

Python 수업 5회차

지난시간 - graph의 assert 는 예외처리..

```
g_1 = tf.Graph()
with g_1.as_default():
    # Operation created in this scope will be added to g_1.
    c = tf.constant(5)
```

`assert` 문은 python 뿐만 아니라 다른 언어에서도 있는 기능입니다

보통

```
# Session
sess_1 = tf.Session(g_1)
```

`assert condition`

```
g_2 = tf.Graph()
with g_2.as_default():
    # Operations created in this scope will be added to g_2.
    d = tf.constant("Node in g_2")
```

처럼 쓰고 프로그램에게 "이 condition에 맞지 않으면 error를 내줘!"라는 의미로 씁니다.

```
# Alternatively, you can pass a graph to the constructor. See http://www.tensorflow.org/api\_docs/python/tf/Session for more details.
# `sess_2` will run operations from `g_2`.
sess_2 = tf.Session(graph=g_2)
```

```
assert c.graph is g_1
assert sess_1.graph is g_1
```

```
assert d.graph is g_2
assert sess_2.graph is g_2
```

```
# 그래프 준비
a = 3
# 세션 실행단계
a = 5
print(a)
assert a is 3
```

5

```
Traceback (most recent call last):
  File "/Users/youngjae/PycharmProject/...", line ..., in <module>
    assert a is 3
AssertionError
```

지난시간 - 로지스틱 회귀

로지스틱 회귀의 목적은 일반적인 **회귀 분석**의 목표와 동일하게 **종속 변수**와 독립 변수간의 관계를 구체적인 함수로 나타내어 향후 예측 모델에 사용하는 것이다. 이는 독립 변수의 선형 결합으로 종속 변수를 설명한다는 관점에서는 선형 회귀 분석과 유사하다. 하지만 로지스틱 회귀는 **선형 회귀** 분석과는 다르게 종속 변수가 범주형 데이터를 대상으로 하며 입력 데이터가 주어졌을 때 해당 데이터의 결과가 특정 분류로 나뉘기 때문에 일종의 분류 (classification) 기법으로도 볼 수 있다.

오늘의 목표 - 아래 코드 이해하기

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

# Dataset loading
mnist = input_data.read_data_sets("./samples/MNIST_data/", one_hot=True)

# Set up model
x = tf.placeholder(tf.float32, [None, 784])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
y = tf.nn.softmax(tf.matmul(x, W) + b)

y_ = tf.placeholder(tf.float32, [None, 10])

cross_entropy = -tf.reduce_sum(y_*tf.log(y))
train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_entropy)
```

```
# Session
```

```
init = tf.global_variables_initializer()
```

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

```
sess.run(init)
```

```
# Learning
```

```
for i in range(1000):
```

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

```
    sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
```

```
# Validation
```

```
correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
```

```
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

```
# Result should be approximately 91%.
```

```
print(sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels}))
```

소프트맥스 회귀란?

- ▶ 뉴런의 출력 값을 정규화하는 함수.
- ▶ 멀티 클래스(multi-class) 분류인 경우 소프트맥스 함수를 자주 사용.
 - ▶ 다변량 로지스틱
- ▶ 각 분류별로 결과 값을 구한 뒤, 0~1 사이의 값으로 변환해주는 함수.
 - ▶ 확률화

$$\begin{bmatrix} w_{a1} & w_{a2} & w_{a3} \\ w_{b1} & w_{b2} & w_{b3} \\ w_{c1} & w_{c2} & w_{c3} \end{bmatrix} \times \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix} = \begin{bmatrix} y1 \\ y2 \\ y3 \end{bmatrix} \xrightarrow{\text{소프트맥스}} S(y_i) = \frac{e^{y_j}}{\sum_j e^{y_j}} \xrightarrow{\text{소프트맥스}} \begin{bmatrix} p1 \\ p2 \\ p3 \end{bmatrix}$$

소프트맥스

$p1 + p2 + p3 = 1.0$

```
#### SoftMax ####
```

```
import numpy as np
```

```
def softmax(x):  
    ex = np.exp(x)  
    return ex / ex.sum()
```

```
x = np.array([1.0, 1.0, 2.0])
```

```
y = softmax(x)
```

```
print(y)
```

```
[ 0.21194156  0.21194156  0.57611688]
```

크로스 엔트로피 손실함수란?

- ▶ 신경망의 손실함수로 사용되는 함수.
- ▶ 손실함수란 - 모델의 결과 값과 실제 대상 값 사이의 차이.
 - ▶ 이 간극을 줄이는 과정이 학습이다.

Mnist란?

- ▶ 머신러닝과 텐서플로우의 기본적인 내용을 연습을 목적으로 하는 튜토리얼용 데이터.
- ▶ Tensorflow 를 통해 쉽게 load해올 수 있다.
 - ▶ 참조
<https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/tutorials/mnist>
- ▶ 28 X 28(= 784) 크기의 0~9 까지의 손글씨가 numpy array로 들어있다.

Mnist 가져오기(load)

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

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

```
def read_data_sets(train_dir,  
                    fake_data=False,  
                    one_hot=False,  
                    dtype=dtypes.float32,  
                    reshape=True,  
                    validation_size=5000,  
                    seed=None,  
                    source_url=DEFAULT_SOURCE_URL):  
  
    train = DataSet(train_images, train_labels, **options)  
    validation = DataSet(validation_images, validation_labels, **options)  
    test = DataSet(test_images, test_labels, **options)  
  
    return base.Datasets(train=train, validation=validation, test=test)
```

Mnist dataset 파악하기 - 데이터 크기

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

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

```
trainimg = mnist.train.images #이미지의 픽셀 데이터
```

```
trainlabel = mnist.train.labels # 이미지의 정답들
```

```
testimg = mnist.test.images
```

```
testlabel = mnist.test.labels
```

```
print(trainimg.shape)
```

```
print(trainlabel.shape)
```

```
print(testimg.shape)
```

```
print(testlabel.shape)
```

```
df = pd.DataFrame(mnist.train.images).head(10)
```

```
df.head()
```

```
Extracting ./samples/MNIST_data/train-images-idx3-ubyte.gz
Extracting ./samples/MNIST_data/train-labels-idx1-ubyte.gz
Extracting ./samples/MNIST_data/t10k-images-idx3-ubyte.gz
Extracting ./samples/MNIST_data/t10k-labels-idx1-ubyte.gz
(55000, 784)
(55000, 10)
(10000, 784)
(10000, 10)
```

	0	1	2	3	4	5	6	7	8	9	...	774	775	776	777	778	779	780	781	782
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

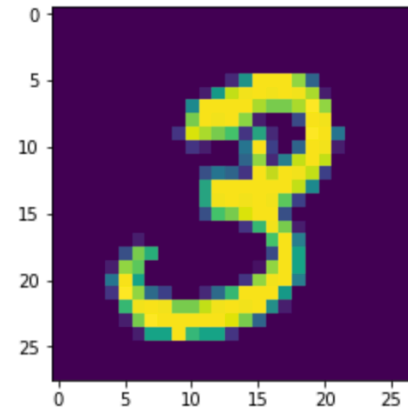
5 rows × 784 columns

Mnist dataset 파악하기 - 그림으로

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

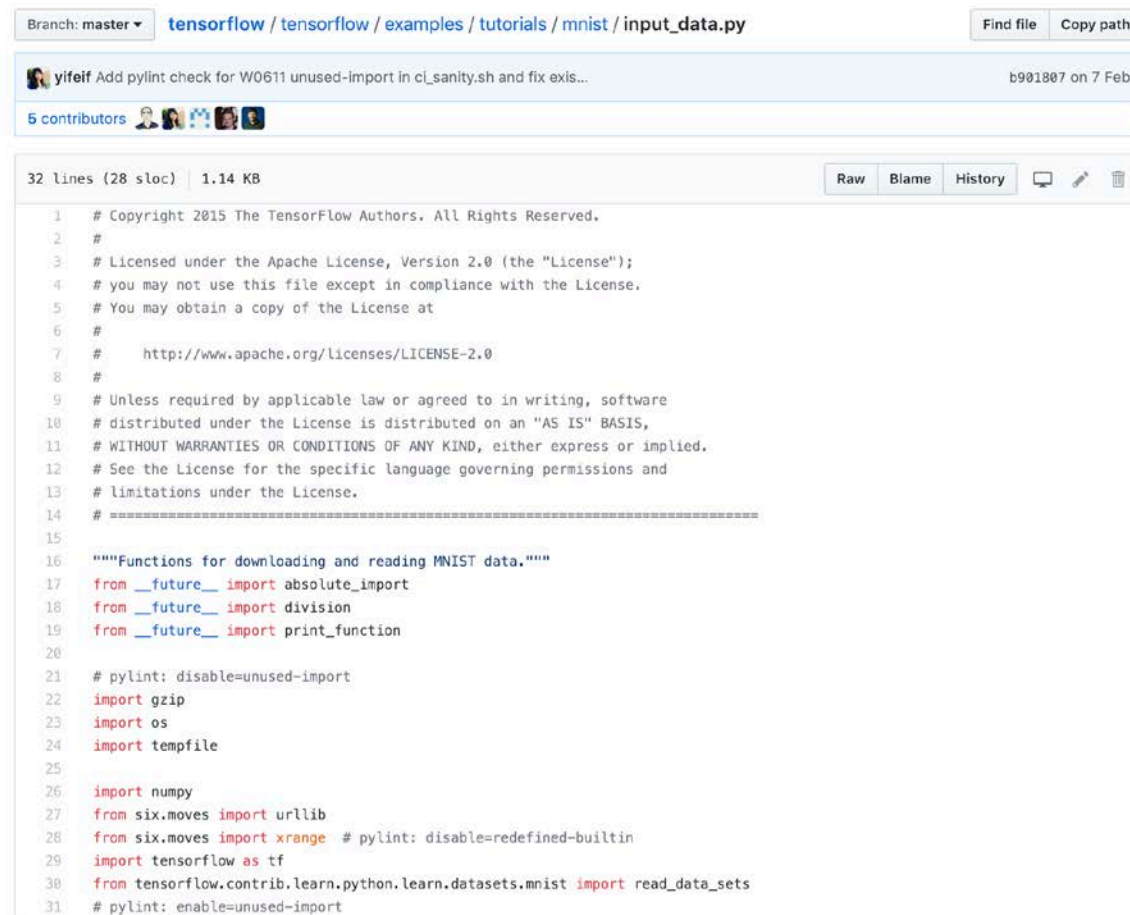
mnist = input_data.read_data_sets("./samples/MNIST_data/", one_hot=True)
arr= np.array(mnist.train.images[1])
arr.shape = (28,28) # 784 = 28 X 28 이다
# imshow는 2차원 자료의 크기를 색깔로 표시
plt.imshow(arr)
```

<matplotlib.image.AxesImage at 0xb43669e48>



프로그램 짜보기 - 1. Dataset loading

import tensorflow as tf
from tensorflow.examples.



```
1 # Copyright 2015 The TensorFlow Authors. All Rights Reserved.
2 #
3 # Licensed under the Apache License, Version 2.0 (the "License");
4 # you may not use this file except in compliance with the License.
5 # You may obtain a copy of the License at
6 #
7 # http://www.apache.org/licenses/LICENSE-2.0
8 #
9 # Unless required by applicable law or agreed to in writing, software
10 # distributed under the License is distributed on an "AS IS" BASIS,
11 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12 # See the License for the specific language governing permissions and
13 # limitations under the License.
14 # =====
15
16 """Functions for downloading and reading MNIST data."""
17 from __future__ import absolute_import
18 from __future__ import division
19 from __future__ import print_function
20
21 # pylint: disable=unused-import
22 import gzip
23 import os
24 import tempfile
25
26 import numpy
27 from six.moves import urllib
28 from six.moves import xrange # pylint: disable=redefined-builtin
29 import tensorflow as tf
30 from tensorflow.contrib.learn.python.learn.datasets.mnist import read_data_sets
31 # pylint: enable=unused-import
```

프로그램 짜보기 - 1. Dataset loading

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

#one_hot 이란?

- **Incoding**방식. 해당하는 특성에만 1을, 나머지면 0을 할당하는 방법.

	A	B	C	D	E	F	G	H	I
1	Original data:			One-hot encoding format:					
2	id	Color		id	White	Red	Black	Purple	Gold
3	1	White		1	1	0	0	0	0
4	2	Red		2	0	1	0	0	0
5	3	Black		3	0	0	1	0	0
6	4	Purple		4	0	0	0	1	0
7	5	Gold		5	0	0	0	0	1
8									
9									

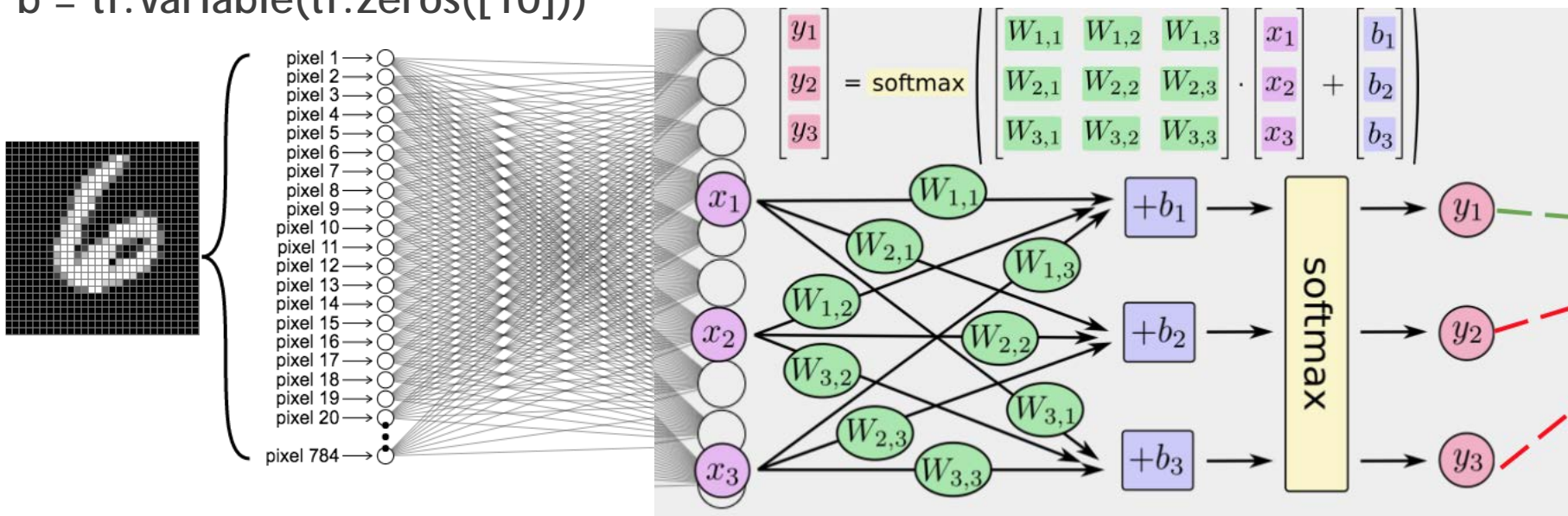
프로그램 짜보기 - 2. 모델 구현

```
x = tf.placeholder(tf.float32, [None, 784])
```

x는 784차원의 벡터로 변형된 MNIST 이미지의 데이터를 넣을 변수.

```
W = tf.Variable(tf.zeros([784, 10]))
```

```
b = tf.Variable(tf.zeros([10]))
```

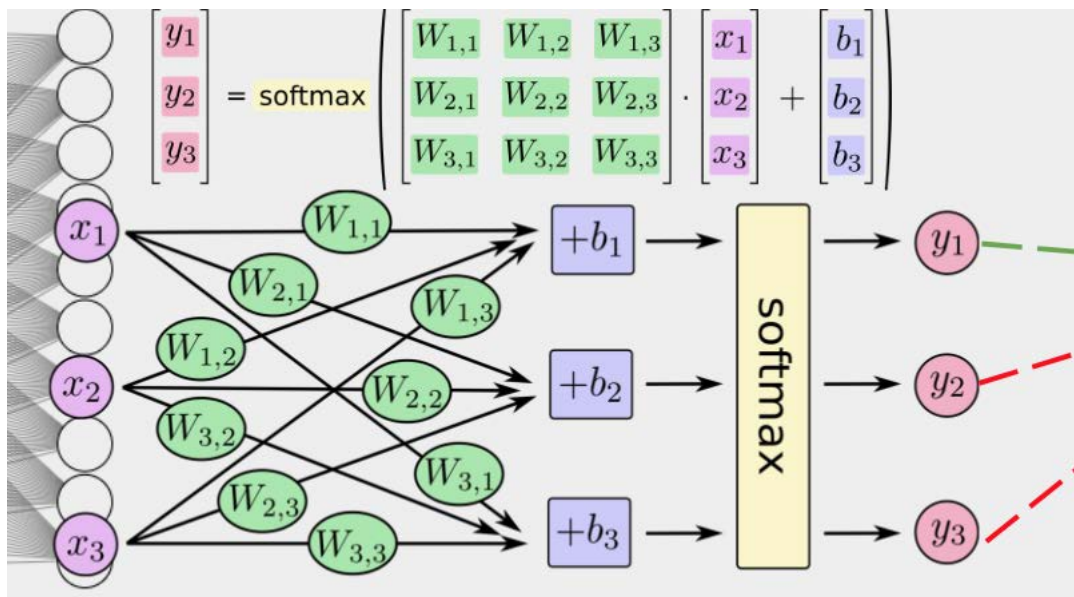


프로그램 짜보기 - 2. 모델 구현

#softmax 함수 구현.

```
y = tf.nn.softmax(tf.matmul(x, W) + b)
```

한줄로도 가능. X와 W를 곱하여 편향을 더한것.



프로그램 짜보기 - 3. 학습 구현

#올바른 실제 답을 넣기 위한 새로운 placeholder

```
y_ = tf.placeholder(tf.float32, [None, 10])
```

교차 엔트로피 $-\sum y' \log(y)$ 를 구현 (축약되었음을 참고..)

```
cross_entropy = -tf.reduce_sum(y_*tf.log(y))
```

tf.log는 y의 각 원소의 로그 값을 계산합니다.

그 다음, y_의 각 원소를 tf.log(y)의 해당하는 원소들과 곱합니다.

tf.reduce_sum으로 텐서의 모든 원소를 더합니다.

프로그램 짜보기 - 3. 학습 구현

학습도를 0.01로 준 경사 하강법 (gradient descent) 알고리즘

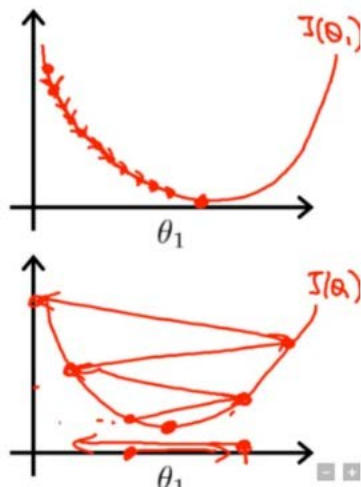
```
train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_entropy)
```

```
new_weight = existing_weight - learning_rate * gradient
```

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

If α is too small, gradient descent can be slow.

If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.



Gradient descent with small (top) and large (bottom) learning rates. Source: Andrew Ng's Machine Learning course on Coursera

프로그램 짜보기 - 4. 변수 초기화

```
init = tf.global_variables_initializer()
```

프로그램 짜보기 - 5. 세션 실행

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

프로그램 짜보기 - 6. 학습 실행

```
for i in range(1000):
```

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

```
    sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
```

1000번 반복

각 반복 단계마다, 학습 세트로부터 100개의 무작위 데이터들의 일괄 처리(batch)들을 가져옴.

placeholders를 대체하기 위한 일괄 처리 데이터에 train_step Feeding

프로그램 짜보기 - 7. 모델 평가하기

```
correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))  
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

`tf.argmax(input, axis=None, name=None, dimension=None)`

`tf.argmax`는 특정한 축(= 1차원)을 따라 가장 큰 원소의 색인을 알려줌

즉, 훈련데이터 `y`가 실제 데이터 `y_`와 같은지 `equal`로 비교. 같으면 `true`, 다르면 `false`

`correct_prediction`는 정답여부를 `true`, `false`로 가지고 있음. `Cast`는 이를 1, 0 으로 바꿔줌.

`reduce_mean`은 이 1과 0 의 값들의 평균을 구해줌.

ex) `[True, False, True, True]`는 `[1,0,1,1]` 이 되고 평균값은 0.75 이 됨

프로그램 짜보기 - 8. 결과 출력

```
print(sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels}))
```

#마지막으로 정확도 accuracy 출력

오늘의 목표 - 전체 다시 보기

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

# Dataset loading
mnist = input_data.read_data_sets("./samples/MNIST_data/", one_hot=True)

# Set up model
x = tf.placeholder(tf.float32, [None, 784])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
y = tf.nn.softmax(tf.matmul(x, W) + b)      #출력 함수

y_ = tf.placeholder(tf.float32, [None, 10]) #실제 정답이 들은 변수

cross_entropy = -tf.reduce_sum(y_*tf.log(y))      # 학습함수(크로스엔트로피 오차함수)
train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_entropy) #경사하강법
```

Session

init = tf.global_variables_initializer()

#변수초기화

sess = tf.Session()

#세션 시작

sess.run(init)

Learning

for i in range(1000):

#학습 1000번 시행

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

#데이터 100개씩 불러와

sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})

#경사하강법

Validation

correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))

#모델평가

accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

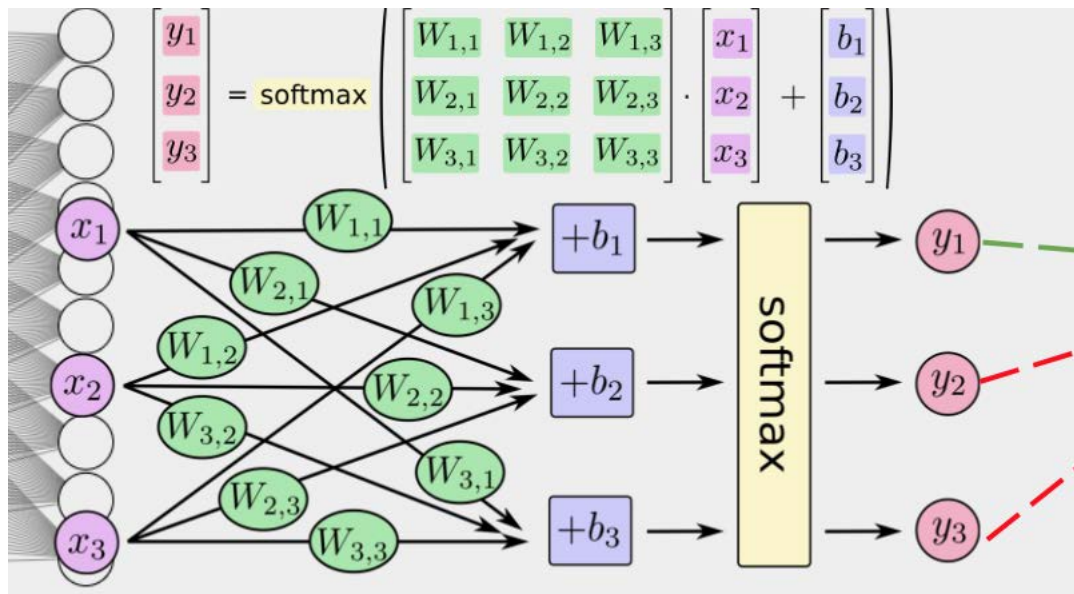
Result should be approximately 91%.

print(sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels}))

#결과 출력

결과 분석.

- ▶ 대략 90% 정도가 나옴.
- ▶ 좋은 모델인가?
 - ▶ 아니다.
- ▶ 최적화 시키는 방법은?
 - ▶ 층 쌓기, 크로스엔트로피, 경사하강법, 학습횟수, 데이터셋의 분리, 오버피팅 등...
- ▶ 좀더 최적화 시키면 99% 이상도 가능하다.



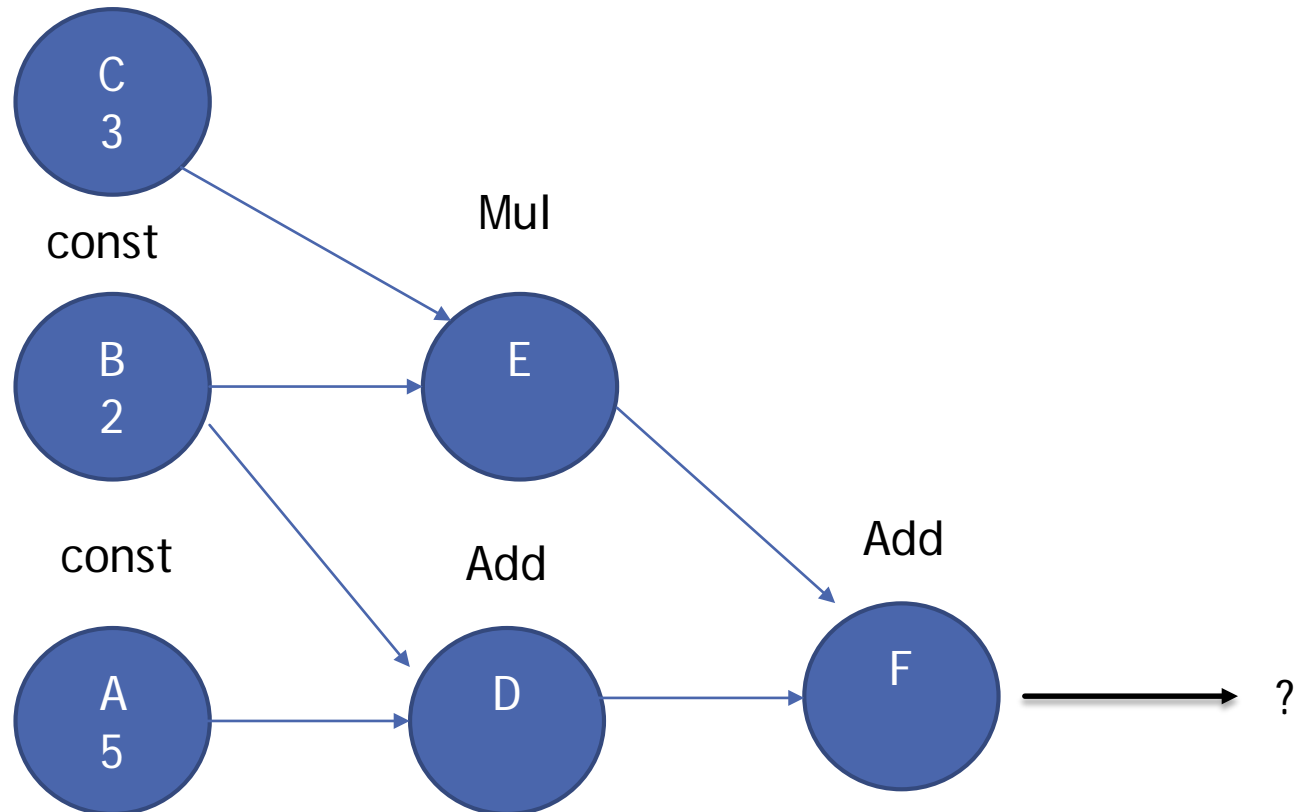
다음시간 숙제1

- ▶ 이 모델에서 정확도를 92%이상으로 끌어올리도록 최적화를 시켜 오기

숙제2

- ▶ 실행시 placeholder 변수인 C에 3이란 값을 feed하여 계산하는 아래의 계산그래프를 Tensorflow로 작성하고, F값을 print 하시오.

Placeholder



숙제3 -최소 아래의 개념 이해하고 오기

- ▶ CNN의
 - ▶ 합성곱계층(conv2d)
 - ▶ 풀링계층(max_pool_2x2)
 - ▶ 활성화 함수(Relu)
- ▶ 오버피팅을 막는 최적화 방법
 - ▶ 드롭아웃(Dropout)

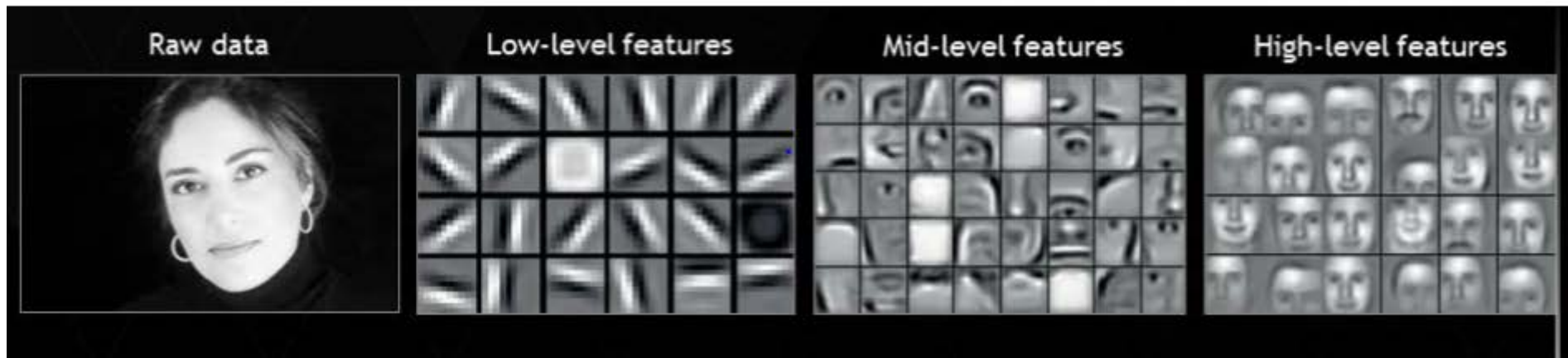
합성곱 신경망 (CNN)

(Convolutional neural network)

합성곱 신경망(CNN) 이란?

▶ Convolutional Neural Network

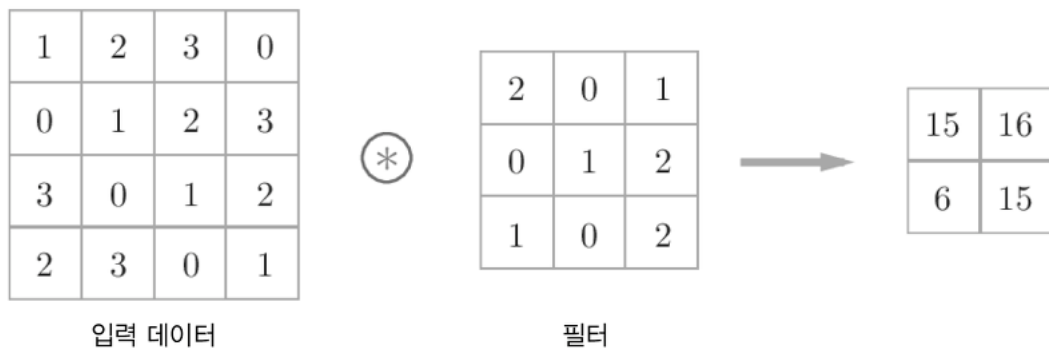
- ▶ 초기 layer에서는 edge나 line을 학습하고, 그 다음 이미지의 high-level 표현을 습득한다.
- ▶ 시각적 이미지를 분석하는 데 주로 사용되는 딥러닝 알고리즘.



CNN의 핵심인 두 계층

▶ 합성곱(convolutional) 계층

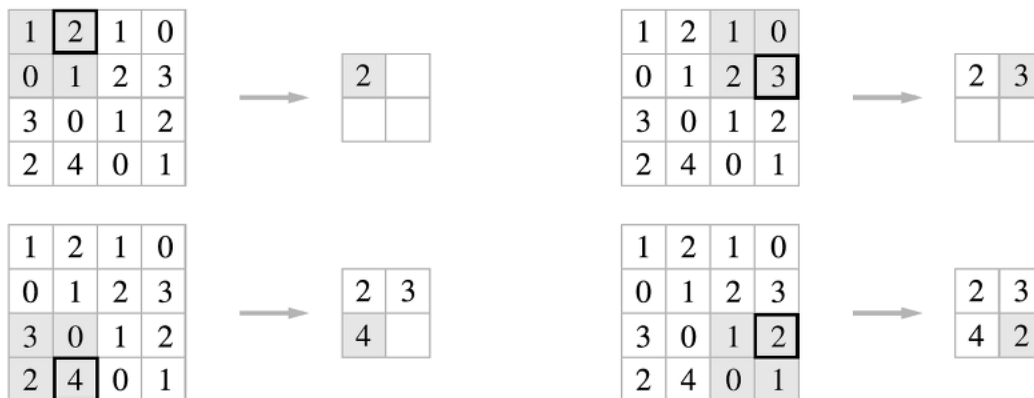
- ▶ 필터를 이용해 합성곱 연산을 수행한다. 필터(커널 = 가중치)의 윈도우를 일정 간격으로 이동해가며 입력 데이터에 적용한다.
- ▶ 딥러닝과 달리 데이터 형상이 유지됨.



CNN의 핵심인 두 계층

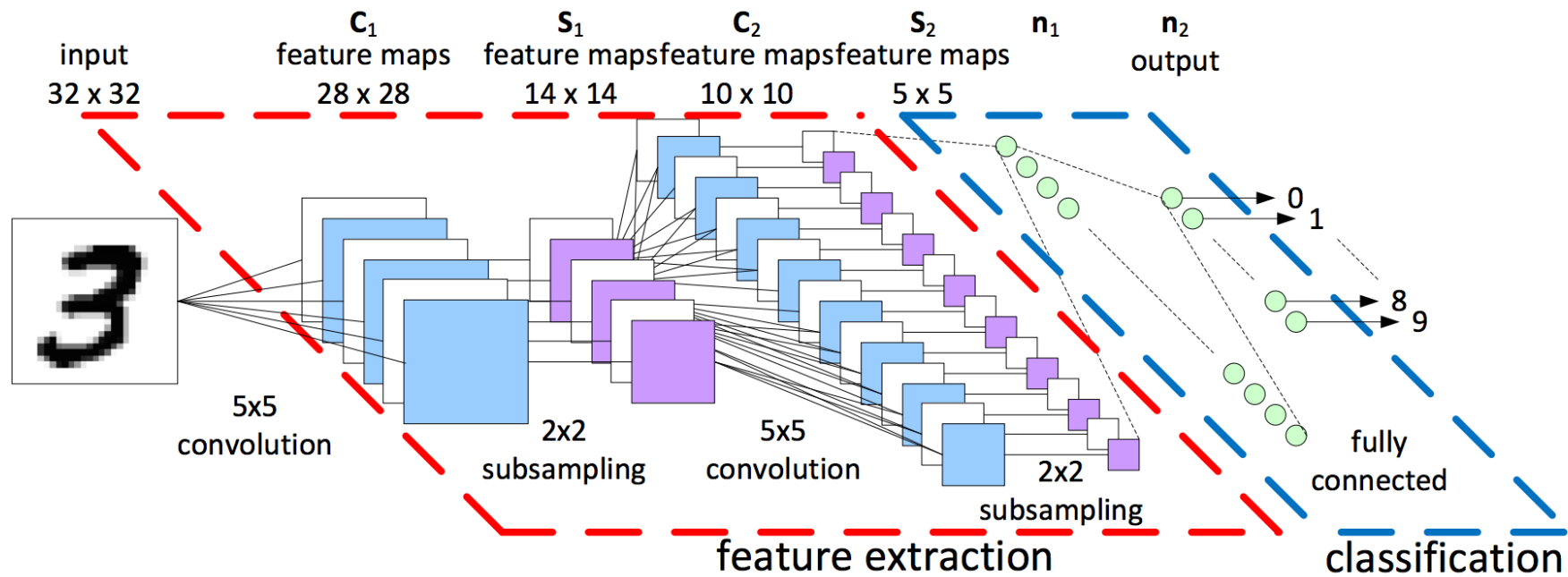
▶ 풀링(pooling) 계층

- ▶ 풀링은 2차원 데이터의 세로 및 가로 방향의 공간을 줄이는 연산
- ▶ 풀링에는 최대 풀링(Max Pooling), 평균 풀링(Average Pooling) 등이 있다.
- ▶ 최대 풀링은 대상 영역에서 최댓값을 취하는 연산, 평균 풀링은 대상 영역의 평균을 계산한다. 이미지 인식 분야에서는 주로 최대 풀링을 사용한다.



Mnist에 적용

- ▶ 합성곱층과 풀링층을 거쳐 사이즈를 줄인뒤, fully Connected로 최종결과값 도출
- ▶ 그리고 softmax를 사용해 분류(classification)



읽어보기

- ▶ <https://codeonweb.com/entry/f50e23df-0f23-4e56-95a6-efb9981716f7>
- ▶ <https://tensorflow.blog/5-%ED%85%90%EC%84%9C%ED%94%8C%EB%A1%9C%EC%9A%B0-%EB%8B%A4%EC%A4%91-%EB%A0%88%EC%9D%B4%EC%96%B4-%EB%89%B4%EB%9F%B4-%EB%84%A4%ED%8A%B8%EC%9B%8C%ED%81%AC-first-contact-with-tensorflow/>