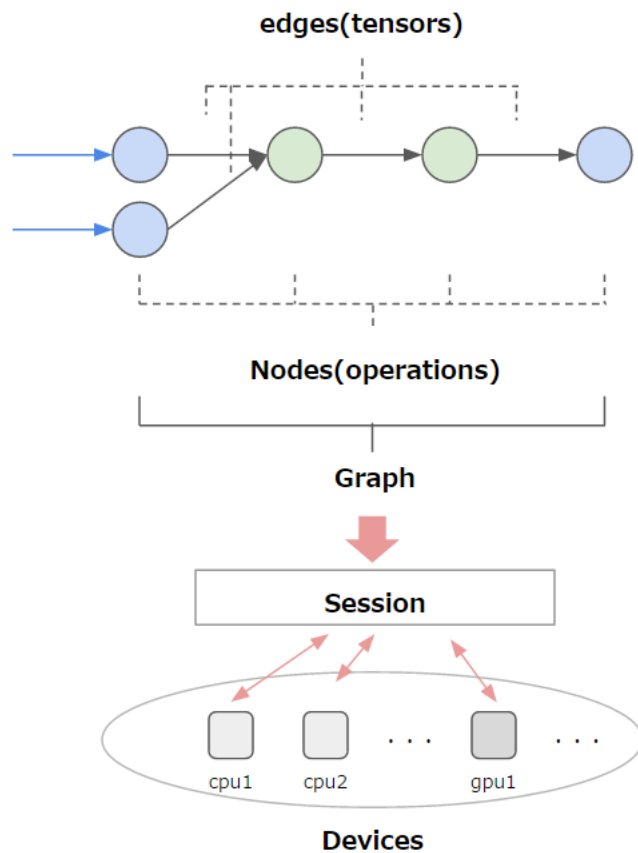


Python 수업 4회차

1. 포함관계표현



Device > Session > graph > 점&선

2. 머신러닝 프로그램의 단계

- ▶ 1. 구성 단계(construction phase)
 - ▶ 보통 graph를 조립함
- ▶ 2. 실행 단계(execution phase)
 - ▶ session을 이용해 graph의 작업(op)을 실행시키는 단계
- ▶ Ex) 뉴럴 네트워크를 표현하고 학습하는 프로그램
 - ▶ 구성 단계 : graph를 만들기
 - ▶ 실행 단계 : graph의 훈련용 작업들(set of training ops)을 반복해서 실행

3. Tensorflow 그래프에서 점과 선이 의미하는것

- ▶ 점 : 연산
- ▶ 선 : 데이터

4. 프로그램 구성 과정

- ▶ 1. Dataset loading : 손글씨 데이터 들을 loading해 온다.
- ▶ 2. Set up model : 그래프 구성을 위한 변수와 함수등을 만든다.
- ▶ 3. Session : 세션을 준비하고, 변수를 초기화 시킨다.
- ▶ 4. Learning : 적당한 횟수로 모델을 학습시킨다.
- ▶ 5. Validation : 학습한 모델이 얼마나 정확한지 평가한다.
- ▶ 6. Result : 결과를 출력한다.

Name 이 왜 필요한가요?

- ▶ 전반적으로 그래프의 큰 그림을 볼때 직관을 제공해준다.
 - ▶ 이런 느낌 때문이지 않을까.

```
import tensorflow as tf
```

```
my_const = tf.constant([1.0, 2.0], name="my_const")
```

```
your_const = tf.constant([2.0, 3.0], name="your_const")
```

```
print (tf.get_default_graph().as_graph_def())
```

Fetches & Feeds 그리고 Placeholder

Placeholder는 한국어로는 안내문구? 정도가 될듯.

Fetches란? - 가져오기?

- ▶ Tensorflow의 run의 인수에 변수를 지정해 특정 노드(작업)의 실행을 요청하는 것.
 - ▶ Ex) sess.run(init)
- ▶ Tensorflow는 노드 간 필요한 노드만 연산한다.
- ▶ 작업의 결과를 가져오기 위해 Session 오브젝트에서 run()을 호출해서 graph를 실행하고 tensor로 결과값을 끌어냅니다.
- ▶ 하나의 노드 뿐 아니라 복수의 tensor를 받아올 수도 있습니다.

이 코드에서 a의 값을 세션실행 후 바꾸고 싶다면?

```
import tensorflow as tf
```

```
a = tf.constant(5,name='input_a')
```

```
b = tf.constant(7,name='input_b')
```

```
c = tf.multiply(a,b,name='mul_c')
```

```
d = tf.add(a,b,name='add_d')
```

```
e = tf.add(c,d,name='add_e')
```

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

```
print(sess.run(e))
```

Feed dictionary

```
import tensorflow as tf
```

```
a = tf.constant(5,name='input_a')
```

```
b = tf.constant(7,name='input_b')
```

```
c = tf.multiply(a,b,name='mul_c')
```

```
d = tf.add(a,b,name='add_d')
```

```
e = tf.add(c,d,name='add_e')
```

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

```
print(sess.run(e,feed_dict={a:10}))
```

문제 제기

- ▶ 위의 예제를 잘보면 알겠지만 만약 값을 Feed Dictionary를 사용한다면 위치를 값을 지정해주는건 무의미하다.
- ▶ 어쨌든 뒤에 값을 입력받는데 앞에 뭐하러 값을 선언하는가?
- ▶ 이렇게 뒤에 값을 받아서 집어넣기만 하는 값이 있다면(즉 input으로만 값을 받는다면) 굳이 노드에 값을 선언할 필요가 없다는 뜻이된다.
- ▶ 이런 경우 값을 비워두고 input받는 데이터라는 것을 명시적으로 선언하는 방법이 있다.
- ▶ 그게 바로 Place Holder이다.

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

Placeholder

- ▶ 나중에 채워질 빈 변수.

```
import tensorflow as tf
```

```
a = tf.placeholder(tf.int32, shape=[])
```

```
b = tf.constant(7, name='input_b')
```

```
c = tf.multiply(a, b, name='mul_c')
```

```
d = tf.add(a, b, name='add_d')
```

```
e = tf.add(c, d, name='add_e')
```

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

```
print(sess.run(e, feed_dict={a: 10}))
```



tf.placeholder(tf.int32, shape = [3, None])
과 같이 크기를 확신할 수 없는 경우 none을 써주
기도 한다.

그밖에 알아두면 좋은 것

데이터 자료형

수학 작업, 초기화함수

디바이스, 그래프, 세션

데이터 자료형

▶ 많다.

Data type	Python type	Description
DT_FLOAT	<code>tf.float32</code>	32 bits floating point.
DT_DOUBLE	<code>tf.float64</code>	64 bits floating point.
DT_INT8	<code>tf.int8</code>	8 bits signed integer.
DT_INT16	<code>tf.int16</code>	16 bits signed integer.
DT_INT32	<code>tf.int32</code>	32 bits signed integer.
DT_INT64	<code>tf.int64</code>	64 bits signed integer.
DT_UINT8	<code>tf.uint8</code>	8 bits unsigned integer.
DT_UINT16	<code>tf.uint16</code>	16 bits unsigned integer.
DT_STRING	<code>tf.string</code>	Variable length byte arrays. Each element of a Tensor is a byte array.
DT_BOOL	<code>tf.bool</code>	Boolean.
DT_COMPLEX64	<code>tf.complex64</code>	Complex number made of two 32 bits floating points: real and imaginary parts.
DT_COMPLEX128	<code>tf.complex128</code>	Complex number made of two 64 bits floating points: real and imaginary parts.
DT_QINT8	<code>tf.qint8</code>	8 bits signed integer used in quantized Ops.
DT_QINT32	<code>tf.qint32</code>	32 bits signed integer used in quantized Ops.
DT_QUINT8	<code>tf.quint8</code>	8 bits unsigned integer used in quantized Ops.

수학 작업(Op)

▶ 많다.

Category	Examples
Element-wise mathematical operations	Add, Sub, Mul, Div, Exp, Log, Greater, Less, Equal, ...
Array operations	Concat, Slice, Split, Constant, Rank, Shape, Shuffle, ...
Matrix operations	MatMul, MatrixInverse, MatrixDeterminant, ...
Stateful operations	Variable, Assign, AssignAdd, ...
Neural network building blocks	SoftMax, Sigmoid, ReLU, Convolution2D, MaxPool, ...
Checkpointing operations	Save, Restore
Queue and synchronization operations	Enqueue, Dequeue, MutexAcquire, MutexRelease, ...
Control flow operations	Merge, Switch, Enter, Leave, NextIteration

초기화 함수

텐서플로 연산	설명
<code>tf.constant(value)</code>	Value 값 으로 채워진 텐서 생성
<code>tf.fill(shape, value)</code>	Shape 형태의 텐서만들고, value 값으로 초기화
<code>tf.zeros(shape)</code>	Shape 형태의 텐서 만들고, 모든 값 0으로 초기화
<code>tf.zeros_like(tensor)</code>	Tensor와 동일한 타입과 형태의 텐서 만들고 모든 값 0
<code>tf.ones(shape)</code>	Shape 형태의 텐서 만들고, 모든 원소 값 1로 초기화
<code>tf.ones_like(tensor)</code>	Tensor와 동일한 타입과 형태의 텐서 만들고 모든 값 1
<code>tf.random_normal(shape, mean, stddev)</code>	정규분포를 따르는 난수 생성
<code>tf.truncated_normal(shape, mean, stddev)</code>	절단 정규분포 를 따르는 난수 생성
<code>tf.random_uniform(shape, minval, maxval)</code>	Minval이상 maxval 미만의 균등 분포 값 생성
<code>tf.random_shuffle(tensor)</code>	첫번째 차원에 따라 텐서를 무작위로 뒤섞음

디바이스(Device)

```
# Operations created outside either context will run on the "best possible"
# device. For example, if you have a GPU and a CPU available, and the operation
# has a GPU implementation, TensorFlow will choose the GPU.
weights = tf.random_normal(...)

with tf.device("/device:CPU:0"):
    # Operations created in this context will be pinned to the CPU.
    img = tf.decode_jpeg(tf.read_file("img.jpg"))

with tf.device("/device:GPU:0"):
    # Operations created in this context will be pinned to the GPU.
    result = tf.matmul(weights, img)
```

세션(Session)

```
# Create a default in-process session.  
with tf.Session() as sess:  
    # ...  
  
# Create a remote session.  
with tf.Session("grpc://example.org:2222"):  
    # ...
```

대화형 세션(Interactive session)

- ▶ 지금까지는 [Session](#)을 실행시키고 [Session.run\(\)](#) 메서드를 이용해왔다.
- ▶ 하지만 python은 interpreter 언어이기에 한줄한줄 치고 바로 응답받는 편의성을 제공한다.
- ▶ 따라서 jupyter나 command console등을 이용하여 대화형으로 프로그래밍을 하고 싶을때 이용.
- ▶ `tf.InteractiveSession()`을 사용해 시작.
- ▶ `Sess.run()` 대신 [Tensor.eval\(\)](#) 와 [Operation.run\(\)](#) 를 이용하자.

대화형 세션 예시.

```
# 인터랙티브 TensorFlow Session을 시작해봅시다.  
# Enter an interactive TensorFlow Session.  
import tensorflow as tf  
sess = tf.InteractiveSession()  
  
x = tf.Variable([1.0, 2.0])  
a = tf.constant([3.0, 3.0])  
  
# 초기화 op의 run() 메서드를 이용해서 'x'를 초기화합니다.  
# Initialize 'x' using the run() method of its initializer op.  
x.initializer.run()  
  
# 'x'에서 'a'를 빼는 작업을 추가하고 실행시켜서 결과를 봅시다.  
# Add an op to subtract 'a' from 'x'. Run it and print the result  
sub = tf.subtract(x, a)  
print(sub.eval())  
# ==> [-2. -1.]  
  
# 실행을 마치면 Session을 닫읍시다.  
# Close the Session when we're done.  
sess.close()
```

그래프(Graph)

```
g_1 = tf.Graph()
with g_1.as_default():
    # Operations created in this scope will be
    c = tf.constant("Node in g_1")

    # Sessions created in this scope will run on g_1
    sess_1 = tf.Session()

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

    # Alternatively, you can pass a graph when creating a session
    # `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
```

```
with tf.Graph().as_default() as g:
    hp = tf.contrib.training.HParams(
        learning_rate=0.005,
        max_steps=500,
    )
    train_ds = setup_mnist_data(True, 50)
    test_ds = setup_mnist_data(False, 1000)
    (train_losses, test_losses, train_accuracies,
     test_accuracies) = train(train_ds, test_ds, hp)

    init = tf.global_variables_initializer()

    with tf.Session(graph=g) as sess:
        sess.run(init)
        (train_losses, test_losses, train_accuracies,
         test_accuracies) = sess.run([train_losses, test_losses, train_accuracies,
                                       test_accuracies])

plt.title('MNIST train/test losses')
plt.plot(train_losses, label='train loss')
plt.plot(test_losses, label='test loss')
plt.legend()
plt.xlabel('Training step')
plt.ylabel('Loss')
plt.show()

plt.title('MNIST train/test accuracies')
plt.plot(train_accuracies, label='train accuracy')
plt.plot(test_accuracies, label='test accuracy')
plt.legend(loc='lower right')
plt.xlabel('Training step')
plt.ylabel('Accuracy')
plt.show()
```

기본은 끝났다.
본격적으로 짜보자!

오늘의 목표

- ▶ 소프트 맥스 회귀란?
- ▶ 크로스엔트로피 손실함수란?
- ▶ Mnist란?
- ▶ Mnist의 분류학습 프로그램 이해하기.

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))
```

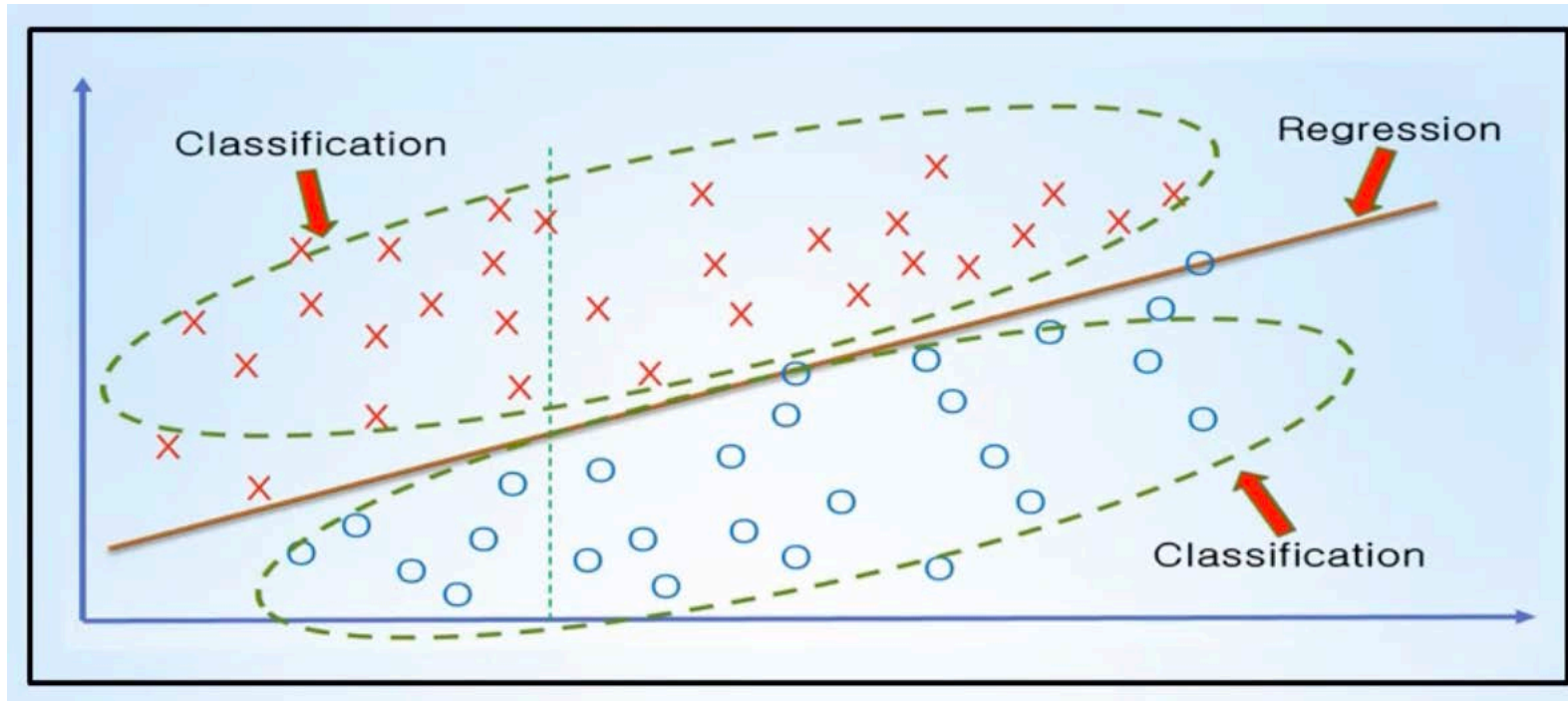
```
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

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

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

Regression(회귀) 와 Classification(분류)

- ▶ 최적의 직선을 회귀로 찾고, 이를 기준으로 분류를 함
 - ▶ 아래는 logistic regression의 예시



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

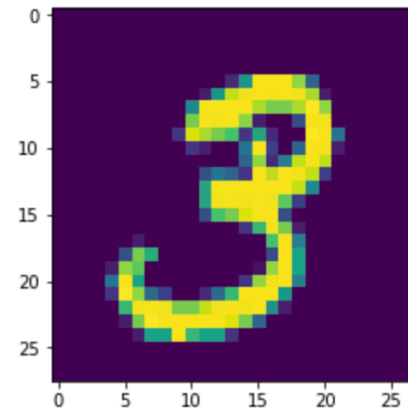
5 rows × 784 columns

Mnist dataset 파악하기 - 그림으로

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

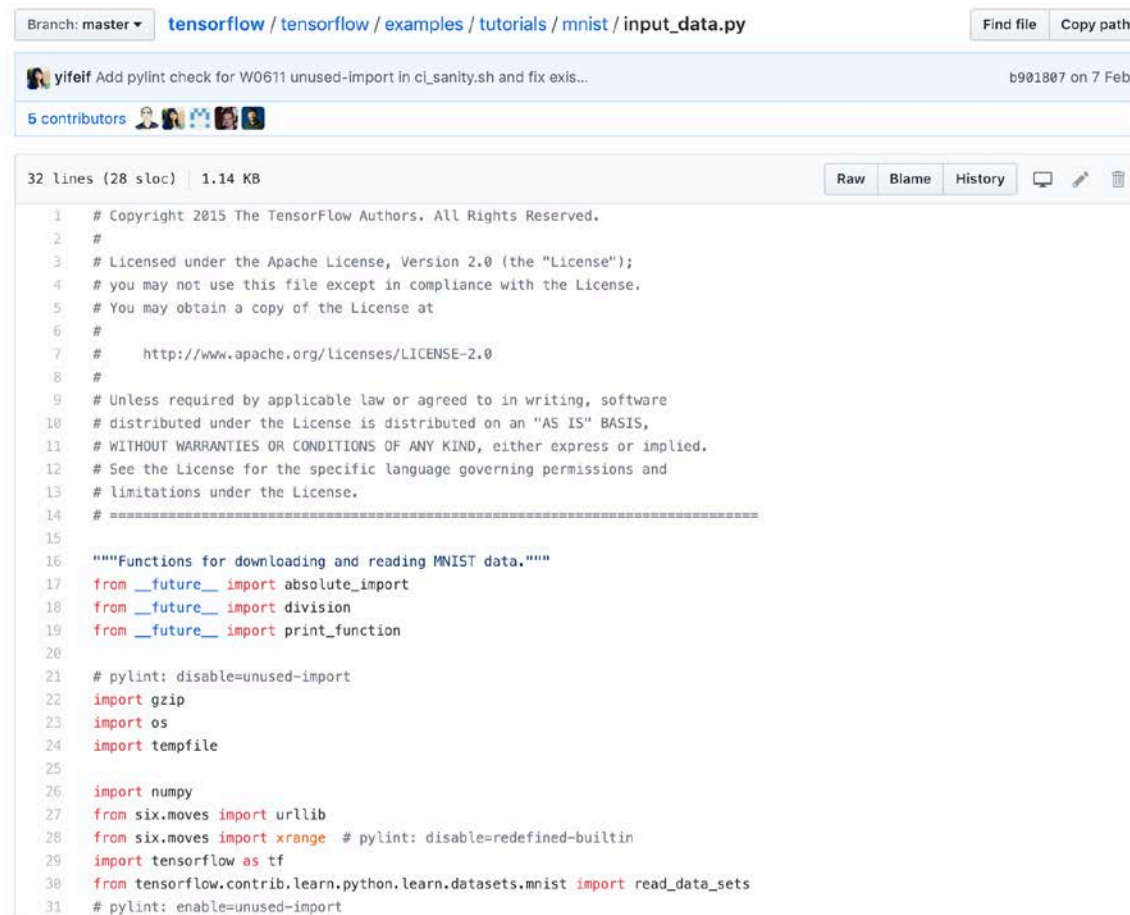
mnist = input_data.read_data_sets("./samples/MNIST_data/", one_hot=True)
arr= np.array(mnist.train.images[1])
arr.shape = (28,28) # 784 = 28 X 28 이다
# imshow는 2차원 자료의 크기를 색깔로 표시
plt.imshow(arr)
```

<matplotlib.image.AxesImage at 0xb43669e48>



프로그램 짜보기 - 1. Dataset loading

import tensorflow as tf
from tensorflow.examples.



```
Branch: master tensorflow / tensorflow / examples / tutorials / mnist / input_data.py Find file Copy path

yifeif Add pylint check for W0611 unused-import in ci_sanitiy.sh and fix exis... b901807 on 7 Feb
5 contributors

32 lines (28 sloc) 1.14 KB Raw Blame History

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]))
```

#W 는 784차원의 이미지 벡터를 곱해서 각 클래스에 대한 가중치를 나타내는 10차원 벡터

#b 는 10차원 벡터에 더하기 위한 편향.

#위의 값 두개는 학습해 나갈것이므로 초기값을 0 으로 해도 무관하다고 함.

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 답을 넣기 위한 새로운 placeholder

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

교차 엔트로피 $-\sum y' \log(y)$ 를 구현

```
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)
```

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.argmax는 특정한 축을 따라 가장 큰 원소의 색인을 알려줌

즉, 훈련데이터 y가 실제 데이터y_와 같은지 equal로 비교. 같으면 true, 다르면 false

correct_prediction는 정답여부를 true, false로 가지고 있음. Cast는 이를 1, 0 으로 바꿔줌.

reduce_mean은 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

결과 분석.

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