2 = tf.get_variable(name='W2', shape=[3,3,32,64],
initializer=tf.contrib.layers.variance_scaling_initializer())
L2 = tf.nn.conv2d(L1, W2, strides=[1, 1, 1, 1], padding='SAME')
z2 = tf.contrib.layers.batch_norm(L2, is_training=isTrain) # 배치 정규화
L2 = tf.nn.relu(z2)
L2 = tf.nn.max_pool(L2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

print(W2.shape)
print(L2.shape)

W3 = tf.Variable(tf.random_normal([32 * 32 * 64, 256], stddev=0.01))
#W3 = tf.get_variable(name='W3', shape=[32*32*64,256],
initializer=tf.contrib.layers.variance_scaling_initializer())
L3 = tf.reshape(L2, [-1, 32 * 32 * 64])
L3 = tf.matmul(L3, W3)
z3 = tf.contrib.layers.batch_norm(L3, is_training=isTrain) # 배치 정규화
L3 = tf.nn.relu(z3)
L3 = tf.nn.dropout(L3, keep_prob)

print(W3.shape)
print(L3.shape)

W4 = tf.Variable(tf.random_normal([256, 2], stddev=0.01))
#W4 = tf.get_variable(name='W4', shape=[256,2],
initializer=tf.contrib.layers.variance_scaling_initializer())
y2 = tf.matmul(L3,W4) # 내적
y_hat = tf.nn.softmax(y2)

print(W4.shape)
print(y2.shape)

#예측값 출력
y_predict = tf.argmax(y_hat, axis = 1)
```

```
# 라벨을 저장하기 위한 변수 생성
y_onehot = tf.placeholder("float", [None,2])
y_label = tf.argmax(y_onehot, axis = 1)

# 정확도를 출력하기 위한 변수 생성
correct_prediction = tf.equal(y_predict, y_label)
acc = tf.reduce_mean(tf.cast(correct_prediction,"float"))

# 교차 엔트로피 오차 함수
loss = -tf.reduce_sum(y_onehot * tf.log(y_hat), axis = 1)

# SGD 경사 감소법
# optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.05)

update_ops = tf.get_collection(tf.GraphKeys.UPDATE_OPS)
with tf.control_dependencies(update_ops):
    # Ensures that we execute the update_ops before performing the train_step
    Train = tf.train.AdamOptimizer(learning_rate=0.0001).minimize(loss)



# 변수 초기화
init = tf.global_variables_initializer()

train_acc_list = []
test_acc_list = []

with tf.Session() as sess:

    sess.run(init)

    for i in range(10000):

        trainX, trainY = loader4.shuffle_batch(trainX, trainY)
        testX, testY = loader4.shuffle_batch(testX, testY)

        train_xs, train_ys = loader4.next_batch(trainX,trainY, 0, 40) # 훈련 데이터
        test_xs, test_ys = loader4.next_batch(testX, testY, 0, 40) # 테스트 데이터



        sess.run(Train, feed_dict={x : train_xs, y_onehot : train_ys, keep_prob: 0.9, isTrain:True})

        if i % 600 == 0:

            train_acc = sess.run(acc, feed_dict={x: train_xs, y_onehot: train_ys, keep_prob:
```

```
1.0,isTrain:False})
            test_acc = sess.run(acc, feed_dict={x: test_xs, y_onehot: test_ys, keep_prob: 1.0,
isTrain:False})

            print(i / 600 + 1, 'epoch train acc:', train_acc, ', test acc:', test_acc)

            train_acc_list.append(train_acc)  # 10000/600 개  16개 # 정확도가 점점 올라감
            test_acc_list.append(test_acc)  # 10000/600 개 16개 # 정확도가 점점 올라감



# 그래프 그리기
markers = {'train': 'o', 'test': 's'}
x = np.arange(len(train_acc_list))
plt.plot(x, train_acc_list, label='train acc')
plt.plot(x, test_acc_list, label='test acc', linestyle='--')
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.ylim(0, 1.0)
plt.legend(loc='lower right')
plt.show()
```

---

07 위의 신경망들 튜닝하기

2018년 9월 20일 목요일 오전 10:15

포트폴리오를 더 빛내기 위한 Tip

정확도를 높이는 것만이 중요한 것은 아니다.

1. 분류가 잘 안되는 사진을 따로 골라내서 그 사진들은
   사람이 분류하게 하고 분류가 잘 안되는 사진들을 빼낸
   신경망을 정확도가 아주 높기때문에 그 신경망을 사람이
   반복해서 단순 작업해야하는 업무를 줄여줄수 있도록
   사용한다.

2. 이미지를 신경망에 넣기 전에 openCV 의 함수를 이용해서

    1. 이미지 회전 함수
    2. 이미지의 밝기를 조절하는 함수
    3. 이미지의 배경을 변경하는 함수
    4. 이미지의 대조를 변경하는 함수
    5. 기타 함수

   이미지를 신경망에 넣기전에 위의 함수들중에 하나를 랜덤으로
   선택하게 해서 이미지를 변형해서 신경망에 입력하면
   더 좋은 결과가 나온다.

cifar10 신경망 튜닝

문제 91) cifar10 이미지 신경망이 발산한 이유를 알아내고 발산하지 않도록 튜닝하시오

답)

용함수를 교차 엔트로피 함수 ----> 평균제곱 오차로
변경하면 됨.

```
# 교차 엔트로피 오차 함수
loss = -tf.reduce_sum(y_onehot * tf.log(y_hat), axis = 1)
```

↓ 변경

```
loss= tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=y3, labels=y_onehot))
```
