# 0. 텐서플로우 설치

2018년 9월 10일 월요일 오전 10:07

## ■ 텐서플로우 설치

텐서플로우는 OS 64비트에 맞게 지원해줍니다.

콘솔로 설치 방법

**<설치> ---------- 파이썬**

1. Python-3.5.1.tgz 파일을 다운로드 합니다.
2. Root 권한 계정으로 로그인합니다.
  # su -
3. Home/oracle/ 아래에 tgz파일을 내려놓은 후, 압축해제를 합니다.
  # tar xvzf Python-3.5.1.tgz
4. 압축해제된 Python 폴더에 들어가서 configure파일을 실행합니다.
  # cd Python-3.5.1
  # ./configure  #빌드준비
5. 컴파일을 합니다.
  # make  # 빌드
6. make를 통해 만들어진 설치파일(setup)을 설치합니다.
  # make altinstall  #install이 아닌 altinstall인 이유는 2.X와 병행해서 사용하기 위해 설치하는 방법.

<실행파일 바꿔주기>

기존 python이 설치되어 있는 버전은 2.X.X일 것이다.
이를 old_python으로 규정하고, 새로운 버전(3.5.1)로 접속할 수 있도록
파일경로를 재설정해준다.

1. 기존 python파일의 경로를 확인해본다.
 # ls -l /usr/bin/python*
    아마도 2.X.X 버전이 나올 것이다.
2. 방금 설치한 3.5.1 파일의 경로를 확인해본다.
 # ls -l /usr/local/bin/python-3.5.1
3. 기존에 있는 python을 old_python으로 이름을 바꿔주고,
  새로설치한 파일을 기존 python 경로로 이동시켜준다.

  # mv /usr/bin/python /usr/bin/python_old
  # cp /usr/local/bin/python3.5 /usr/bin/python

4. Python 접속이 잘 되는지 확인해본다.
# python 또는
# python -V
 >> Python 3.5.1

**<가상환경 구성하기>**

1. 먼저 우리 프로젝트명을 정하여서 다음과 같이 폴더를 home/oracle아래에 만듭니다.
 $ mkdir tensor_project
2. 생성한 폴더 안으로 이동하여, 가상환경을 만들어줍니다.
 $ cd tensor_project

$ python -m venv myvenv
>>> 만약에 위 가상환경 만드는 메소드에 오류가 발생한다면 아래와 같이 진행..
sudo apt-get install python3-venv (전 개인적으로 이 방법이 잘 안됐어요)
오류없이 정상적으로 작동이 되었다면 다음과 같은 폴더가 생성됩니다.
drwxr-xr-x 5 oracle oinstall 4096 Aug 7 13:50 myvenv << 얘가 우리의 가상환경 폴더입니다.

3. 가상환경 사용하기
activate 파일을 실행하면됩니다.
$ source myvenv/bin/activate 또는
$ . myvenv/bin/activate

정상적으로 됐다면, 콘솔의 프롬프트 앞에 (myvenv) 라는 접두어가 붙어있을 것입니다.
그렇다면 정상적으로 가상환경이 시작되었음을 알 수 있습니다.
콘솔의 프롬프트 앞에(myvenv)접두어가 붙어있다면 virtualenv가 시작되었음을 알 수 있어요.
가상환경에서 작업 할 때,python은 자동으로 올바른 버전을 참조하므로 python3 대신 python를 사용할 수 있습니다.

## 가상환경 나오려면?

(myvenv) $ deactivate

< 텐서플로우 설치 >

pip기능을 이용해서 텐서플로우를 설치하면 되는데, 그 전에 라이브러리 업데이트 관련 모듈 3개를 최신업데이트 해줘야 함

wheel, setup-tools, pip << 이 3개

pip install --upgrade --trusted-host pypi.python.org pip #pip를 설치할 건데, 인증허락해주면서 설치해줘!
pip install --upgrade --trusted-host pypi.python.org setup-tools
pip install --upgrade --trusted-host pypi.python.org wheel
위 3개가 최신화가 되었으면 텐서플로우 를 설치한다.
## Python 3.5 CPU only url :
https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.2.1-cp35-cp35m-linux_x86_64.whl
위 url주소를 다음과 같이 쓴다.

pip install --upgrade (url주소)

( 위 작업에서 오류가 날 확률이 많습니다..ㅠㅠ)

잘 설치되었다면 파이썬에 들어가서 import tensorflow as tf 해보시면 됩니다.

---

예제

```
a = tf.placeholder("float") # 공간을 만든다.
b = tf.placeholder("float")
y = tf.multiply(a,b)  #곱을 한다.
```

```
with tf.Session() as sess:
    sess.run(y, feed_dict={a:10,b:32})
```

# 텐서플로우란?

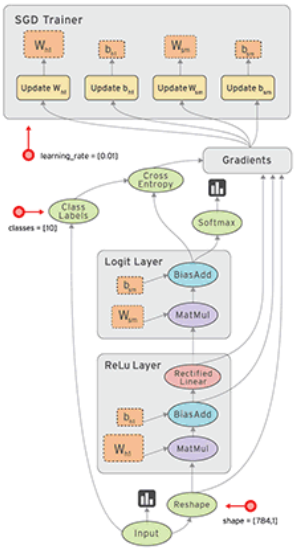
2018년 9월 10일 월요일 오전 9:59

## 1. Tensorflow 란

텐서 플로우는 기계학습과 딥러닝을 위해 구글에서 만든 오픈소스 라이브러리이다.

**■ 텐서플로우 특징**

1. **데이터 플로우 그래프를 통한 풍부한 표현력**
   - 데이터 플로우 그래프 방식을 사용하였고 그래프를 시각화 하려면 ? --> " 텐서 보드를 사용하면 된다."




2. **코드 수정없이 CPU/GPU 모드로 동작**
   2016/11/29에 TensorFlow v0.12.0 RCO가 업데이트 되었고 2016/11/29에 나온 버젼의 핵심 변경사항은 Window에서 GPU 버전의 텐써플로우를 지원한다는 것이다. 예젠에는 Ubuntu에서만 가능하던 GPU 버전도 윈도우에서 설치가 가능하게 되었다.

**■ 텐서플로우 용어 설명**

| 용어 | 설명 |
|---|---|
| **오퍼레이션 (Operation)** | - 그래프 상의 노드는 오퍼레이션(op)로 불린다.<br>- op는 하나 이상의 텐서를 받을 수 있다.<br>- op는 계산을 수행하고, 결과를 하나 이상의 텐서로 반환할 수 있다. |
| **텐서 (Tensor)** | - 내부적으로 모든 데이터는 텐서를 통해 표현된다.<br>- 텐서는 일종의 다차원 배열인데, 그래프 내의 오퍼레이션간에 텐서가 전달된다. |
| **세션 (Session)** | - 그래프를 실행하기 위해서는 세션 객체가 필요하다.<br>- 세션은 op의 실행환경을 캡슐화 한 것이다. |
| **변수 (Variable)** | - 변수는 그래프의 실행시, 파라미터를 저장하고 갱신하는데 사용된다.<br>- 메모리상에서 텐서를 저장하는 버퍼 역할을 한다. |

**■ 그래프 구조**

| 용어 | 설명 |
|---|---|
| **노드** | 노드는 수학적 계산, 데이터 입/출력, 그리고 데이터의 읽기/저장 등의 작업을 수행합니다. |
| **엣지** | 엣지는 노드들 간 데이터의 입출력 관계를 나타냅니다.<br>엣지는 동적 사이즈의 다차원 데이터 배열(=텐서)을 실어나르는데, 여기에서 텐서플로우라는 이름이 지어졌습니다. |
| **텐서** | 텐서(Tensor)는 과학과 공학 등 다양한 분야에서 이전부터 쓰이던 개념입니다. 수학에서는 임의의 기하 구조를 좌표 독립적으로 표현하기 위한 표기법으로 알려져 있지만, 분야마다 조금씩 다른 의미로 사용됩니다. 여기에서는 *학습 데이터가 저장되는 다차원 배열* 정도로 이해하시면 되겠습니다. |

■ **텐서플로우 설치 (window anaconda)**

- 아나콘다 프롬프트 창에서 pip install tensorflow

■ **Tensorflow 실행 구조**

**Session은 fetch와 feed 2가지 방법으로 처리한다.**

1. **feed --> placeholder에 값을 넣어 실행하는 방법**
   - **Session은 feed일 경우는 반드시 feed_dict로 처리값을 할당 해야 한다.**

```
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}))
```

2. **fetch --> 연산 결과를 fetch(가져오는) 방법**

■ **python / Tensorflow 문법 비교**

| Python | Tensorflow |
|---|---|
| for i in range(5):<br>  print(i+1) | import tensorflow as tf<br><br>x = tf.Variable(0, name='x')<br>model = tf.global_variables_initializer()<br><br>with tf.Session() as sess:<br>  for i in range(5):<br>  sess.run(model)<br>  x = x + 1<br>  print(sess.run(x)) |

■ **numpy / Tensorflow 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,:] |

■ 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_hot), axis=1)

■ 텐서플로우에서 가중치 초기화 할 때 주의사항

쥬피터 노트북과 스파이더는 오류가 나므로

tf.reset_default_graph() 를 맨 위에다가 적어줘야 한다.
텐서 플로우는 그래프를 메모리에 올려서 실행하게 되는데 파이참은 코드를 매번 실행할때마다 메모리는 지워주는데 스파이더나 쥬피터는 대화형 방식이라 실행할때 마다 메모리에 그래프가 누적되어서 맨위에 tf.reset_default_graph()를 적어줘야 한다.

■ 텐서플로우로 배치 정규화 구현

배치 정규화 ? --> 신경망 학습시 가중치의 값의 데이터가 골고루 분산될 수 있도록 강제하는 장치이다.

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

■ 훈련할 때 만들었던 최적의 감마와 베타를 테스트할때 적용하는 코드

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

훈련할 때 학습되면서 배치 정규화의 최적의 감마와 베타를 생성한다.

■ 언더피팅을 막기 위한 방법

1. 가중치 초기화값 건정
2. 배치 정규화

■ 오버피팅을 막기 위한 방법

1. 드롭아웃

**텐서플로우 기본 실습 (1)**

```
import tensorflow as tf

sess = tf.Session()
hello = tf.constant('Hello, Tensorflow')
print(sess.run(hello))  #바이너리 타입 (b)
print(str(sess.run(hello),encoding="utf-8"))  #유니코드 타입

**결과
b'Hello, Tensorflow'
Hello, Tensorflow

##설명
파이썬 3버전은 문자열 unicode가 기본이므로 str에서 encoding 처리를 해줘야 binary 타입을 unicode type 으로 반환한다.

위에서 변수를 정의했지만, 실행은 session 객체와 run 메소드를 사용할 때 계산이 되어 실행한다.
```

**텐서플로우 기본 실습 (2)**

```
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 방정식 x+5로 정의

model = tf.global_variables_initializer() #변수를 초기화 하겠다.(변수사용하기 위해 필수)

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

**결과
40
```

**텐서플로우 기본 실습 (3) :**
**텐서플로우는 빌딩 구조와 실행구조(Session)가 분리되어 있다**

```
import tensorflow as tf

x2 = tf.linspace(-1.0, 1.0, 10)  # -1 ~1 사이의 숫자 중 10개로 나눠서 출력
print(x2)  # Tensor("LinSpace_2:0", shape=(10,), dtype=float32)
g=tf.get_default_graph()
print([op.name for op in g.get_operations()])

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

**결과
Tensor("LinSpace:0", shape=(10,), dtype=float32)
['LinSpace/start', 'LinSpace/stop', 'LinSpace/num', 'LinSpace']
[-1.         -0.7777778  -0.5555556  -0.3333333  -0.1111111   0.11111116
  0.33333337  0.5555556   0.7777778   1.        ]
```

**텐서플로우 기본 실습 (4) :**

```
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)

**결과
Tensor("Const:0", shape=(), dtype=string)
Tensor("Add:0", shape=(), dtype=int32)
```

## 2. 예제

**문제 1.** 위에서 만든 텐서 그래프를 실행 하시오.

```
import tensorflow as tf

############ 모델 구성하는 부분
hello = tf.constant('Hello, Tensorflow!')
print(hello)

a = tf.constant(10)
b = tf.constant(32)
```

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

############모델 실행하는 부분
sess = tf.Session()
print(sess.run(hello))
print(sess.run([a,b,c]))
```

**결과

```
Tensor("Const_12:0", shape=(), dtype=string)
b'Hello, Tensorflow!'
[10, 32, 42]
```

**설명

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

| **문제 2.** | 아래의 모델(그래프)를 실행 하시오. (with 절을 사용할 것.) |
|---|---|

```
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))
```

**결과

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

| **문제 3.** | 구구단 2단을 텐써로 출력하시오. |
|---|---|

```
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:
    sess.run(model)
    for _ in range(8):
```

```
        y = y+1
        z = tf.multiply(x,y)
        print(sess.run(x), '*' ,sess.run(y),'=',sess.run(z))
```

**결과

```
2 * 2 = 4
2 * 3 = 6
...
2 * 8 = 16
2 * 9 = 18
```

| **문제 4.** | (점심시간 문제) 구구단 2단 부터 9단까지 출력 하시오. |
|---|---|

```
import tensorflow as tf

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

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

**결과

```
2 * 2 = 4
2 * 3 = 6
...
9 * 8 = 72
9 * 9 = 81
```

| **문제 5.** | zero와 숫자 1을 채워넣는 배열을 생성하는 아래의 numpy 문법을 tensor로 구현 하시오. |
|---|---|

```
# 1. numpy
import numpy as np

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

print(a)
print(b,end='\n\n')

#tensorflow
import tensorflow as tf

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

**결과

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

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

| **문제 6.** | 아래의 numpy 문법을 Tensorflow로 구현 하시오. |
|---|---|

```
#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

b = tf.argmax(a,axis=0)

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

****결과**

```
3
3
```

| **문제 7.** | 아래의 numpy 문법을 tensorflow로 바꾸시오. |
|---|---|

```
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]]])

print(np.sum(a,axis=0), end='\n\n')

d = tf.reduce_sum(a, reduction_indices=[0])

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

****결과**

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

**\#\#설명**

```
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]]])
```

같은 색깔의 숫자를 더한 값

| **문제 8.** | 아래의 numpy 문법을 tensorflow로 변경 하시오. |
|---|---|

```
import numpy as np
import tensorflow as tf

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

print(b.shape) # (12,12)


c = tf.reshape(a,(12,12))

with tf.Session() as sess:
   print(sess.run(c))
   print(c.get_shape())  # (12,12)
```

****결과**

```
(12, 12)
[[  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 단층 신경망 이해에 중요 문법)<br>아래의 numpy 배열의 열단위 sum을 출력 하시오. |
|---|---|

```
import numpy as np
import tensorflow as tf

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

****결과**

```
[3 5 7]
```

| 문제 10. | (텐서플로우로 구현한 단층 신경망 이해에 중요 문법)<br>숫자 0과 1로 채워진 2행 3열 행렬을 만들고 두 행렬의 합을 출력 하시오. (텐서플로우로 구현) |
|---|---|

```
import numpy as np
import tensorflow as tf

b1 = tf.zeros([2,3])
b2 = tf.ones([2,3],'float')

result = tf.add(b1,b2)

with tf.Session() as sess:
   print(sess.run(b1))
   print(sess.run(b2),end ='\n\n')
   print(sess.run(result))
```

**결과

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

| 문제 11. | 숫자 2로 채워진 2x3 행렬을 x라는 변수로 만들고 숫자 3으로 채워진 3x2 행렬을 y로 만든 후에 x행렬과 y행렬의 내적을 구하시오. |
|---|---|

```
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)

v1 = np.array([[2,2,2],[2,2,2]])
v2 = np.array([[3,3],[3,3],[3,3]])

with tf.Session() as sess:
   print(sess.run(result,feed_dict={x:v1,y:v2}))
```

##다른 방법

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

sess=tf.Session()
print(sess.run(tf.matmul(x,y)))
```

**결과

```
[[18. 18.]
 [18. 18.]]
```

| 문제 12. | |
|---|---|

```
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]

sess = tf.Session()
a=tf.cast(correct_prediction, "float")  # float형으로 변환
print(sess.run(a))
```

**결과

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

| 문제 13. | 위의 출력된 결과에서 전체의 갯수중에 1이 몇개나 되는지, 즉 정확도를 출력 하시오.<br>(전부 더해서 전체 수로 나눈다) |
|---|---|

```
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]

sess = tf.Session()
a = tf.cast(correct_prediction, "float")
b = tf.reduce_mean(a)
print(sess.run(b))
```

**결과

```
0.93
```

### # mnist 데이터로 단층 신경망 구현하기

| 문제 14. | 텐서 플로우에 기본적으로 내장되어 있는 mnist 데이터를 가져오시오. |
|---|---|

```
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)
```

**결과

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

| 문제 15. | 위의 mnist 데이터 중에 train 데이터의 라벨을 one hot encoding 하지 말고 숫자로 100개의 |
|---|---|

| | 라벨을 가져 오시오. |
|---|---|

```
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_ys)
```

**결과

```
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
[9 0 1 4 6 5 5 5 3 2 9 5 0 2 3 3 9 7 2 2 4 4 3 5 6 7 3 7 2 9 1 6 8 7 3 5 0
 6 2 5 2 4 9 3 9 6 2 7 7 0 9 9 9 9 9 8 8 0 4 2 3 9 8 8 7 4 8 6 1 7 2 8 4 3
 9 2 1 3 6 3 7 8 5 3 7 9 4 9 2 7 9 1 5 2 0 4 1 2 5 1]
```

| **문제 16.** | 이번에는 test 데이터와 test 데이터의 라벨 100개를 가져오는데 shape만 출력 하시오. |
|---|---|

```
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)
print(batch_yt.shape)
```

**결과

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

| **문제 17.** | 숫자 2로 채워진 행렬 2x3 행렬을 텐서플로우로 출력 하시오. |
|---|---|

```
import tensorflow as tf

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

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

**&&다른방법**

```
x = tf.placeholder('float',[2,3])

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

**결과

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

| 문제 18. | 아래의 None의 의미가 무엇인지 테스트 하시오. |
|---|---|

```
import tensorflow as tf

x = tf.placeholder('float',[None,3])
y = tf.placeholder('float',[None,3])
with tf.Session() as sess:
    print(sess.run(x, feed_dict = {x:[[2,2,2],[2,2,2]]}))
    print(sess.run(x, feed_dict = {x:[[2,2,2],[2,2,2],[2,2,2]]}))
```

**결과

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

| 문제 19. | Mnist 데이터 784(28x28) 개에 맞게 x변수를 선언 하고 배치로 입력될 데이터의 갯수는 몇개든 상관없게 None으로 변수를 만들고 Mnist 데이터를 x변수에 1개를 담고 출력해 보시오. |
|---|---|

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

mnist = input_data.read_data_sets('MNIST_data/',one_hot=False)
x = tf.placeholder('float',[None,784])
batch_xs, batch_ys = mnist.train.next_batch(100)
sess=tf.Session()
print(sess.run(x, feed_dict={x:batch_xs}).shape)
```

**결과

```
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
(100, 784)
```

| 문제 20. | 위의 코드를 수정해서 훈련 데이터 100개 뿐만 아니라 훈련데이터 라벨 100개도 출력 하시오. |
|---|---|

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

mnist = input_data.read_data_sets('MNIST_data/',one_hot=True)
x = tf.placeholder('float',[None,784])
y = tf.placeholder('float',[None,10])
batch_xs, batch_ys = mnist.train.next_batch(100)
print(batch_ys.shape)
sess=tf.Session()
print(sess.run(x, feed_dict={x:batch_xs}).shape)
print(sess.run(y, feed_dict={y:batch_ys}).shape)
```

**결과

```
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
(100, 10)
```

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

| 문제 21. | (텐서플로우로 가중치를 랜덤하게 생성하는 방법) 2x3 행렬로 -1에서 1사이의 난수를 생성하는 변수를 w로 생성하고 실행 하시오. |
|---|---|

```
import tensorflow as tf

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

sess= tf.Session()
sess.run(init)
print(sess.run(W))

**결과
[[-0.48063493   0.268353     0.6888411 ]
 [-0.8306036   -0.9485159   -0.08402276]]
```

| 문제 22. | 위에서 배치 단위로 불러오는 입력 데이터 100x784와 내적할 가중치 행렬 W를 784x50으로 생성하시오. |
|---|---|

```
import tensorflow as tf

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

sess= tf.Session()
sess.run(init)
print(sess.run(W).shape)

**결과
(784, 50)
```

| 문제 23. | 위에서 만든 입력값 (100x784)와 지금 만든 가중치 (784x50) 행렬과 내적을 한 결과를 출력 하시오. |
|---|---|

```
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) #100개씩 가져옴
batch_xt, batch_yt =mnist.test.next_batch(100) #100개씩 가져옴

x=tf.placeholder('float',[None,784])
y=tf.placeholder('float',[None,10])
with tf.Session()
W = tf.Variable(tf.random_uniform([784,50],-1,1),name='W')
init = tf.global_variables_initializer()
sess.run(init)
dot = tf.matmul(sess.run(x,feed_dict={x:batch_xs}),W)
print(sess.run(dot))

**결과
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
```

```
[[-11.031454    12.481312     3.4423923  ...   2.3007083    1.5383326
  -10.084766  ]
 [ -4.067885     1.1794797   -0.38498434 ...   2.7464216    7.253793
   -5.3992243 ]
 [ -4.1127014    0.976995    -1.6660548  ...  -7.4104605    3.1723354
    6.4654183 ]
 ...
 [-10.123704     6.9746327    0.3111267  ...  -4.637444     2.9501085
   -7.6628866 ]
 [-10.560861     4.461418     4.898665   ...   3.1114984   -0.67581
   -2.401013  ]
 [ -5.609081    10.136238     8.418104   ...   1.5537186    0.8870498
  -13.048459  ]]
```

**문제 24.** 1x50으로 bias를 생성하는데 변수를 b로 해서 생성하고 숫자를 다 1로 채우시오.

```
b1=tf.fill([1,50],1)

b2=tf.Variable(tf.ones([50]))  # 두 가지 방법

init = tf.global_variables_initializer()
with tf.Session() as sess:
   sess.run(init)
   print(sess.run(b1))
   print(sess.run(b2))

**결과
[[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 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. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1.]
```

**문제 25.** 문제 23의 결과에 편향 b를 더하시오.

```
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) #100개씩 가져옴
batch_xt, batch_yt =mnist.test.next_batch(100) #100개씩 가져옴

x=tf.placeholder('float',[None,784])
y=tf.placeholder('float',[None,10])
W = tf.Variable(tf.random_uniform([784,50],-1,1),name='W')
b=tf.Variable(tf.ones([50]))

init = tf.global_variables_initializer()
with tf.Session() as sess:
   sess.run(init)
   dot = tf.matmul(sess.run(x,feed_dict={x:batch_xs}),W)
   print(sess.run(dot+b))

**결과
[[  0.150374     4.0734644   -6.9241867  ...  -0.44436026   7.01465
    3.4225602 ]
 [ -6.8455863    0.6940498    0.39944017 ...   2.0570948    7.2407975
```

```
  -5.3480673 ]
 [  0.1631552   -3.9710321    1.7974918  ...   1.1024354    2.089592
    0.4081843 ]
 ...
 [  2.9227896    5.213005     2.8286893  ...  -4.147062     1.1163049
    1.1538167 ]
 [  8.032697     4.6191397  -11.295286   ...  -2.2122655   15.267454
   -3.4942698 ]
 [  1.2306155   -4.839216     2.8810546  ...  -2.441917     4.3707223
   -5.974695  ]]
```

| **문제 26.** | 문제 25번에서 구한 가중의 합의 결과인 y값을 시그모이드 함수를 통과 시켜 보시오. |
|---|---|

```
import tensorflow as tf
import numpy as np
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) #100개씩 가져옴
batch_xt, batch_yt =mnist.test.next_batch(100) #100개씩 가져옴

x=tf.placeholder('float',[None,784])
y=tf.placeholder('float',[None,10])
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}))
```

**결과

```
[[9.9913174e-01 9.6712309e-01 2.4800710e-01 ... 5.5985886e-01
  9.1793048e-01 1.5657598e-01]
 [1.8016336e-02 9.9998260e-01 9.6165621e-01 ... 9.8969650e-01
  2.8183427e-01 1.8325387e-01]
 [2.3745106e-05 9.1831970e-01 9.8789984e-01 ... 9.7842878e-01
  3.6996615e-01 9.3301362e-01]
 ...
 [8.0221687e-03 9.5601988e-01 2.4761687e-01 ... 7.2795379e-01
  9.7270614e-01 5.9634531e-01]
 [7.9505928e-02 9.1336262e-01 5.9107399e-01 ... 9.7013974e-01
  7.7097583e-01 7.8705567e-01]
 [9.8381585e-01 9.0439126e-02 9.8974890e-01 ... 1.6445253e-02
  5.6850899e-04 9.2558467e-01]]
```

| **문제 27.** | 위의 활성화 함수를 Relu로 변경해 보시오. |
|---|---|

```
import tensorflow as tf
import numpy as np
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) #100개씩 가져옴
```

```
batch_xt, batch_yt =mnist.test.next_batch(100) #100개씩 가져옴

x=tf.placeholder('float',[None,784])
y=tf.placeholder('float',[None,10])
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}))
```

**결과

```
[[ 0.          0.          0.          ... 13.923153    7.07159
   1.5169069 ]
 [ 4.2751684   0.          0.          ...  1.3506203   9.296968
   0.        ]
 [ 0.          0.          0.          ...  0.          2.6707008
   0.        ]
 ...
 [ 2.3094802   0.          0.          ...  1.9258523   6.3225584
   0.        ]
 [ 0.          0.          0.          ... 11.384467    0.
   0.        ]
 [ 2.6642718   2.3731618   0.70178914 ...  1.7504079   8.550774
   0.        ]]
```

| **문제 28.** | 위에서 출력한 가중의 y_hat의 결과를 softmax 함수를 통과 시킨 결과가 어떻게 되는지 출력하시오. |
|---|---|

```
import tensorflow as tf
import numpy as np
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) #100개씩 가져옴
batch_xt, batch_yt =mnist.test.next_batch(100) #100개씩 가져옴

x=tf.placeholder('float',[None,784])
y=tf.placeholder('float',[None,10])
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}))
```

**결과

```
[[4.46418397e-07 1.41125638e-03 9.26414723e-05 1.18857209e-07
  3.72706051e-03 2.60583212e-04 9.94322777e-01 6.03546880e-07
  1.83168784e-04 1.41395878e-06]
....
 [1.35094323e-03 1.61422031e-06 6.92922482e-03 2.68043572e-04
  1.44986856e-10 3.74985695e-01 6.17917567e-06 6.11933410e-01
  4.52224584e-03 2.73552541e-06]]
```

(100, 10)

| 문제 29. | 위에서 출력한 가중의 y_hat의 결과를 softmax 함수를 통과 시킨 결과가 어떻게 되는지 출력하시오. |
|---|---|

```
import tensorflow as tf
import numpy as np
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) #100개씩 가져옴
batch_xt, batch_yt =mnist.test.next_batch(100) #100개씩 가져옴

x=tf.placeholder('float',[None,784])
y=tf.placeholder('float',[None,10])
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)
arg_mx = tf.argmax(y_hat,axis=1)

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

**결과

```
[3 0 5 3 3 3 0 5 5 6 3 0 3 6 3 5 3 6 2 3 5 5 0 2 6 5 3 6 5 0 3 2 6 3 6 5 6
 5 5 3 2 5 3 3 5 0 5 3 2 3 2 6 3 8 5 2 5 6 5 5 5 3 3 3 3 3 3 5 6 2 4 3 5 6
 6 3 3 5 0 2 3 6 0 0 0 5 5 3 6 3 8 5 5 5 5 5 5 4 6 3]
```

| 문제 30. | 위의 코드에 라벨을 가져오는 코드를 추가해서 정확도를 출력 하시오. |
|---|---|

```
import tensorflow as tf
import numpy as np
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) #100개씩 가져옴
batch_xt, batch_yt =mnist.test.next_batch(100) #100개씩 가져옴
x=tf.placeholder('float',[None,784])
#y=tf.placeholder('float',[None,10])
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)
x_predict = tf.argmax(y_hat,axis=1)

correct_prediction = tf.equal(x_predict,batch_ys)
a = tf.cast(correct_prediction, "float")
acc = tf.reduce_mean(a)

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

**결과

```
0.07
```

| 문제 31. | 교차 엔트로피 함수를 문제 30번 코드에 추가하고 loss를 출력 하시오. |
|---|---|

```
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"))

init = tf.global_variables_initializer()

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}))

##############################################
# 그외 옵티마이저
# optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
#러닝 레이트가 학습되면서 자동 조절되는 경사하강법
# optimizer = tf.train.AdagradOptimizer(learning_rate=0.01)

# optimizer = tf.train.MomentumOptimizer(learning_rate=0.01)
#관성을 이용해서 local minima에 안빠지게 하는 경사 감소법
# optimizer = tf.train.AdamOptimizer(learning_rate=0.01)
#Adagrade의 장점 + Momentum의 장점
##############################################
```

**결과

```
[1 2 2 7 8 7 1 2 7 7 1 1 7 9 2 2 0 2 9 7 1 9 2 8 0 0 1 7 1 1 1 8 2 3 1 3 2
 9 1 7 7 1 7 3 7 3 3 2 8 0 7 8 0 3 1 7 9 8 7 7 1 3 7 1 3 7 1 3 7 2 8 1 3 5
 2 2 2 0 1 3 7 2 1 7 2 7 7 9 3 9 0 7 3 3 3 9 1 3 3 7]
[1 0 8 6 4 8 2 2 4 4 1 1 7 3 8 3 6 8 9 1 8 0 6 3 4 3 2 4 2 5 2 4 5 7 3 8 0
 0 1 1 7 8 0 3 6 7 0 4 4 4 0 9 6 5 3 0 9 3 0 8 1 5 6 8 7 0 1 7 7 0 1 6 7 1
 5 8 8 6 2 0 6 4 5 9 8 0 6 2 3 3 6 5 7 7 2 3 6 0 4 4]
0.14
```

| 문제 32. | 문제 31번 코드에 SGD 경사하강법 코드를 사용해서 구현하시오. |
|---|---|

```
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)

#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()

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}))
```

**결과

```
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
0.28
```

| 문제 33. | 정확도를 출력 하시오. |
|---|---|

```
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)

#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()

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}))
```

**결과

0.28

| **문제 34.** | 위의 코드 중 아래의 3개 코드를 for loop문을 이용해서 반복수행해서 1에폭 돌게 하여라 |
|---|---|

```
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)

#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()

sess = tf.Session()
sess.run(init)


# print (sess.run(y_predict, feed_dict={x:batch_xs}))
# print (sess.run(y_label, feed_dict={y_onehot:batch_ys}))
for i in range(1,601):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train,feed_dict={x: batch_xs,y_onehot:batch_ys})
    sess.run(accuracy, feed_dict={x:batch_xs,y_onehot:batch_ys})

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

**결과

0.9

| 문제 35. | 위의 코드를 수정해서 아래와 같이 결과가 출력되게 하시오.<br>1 에폭 정확도 : 0.9<br>2 에폭 정확도 : 0.92<br>3 에폭 정확도 : 0.94<br>.. |
|---|---|

```
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)

#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()

sess = tf.Session()
sess.run(init)



# print (sess.run(y_predict, feed_dict={x:batch_xs}))
```

```
# print (sess.run(y_label, feed_dict={y_onehot:batch_ys}))
for i in range(1,10000):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train,feed_dict={x: batch_xs,y_onehot:batch_ys})
    sess.run(accuracy, 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}))
```

**결과

1.0 에폭 정확도 0.93
2.0 에폭 정확도 0.92
3.0 에폭 정확도 0.95
4.0 에폭 정확도 0.94
5.0 에폭 정확도 0.9
6.0 에폭 정확도 0.93
7.0 에폭 정확도 0.94
8.0 에폭 정확도 0.9
9.0 에폭 정확도 0.93
10.0 에폭 정확도 0.95
11.0 에폭 정확도 0.95
12.0 에폭 정확도 0.97
13.0 에폭 정확도 0.94
14.0 에폭 정확도 0.95
15.0 에폭 정확도 0.95
16.0 에폭 정확도 0.98

| 문제 36. | 러닝 레이트를 0.05로 학습시키고 정확도를 확인 하시오. |
|---|---|

```
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)

#SGD 경사 감소법 구현
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.005)

#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()

sess = tf.Session()
sess.run(init)

# print (sess.run(y_predict, feed_dict={x:batch_xs}))
# print (sess.run(y_label, feed_dict={y_onehot:batch_ys}))
for i in range(1,10000):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train,feed_dict={x: batch_xs,y_onehot:batch_ys})
    sess.run(accuracy, 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}))
```

**결과

1.0 에폭 정확도 0.84
2.0 에폭 정확도 0.89
3.0 에폭 정확도 0.92
4.0 에폭 정확도 0.91
5.0 에폭 정확도 0.88
6.0 에폭 정확도 0.95
7.0 에폭 정확도 0.95
8.0 에폭 정확도 0.93
9.0 에폭 정확도 0.95
10.0 에폭 정확도 0.98
11.0 에폭 정확도 0.96
12.0 에폭 정확도 0.95
13.0 에폭 정확도 0.9
14.0 에폭 정확도 0.98
15.0 에폭 정확도 0.91
16.0 에폭 정확도 0.9

| 문제 37. | 경사 감소법을 adam으로 변경해서 학습 시키고 정확도를 확인 하시오. |
|---|---|

```
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)

#SGD 경사 감소법 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"))

init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)

# print (sess.run(y_predict, feed_dict={x:batch_xs}))
# print (sess.run(y_label, feed_dict={y_onehot:batch_ys}))
for i in range(1,10000):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train,feed_dict={x: batch_xs,y_onehot:batch_ys})
    sess.run(accuracy, 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}))

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

**결과

1.0 에폭 정확도 0.89
2.0 에폭 정확도 0.91
3.0 에폭 정확도 0.89
4.0 에폭 정확도 0.91
5.0 에폭 정확도 0.12
6.0 에폭 정확도 0.04
7.0 에폭 정확도 0.06
8.0 에폭 정확도 0.11
9.0 에폭 정확도 0.09
10.0 에폭 정확도 0.13
11.0 에폭 정확도 0.11
12.0 에폭 정확도 0.1
13.0 에폭 정확도 0.1
14.0 에폭 정확도 0.08
15.0 에폭 정확도 0.14
16.0 에폭 정확도 0.14

| 문제 37. | 경사 감소법을 adam으로 변경해서 학습 시키고 정확도를 확인 하시오. |
|---|---|

```
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)

#SGD 경사 감소법 구현
optimizer = tf.train.AdamOptimizer(learning_rate=0.01) # 학습률이 높아서 발산해버림

#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()

sess = tf.Session()
sess.run(init)

# print (sess.run(y_predict, feed_dict={x:batch_xs}))
# print (sess.run(y_label, feed_dict={y_onehot:batch_ys}))
for i in range(1,10000):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train,feed_dict={x: batch_xs,y_onehot:batch_ys})
    sess.run(accuracy, 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}))

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

**결과

1.0 에폭 정확도 0.89
2.0 에폭 정확도 0.91
3.0 에폭 정확도 0.89
4.0 에폭 정확도 0.91
5.0 에폭 정확도 0.12
6.0 에폭 정확도 0.04
7.0 에폭 정확도 0.06
8.0 에폭 정확도 0.11
9.0 에폭 정확도 0.09
10.0 에폭 정확도 0.13
11.0 에폭 정확도 0.11
12.0 에폭 정확도 0.1
13.0 에폭 정확도 0.1
14.0 에폭 정확도 0.08
15.0 에폭 정확도 0.14
16.0 에폭 정확도 0.14

| 문제 38. | 문제 37번 코드에 가중치 초기화 he를 사용해서 정확도를 확인하시오. |
|---|---|

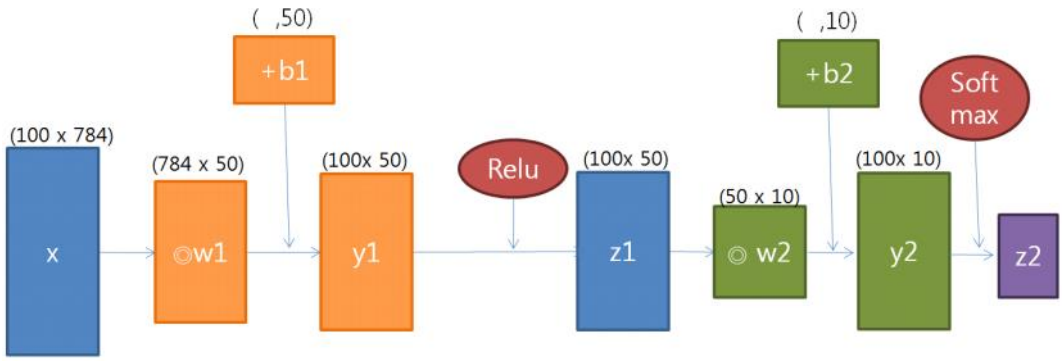
```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/",one_hot=True)

tf.reset_default_graph()

# 계층 생성해서 예측값 출력하는 코드
x = tf.placeholder("float",[None,784])
#W = tf.Variable(tf.random_uniform([784,10],-1,1), name="W")
W = tf.get_variable(name='W', shape=[784, 10], initializer=tf.contrib.layers.variance_scaling_initializer())
# he 초기값
```

```
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)

#SGD 경사 감소법 구현
optimizer = tf.train.AdamOptimizer(learning_rate=0.001) # 학습률이 높아서 발산해버림

#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()

sess = tf.Session()
sess.run(init)

# print (sess.run(y_predict, feed_dict={x:batch_xs}))
# print (sess.run(y_label, feed_dict={y_onehot:batch_ys}))
for i in range(1,10000):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train,feed_dict={x: batch_xs,y_onehot:batch_ys})
    sess.run(accuracy, 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}))
```

**결과

1.0 에폭 정확도 0.94
2.0 에폭 정확도 0.93
3.0 에폭 정확도 0.93
4.0 에폭 정확도 0.89
5.0 에폭 정확도 0.9
6.0 에폭 정확도 0.94
7.0 에폭 정확도 0.95
8.0 에폭 정확도 0.93
9.0 에폭 정확도 0.97
10.0 에폭 정확도 0.95
11.0 에폭 정확도 0.87
12.0 에폭 정확도 0.93
13.0 에폭 정확도 0.91
14.0 에폭 정확도 0.93
15.0 에폭 정확도 0.92
16.0 에폭 정확도 0.93

| 문제 39. | 위의 단층 코드를 다층(2층)으로 변경 하시오. |
|---|---|



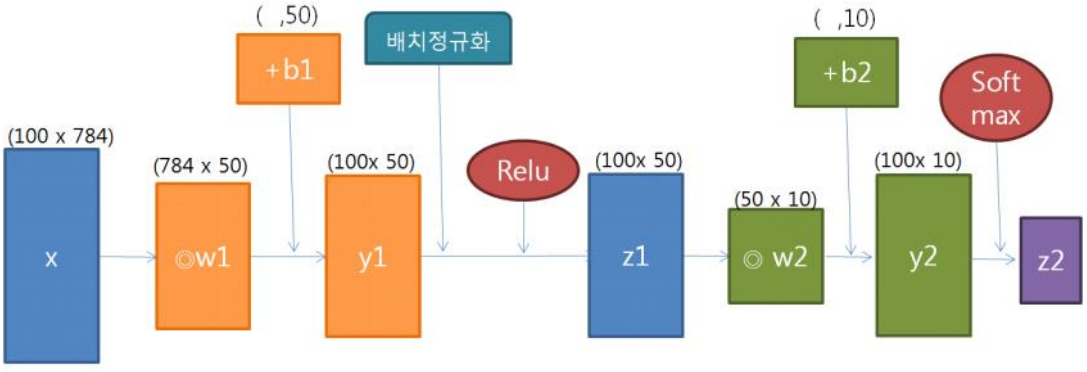

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/",one_hot=True)

tf.reset_default_graph()

# 계층 생성해서 예측값 출력하는 코드

###은닉1층
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
Z1 = tf.nn.relu(y1)


###은닉2층
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(Z1,W2) + b2
Z2 = tf.nn.softmax(y2)

y_predict = tf.argmax(Z2, 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(Z2), axis=1)

#SGD 경사 감소법 구현
optimizer = tf.train.AdamOptimizer(learning_rate=0.001) # 학습률이 높아서 발산해버림

#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()

sess = tf.Session()
sess.run(init)

# print (sess.run(y_predict, feed_dict={x:batch_xs}))
```

```
# print (sess.run(y_label, feed_dict={y_onehot:batch_ys}))
for i in range(1,10000):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train,feed_dict={x: batch_xs,y_onehot:batch_ys})
    sess.run(accuracy, 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}))
```

**결과

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

| 문제 40. | 문제 39번 다층 신경망 코드에 배치 정규화 코드를 추가 하시오. |
|---|---|




```
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 생성해서 예측값 출력하는 코드

###은닉1층
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)     #배치 정규화
```

```
Z1 = tf.nn.relu(batch_z1)

###은닉2층
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(Z1,W2) + b2
Z2 = tf.nn.softmax(y2)

y_predict = tf.argmax(Z2, 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(Z2), axis=1)

#SGD 경사 감소법 구현
optimizer = tf.train.AdamOptimizer(learning_rate=0.001)  # 학습률이 높아서 발산해버림

#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()

sess = tf.Session()
sess.run(init)

# print (sess.run(y_predict, feed_dict={x:batch_xs}))
# print (sess.run(y_label, feed_dict={y_onehot:batch_ys}))
for i in range(1,10000):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train,feed_dict={x: batch_xs,y_onehot:batch_ys})
    sess.run(accuracy, 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}))
```

**결과

1.0 에폭 정확도 0.99
2.0 에폭 정확도 0.95
3.0 에폭 정확도 0.97
4.0 에폭 정확도 0.95
5.0 에폭 정확도 0.98
6.0 에폭 정확도 1.0
7.0 에폭 정확도 0.97
8.0 에폭 정확도 0.98
9.0 에폭 정확도 0.99
10.0 에폭 정확도 0.98
11.0 에폭 정확도 0.98
12.0 에폭 정확도 0.98
13.0 에폭 정확도 0.98
14.0 에폭 정확도 1.0

15.0 에폭 정확도 0.99
16.0 에폭 정확도 0.99

#훈련하는 신경망에 테스트를 하는 코드를 추가

| 문제 41. | 지금 현재까지의 코드에는 신경망을 훈련만 시키는 코드였는데 테스트까지 진행해서 오버피팅이 발생했는지 확인할 수 있도록 에폭마다 훈련 데이터의 정확도와 테스트 데이터의 정확도를 같이 출력 하시오. |
|---|---|

```
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 생성해서 예측값 출력하는 코드

###은닉1층
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)     #배치 정규화
Z1 = tf.nn.relu(batch_z1)

###은닉2층
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(Z1,W2) + b2
Z2 = tf.nn.softmax(y2)

y_predict = tf.argmax(Z2, 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(Z2), axis=1)

#SGD 경사 감소법 구현
optimizer = tf.train.AdamOptimizer(learning_rate=0.001)  # 학습률이 높아서 발산해버림

#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()

sess = tf.Session()
sess.run(init)

for i in range(0,10000):
    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(accuracy, feed_dict={x :train_xs, y_onehot: train_ys}))
        print(i / 600 + 1 ,'ecpo test acc:', sess.run(accuracy, feed_dict={x :test_xs, y_onehot: test_ys}))
        print("============================================================")
```

**결과

```
1.0 ecpo train acc: 0.21
1.0 ecpo test acc: 0.18
========================================================================
2.0 ecpo train acc: 0.89
2.0 ecpo test acc: 0.91
========================================================================
....
16.0 ecpo train acc: 1.0
16.0 ecpo test acc: 0.99
========================================================================
17.0 ecpo train acc: 1.0
17.0 ecpo test acc: 0.93
========================================================================
```

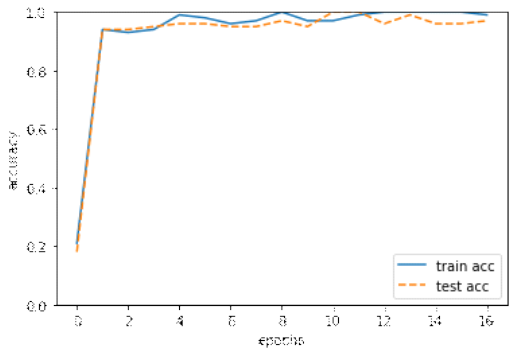
| 문제 42. | train, test의 결과를 그래프로 출력하시오. |
|---|---|

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

tf.reset_default_graph()

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

# 계층 생성
x = tf.placeholder("float" ,[None, 784])
W1 = tf.get_variable(name='W1', shape=[784, 50], initializer=tf.contrib.layers.variance_scaling_initializer())
b1 = tf.Variable(tf.ones([50]))
W2 = tf.get_variable(name='W2', shape=[50, 10], initializer=tf.contrib.layers.variance_scaling_initializer())
b2 = tf.Variable(tf.ones([10]))

z1 = tf.matmul(x, W1) + b1
batch_z1 = tf.contrib.layers.batch_norm(z1, True)
r1 = tf.nn.relu(batch_z1)
z2 = tf.matmul(r1, W2) + b2

y_hat = tf.nn.softmax(z2)
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_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:
         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})

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

         print("================================================================================")

# 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()
```

**결과

```
1.0 ecpo train acc: 0.21
1.0 ecpo test acc: 0.18
================================================================================
2.0 ecpo train acc: 0.89
2.0 ecpo test acc: 0.91
================================================================================
....
16.0 ecpo train acc: 1.0
16.0 ecpo test acc: 0.99
================================================================================
17.0 ecpo train acc: 1.0
17.0 ecpo test acc: 0.93
================================================================================
```




#드롭아웃

| **문제 43.** | (점심시간 문제) 위의 코드가 오버피팅이 발생하지 않도록 dropout을 적용 하시오. |
|---|---|

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

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

tf.reset_default_graph()

# 계층 생성해서 예측값 출력하는 코드

###은닉1층
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)    #배치 정규화
Z1 = tf.nn.relu(batch_z1)


###은닉2층
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(Z1,W2) + b2

#드롭아웃
keep_prob=tf.placeholder('float')
dropout = tf.nn.dropout(y2, keep_prob)

Z2 = tf.nn.softmax(dropout)

y_predict = tf.argmax(Z2, 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(Z2), axis=1)

#SGD 경사 감소법 구현
optimizer = tf.train.AdamOptimizer(learning_rate=0.001) # 학습률이 높아서 발산해버림
```

```
#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()

sess = tf.Session()
sess.run(init)

train_acc_list=[]
test_acc_list =[]

for i in range(0,10000):
    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})

    if i % 600 == 0:
        train_acc = sess.run(accuracy, feed_dict={x: train_xs, y_onehot: train_ys, keep_prob:0.5})
        test_acc = sess.run(accuracy, feed_dict={x: test_xs, y_onehot: test_ys, keep_prob:1.0})
                                                                    # 테스트할땐 드롭아웃 안한다 (=1)
        print(i / 600 + 1, 'ecpo train acc:', train_acc, 'ecpo 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

        print("============================================================")
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()
```

**결과

1.0 ecpo train acc: 0.08 ecpo test acc: 0.1

==========

2.0 ecpo train acc: 0.52 ecpo test acc: 0.95

==========

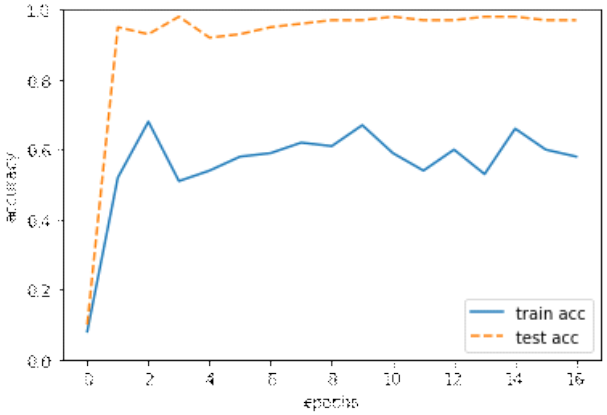
....

16.0 ecpo train acc: 0.6 ecpo test acc: 0.97

==========

17.0 ecpo train acc: 0.58 ecpo test acc: 0.97

==========



#배치정규화

| 문제 44. | 배치정규화도 훈련시에는 배치정규화를 켜고 테스트 시에는 배치정규화를 끄게 코드를 구현하시오. ( 훈련시에 만들었던 최적의 감마와 베타값을 사용하기 위해서) |
|---|---|

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

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

tf.reset_default_graph()

# 계층 생성해서 예측값 출력하는 코드

###은닉1층
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

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

#batch_z1 = tf.contrib.layers.batch_norm(y1,True)     #배치 정규화
batch_z1 = tf.contrib.layers.batch_norm(y1, is_training=isTrain) # 배치 정규화
Z1 = tf.nn.relu(batch_z1)


###은닉2층
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(Z1,W2) + b2


#드롭아웃
keep_prob=tf.placeholder('float')
dropout = tf.nn.dropout(y2, keep_prob)

Z2 = tf.nn.softmax(dropout)

y_predict = tf.argmax(Z2, 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(Z2), axis=1)

#SGD 경사 감소법 구현
optimizer = tf.train.AdamOptimizer(learning_rate=0.001) # 학습률이 높아서 발산해버림

#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()

sess = tf.Session()
sess.run(init)

train_acc_list=[]
test_acc_list =[]

for i in range(0,10000):
    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:
        train_acc = sess.run(accuracy, feed_dict={x: train_xs, y_onehot: train_ys, keep_prob:0.5, isTrain:False})
        test_acc = sess.run(accuracy, feed_dict={x: test_xs, y_onehot: test_ys, keep_prob:1.0, isTrain:False})
        print(i / 600 + 1, 'ecpo train acc:', train_acc, 'ecpo 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

        print("============================================================")

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()
```

**결과

1.0 ecpo train acc: 0.1 ecpo test acc: 0.16

==========

2.0 ecpo train acc: 0.21 ecpo test acc: 0.74

==========

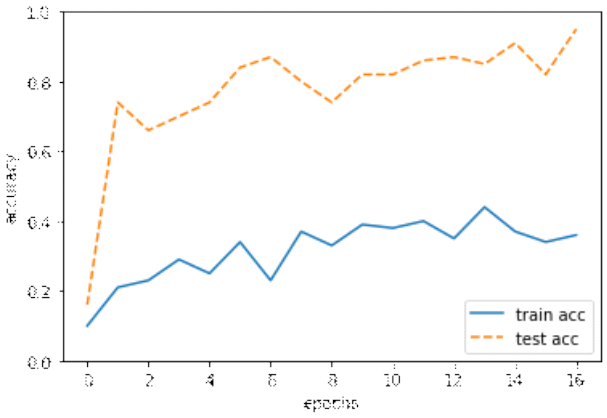
....

16.0 ecpo train acc: 0.34 ecpo test acc: 0.82

==========

17.0 ecpo train acc: 0.36 ecpo test acc: 0.95

==========




| 문제 45. | 훈련 시에 배치 정규화로 만들어진 최적의 베타와 감마를 계속 잘 유지시킬 수 있도록 위의 코드에 아래 코드를 추가하시오.<br><br>**update_ops = tf.get_collection(tf.GraphKeys.UPDATE_OPS)** |
|---|---|

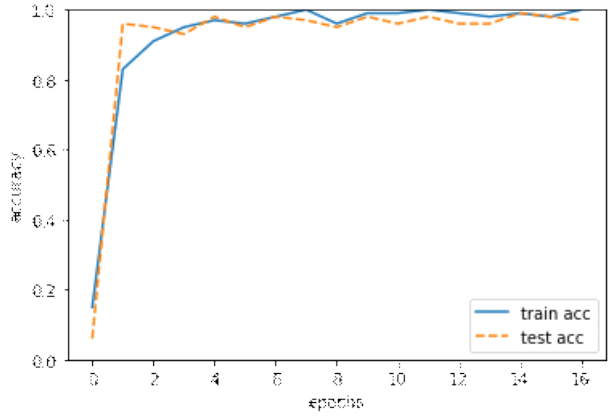
```
    with tf.control_dependencies(update_ops):
        # Ensures that we execute the update_ops before performing the train_step
        train = tf.train.AdamOptimizer(learning_rate=0.001).minimize(loss)
```

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

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

tf.reset_default_graph()

# 계층 생성해서 예측값 출력하는 코드

###은닉1층
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

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

#batch_z1 = tf.contrib.layers.batch_norm(y1,True)    #배치 정규화
batch_z1 = tf.contrib.layers.batch_norm(y1, is_training=isTrain) # 배치 정규화
Z1 = tf.nn.relu(batch_z1)


###은닉2층
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(Z1,W2) + b2


#드롭아웃
keep_prob=tf.placeholder('float')
dropout = tf.nn.dropout(y2, keep_prob)

Z2 = tf.nn.softmax(dropout)

y_predict = tf.argmax(Z2, 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(Z2), axis=1)

#SGD 경사 감소법 구현
#optimizer = tf.train.AdamOptimizer(learning_rate=0.001) # 학습률이 높아서 발산해버림
#학습 오퍼레이션 정의
#train = optimizer.minimize(loss)

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)

# 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()

sess = tf.Session()
sess.run(init)

train_acc_list=[]
test_acc_list =[]

for i in range(0,10000):
    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:
        train_acc = sess.run(accuracy, feed_dict={x: train_xs, y_onehot: train_ys, keep_prob:1.0, isTrain:False})
        test_acc = sess.run(accuracy, feed_dict={x: test_xs, y_onehot: test_ys, keep_prob:1.0, isTrain:False})
        print(i / 600 + 1, 'ecpo train acc:', train_acc, 'ecpo 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

        print("============================================================")
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()
```

**결과

```
1.0 ecpo train acc: 0.1 ecpo test acc: 0.07
================================================================================
2.0 ecpo train acc: 0.87 ecpo test acc: 0.89
================================================================================
....
16.0 ecpo train acc: 0.98 ecpo test acc: 0.96
================================================================================
17.0 ecpo train acc: 0.99 ecpo test acc: 0.97
================================================================================
```




| 문제 46. | 2신경망을 3층 신경망으로 변경하고 정확도를 확인 하시오.<br>입력측 (784) --> 은닉1층 (100) --> 은닉2층(100) --> 출력층(10) |
|---|---|

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

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

tf.reset_default_graph()

# 계층 생성해서 예측값 출력하는 코드

###은닉1층
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

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

#batch_z1 = tf.contrib.layers.batch_norm(y1,True)    #배치 정규화
batch_z1 = tf.contrib.layers.batch_norm(y1, is_training=isTrain) # 배치 정규화
Z1 = tf.nn.relu(batch_z1)


###은닉2층
W2 = tf.get_variable(name='W2', shape=[100, 50], initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b2 = tf.Variable(tf.ones([50]))
y2 = tf.matmul(Z1,W2) + b2

batch_y2 = tf.contrib.layers.batch_norm(y2, is_training=isTrain) # 배치 정규화
Z2 = tf.nn.relu(batch_y2)

###은닉3층
W3 = tf.get_variable(name='W3', shape=[50, 10], initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b3 = tf.Variable(tf.ones([10]))
y3 = tf.matmul(Z2,W3) + b3

#드롭아웃
keep_prob=tf.placeholder('float')
dropout = tf.nn.dropout(y3, keep_prob)

Z3 = tf.nn.softmax(dropout)

y_predict = tf.argmax(Z3, 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(Z3), axis=1)

#SGD 경사 감소법 구현
#optimizer = tf.train.AdamOptimizer(learning_rate=0.001)  # 학습률이 높아서 발산해버림

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)

#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()

sess = tf.Session()
sess.run(init)

train_acc_list=[]
test_acc_list =[]

for i in range(0,10000):
    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:
        train_acc = sess.run(accuracy, feed_dict={x: train_xs, y_onehot: train_ys, keep_prob:1.0, isTrain:False})
        test_acc = sess.run(accuracy, feed_dict={x: test_xs, y_onehot: test_ys, keep_prob:1.0, isTrain:False})
        print(i / 600 + 1, 'ecpo train acc:', train_acc, 'ecpo 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

        print("============================================================")
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()
```

**결과

1.0 ecpo train acc: 0.14 ecpo test acc: 0.13

==========

2.0 ecpo train acc: 0.78 ecpo test acc: 0.87

==========

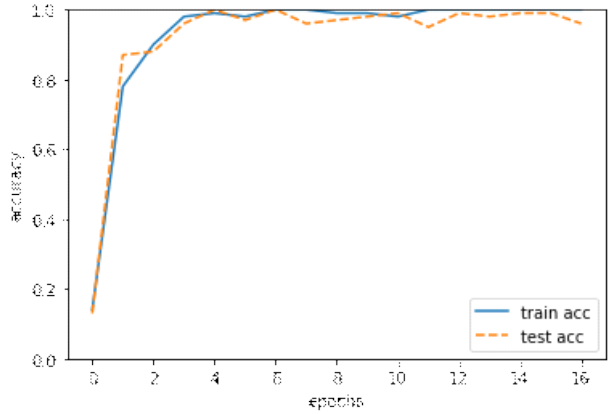
....

16.0 ecpo train acc: 1.0 ecpo test acc: 0.99

==========

17.0 ecpo train acc: 1.0 ecpo test acc: 0.96

==========



| 문제 47. | 문제 46번 코드를 cnn 코드로 구현 하시오.<br>입력층 --> 은닉1층(conv-pool) --> 은닉2층(conv-pool) --> 은닉3층 ----> 출력층 |
|---|---|

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

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

tf.reset_default_graph()

# 계층 생성해서 예측값 출력하는 코드

###은닉1층
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

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

#batch_z1 = tf.contrib.layers.batch_norm(y1,True)    #배치 정규화
batch_z1 = tf.contrib.layers.batch_norm(y1, is_training=isTrain) # 배치 정규화
Z1 = tf.nn.relu(batch_z1)


###은닉2층
W2 = tf.get_variable(name='W2', shape=[100, 50], initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b2 = tf.Variable(tf.ones([50]))
y2 = tf.matmul(Z1,W2) + b2

batch_y2 = tf.contrib.layers.batch_norm(y2, is_training=isTrain) # 배치 정규화
Z2 = tf.nn.relu(batch_y2)

###은닉3층
W3 = tf.get_variable(name='W3', shape=[50, 10], initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b3 = tf.Variable(tf.ones([10]))
y3 = tf.matmul(Z2,W3) + b3

#드롭아웃
keep_prob=tf.placeholder('float')
dropout = tf.nn.dropout(y3, keep_prob)

Z3 = tf.nn.softmax(dropout)

y_predict = tf.argmax(Z3, 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(Z3), axis=1)
```

```
#SGD 경사 감소법 구현
#optimizer = tf.train.AdamOptimizer(learning_rate=0.001)  # 학습률이 높아서 발산해버림

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)

#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()

sess = tf.Session()
sess.run(init)

train_acc_list=[]
test_acc_list =[]

for i in range(0,10000):
    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:
        train_acc = sess.run(accuracy, feed_dict={x: train_xs, y_onehot: train_ys, keep_prob:1.0, isTrain:False})
        test_acc = sess.run(accuracy, feed_dict={x: test_xs, y_onehot: test_ys, keep_prob:1.0, isTrain:False})
        print(i / 600 + 1, 'ecpo train acc:', train_acc, 'ecpo 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

        print("============================================================")
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()
```

**결과

1.0 ecpo train acc: 0.14 ecpo test acc: 0.13

==========

2.0 ecpo train acc: 0.78 ecpo test acc: 0.87

==========

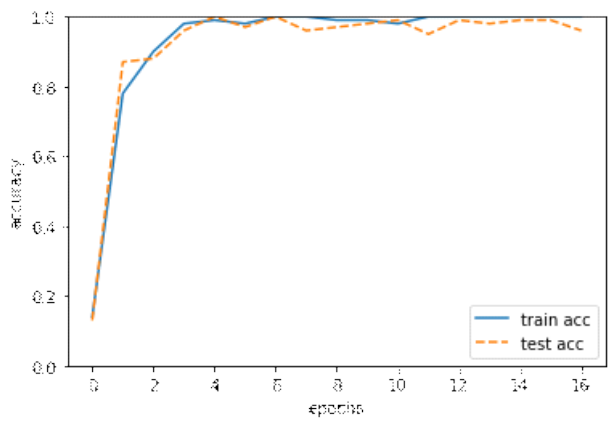
....

16.0 ecpo train acc: 1.0 ecpo test acc: 0.99

==========

17.0 ecpo train acc: 1.0 ecpo test acc: 0.96

==========




| 문제 47++ | cnn 적용 코드 |
|---|---|

```
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()

x = tf.placeholder(tf.float32, [None, 28, 28, 1])
y = tf.placeholder(tf.float32, [None, 10])
keep_prob = tf.placeholder(tf.float32)

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')
L1 = tf.nn.relu(L1)
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))
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()
```

****결과**

1.0 ecpo train acc: 0.14 ecpo test acc: 0.13

==========

2.0 ecpo train acc: 0.78 ecpo test acc: 0.87

==========

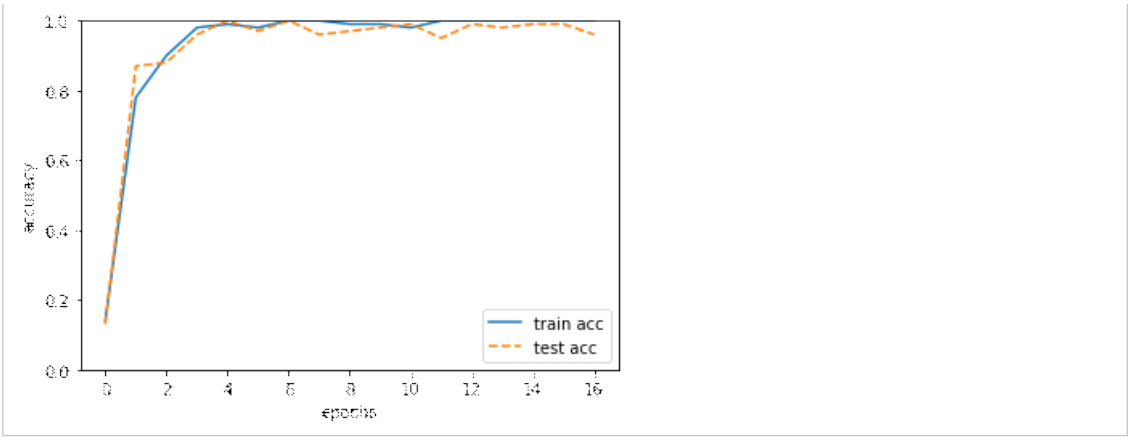
....

16.0 ecpo train acc: 1.0 ecpo test acc: 0.99

==========

17.0 ecpo train acc: 1.0 ecpo test acc: 0.96

==========


1.0
0.8
0.6
0.4
0.2
0.0
accuracy
0
2
4
6
8
10
12
14
16
epochs
train acc
test acc

# cifar10 실습

2018년 9월 14일 금요일 오전 10:25

**■ cifar10 데이터를 신경망에 로드하는 데이터 전처리 코드 작성**

- cifar10 소개

  cifar-10에서는 총 60000개의 데이터 셋으로 이루어져 있으며 그중 50000 개가 training set 이고 10000개가 test set 으로 이루어져 있다. class는 airplain 부터 truck 까지 10개로 구성되어 있다.

  https://www.cs.toronto.edu/~kriz/cifar.html

class 종류

| | |
|---|---|
| 1. 비행기 | 6. 개 |
| 2. 자동차 | 7. 개구리 |
| 3. 새 | 8. 말 |
| 4. 고양이 | 9. 양 |
| 5. 사슴 | 10. 트럭 |

**■ 이미지 데이터 신경망에 로드하기 위해 반드시 알아야 하는 내용**

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

| **문제 48.** | test100이란 폴더를 만들고 test 이미지 100개를 이 폴더에 복사하고 복사한 이미지를 아래와 같이 불러오는 함수를 생성 하시오. (c:₩₩data₩₩test100) |
|---|---|

```
import  os

test_image='c:₩₩data₩₩test100'

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

print ( image_load(test_image) )
```

**결과

['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.

```
import os
import re

test_image='c:₩₩data₩₩test100'

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

print (image_load(test_image))
```

****결과**

['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']

**문제 50.** 아래의 결과를 정렬해서 출력되게 하시오.

```
import os
import re

test_image='c:₩₩data₩₩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))
```

****결과**

[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]

| **문제 51.** | 문제 50번에 나온 결과에 png를 붙여서 아래와 같이 결과가 출력되게 하시오. |
|---|---|

```
import os
import re

test_image='c:₩₩data₩₩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()

    for j,v in enumerate(file_name):
        file_name[j] = str(v) +'.png'
    return  file_name

print (image_load(test_image))
```

****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']

| **문제 52.** | 이미지 이름 앞에 절대경로가 아래처럼 붙게 하시오. |
|---|---|

```
import os
import re

test_image='c:₩₩data₩₩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()

    for j,v in enumerate(file_name):
        file_name[j] = test_image + str(v) +'.png'

    return  file_name
```

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

**결과

['c:₩₩data₩₩test100₩₩1.png', 'c:₩₩data₩₩test100₩₩2.png', 'c:₩₩data₩₩test100₩₩3.png', 'c:₩₩data₩₩test100₩₩4.png', 'c:₩₩data₩₩test100₩₩5.png', 'c:₩₩data₩₩test100₩₩6.png', 'c:₩₩data₩₩test100₩₩7.png', ... ]

| **문제 53.** | 위의 이미지들을 cv2.imread 함수를 이용해서 숫자 리스트로 변환 하시오. |
|---|---|

```
import os
import re
import cv2

test_image='c:₩₩data₩₩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()

    for j,v in enumerate(file_name):
        file_name[j] = test_image + str(v) +'.png'

    return  file_name

img_lst=[]
for i in image_load(test_image):
    img_lst.append(cv2.imread(i))

print(img_lst)
```

**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],
         ...,
         [ 31,  95, 136],
         [ 32,  91, 125],
         [ 34,  88, 119]],

          .....
```

| **문제 54.** | 위의 이미지들을 cv2.imread 함수를 이용해서 숫자 리스트로 변환 하시오. |
|---|---|

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

test_image ='c:₩₩data₩₩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()
    data  = []
    for i in file_name:
        file = path + str(i) + '.png'
        data.append(file)
    image = []
    for j in data:
        img = cv2.imread(j)
        image.append(img)
    return np.array(image)

print (image_load(test_image))
```

**결과

```
[[[[ 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]
   ...
   [ 31  95 136]
   [ 32  91 125]
   [ 34  88 119]]

  .....
```

| **문제 55.** | test_label.csv 파일을 c:₩₩data₩₩test100₩₩ 경로에 복사하고 결과가 아래와 같이 출력될 수 있도록 함수를 생성하시오. |
|---|---|

| | |
|---|---|
| | 결과 :<br>['3'], ['8'], ['0'] , .... |

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

test_image ='c:₩₩data₩₩test100₩₩'
test_label ='c:₩₩data₩₩test100₩₩test_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()
    data  = []
    for i in file_name:
        file = path + str(i) + '.png'
        data.append(file)
    image = []
    for j in data:
        img = cv2.imread(j)
        image.append(img)
    return np.array(image)

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

    for i in labeldata:
        labellist.append(i)

    return labellist

print(label_load(test_label))
```

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

| **문제 56.** | 위의 숫자 리스트를 numpy 배열로 변환 하시오. |
|---|---|

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

```
import csv

test_image ='c:₩₩data₩₩test100₩₩'
test_label ='c:₩₩data₩₩test100₩₩test_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()
    data  = []
    for i in file_name:
        file = path + str(i) + '.png'
        data.append(file)
    image = []
    for j in data:
        img = cv2.imread(j)
        image.append(img)
    return np.array(image)

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

    for i in labeldata:
        labellist.append(i)

    return np.array(labellist)

print(label_load(test_label))
```

**결과

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

| **문제 57.** | 위의 결과가 문자가 아니라 숫자로 출력되게 변환 하시오. |
|---|---|

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

```
test_image ='c:₩₩data₩₩test100₩₩'
test_label ='c:₩₩data₩₩test100₩₩test_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()
    data  = []
    for i in file_name:
        file = path + str(i) + '.png'
        data.append(file)
    image = []
    for j in data:
        img = cv2.imread(j)
        image.append(img)
    return np.array(image)


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))

# print (image_load(test_image))
```

**결과

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

**# one hot encoding 된 데이터에서 numpy로 최대 인덱스를 가져오는 방법**

| **문제 58.** | 아래의 결과를 출력해보시오.<br>[0, 0, 0, 0, 1, 0, 0, 0, 0, 0] |
|---|---|

```
import numpy as np

print(np.eye(10)[4])
```

**결과
[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]

| **문제 59.** | 문제 57번에서 가져온 숫자 리스트를 가지고 아래와 같이 one hot encoding 된 결과를 출력 하시오. |
|---|---|

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

test_image ='c:₩₩data₩₩test100₩₩'
test_label ='c:₩₩data₩₩test100₩₩test_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()
    data  = []
    for i in file_name:
        file = path + str(i) + '.png'
        data.append(file)
    image = []
    for j in data:
        img = cv2.imread(j)
        image.append(img)
    return np.array(image)


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)
    label = np.eye(10)[label]

    return label

print(label_load(test_label))
```

**결과

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

| **문제 60.** | 위의 차원은 3차원인데 우리는 2차원으로 줄여야 한다. 왜냐하면 cnn코드에서 라벨이 입력될 때는 아래처럼 2차원이기 때문이다. 그래서 차원을 줄여서 출력하게끔 위의 함수를 수정 하시오.<br>tf.placeholder('float',[None,10])<br><br>테스트 해야할 내용 :<br>import numpy as np<br>x = np.array([[[0], [1], [2]]])<br>print(x.shape) # (1,3,1) (채널, 행, 열)<br>print(np.squeeze(x).shape) # (3,)<br>print(np.squeeze(x, axis=0).shape) #(3,1) (axis = 0 : 열 )<br>print(np.squeeze(x, axis=2).shape) # (1,3) |
|---|---|

**결과

| **문제 61.** | 라벨의 차원을 3차원에서 2차원으로 줄이시오.<br>(10000, 1 , 10) --> (10000,10) |
|---|---|

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

test_image ='c:\\data\\test100\\'
test_label ='c:\\data\\test100\\test_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()
    data  = []
    for i in file_name:
        file = path + str(i) + '.png'
        data.append(file)
    image = []
    for j in data:
        img = cv2.imread(j)
        image.append(img)
```

```
    return np.array(image)


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)
    label = np.eye(10)[label]
    label = np.squeeze(label, axis=1)

    return label.shape

print(label_load(test_label))
```

**결과

```
(10000, 10)
```

| 문제 62 | 오전에 만들었던 두가지 함수(image_load, label_load) 를 loader2.py 라는 파이썬 코드에 저장하고 아래와 같이 loader2.py 를 import 한후에 cifar10 전체 데이터를 로드하는 코드를 구현하시오 ! |
|---|---|

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

test_image ='c:\\data\\cifar10\\test\\'
test_label ='c:\\data\\cifar10\\test_label.csv'

train_image ='c:\\data\\cifar10\\train\\'
train_label ='c:\\data\\cifar10\\train_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()
    data  = []
    for i in file_name:
        file = path + str(i) + '.png'
        data.append(file)
    image = []
    for j in data:
        img = cv2.imread(j)
        image.append(img)
    return np.array(image)

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)
    label = np.eye(10)[label]
    label = np.squeeze(label, axis=1)

    return label

print("LOADING DATA")
start = time.time()
trainX = image_load(train_image)
print(trainX.shape) # (50000, 32, 32,3)
trainY = label_load(train_label)
print(trainY.shape) # (50000, 10)
testX = image_load(test_image)
print(testX.shape) # (10000,32, 32, 3)
testY = label_load(test_label)
print(testY.shape) # (10000, 10)

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

****결과**

```
LOADING DATA
(50000, 32, 32, 3)
(50000, 10)
(10000, 32, 32, 3)
(10000, 10)
image load time: 42.30
```

#cifar10을 위한 next_batch 함수 생성방법

| **문제 63.** | test100 폴더 밑에 10000개의 데이터 중 10개만 출력 하시오. |
|---|---|

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

test_image ='c:\\data\\cifar10\\test\\'
test_label ='c:\\data\\cifar10\\test_label.csv'

train_image ='c:\\data\\cifar10\\train\\'
train_label ='c:\\data\\cifar10\\train_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()
    data  = []
    for i in file_name:
        file = path + str(i) + '.png'
        data.append(file)
    image = []
    for j in data:
        img = cv2.imread(j)
        image.append(img)
    return np.array(image)

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)
    label = np.eye(10)[label]
    label = np.squeeze(label, axis=1)

    return label

print("LOADING DATA")
start = time.time()
trainX = image_load(train_image)
print(trainX.shape) # (50000, 32, 32,3)
trainY = label_load(train_label)
print(trainY.shape) # (50000, 10)
testX = image_load(test_image)
print(testX.shape) # (10000,32, 32, 3)
testY = label_load(test_label)
print(testY.shape) # (10000, 10)

end = time.time()
print(trainX[0:10])
```

**결과

```
LOADING DATA
(50000, 32, 32, 3)
(50000, 10)
(10000, 32, 32, 3)
(10000, 10)
[[[[ 63   62   59]
   [ 45   46   43]
   [ 43   48   50]
   ...
   [108 132 158]
   [102 125 152]
```

```
   [103 124 148]]
[[ 20  20  16]
   [  0   0   0]
   [  0   8  18]
   ...
```

| 문제 64. | 이번에는 라벨도 배치 사이즈 만큼 같이 출력할 수 있도록 next_batch 함수에 코드를 추가해서 아래와 같이 출력되게 하시오. |
|---|---|

```
import time


test_image ='c:\\data\\cifar10\\test\\'
test_label ='c:\\data\\cifar10\\test_label.csv'

train_image ='c:\\data\\cifar10\\train\\'
train_label ='c:\\data\\cifar10\\train_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()
    data  = []
    for i in file_name:
        file = path + str(i) + '.png'
        data.append(file)
    image = []
    for j in data:
        img = cv2.imread(j)
        image.append(img)
    return np.array(image)


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)
    label = np.eye(10)[label]
    label = np.squeeze(label, axis=1)

    return label


import time
```

```
test_image ='c:\\data\\cifar10\\test\\'
test_label ='c:\\data\\cifar10\\test_label.csv'

train_image ='c:\\data\\cifar10\\train\\'
train_label ='c:\\data\\cifar10\\train_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()
    data  = []
    for i in file_name:
        file = path + str(i) + '.png'
        data.append(file)
    image = []
    for j in data:
        img = cv2.imread(j)
        image.append(img)
    return np.array(image)


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)
    label = np.eye(10)[label]
    label = np.squeeze(label, axis=1)

    return 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,0,100))
```

**결과

```
(array([[[[ 49, 112, 158],
          [ 47, 111, 159],
          [ 51, 116, 165],
          ...,
```

```
        [ 36,  95, 137],
        [ 36,  91, 126],
        [ 33,  85, 116]],

       ], dtype=uint8), array([[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., 1., 0.],
              [1., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
              [0., 0., 0., 0., 0., 0., 1., 0., 0., 0.],
              [0., 0., 0., 0., 0., 0., 1., 0., 0., 0.],
              [0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
              [0., 0., 0., 0., 0., 0., 1., 0., 0., 0.],
  ...
```

| **문제 65.** | |
|---|---|

**결과

| **문제 66.** | 아래의 코드를 실행해 보시오. |
|---|---|

```
import random
import numpy as np

print(np.arange(10))
```

**결과
[0 1 2 3 4 5 6 7 8 9]

| **문제 67.** | 위의 숫자 10개가 랜덤으로 섞여서 출력되게 하시오. (random.shuffle를 사용해서 구현) |
|---|---|

```
import random
import numpy as np

a=np.arange(10)
random.shuffle(a)
print(a)
```

**결과
[2 7 0 9 8 5 4 3 6 1]

| **문제 68.** | 위의 코드를 이용해서 shuffle_batch 함수를 만들어서 입력된 데이터가 shuffle 되게 하시오 ! |
|---|---|

```
import time
import numpy as np
```

```
import random

test_image ='c:\\data\\cifar10\\test\\'
test_label ='c:\\data\\cifar10\\test_label.csv'

train_image ='c:\\data\\cifar10\\train\\'
train_label ='c:\\data\\cifar10\\train_label.csv'

print("LOADING DATA")

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()
    data  = []
    for i in file_name:
        file = path + str(i) + '.png'
        data.append(file)
    image = []
    for j in data:
        img = cv2.imread(j)
        image.append(img)
    return np.array(image)


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)
    label = np.eye(10)[label]
    label = np.squeeze(label, axis=1)

    return 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


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

print(shuffle_batch(testX[0:2],testY[0:2]))  #2개의 순서만 랜덤으로 ..
```

****결과**

LOADING DATA

```
(array([[[[235, 235, 235],
          [231, 231, 231],
          [232, 232, 232],
          ...,
          [233, 233, 233],
          [233, 233, 233],
          [232, 232, 232]],
         ....
         , dtype=uint8),
         array([[[0., 0., 0., 0., 0., 0., 0., 1., 0.],
                 [0., 0., 0., 1., 0., 0., 0., 0., 0.]]))
```

| **문제 69.** | 위에서 만든 next_batch 함수와 shuffle_batch 함수를 loader2.py에 추가 하시오. |
|---|---|

```
#loder2.py

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

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()
    data  = []
    for i in file_name:
        file = path + str(i) + '.png'
        data.append(file)
    image = []
    for j in data:
        img = cv2.imread(j)
        image.append(img)
    return np.array(image)

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)
    label = np.eye(10)[label]
    label = np.squeeze(label, axis=1)

    return 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

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
```

| **문제 70.** | 기존 mnist 데이터를 cnn 신경망에 놓는 코드에 mnist 대신에 cifar10 데이터를 이용해서 학습시키고 정확도 그래프를 볼 수 있도록 코드를 완성 시키시오.<br><br>1. mnist 데이터 cnn 신경망 코드<br><br>--> 수정<br>1. cnn 신경망의 데이터 행렬<br>2. |
|---|---|

```
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)

# cifar10 데이터 로드
train_image = 'D:\\a\\cifar10\\train100\\'
train_label = 'D:\\a\\cifar10\\train_label.csv'
test_image = 'D:\\a\\cifar10\\test100\\'
test_label = 'D:\\a\\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)
testX, testY = loader2.shuffle_batch(testX, testY)

tf.reset_default_graph()

x = tf.placeholder(tf.float32, [None, 32, 32, 1])
y = tf.placeholder(tf.float32, [None, 10])
keep_prob = tf.placeholder(tf.float32)
```

```
W1 = tf.Variable(tf.random_normal([3, 3, 3, 32], stddev=0.01))
L1 = tf.nn.conv2d(x, W1, strides=[1, 1, 1, 1], padding='SAME')
L1 = tf.nn.relu(L1)
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))
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, 8 *8 * 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 = loader2.next_batch(trainX,trainY,0,100) # 훈련 데이터
        test_xs, test_ys = loader2.next_batch(testX, testY, 0, 100) # 테스트 데이터

       #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)   #cifar10은 사진이라 필요없다

        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()

**결과
LOADING DATA
(array([[[[235, 235, 235],
          [231, 231, 231],
          [232, 232, 232],
          ...,
          [233, 233, 233],
          [233, 233, 233],
          [232, 232, 232]],
         ....
        , dtype=uint8),
        array([[0., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
               [0., 0., 0., 1., 0., 0., 0., 0., 0., 0.]]))
```

| 문제 71. | 개 고양이 사진이 있는 폴더를 만들고 그 폴더에 개 사진 100장과 고양이 사진 100장을 넣고 아래와 같이 불러오는 함수를 생성 하시오. |
|---|---|

```
import os
```

```
test_images ='./catdog'

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

print ( image_load(test_images) )
```

**결과
[' (198).jpeg', ' (48).jpeg', ' (128).jpeg', ' (39).jpeg', ' (159).jpeg', ' (83).jpeg', ' (117).jpeg', ' (65).jpeg', ' (76).jpeg', ' (116).jpeg', ' (126).jpeg', ' (64).jpeg', ....

| 문제 72. | 위의 결과에서 아래와 같이 숫자만 출력되게 하시오. |
|---|---|

```
import os
import re

test_images ='./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)

   return  file_name

print ( image_load(test_images) )
```

**결과
[198, 48, 128, 39, 159, 83, 117, 65, ... ]

| 문제 73. | 72번의 결과를 정렬되게 하시오. |
|---|---|

```
import os
import re

test_images ='./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) )
```

**결과
[1, 2, 3, 4, 5, 6,  ... ]