

목차

2018년 11월 27일 화요일 오후 6:47

- 텐서 플로우

- 텐서 플로우의 특징?
- 텐서 플로우 용어 설명
- 텐서 플로우 설치
- 텐서 플로우 실습1
- 텐서 플로우 실습2
- 텐서 플로우 실습3
- 텐서 플로우 실습4
- 텐서 플로우 실행구조
- 파이썬 기본문법과 텐서 플로우 문법 비교
- 텐서 플로우로 구현한 단층 신경망 중요한 문법1
- 텐서 플로우로 구현한 단층 신경망 중요한 문법2
- 텐서 플로우의 cast 함수의 이해

- Mnist데이터로 단층 신경망 구현하기

- 텐서 플로우로 가중치를 랜덤으로 생성하는 방법
- 텐서 플로우로 구현하는 비용 함수
- 경사감소법을 텐서 플로우로 구현하는 방법
- 텐서 플로우로 배치 정규화 구현
- 텐서 플로우로 CNN 구현하기

- Cifar 10 이미지 분류

- cifar10 데이터를 신경망에 로드하는 데이터 전처리 코드 작성
- 이미지 데이터 신경망에 로드하기 위해 반드시 알아야하는 내용
- one hot encoding 된 데이터에서 numpy 로 최대 인덱스 가져오는 방법
- data를 shuffle 하는 함수 생성

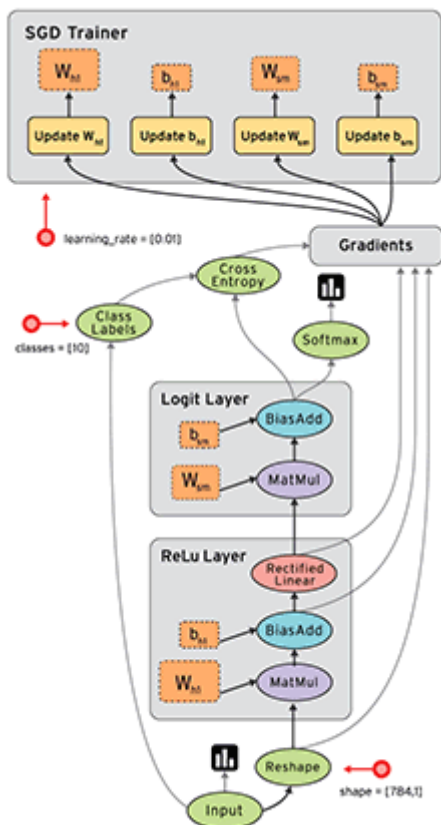
텐서 플로우

2018년 9월 10일 월요일 오전 9:56

텐서 플로우(TensorFlow)는 기계학습과 딥러닝을 위해 구글에서 만든 오픈소스 라이브러리이다.

텐서 플로우의 특징?

1. 데이터 플로우 그래프를 통한 풍부한 표현력



데이터 플로우 그래프(Data Flow Graph) 방식을 사용하였고 그래프를 시각화 하려면?

" 텐서 보드를 사용하면 된다 "

2. 코드 수정없이 CPU/GPU 모드로 동작

한국 시간으로 2016년 11월29일에 TensorFlow v0.12.0 RCO로 업데이트 되었고 2016년 11월29일에

나온 버전의 핵심 변경사항은 Window에서 GPU버전에 텐서 플로우를 지원한다는 것. 예전에는 Ubuntu에서만 가능하던 GPU버전도 윈도우에서 설치가 가능하게 되었다.

텐서 플로우 용어 설명

1. 오퍼레이션(Operation) : 그래프 상의 노드는 오퍼레이션(op)이라 불린다.

오퍼레이션은 하나 이상의 텐서를 받을 수 있다.

계산을 수행하고, 결과를 하나 이상의 텐서로 반환할 수 있다.

2. 텐서(Tensor) : 내부적으로 모든 데이터는 텐서를 통해 표현된다. 텐서는 일종의 4차원 배열인데, 그래프

내의 오퍼레이션 간에 텐서가 전달된다.

3. 세션(session) : 그래프를 실행하기 위해서는 세션 객체가 필요하다. 세션은 오퍼레이션의 실행 환경을

캡슐화 한 것이다.

4. 변수(Variable) : 그래프의 실행시, 파라미터를 저장하고 갱신하는데 사용된다. 메모리 상에서 텐서를 저장

하는 버퍼 역할을 한다.

텐서 플로우 설치

1. pip install tensorflow

텐서 플로우 실습1

```
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 무자열 unicode가 기본으로 str에서 encoding 처리를 해줘야 b타입을 unicode type을 반환한다.

설명 : 위에서 변수를 정의했지만, 실행이 정의한 시점에서 실행되는 것은 아니다. session 객체와 run 메소드를 사용할때 계산이 되어 실행이 된다.

텐서 플로우 실습2

```
import tensorflow as tf
```

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

```
model = tf.global_variables_initializer() #변수를 초기화 하겠다
```

```
with tf.Session() as sess: #값을 계산하기 위해 세션 생성
    sess.run(model) #위에서 초기화한 model을 실행
    print(sess.run(y)) #변수 y를 실행하면서 현재값 출력
```

텐서 플로우 실습3

텐서 플로는 빌딩 구조와 실행구조(Session)가 분리가 되어 있어 이해하는 실습

```
import tensorflow as tf
```

```
x2=tf.linspace(-1.0, 1.0, 10) #-1~1 사이의 숫자중에 10개를 랜덤으로 출력
```

```
#x2=tf.global_variables_initializer()
```

```
print(x2) #만든 설계를 생성
```

```
Tensor("LinSpace_1:0", shape=(10,), dtype=float32)
```

```
g=tf.get_default_graph()
```

```
print([op.name for op in g.get_operations()])
```

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

```
sess=tf.Session()
```

```
print(sess.run(x2))
```

```
[-1.          -0.7777778 -0.5555556 -0.3333333 -0.1111111  0.1111116
  0.3333333  0.5555556  0.7777778  1.          ]
```

```
sess.close()
```

텐서 플로우 실습4

```
import tensorflow as tf
```

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

```
print(hello)
```

```
a=tf.constant(10)
```

```
b=tf.constant(32)
```

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

```
print(c)
```

```
Tensor("Const:0", shape=(), dtype=string)
```

```
Tensor("Add:0", shape=(), dtype=int32)
```

문제1. 위에서 만든 텐서 그래프를 실행하시오.

답)

```
import tensorflow as tf
```

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

```
a=tf.constant(10)
```

```
b=tf.constant(32)
```

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

#모델을 구성하는 부분

#-----

#모델을 실행하는 부분

```
sess=tf.Session()
```

```
print(sess.run(hello))
```

```
print(sess.run([a,b,c]))
```

```
sess.close()
b'Hello Tensorflow!'
[10, 32, 42]
```

모델을 구성하는 부분과 실행하는 부분을 분리하여 프로그램을 깔끔하게 작성할 수 있다.

문제2. 아래의 모델(그래프)를 with 절을 사용해서 실행하시오.

보기)

```
import tensorflow as tf
```

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

답)

```
import tensorflow as tf
```

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

with tf.Session() as sess: #값을 계산하기 위해 세션 생성

```
print('a=',sess.run(a))
print('b=',sess.run(b))
print('c=',sess.run(c))
print('d=',sess.run(d))
print('e=',sess.run(e))
print('f=',sess.run(f))
print('g=',sess.run(g))
print('h=',sess.run(h))
```

```
a= 3
b= 9
c= 9
d= 54
e= 20
f= 1
g= 63
h= 63
```

텐서 플로우 실행구조

"session은 fetch와 feed 2가지 방법으로 처리 "

feed : placeholder 에 값을 넣어 실행하는 방법

fetch : 연산 결과를 fetch(가져오는) 방법

파이썬 기본문법과 텐서 플로우 문법 비교

1에서 5까지의 숫자를 출력한다.

파이썬 기본

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

텐서 플로우

```
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+=1
        print(sess.run(x))
```

문제3. 구구단 2단을 텐서로 출력하시오.

답1)

```
import tensorflow as tf

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

with tf.Session() as sess:
    for i in range(9):
        sess.run(model)
        x+=1
        print('2 x',sess.run(x),'=',sess.run(x)*2)
```

답2)

```
import tensorflow as tf

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

with tf.Session() as sess:
    for i in range(9):
        sess.run(model)
        mul=tf.multiply(x,y)
        print(sess.run(y),'x',sess.run(x),'=',sess.run(mul))
        x+=1
```

```

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

```

문제4. 2단부터 9단까지 출력하시오.

답)

```
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}))
```

```
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
```

```
8 x 9 = 72
```

```
9 x 1 = 9
```

```
9 x 2 = 18
```

```
9 x 3 = 27
```

```
9 x 4 = 36
```

```
9 x 5 = 45
```

```
9 x 6 = 54
```

```
9 x 7 = 63
```

```
9 x 8 = 72
```

```
9 x 9 = 81
```

■ Tensorflow 의 실행에 feed 예시

Session 은 feed 일 경우는 반드시 feed_dict 으로 처리값을 할당해야한다.

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

문제5. zero 와 숫자 1 을 채워넣는 배열을 생성하는 아래의
numpy 문법을 tensor 로 구현하시오 !

1. numpy

```

import numpy as np

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

print (a)
print (b)

```

답 :

```

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

문제6. 아래의 numpy 문법을 텐서 플로우로 구현하시오.

보기)

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

답)

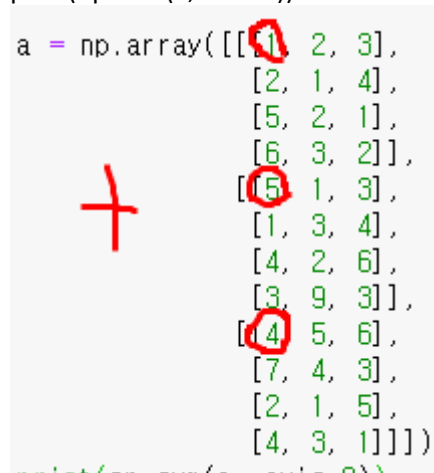
```
import tensorflow as tf
import numpy as np
```

```
a=np.array([0,0,0,1,0,0,0,0,0])
b=tf.argmax(a, axis=0)
with tf.Session() as sess:
    print(sess.run(b))
```

문제7. 아래의 numpy문법을 텐서 플로우로 구현하시오.

보기)

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



```
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]]])
```

답)

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

```
d=tf.reduce_sum(a,reduction_indices=[0])
with tf.Session() as sess:
    print(sess.run(d))
```

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

문제8. 아래의 numpy 문법을 텐서 플로우로 변경하시오.

보기)

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

답)

```
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)
```

텐서 플로우로 구현한 단층 신경망 중요한 문법1

문제9. 아래의 numpy 배열의 열단위 sum을 출력하시오.

보기)

```
import tensorflow as tf
import numpy as np

x=np.arange(6).reshape(2,3)
print(x)
```

```
[[0 1 2]
 [3 4 5]]
```

답)

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

```
x=np.arange(6).reshape(2,3)
d=tf.reduce_sum(x,reduction_indices=[0])
with tf.Session() as sess:
    print(sess.run(d))
[3 5 7]
```

텐서 플로우로 구현한 단층 신경망 중요한 문법2

문제10. 숫자 0으로 채워진 2행 3열 행렬을 만들고 숫자 1로 채워진 2행 3열 행렬을 만들고 두 행렬의

합을 출력하시오.

답)

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

```
b1=tf.zeros([2,3])
b2=tf.ones([2,3],"float")
res=tf.add(b1,b2)
```

```
with tf.Session() as sess:
    # print(sess.run(b1))
    # print(sess.run(b2))
```

```
    print(sess.run(res))
[[1. 1. 1.]
 [1. 1. 1.]]
```

문제11. 숫자 2로 채워진 2x3행렬을 x라는 변수로 만들고 숫자3으로 채워진 3x2행렬을 y로 만든후에

x행렬과 y행렬의 내적을 구하시오.

답1)

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

```
x=tf.placeholder('float',[2,3])
y=tf.placeholder('float',[3,2])
res=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(res, feed_dict={ x:[[2,2,2],[2,2,2]], y:[[3,3],[3,3],[3,3]]}))
[[2. 2. 2.]
 [2. 2. 2.]]
[[3. 3.]
 [3. 3.]
 [3. 3.]]
[[18. 18.]
 [18. 18.]]
```

답2)

```
x=tf.fill([2,3],2)
y=tf.fill([3,2],3)

sess=tf.Session()
print(sess.run(tf.matmul(x,y)))
#이렇게하면 session.close()를 해줘야함
```

답3)

```
a=tf.placeholder('float')
b=tf.placeholder('float')
x=tf.fill([2,3],a)
y=tf.fill([3,2],b)
result=tf.matmul(x,y)
sess=tf.Session()
print(sess.run(result, feed_dict={a:2,b:3}))
```

복습

1. 텐서 플로우를 사용했을때의 이점?

- 코드가 보기 쉽다. (모델을 생성하는 부분(오퍼레이션, 변수)
모델을 실행하는 부분(세션))
- 신경망 구현에 필요한 모든 함수들이 다 내장 되어있다.
- 속도가 빠르다.
- GPU를 사용할 수 있다.

텐서 플로우의 cast 함수의 이해

문제12. 아래의 배열의 True를 1로 변경하고 False를 0으로 변경시키시오.

답)

```
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]
```

```
a = tf.cast(correct_prediction, 'float')
with tf.Session() as sess:
    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. 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.
0. 1. 1. 1.]
```

문제13. 위의 출력된 결과에서 전체의 개수중에서 1이 몇 개나 되는지 정확도를 출력하시오.

답)

[illegible]

```
accuracy = tf.reduce_mean(tf.cast(correct_prediction, 'float'))
with tf.Session() as sess:
    print(sess.run(accuracy))
```

0.93

Mnist데이터로 단층 신경망 구현하기

2018년 9월 11일 화요일 오전 10:05

문제14. 텐서 플로우에 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)
```

문제15. 위의 mnist 데이터중에 train 데이터의 라벨을 one hot encoding 하지말고, 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)

#print(batch_xs.shape)
print(batch_ys) #훈련데이터 라벨
[9 7 5 9 4 7 3 4 1 9 5 9 1 3 3 6 3 2 9 6 1 6 7 9 5 2 6 2 1 7 1 3 1 8 7 3 3
 7 4 0 0 0 1 1 2 3 5 0 8 8 8 5 1 3 6 9 8 6 3 9 3 3 4 8 6 8 5 6 8 1 8 0 9 3
 4 7 1 6 8 0 9 7 9 6 3 5 9 3 5 0 8 8 2 2 4 6 8 4 8 7]
```

문제16. 이번에는 test 데이터와 test데이터의 라벨 100개를 가져오는데 shape만 출력하시오.

```
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)
batch_xt, batch_yt = mnist.test.next_batch(100)
print(batch_xt.shape) #테스트 데이터
(100, 784)
```

설명 : mnist 데이터는 훈련 데이터 6만장과 테스트 데이터 1만장으로 구성되어 있음.

문제17. 숫자 2로 채워진 행렬 2x3행렬을 텐서 플로우로 출력하시오.

답1)

```
import tensorflow as tf
a=tf.placeholder('float')
x=tf.fill([2,3],a)
with tf.Session() as sess:
    print(sess.run(x,feed_dict={a:2}))
[[2. 2. 2.]
 [2. 2. 2.]]
```

답2)

```
import tensorflow as tf
b=tf.placeholder('float',[None,3])

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

문제18. 아래의 None의 의미가 무엇인지 테스트 하시오.

```
보기)
import tensorflow as tf
b=tf.placeholder('float',[None,3])

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

설명 : 앞의 행의 개수가 몇 개로 들어오던 관계 없다는 뜻

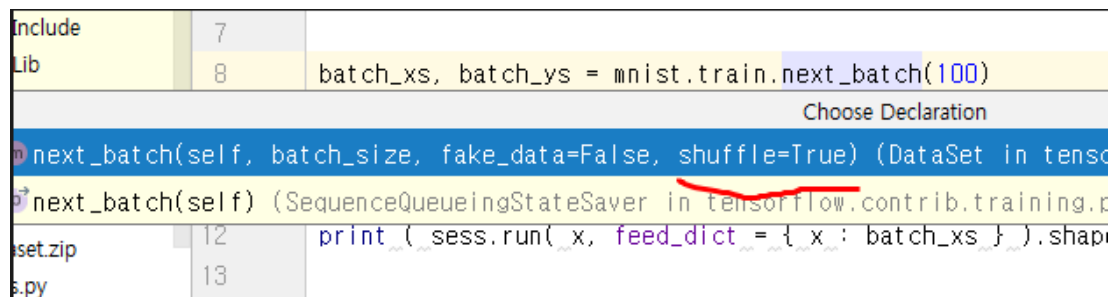
```
print(sess.run(b,feed_dict={b:[[2,2,2],[2,2,2]]}))
print(sess.run(b,feed_dict={b:[[2,2,2],[2,2,2],[2,2,2]]}))
[[2. 2. 2.]
 [2. 2. 2.]]
[[2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]]
```

문제19. mnist 데이터 784개의 맞게 x변수를 선언하고 배치로 입력될 데이터의 개수는 몇 개 이든지 상관없게 None으로 변수를 만들고 Mnist 데이터를 x변수에 100개를 담고 출력하 시오.

```
답)
import tensorflow as tf
x=tf.placeholder('float',[None,784])
batch_xs, batch_ys=mnist.train.next_batch(100) #next_batch:자동 shuffle 해서 뽑아짐

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

```
(100, 784)
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```



문제20. 위의 코드를 수정해서 훈련 데이터 100개 뿐만 아니라 훈련 데이터 라벨 100개로 출력하도록 코드를 수정하시오.

답)

```
import tensorflow as tf
x=tf.placeholder('float',[None,784])
y=tf.placeholder('float',[None,10])
mnist=input_data.read_data_sets("MNIST_data/",one_hot=True)
```

```
batch_xs, batch_ys=mnist.train.next_batch(100) #next_batch:자동 shuffle 해서 뽑아짐
batch_xt, batch_yt=mnist.test.next_batch(100)
```

```
with tf.Session() as sess:
    #print(sess.run(x,feed_dict={x:batch_xs}).shape)
    #print(sess.run(y,feed_dict={y:batch_ys}))
    print(sess.run(y,feed_dict={y:batch_ys}))
[[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 1.]
```

텐서 플로우로 가중치를 랜덤으로 생성하는 방법

문제21. 2x3행렬로 -1에서 1사이의 난수를 생성하는 변수를 W로 생성하고, 실행하시오.

답)

```
import tensorflow as tf
```

```
W=tf.Variable(tf.random_uniform([2,3],-1,1), name='W')
```

```
init=tf.global_variables_initializer() #변수 초기화를 해줘야한다.
```



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

```
[[ 0.8885119  0.01650453  0.39555407]
 [ 0.15406442 -0.9503174  -0.37816906]]
```

문제22. 위에서 배치 단위로 불러오는 입력 데이터 100x784와 내적할 가중치 행렬 W(784x50)을 생성.

답)

```
import tensorflow as tf

W=tf.Variable(tf.random_uniform([784,50],-1,1), name='W')
init=tf.global_variables_initializer()

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

```
[[[-0.9303236 -0.56139874  0.7627468 ... -0.71256113  0.6228123
  -0.74620175]
 [-0.21058035 -0.6232705  0.5696466 ... -0.67894053  0.4980166
  -0.6798265 ]
 [-0.02601695  0.37535262  0.5877044 ...  0.7947562 -0.8137593
  0.14077902]
 ...
 [-0.57022357  0.15898228 -0.665324 ...  0.5567727 -0.78723526
  -0.95187163]
 [ 0.9205117  0.11303067  0.56724524 ... -0.16901302  0.19513202
  -0.8802395 ]
 [ 0.66147923  0.7721181  0.61196613 ... -0.26731634  0.8020432
  0.6147742 ]]
```

문제23. 위에서 만든 입력값(100x784)와 지금 만든 가중치(784x50)행렬과 내적을 한 결과를 출력하시오.

답)

```
import tensorflow as tf

x=tf.placeholder('float',[None,784])
mnist=input_data.read_data_sets("MNIST_data/",one_hot=True)

W=tf.Variable(tf.random_uniform([784,50],-1,1), name='W')
init=tf.global_variables_initializer()
y=tf.matmul(x,W)

with tf.Session() as sess:
    sess.run(init)

    batch_xs, batch_ys=mnist.train.next_batch(100) #next_batch:자동 shuffle 해서 뽑아짐
    print(sess.run(y,feed_dict={x:batch_xs}))
```

문제24. 1x50으로 bias를 생성하는데 변수를 b로 해서 생성하고 숫자를 다 1로 채우시오.

답)

```
import tensorflow as tf
b=tf.ones([1,50])
```



```

init=tf.global_variables_initializer()
y=tf.matmul(x,W) + b
y_hat=tf.nn.relu(y)

```

```

with tf.Session() as sess:
    sess.run(init)

    batch_xs, batch_ys=mnist.train.next_batch(100) #next_batch:자동 shuffle 해서 뽑아짐
    print(sess.run(y_hat,feed_dict={x:batch_xs}))
    print(sess.run(y_hat,feed_dict={x:batch_xs}).shape)

```

문제28. 위에서 출력한 y_hat의 결과를 softmax 함수를 통과 시킨 결과를 출력하시오.

답)

```

import tensorflow as tf

mnist=input_data.read_data_sets("MNIST_data/",one_hot=True)
x=tf.placeholder('float',[None,784])
b=tf.ones([1,10]) #단층이라 출력층으로 바로가도록 10
W=tf.Variable(tf.random_uniform([784,10],-1,1), name='W') #단층이라 출력층으로 바로가
    #도록 10
init=tf.global_variables_initializer()
y=tf.matmul(x,W) + b
#y_hat=tf.nn.relu(y) #출력층으로 바로가서 빼줘도된다.
y_res=tf.nn.softmax(y)

with tf.Session() as sess:
    sess.run(init)

    batch_xs, batch_ys=mnist.train.next_batch(100) #next_batch:자동 shuffle 해서 뽑아짐
    print(sess.run(y_res,feed_dict={x:batch_xs}))
    print(sess.run(y_res,feed_dict={x:batch_xs}).shape)
(100, 10)

```

**문제29. 텐서 플로우의 argmax 함수를 이용해서 위에서 출력된 100x10 확률 벡터들의 최
대요소 인덱스를 출력하시오.**

답)

```

import tensorflow as tf

mnist=input_data.read_data_sets("MNIST_data/",one_hot=True)
x=tf.placeholder('float',[None,784])
b=tf.ones([1,10])
W=tf.Variable(tf.random_uniform([784,10],-1,1), name='W')
init=tf.global_variables_initializer()
y=tf.matmul(x,W) + b
#y_hat=tf.nn.relu(y) #출력층으로 바로가서 빼줘도된다.
y_res=tf.nn.softmax(y)
y_am=tf.argmax(y_res, axis=1)

with tf.Session() as sess:
    sess.run(init)

    batch_xs, batch_ys=mnist.train.next_batch(100) #next_batch:자동 shuffle 해서 뽑아짐
    print(sess.run(y_am,feed_dict={x:batch_xs}))

```

```
print(sess.run(y_am, feed_dict={x: batch_xs}).shape)

[[1 1 2 9 0 2 2 5 9 9 3 3 5 1 3 6 5 2 9 7 2 3 9 9 9 9 9 6 6 5 1 1 2 2 1
 3 1 2 1 7 2 1 1 5 2 9 1 1 1 3 3 9 1 3 1 1 9 1 9 5 5 2 2 9 9 1 1 3 2 5 5 1
 5 3 1 9 9 9 2 5 1 9 2 2 9 8 9 1 5 9 5 1 3 1 3 9 1 5]
(100,)]
```

문제30. 위의 코드에 라벨을 가져오는 코드를 추가해서 정확도를 추가하시오.

(예상 숫자 100개와 라벨 숫자 100개를 비교해서 정확도 출력)

힌트)

```
correct_prediction=tf.equal(y_am, y_label)
```

답)

```
import tensorflow as tf
```

```
batch_xs, batch_ys = mnist.train.next_batch(100) #next_batch: 자동 shuffle 해서 뽑아짐
batch_xt, batch_yt = mnist.test.next_batch(100)
mnist=input_data.read_data_sets("MNIST_data/", one_hot=False)
```

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

```
y_hat=tf.nn.softmax(y)
y_predict=tf.argmax(y_hat, axis=1)
```

```
correct_prediction=tf.equal(y_predict, batch_ys)
print(correct_prediction)
a=tf.cast(correct_prediction, 'float')
acc=tf.reduce_mean(a)
with tf.Session() as sess:
    sess.run(init)
    print(sess.run(acc, feed_dict={x: batch_xs}))
```

텐서 플로우로 구현하는 비용 함수

1. 최소 제곱 오차 함수 (mean square error)

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

2. 교차 엔트로피 오차 함수(cross entropy error)

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

문제31. 교차엔트로피 함수를 문제30번 코드에 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)
```

```
# 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 답을 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"))

init = tf.global_variables_initializer()

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

```

경사감소법을 텐서 플로우로 구현하는 방법

#####

그외 옵티마이저

optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)

SGD:

미니배치만큼 랜덤으로 데이터를 추출해서 확률적으로 경사를 감소하여 global minima로 찾아가는 방법

단점: Local minima에 잘 빠진다.

optimizer = tf.train.AdagradOptimizer(learning_rate=0.01)

러닝 레이트가 학습되면서 자동 조절되는 경사 하강법

optimizer = tf.train.MomentumOptimizer(learning_rate=0.01)

관성을 이용해서 Local minima에 빠지지 않게 하는 방법

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

adagrad의 장점 + Momentum의 장점

#####

문제32. 문제31번 코드에 SGD 경사 감소법 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 원한 인코딩해서 답을 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)

#SGD 경사 감소법 구현
optimizer = tf.train.GradientDescentOptimizer(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"))

init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)

    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train, feed_dict={x:batch_xs, y_onehot:batch_ys})

```

문제33. 위의 코드의 정확도를 출력하시오.

```

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

```

```

Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
0.15

```

문제34. 위의 코드중 아래의 3줄의 코드를 for loop문을 이용해서 반복수행해서 1에폭 돌게

하시오.

```
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)

```
with tf.Session() as sess:
    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}))
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
0.17
0.32
0.34
0.4
0.42
0.5
0.58
0.59
0.6
0.51
0.66
0.69
0.62
```

문제35. 위의 코드를 수정해서 아래와 같이 결과가 출력되게 하시오.

```
보기)
1 에폭 : 정확도 0.9
2 에폭 : 정확도 0.92
3 에폭 : 정확도 0.94
.
.
```

답)

```
with tf.Session() as sess:
    sess.run(init)
    for i in range(1,10001):
        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 (i/600,'에폭 정확도 :',sess.run(accuracy,
            feed_dict={x:batch_xs,y_onehot:batch_ys}))
```

```

1.0 에폭 정확도 : 0.94
2.0 에폭 정확도 : 0.94
3.0 에폭 정확도 : 0.91
4.0 에폭 정확도 : 0.92
5.0 에폭 정확도 : 0.95
6.0 에폭 정확도 : 0.92
7.0 에폭 정확도 : 0.97
8.0 에폭 정확도 : 0.94
9.0 에폭 정확도 : 0.94
10.0 에폭 정확도 : 0.99
11.0 에폭 정확도 : 0.96
12.0 에폭 정확도 : 0.96
13.0 에폭 정확도 : 0.96
14.0 에폭 정확도 : 0.94
15.0 에폭 정확도 : 0.97
16.0 에폭 정확도 : 0.93

```

문제36. 러닝 레이트를 0.05로 하고 학습시키고 정확도를 확인하시오.

답)

```
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.05)
```

문제37. 경사 감소법을 adam으로 변경해서 학습시키고 정확도를 확인하시오.

답)

#SGD 경사 감소법 구현

```
optimizer = tf.train.AdagradOptimizer(learning_rate=0.01)
```

```

1.0 에폭 정확도 : 0.69
2.0 에폭 정확도 : 0.8
3.0 에폭 정확도 : 0.78
4.0 에폭 정확도 : 0.87
5.0 에폭 정확도 : 0.86
6.0 에폭 정확도 : 0.87
7.0 에폭 정확도 : 0.84
8.0 에폭 정확도 : 0.86
9.0 에폭 정확도 : 0.85
10.0 에폭 정확도 : 0.88
11.0 에폭 정확도 : 0.87
12.0 에폭 정확도 : 0.84
13.0 에폭 정확도 : 0.85
14.0 에폭 정확도 : 0.91
15.0 에폭 정확도 : 0.91
16.0 에폭 정확도 : 0.92

```

문제39. 위의 단층 코드를 다층(2층)으로 변경하시오.

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
```

```
#입력층
```

```
#은닉 1층
```



```

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 # 내적
y1_relu = tf.nn.relu(y1) # 렐루 활성화 함수 사용

# 출력 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 # 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=0.001)

# 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(10000):
        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(i / 600 + 1,'epoch acc:', sess.run(acc, feed_dict={x:batch_xs, y_onehot: batch_ys}))

```

```

1.0 ecpo acc: 0.14
2.0 ecpo acc: 0.95
3.0 ecpo acc: 0.93
4.0 ecpo acc: 0.96
5.0 ecpo acc: 0.91
6.0 ecpo acc: 0.97
7.0 ecpo acc: 0.99
8.0 ecpo acc: 0.97
9.0 ecpo acc: 0.98
10.0 ecpo acc: 1.0
11.0 ecpo acc: 0.97
12.0 ecpo acc: 0.99
13.0 ecpo acc: 0.96
14.0 ecpo acc: 0.98
15.0 ecpo acc: 0.98
16.0 ecpo acc: 1.0
17.0 ecpo acc: 0.99

```

텐서 플로우로 배치 정규화 구현

배치 정규화 ?

신경망 학습시 가중치의 값의 데이터가 골고루 분산 될 수 있도록 하는 것을 강제하는 장치

구현코드

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

Relu들어가기 직전에 넣어준다.

문제40. 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)

tf.reset_default_graph()
# 계층 생성
#입력층
#은닉 1층
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 # 내적
batch_y1=tf.contrib.layers.batch_norm(y1,True)
y1_relu = tf.nn.relu(y1) # 렐루 활성화 함수 사용

# 출력 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 # 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=0.001)

# 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(10000):
        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(i / 600 + 1, 'epoch acc:', sess.run(acc, feed_dict={x: batch_xs, y_onehot: batch_ys}))

```

```

1.0 ecpo acc: 0.17
2.0 ecpo acc: 0.93
3.0 ecpo acc: 0.95
4.0 ecpo acc: 0.99
5.0 ecpo acc: 0.97
6.0 ecpo acc: 0.96
7.0 ecpo acc: 0.98
8.0 ecpo acc: 0.97
9.0 ecpo acc: 0.98
10.0 ecpo acc: 1.0
11.0 ecpo acc: 0.98
12.0 ecpo acc: 0.98
13.0 ecpo acc: 0.99
14.0 ecpo acc: 0.99
15.0 ecpo acc: 1.0
16.0 ecpo acc: 0.98
17.0 ecpo acc: 0.99

```

훈련할때 만들었던 최적의 감마와 베타를 테스트 할때 적용하는 코드

```

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

```

훈련할때 학습되면서 배치 정규화의 최적의 감마와 베타를 적용한다.

훈련하는 신경망에 테스트를 하는 코드를 추가

문제41. 지금 현재까지의 코드에는 신경망을 훈련만 시키는 코드였는데 테스트까지 진행해서 오버피팅이 발생했는지 확인할 수 있도록 에폭마다 훈련 데이터의 정확도와 테스트 데이터의 정확도를 같이 출력하시오.

답)

```

with tf.Session() as sess:
    sess.run(init)
    for i in range(10000):
        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(i / 600 + 1, 'ecpo train_acc:', sess.run(acc, feed_dict={x: train_xs, y_onehot: train_ys}))
            print(i / 600 + 1, 'ecpo test_acc:', sess.run(acc, feed_dict={x: test_xs, y_onehot: test_ys}))

```

문제42. 위의 결과를 그래프로 시각화 하시오.

답)

```

with tf.Session() as sess:
    sess.run(init)
    for i in range(10000):
        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(i / 600 + 1, 'ecpo train_acc:', sess.run(acc, feed_dict={x: train_xs, y_onehot: train_ys}))

```

```

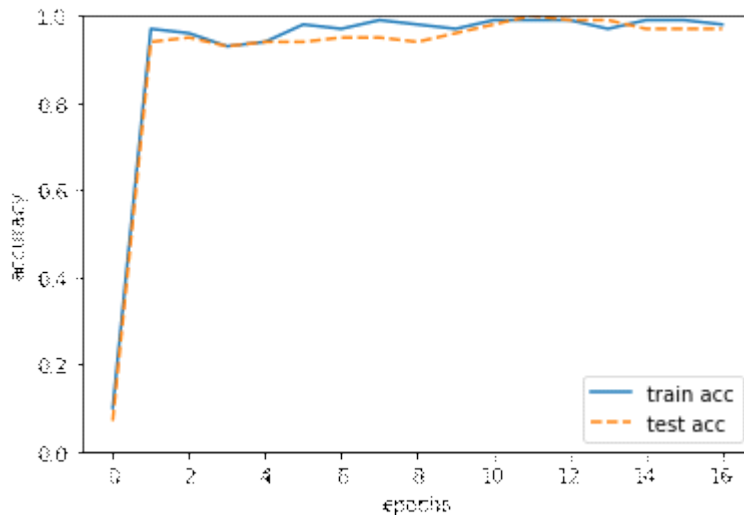
print(i / 600 + 1, 'ecpo test_acc:', sess.run(acc, feed_dict={x:test_xs, y_onehot: test_ys}))
print("=====")

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

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()

```



문제43. 위의 코드가 오버피팅 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(10000):
        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})

```

```

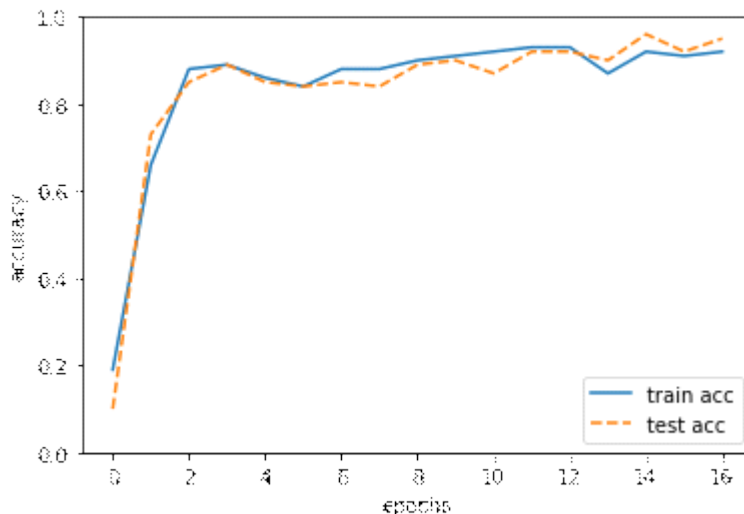
if i % 600 == 0:
    print(i / 600 + 1,'epoch train_acc:', sess.run(acc, feed_dict={x:train_xs, y_onehot:
train_ys,
                                keep_prob:1.0})))#전부 남겨놔라
    print(i / 600 + 1,'epoch test_acc:', sess.run(acc, feed_dict={x:test_xs, y_onehot: test_ys,
                                keep_prob:1.0})))#전부 남겨놔라
    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) # 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()

```



설명 : 노드개수를 개로하면 적어서 선이 왔다갔다하는데 100으로 해주면 어느정도 해결된다.

훈련할때만 keep_prob을 0.5(절반)만하고 정확도를 출력할때는 1.0(전부)으로 해줘야 한다.

그리고 위에 문제에는 오버피팅이 발생해서 러닝레이트를 더 줄여줬다 (0.001->0.0001)

문제44. 배치 정규화를 훈련시에는 배치 정규화를 켜고 테스트시에는 끄도록 코드를 구현하시오.

(훈련시에 만들었던 최적의 감마와 베타값을 사용하기 위해서)

```

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

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

#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,is_training=isTrain)
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()

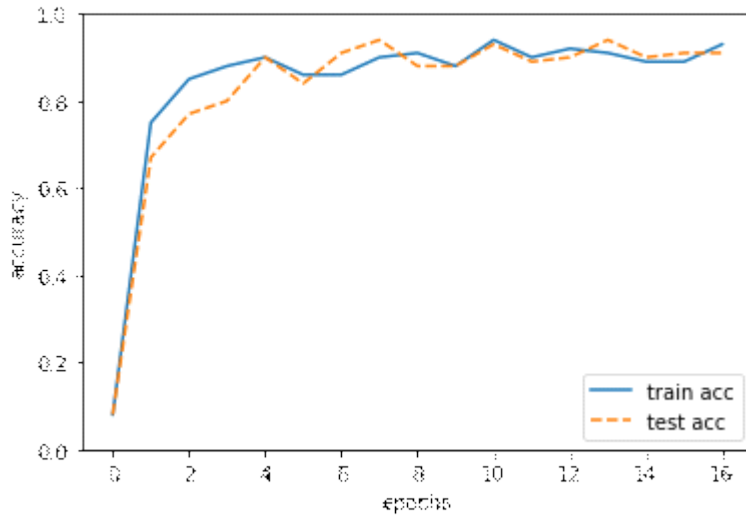
train_acc_list=[]
test_acc_list=[]
with tf.Session() as sess:
    sess.run(init)
    for i in range(10000):
        train_xs, train_ys = mnist.train.next_batch(100)
        test_xs, test_ys = mnist.test.next_batch(100)
        sess.run(Train, feed_dict={x : train_xs, y_onehot : train_ys, keep_prob:0.9, isTrain:True})
        if i % 600 == 0:
            print(i / 600 + 1,'ecpo train_acc :', sess.run(acc, feed_dict={x:train_xs, y_onehot:
train_ys,
                                                    keep_prob:1.0 })))#전부 남겨놔라
            print(i / 600 + 1,'ecpo test_acc:', sess.run(acc, feed_dict={x:test_xs, y_onehot: test_ys,
                                                    keep_prob:1.0 })))#전부 남겨놔라
            print("=====")

            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:True})

            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()

```



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
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

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

#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,is_training=isTrain)
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)

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

# 출력(3층)
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)

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)

# 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(10000):
        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, keep_prob:0.5, isTrain:True})
        if i % 600 == 0:
            print(i / 600 + 1,'epoch train_acc :', sess.run(acc, feed_dict={x:train_xs, y_onehot:
train_ys,
                                keep_prob:1.0 })))#전부 남겨놔라
            print(i / 600 + 1,'epoch test_acc:', sess.run(acc, feed_dict={x:test_xs, y_onehot: test_ys,
                                keep_prob:1.0 })))#전부 남겨놔라
    print("=====")

```

```

        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:True})

    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()

```

문제46. 2층 신경망 --> 3층 신경망으로 변경하고 정확도를 확인하시오.

```

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
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
batch_y1=tf.contrib.layers.batch_norm(y1,is_training=isTrain)
y1_relu = tf.nn.relu(y1) # 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
batch_y2=tf.contrib.layers.batch_norm(y1,is_training=isTrain)

```

```

y2_relu = tf.nn.relu(y2) # 렐루 활성화 함수 사용

#drop out
keep_prob = tf.placeholder('float')
y2_drop=tf.nn.dropout(y2_relu,keep_prob=0.5)

# 출력(3층)
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(y2_drop,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)

# Adam 경사 감소법
# optimizer = tf.train.AdamOptimizer(learning_rate=0.0001)

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)

# 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(10000):
        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, keep_prob:0.5, isTrain:True})
if i % 600 == 0:
    print(i / 600 + 1,'epoch train_acc :', sess.run(acc, feed_dict={x:train_xs, y_onehot:
train_ys,
                                keep_prob:1.0 })))#전부 남겨놔라
    print(i / 600 + 1,'epoch test_acc:', sess.run(acc, feed_dict={x:test_xs, y_onehot: test_ys,
                                keep_prob:1.0 })))#전부 남겨놔라
    print("=====")

    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:True})

    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()

```

텐서 플로우로 CNN 구현하기

```

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: 0.9 ,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()

```

문제47. 문제46번 코드를 오전에 그린 cnn코드로 구현하시오.

```

보기)
입력층 --> 은닉1층 --> 은닉2층 --> 은닉3층 --> 출력층
      (conv,pool) (conv,pool) (fully connected)

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

#mnist 데이터 로드하는 부분
mnist = input_data.read_data_sets('MNIST_data/', one_hot = True)
tf.reset_default_graph()

#입력데이터를 저장할 변수를 선언
x = tf.placeholder(tf.float32, [None, 28, 28, 1]) #cnn이라 입력이미지 그대로 입력
          #None : 배치 개수

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

#은닉 1층(conv-->relu-->pooling) ( (1,28,28) --> (32,14,14) )
W1 = tf.Variable(tf.random_normal([3, 3, 1, 32], stddev=0.01))
L1 = tf.nn.conv2d(x, W1, strides=[1, 1, 1, 1], padding='SAME') #SAME을 쓰면 그대로 나간다.
(28x28)
L1 = tf.nn.relu(L1)

```



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

```
#은닉 2층(conv-->relu-->pooling) ( (32,14,14) --> (64,7,7) )
```

```
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')
```

```
#완전 연결 계층 (64,7,7) --> (100,256)
```

```
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

#cnn에는 아래 두줄이 필요하다.
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, '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()

```

Cifar 10 이미지 분류

2018년 9월 14일 금요일 오전 11:49

cifar10 데이터를 신경망에 로드하는 데이터 전처리 코드 작성

* cifar10 데이터 소개

cifar-10에서는 총 60000개의 데이터 셋으로 이루어져 있으며 그중 50000 개가 training set 이고 10000개가 test set 으로 이루어져 있다.

class는 airplane 부터 truck 까지 10개로 구성되어 있다.

문제48. C:\python_data\cifar10\test100 폴더를 만들고 test 이미지 100개를 이 폴더에 따로 복사하고 복사한 이미지를 아래와 같이 불러오는 함수를 생성하시오 !

답)

```
import os
```

```
test_image='C:\\python_data\\cifar10\\test100'
```

```
def image_load(path):  
    file_list = os.listdir(path)  
    return file_list
```

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

```
['1.png', '10.png', '100.png', '11.png',  
'21.png', '22.png', '23.png', '24.png',  
'34.png', '35.png', '36.png', '37.png',
```

문제49. 위의 함수를 수정해서 아래와 같이 숫자만 출력되게 하시오 !

답)

```
import os  
import re
```

```
test_image='C:\\python_data\\cifar10\\test100'
```

```
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
```

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

```
[1, 10, 100, 11, 12, 13, 14,  
8, 29, 3, 30, 31, 32, 33, 34
```

문제50. 위에 결과를 정렬해서 출력되게하시오 !

```
보기)
import os
import re

test_image='C:\\python_data\\cifar10\\test100'
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

print ( image_load(test_image) )
```

문제51. 문제50번에 나온 결과에 png 를 붙여서 결과가 출력되게 하시오 !

```
답)
import os
import re

test_image='C:\\python_data\\cifar10\\test100'
def image_load(path):
    file_list = os.listdir(path)
    file_name=[]
    for i in file_list:
        file_name.append(int(re.sub('[^0-9]','',i)))
    file_name.sort()

    file_res=[]
    for j in file_name:
        file_res.append('%d.png' %j)
    return file_res

print ( image_load(test_image) )

['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']
```

문제52. 이미지 이름 앞에 절대경로가 붙게 하시오 !

```
답)
import os
import re

test_image='C:\\python_data\\cifar10\\test100'

def image_load(path):
    file_list = os.listdir(path)
    file_name=[]
    for i in file_list:
        file_name.append(int(re.sub('[^0-9]','',i)))
    file_name.sort()
```

```

file_res=[]
for j in file_name:
    file_res.append('%s\\%d.png' %(test_image,j))
return file_res

print ( image_load(test_image))

['C:\\python_data\\cifar10\\test100\\1.png', 'C:\\python_data\\cifar10\\test100\\2
1.png', 'C:\\python_data\\cifar10\\test100\\3.png', 'C:\\python_data\\cifar10\\test100\\4.png', 'C:\\python_data\\cifar10\\test100\\5.png', 'C:\\python_data\\cifar10\\test100\\6.png', 'C:\\python_data\\cifar10\\test100\\7.png', 'C:\\python_data\\cifar10\\test100\\8.png', 'C:\\python_data\\cifar10\\test100\\9.png', 'C:\\python_data\\cifar10\\test100\\10.png', 'C:\\python_data\\cifar10\\test100\\11.png', 'C:\\python_data\\cifar10\\test100\\12.png', 'C:\\python_data\\cifar10\\test100\\13.png', 'C:\\python_data\\cifar10\\test100\\14.png', 'C:\\python_data\\cifar10\\test100\\15.png', 'C:\\python_data\\cifar10\\test100\\16.png', 'C:\\python_data\\cifar10\\test100\\17.png', 'C:\\python_data\\cifar10\\test100\\18.png', 'C:\\python_data\\cifar10\\test100\\19.png', 'C:\\python_data\\cifar10\\test100\\20.png', 'C:\\python_data\\cifar10\\test100\\21.png', 'C:\\python_data\\cifar10\\test100\\22.png', 'C:\\python_data\\cifar10\\test100\\23.png', 'C:\\python_data\\cifar10\\test100\\24.png', 'C:\\python_data\\cifar10\\test100\\25.png', 'C:\\python_data\\cifar10\\test100\\26.png', 'C:\\python_data\\cifar10\\test100\\27.png', 'C:\\python_data\\cifar10\\test100\\28.png', 'C:\\python_data\\cifar10\\test100\\29.png', 'C:\\python_data\\cifar10\\test100\\30.png', 'C:\\python_data\\cifar10\\test100\\31.png', 'C:\\python_data\\cifar10\\test100\\32.png', 'C:\\python_data\\cifar10\\test100\\33.png', 'C:\\python_data\\cifar10\\test100\\34.png', 'C:\\python_data\\cifar10\\test100\\35.png', 'C:\\python_data\\cifar10\\test100\\36.png', 'C:\\python_data\\cifar10\\test100\\37.png', 'C:\\python_data\\cifar10\\test100\\38.png', 'C:\\python_data\\cifar10\\test100\\39.png', 'C:\\python_data\\cifar10\\test100\\40.png', 'C:\\python_data\\cifar10\\test100\\41.png', 'C:\\python_data\\cifar10\\test100\\42.png', 'C:\\python_data\\cifar10\\test100\\43.png', 'C:\\python_data\\cifar10\\test100\\44.png', 'C:\\python_data\\cifar10\\test100\\45.png', 'C:\\python_data\\cifar10\\test100\\46.png', 'C:\\python_data\\cifar10\\test100\\47.png', 'C:\\python_data\\cifar10\\test100\\48.png', 'C:\\python_data\\cifar10\\test100\\49.png', 'C:\\python_data\\cifar10\\test100\\50.png', 'C:\\python_data\\cifar10\\test100\\51.png', 'C:\\python_data\\cifar10\\test100\\52.png', 'C:\\python_data\\cifar10\\test100\\53.png', 'C:\\python_data\\cifar10\\test100\\54.png', 'C:\\python_data\\cifar10\\test100\\55.png', 'C:\\python_data\\cifar10\\test100\\56.png', 'C:\\python_data\\cifar10\\test100\\57.png', 'C:\\python_data\\cifar10\\test100\\58.png', 'C:\\python_data\\cifar10\\test100\\59.png', 'C:\\python_data\\cifar10\\test100\\60.png', 'C:\\python_data\\cifar10\\test100\\61.png', 'C:\\python_data\\cifar10\\test100\\62.png', 'C:\\python_data\\cifar10\\test100\\63.png', 'C:\\python_data\\cifar10\\test100\\64.png', 'C:\\python_data\\cifar10\\test100\\65.png', 'C:\\python_data\\cifar10\\test100\\66.png', 'C:\\python_data\\cifar10\\test100\\67.png', 'C:\\python_data\\cifar10\\test100\\68.png', 'C:\\python_data\\cifar10\\test100\\69.png', 'C:\\python_data\\cifar10\\test100\\70.png', 'C:\\python_data\\cifar10\\test100\\71.png', 'C:\\python_data\\cifar10\\test100\\72.png', 'C:\\python_data\\cifar10\\test100\\73.png', 'C:\\python_data\\cifar10\\test100\\74.png', 'C:\\python_data\\cifar10\\test100\\75.png', 'C:\\python_data\\cifar10\\test100\\76.png', 'C:\\python_data\\cifar10\\test100\\77.png', 'C:\\python_data\\cifar10\\test100\\78.png', 'C:\\python_data\\cifar10\\test100\\79.png', 'C:\\python_data\\cifar10\\test100\\80.png', 'C:\\python_data\\cifar10\\test100\\81.png', 'C:\\python_data\\cifar10\\test100\\82.png', 'C:\\python_data\\cifar10\\test100\\83.png', 'C:\\python_data\\cifar10\\test100\\84.png', 'C:\\python_data\\cifar10\\test100\\85.png', 'C:\\python_data\\cifar10\\test100\\86.png', 'C:\\python_data\\cifar10\\test100\\87.png', 'C:\\python_data\\cifar10\\test100\\88.png', 'C:\\python_data\\cifar10\\test100\\89.png', 'C:\\python_data\\cifar10\\test100\\90.png', 'C:\\python_data\\cifar10\\test100\\91.png', 'C:\\python_data\\cifar10\\test100\\92.png', 'C:\\python_data\\cifar10\\test100\\93.png', 'C:\\python_data\\cifar10\\test100\\94.png', 'C:\\python_data\\cifar10\\test100\\95.png', 'C:\\python_data\\cifar10\\test100\\96.png', 'C:\\python_data\\cifar10\\test100\\97.png', 'C:\\python_data\\cifar10\\test100\\98.png', 'C:\\python_data\\cifar10\\test100\\99.png', 'C:\\python_data\\cifar10\\test100\\100.png']

```

문제53. 위의 이미지들을 cv2.imread 함수를 이용해서 숫자 list 로 변환하시오 !

```

답)
import os
import re
import cv2

test_image='C:\\python_data\\cifar10\\test100'

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

    file_res=[]
    for j in file_name:
        file_res.append('%s\\%d.png' %(test_image,j))

    image=[]
    for k in file_res:
        img=cv2.imread(k)
        image.append(img)
    return image

print ( image_load(test_image))

```

```

b g r
[array([[ 49, 112, 158],
        [ 47, 111, 159],
        [ 51, 116, 165],
        ...,
        [ 36,  95, 137],
        [ 36,  91, 126],
        [ 33,  85, 116]])]

```

문제54. 위의 숫자 list 를 numpy 배열로 변환해서 출력되게 하시오 !

```

답)
import cv2
import numpy as np
import csv

return np.array(image)

```

문제55. test_label.csv 파일을 C:\python_data\cifar10\ 밑에 복사하고 결과가 아래와 같이

출력될 수 있도록 함수를 생성하시오 !

결과 :

['3'], ['8'], ['8'], ['0'], ...

답)

```
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
print(label_load(test_label))
```

문제56. 위의 숫자 list 를 numpy 배열로 변환하시오 !

답)

```
def label_load(path):
    file=open(path)
    label_data=csv.reader(file)
    label_list=[]

    for i in label_data:
        label_list.append(i)
    label=np.array(label_list)
    return label
print(label_load(test_label))

[[ '3' ]
 [ '8' ]
 [ '8' ]
 ...
 [ '5' ]
 [ '1' ]
 [ '7' ]]
```

문제57. 위의 결과가 문자가 아니라 숫자로 출력되게 변환하시오 !

답)

```
def label_load(path):

    file = open(path)
    labeldata = csv.reader(file)
    labellist = []

    for i in labeldata:
        labellist.append(i)

    label = np.array(labellist)
    label = label.astype(int)

    return label

print ( label_load(test_label) )
```

```
[[3]
 [8]
 [8]
 ...
 [5]
 [1]
 [7]]
```

이미지 데이터 신경망에 로드하기 위해 반드시 알아야하는 내용

1. 훈련 데이터 이미지를 numpy 배열로 변환하는 방법
2. 라벨 데이터를 numpy 이미지로 변환하는 방법
3. one hot encoding 된 데이터에서 numpy 로 최대 인덱스 가져오는 방법
4. 배치 처리를 하기 위해 next_batch 함수 만드는 방법

one hot encoding 된 데이터에서 numpy 로 최대 인덱스 가져오는 방법

문제58. 아래의 결과를 출력해보시오 !

보기)

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

답)

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

문제59. 문제57번에서 가져온 숫자 리스트를 가지고 아래와 같이 one hot encoding 된 결과를 출력하시오.

보기)

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

답)

```
def label_load(path):
    file=open(path)
    label_data=csv.reader(file)
    label_list=[]

    for i in label_data:
        label_list.append(i)
    label=np.array(label_list)
    label=label.astype(int)
    label=np.eye(10)[label]
    return label

print(label_load(test_label))
```

```

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

```

문제60. 위의 차원은 3차원인데 우리는 2차원으로 줄여야한다. 왜냐면 `cnn` 코드에서 라벨이 입력될 때는 아래처럼 2차원이기 때문이다. 그래서 차원을 줄이는 함수를 테스트하시오 !

```

Ex)
import numpy as np
x=np.array([[[0],[1],[2]]])
print(x.shape)
print(np.squeeze(x).shape)
print(np.squeeze(x, axis=0).shape)
print(np.squeeze(x, axis=2).shape)
(1, 3, 1)
(3,)
(3, 1)
(1, 3)

```

문제61. 라벨의 차원을 3차원에서 2차원으로 줄이시오.

```

보기)
(10000, 1, 10) ---> ( 10000,10)
0  1  2

```

```

답)
import os
import re
import cv2
import numpy as np
import csv

test_image='C:/python_data/cifar10/test100/'
test_label='C:/python_data/cifar10/test_label.csv'

def label_load(path):
    file=open(path)
    label_data=csv.reader(file)
    label_list=[]

    for i in label_data:
        label_list.append(i)
    label=np.array(label_list)
    label=label.astype(int)
    label=np.eye(10)[label]
    label=np.squeeze(label,axis=1)
    return label.shape

```



```
print(label_load(test_label))
```

문제62. 오전에 만들었던 두가지 함수(image_load, label_load)를 loader2.py라는 파일
썬코드에 저장하고 아래와 같이 loader2.py를 import한 후에 cifar10전체 데이터를
로드하는 코드를 구현하시오.

답)

```
import loader2
import time

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

print("LOADING DATA")
start = time.time()
trainX = loader2.image_load(train_image)
print(trainX.shape) # (50000, 32, 32,3)
trainY = loader2.label_load(train_label)
print(trainY.shape) # (50000, 10)
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))
```

일단 넘어가자

문제63. test100 폴더 밑에 100개의 데이터 중 10개만 출력하시오

답)

data를 shuffle 하는 함수 생성

문제66. 아래의 코드를 실행해보시오

보기)

```
import random
import numpy as np

np.arange(10)
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

문제67. 위의 숫자 10개가 랜덤으로 섞여서 출력되게하시오(random.shuffle 사용)

답)

```
import random
import numpy as np

data=np.arange(10)
```

```
random.shuffle(data)
print(data)
[6 7 0 8 5 3 9 2 1 4]
```

문제68. 위의코드를이용해서shuffle_batch 함수를 만들어서입력된 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))
```


문제73. 아래와 같이 결과를 정렬해서 출력하시오.

보기)

[1,2,3,4,...,199,200]

답)

import os

import re

test_images='C:\python_data\catdog'

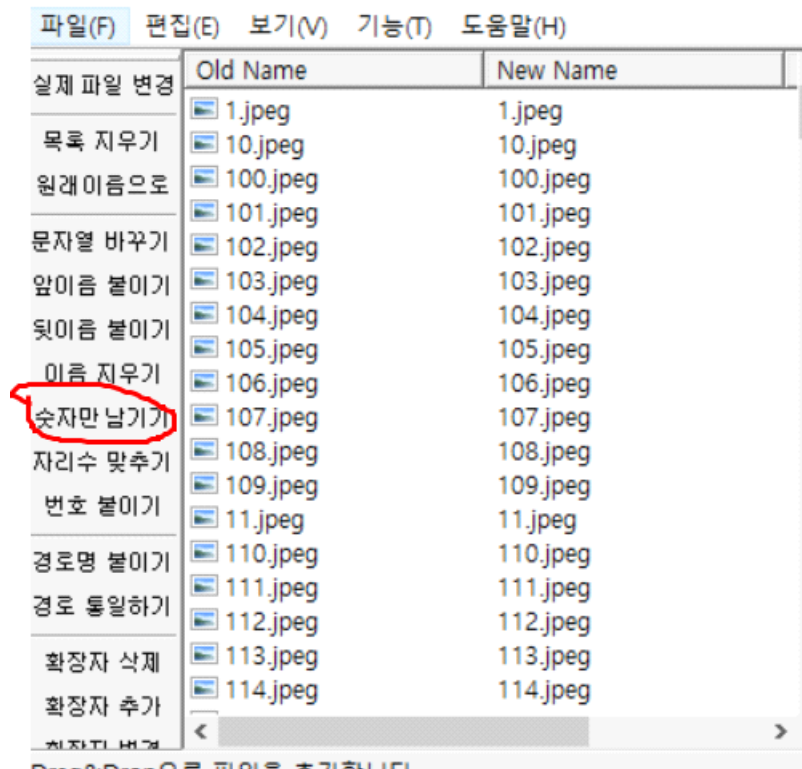
```
def image_load(path):
    file_list = os.listdir(path)
    file_name=[]
    for i in file_list:
        a=int(re.sub('[^0-9]','',i))
        file_name.append(a)
    file_name.sort()
    return file_name
```

print (image_load(test_images))

```
[1, 2, 3, 4, 5, 6,
... ..
196, 197, 198, 199, 200]
```

문제74. darknamer.exe 프로그램을 이용해서 (1).jpeg를 1.jpeg로 전부 변경하시오.

답)



문제75. 문제73번에서 나온 결과에 jpeg를 붙여서 출력되도록 하시오.

답)

import os

import re

```

test_images='C:\python_data\catdog'

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

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

print ( image_load(test_images) )

```

```

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

```

문제76. 위의 결과 앞에 절대경로를 붙여서 출력하시오.

```

답)
import os
import re

test_images='C:\python_data\catdog'

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

    file_res=[]
    for j in file_name:
        file_res.append('%s\%d.jpeg' %(test_images,j))
    return file_res

print ( image_load(test_images) )

```

```

['C:\\python_data\\catdog_1.jpeg', 'C:\\python_data\\catdog_2.jpeg',
 'C:\\python_data\\catdog_3.jpeg', 'C:\\python_data\\catdog_4.jpeg',
 'C:\\python_data\\catdog_5.jpeg', 'C:\\python_data\\catdog_6.jpeg',
 'C:\\python_data\\catdog_7.jpeg', 'C:\\python_data\\catdog_8.jpeg',
 'C:\\python_data\\catdog_9.jpeg', 'C:\\python_data\\catdog_10.jpeg',
 'C:\\python_data\\catdog_11.jpeg', 'C:\\python_data\\catdog_12.jpeg',
 'C:\\python_data\\catdog_13.jpeg', 'C:\\python_data\\catdog_14.jpeg',
 'C:\\python_data\\catdog_15.jpeg', 'C:\\python_data\\catdog_16.jpeg',
 'C:\\python_data\\catdog_17.jpeg', 'C:\\python_data\\catdog_18.jpeg',
 'C:\\python_data\\catdog_19.jpeg', 'C:\\python_data\\catdog_20.jpeg',
 'C:\\python_data\\catdog_21.jpeg', 'C:\\python_data\\catdog_22.jpeg',
 'C:\\python_data\\catdog_23.jpeg', 'C:\\python_data\\catdog_24.jpeg',
 'C:\\python_data\\catdog_25.jpeg', 'C:\\python_data\\catdog_26.jpeg',
 'C:\\python_data\\catdog_27.jpeg', 'C:\\python_data\\catdog_28.jpeg',
 'C:\\python_data\\catdog_29.jpeg', 'C:\\python_data\\catdog_30.jpeg',
 'C:\\python_data\\catdog_31.jpeg', 'C:\\python_data\\catdog_32.jpeg',
 'C:\\python_data\\catdog_33.jpeg', 'C:\\python_data\\catdog_34.jpeg',
 'C:\\python_data\\catdog_35.jpeg', 'C:\\python_data\\catdog_36.jpeg',
 'C:\\python_data\\catdog_37.jpeg', 'C:\\python_data\\catdog_38.jpeg',
 'C:\\python_data\\catdog_39.jpeg', 'C:\\python_data\\catdog_40.jpeg',
 'C:\\python_data\\catdog_41.jpeg', 'C:\\python_data\\catdog_42.jpeg',
 'C:\\python_data\\catdog_43.jpeg', 'C:\\python_data\\catdog_44.jpeg',
 'C:\\python_data\\catdog_45.jpeg', 'C:\\python_data\\catdog_46.jpeg',
 'C:\\python_data\\catdog_47.jpeg', 'C:\\python_data\\catdog_48.jpeg',
 'C:\\python_data\\catdog_49.jpeg', 'C:\\python_data\\catdog_50.jpeg',
 'C:\\python_data\\catdog_51.jpeg', 'C:\\python_data\\catdog_52.jpeg',
 'C:\\python_data\\catdog_53.jpeg', 'C:\\python_data\\catdog_54.jpeg',
 'C:\\python_data\\catdog_55.jpeg', 'C:\\python_data\\catdog_56.jpeg',
 'C:\\python_data\\catdog_57.jpeg', 'C:\\python_data\\catdog_58.jpeg',
 'C:\\python_data\\catdog_59.jpeg', 'C:\\python_data\\catdog_60.jpeg',
 'C:\\python_data\\catdog_61.jpeg', 'C:\\python_data\\catdog_62.jpeg',
 'C:\\python_data\\catdog_63.jpeg', 'C:\\python_data\\catdog_64.jpeg',
 'C:\\python_data\\catdog_65.jpeg', 'C:\\python_data\\catdog_66.jpeg',
 'C:\\python_data\\catdog_67.jpeg', 'C:\\python_data\\catdog_68.jpeg',
 'C:\\python_data\\catdog_69.jpeg', 'C:\\python_data\\catdog_70.jpeg',
 'C:\\python_data\\catdog_71.jpeg', 'C:\\python_data\\catdog_72.jpeg',
 'C:\\python_data\\catdog_73.jpeg', 'C:\\python_data\\catdog_74.jpeg',
 'C:\\python_data\\catdog_75.jpeg', 'C:\\python_data\\catdog_76.jpeg',
 'C:\\python_data\\catdog_77.jpeg', 'C:\\python_data\\catdog_78.jpeg',
 'C:\\python_data\\catdog_79.jpeg', 'C:\\python_data\\catdog_80.jpeg',
 'C:\\python_data\\catdog_81.jpeg', 'C:\\python_data\\catdog_82.jpeg',
 'C:\\python_data\\catdog_83.jpeg', 'C:\\python_data\\catdog_84.jpeg',
 'C:\\python_data\\catdog_85.jpeg', 'C:\\python_data\\catdog_86.jpeg',
 'C:\\python_data\\catdog_87.jpeg', 'C:\\python_data\\catdog_88.jpeg',
 'C:\\python_data\\catdog_89.jpeg', 'C:\\python_data\\catdog_90.jpeg',
 'C:\\python_data\\catdog_91.jpeg', 'C:\\python_data\\catdog_92.jpeg',
 'C:\\python_data\\catdog_93.jpeg', 'C:\\python_data\\catdog_94.jpeg',
 'C:\\python_data\\catdog_95.jpeg', 'C:\\python_data\\catdog_96.jpeg',
 'C:\\python_data\\catdog_97.jpeg', 'C:\\python_data\\catdog_98.jpeg',
 'C:\\python_data\\catdog_99.jpeg', 'C:\\python_data\\catdog_100.jpeg']

```

문제77. 위의 이미지들을 cv2.imread함수를 이용해서 숫자list로 변환하시오.

```

답)
import os
import re
import cv2

test_images='C:\python_data\catdog'

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

file_res=[]
for j in file_name:
    file_res.append('%s\\%d.jpeg'%(test_images,j)) #:\\을 붙여야지 None이 안나온다.
#return file_res

image=[]
for k in file_res:
    img=cv2.imread(k)
    image.append(img)
return image

print ( image_load(test_images) )

```

```

[array([[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],
        [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]]],
      dtype=uint8)

```

문제78. 위의 숫자 list를 numpy배열로 변환해서 출력하시오,

답)

```

import numpy as np
.
.
image=[]
for k in file_res:
    img=cv2.imread(k)
    image.append(img)
return np.array(image)

```

문제79. 라벨 csv를 읽어와서 아래와 같이 결과가 출력되는 함수를 생성하시오.

보기)

```

print(label_load(test_label))

```

결과)

```

['1'],['1'],['1']

```

답)

```

import csv

test_label='C:\python_data\cat_dog_label.csv'

```

```
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
print(label_load(test_label))
```

문제80. 위의 숫자 list를 numpy배열로 변환하시오.

답)

```
import csv
import numpy as np

test_label='C:\python_data\cat_dog_label.csv'

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)
print(label_load(test_label))
```

```
[[ '1' ]
 [ '1' ]
 [ '1' ]
 [ '1' ]
 [ '1' ]
 [ '1' ]
 [ '1' ]
 [ '1' ]]
```

문자 -> 숫자 변환

```
def label_load(path):
    file=open(path)
    label_data=csv.reader(file)
    label_list=[]

    for i in label_data:
        label_list.append(i)
    label = np.array(label_list)
    label = label.astype(int)

    return np.array(label)
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 되도록 생성
ex) mnist : [0 0 0 1 0 0 0 0 0]
개/고양이 : [0 1]

문제81. label_load 함수가 아래와 같이 one hot encoding된 결과로 출력되도록 코드를 추가하시오.

답)

```
def label_load(path):
    file=open(path)
    label_data=csv.reader(file)
    label_list=[]

    for i in label_data:
        label_list.append(i)
    label = np.array(label_list)
    label = label.astype(int)
    label = np.eye(2)[label]

    return np.array(label)
print(label_load(test_label))
```

문제82. 위의 라벨은 3차원인데 cnn 신경망에서 사용하려면 2차원으로 줄여야한다. 그래서 차원을 줄이는 코드를 추가하시오.

보기)

(200, 1, 2) --->(200, 2)

답)

```
def label_load(path):
    file=open(path)
    label_data=csv.reader(file)
    label_list=[]

    for i in label_data:
        label_list.append(i)
    label = np.array(label_list)

    #숫자형으로 변환
    label = label.astype(int)

    #라벨 one hot encoding
    label = np.eye(2)[label]

    #라벨 차원 줄이기
    label=np.squeeze(label,axis=1)
    return label.shape

print(label_load(test_label))

(200, 2)
```

개/고양이 이미지를 batch 단위로 신경망에 입력하는 코드

개100, 고양이100 --->suffle ---> 10장씩 배치로 입력

문제83. cifar10 이미지 신경망 생성할때 사용했던 next_batch함수를 가지고 와서 10개

씩 배치되게 하시오.

답)

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

test_images='C:\python_data\catdog'
test_label='C:\python_data\cat_dog_label.csv'

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

    file_res=[]
    for j in file_name:
        file_res.append('%s\\%d.jpeg'%(test_images,j))
    #return file_res

    image=[]
    for k in file_res:
        img=cv2.imread(k)
        image.append(img)
    return np.array(image)

def label_load(path):
    file=open(path)
    label_data=csv.reader(file)
    label_list=[]

    for i in label_data:
        label_list.append(i)
    label = np.array(label_list)

    #숫자형으로 변환
    label = label.astype(int)

    #라벨 one hot encoding
    label = np.eye(2)[label]

    #라벨 차원 줄이기
    label=np.squeeze(label,axis=1)
    return label

print("LOADING DATA")

testX = image_load(test_images)
testY = label_load(test_label)

def next_batch(data_list,label,idx,batch_size):
```

```

batch1 = data_list[idx * batch_size:idx * batch_size + batch_size]
label2 = label[idx * batch_size:idx * batch_size + batch_size]

return batch1, label2

```

```
print(next_batch(testX,testY,0,10))
```

```

LOADING DATA
(array([[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],
       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],
       ...

```

문제84. cifar10에서 사용했던 shuffle_batch 함수를 생성하시오.

```

답)
import os
import re
import cv2
import numpy as np
import random

test_images='C:\python_data\catdog'
test_label='C:\python_data\cat_dog_label.csv'

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

    file_res=[]
    for j in file_name:
        file_res.append('%s\\%d.jpeg'%(test_images,j))
    #return file_res

    image=[]
    for k in file_res:
        img=cv2.imread(k)
        image.append(img)
    return np.array(image)

def label_load(path):
    file=open(path)
    label_data=csv.reader(file)
    label_list=[]

    for i in label_data:

```

```

        label_list.append(i)
    label = np.array(label_list)

    #숫자형으로 변환
    label = label.astype(int)

    #라벨 one hot encoding
    label = np.eye(2)[label]

    #라벨 차원 줄이기
    label=np.squeeze(label,axis=1)
    return label

print("LOADING DATA")

testX = image_load(test_images)
testY = label_load(test_label)

def next_batch(data_list,label,idx,batch_size):

    batch1 = data_list[idx * batch_size:idx * batch_size + batch_size]
    label2 = label[idx * batch_size:idx * batch_size + batch_size]

    return batch1, label2

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

print( shuffle_batch(testX, testY) )

```

문제85. 위에서 만든 함수 4개를 loader3.py라는 모듈로 저장하시오.

함수 리스트)

1. image_load
2. label_load
3. next_batch
4. shuffle_batch

답)

```

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

# test_image ='C:\python_data\catdog'
# test_label='C:\python_data\cat_dog_label.csv'

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

file_res=[]
for j in file_name:
    file_res.append('%s\\%d.jpeg' %(path,j))
#return file_res

image=[]
for k in file_res:
    img=cv2.imread(k)
    image.append(img)
return np.array(image)

def label_load(path):
    file=open(path)
    label_data=csv.reader(file)
    label_list=[]

    for i in label_data:
        label_list.append(i)
    label = np.array(label_list)

    #숫자형으로 변환
    label = label.astype(int)

    #라벨 one hot encoding
    label = np.eye(2)[label]

    #라벨 차원 줄이기
    label=np.squeeze(label,axis=1)
    return label

#print("LOADING DATA")

# testX = image_load(test_image)
# testY = label_load(test_label)

def next_batch(data_list,label,idx,batch_size):

    batch1 = data_list[idx * batch_size:idx * batch_size + batch_size]
    label2 = label[idx * batch_size:idx * batch_size + batch_size]

    return batch1, label2

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

#print( shuffle_batch(testX, testY) )

```

jpeg를 jpg로 전부 변경하고 돌리시오

문제86. 기존에 cifar cnn코드 + 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 numpy as np
import loader3

train_image = 'C:\\python_data\\catdog\\'
train_label = 'C:\\python_data\\train_4000_label.csv'

test_image = 'C:\\python_data\\catdog\\train_4000\\'
test_label = 'C:\\python_data\\catdog\\train_4000\\cat_dog_label.csv'

print("LOADING DATA")
# cifar10 데이터 로드

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) #drop out을 쓴다.(무엇을뺄지)

# 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')

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')
```

```

W3 = tf.Variable(tf.random_normal([32 * 32 * 64, 256], stddev=0.01))
#W3 = tf.get_variable(name='W3', shape=[8*8*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)

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)

#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 = 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: 0.9, 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()

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

train_image = 'C:\\python_data\\train_4000\\'
train_label = 'C:\\python_data\\train_label_4000.csv'

test_image = 'C:\\python_data\\catdog\\'
test_label = 'C:\\python_data\\cat_dog_label.csv'

print("LOADING DATA")
# cifar10 데이터 로드

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')

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')

W3 = tf.Variable(tf.random_normal([32 * 32 * 64, 256], stddev=0.01))
#W3 = tf.get_variable(name='W3', shape=[8*8*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)

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)

#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=0.001).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: 0.9, 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()

```