

TensorFlow Tutorial

전현호

Contents

1. What is TensorFlow?
2. **Install** TensorFlow
3. **Data type**
4. Linear regression + Plot
5. MNIST
6. Multi Layer Perceptron (NN, FCN)
7. Convolutional NN

What is TensorFlow?



다차원 배열

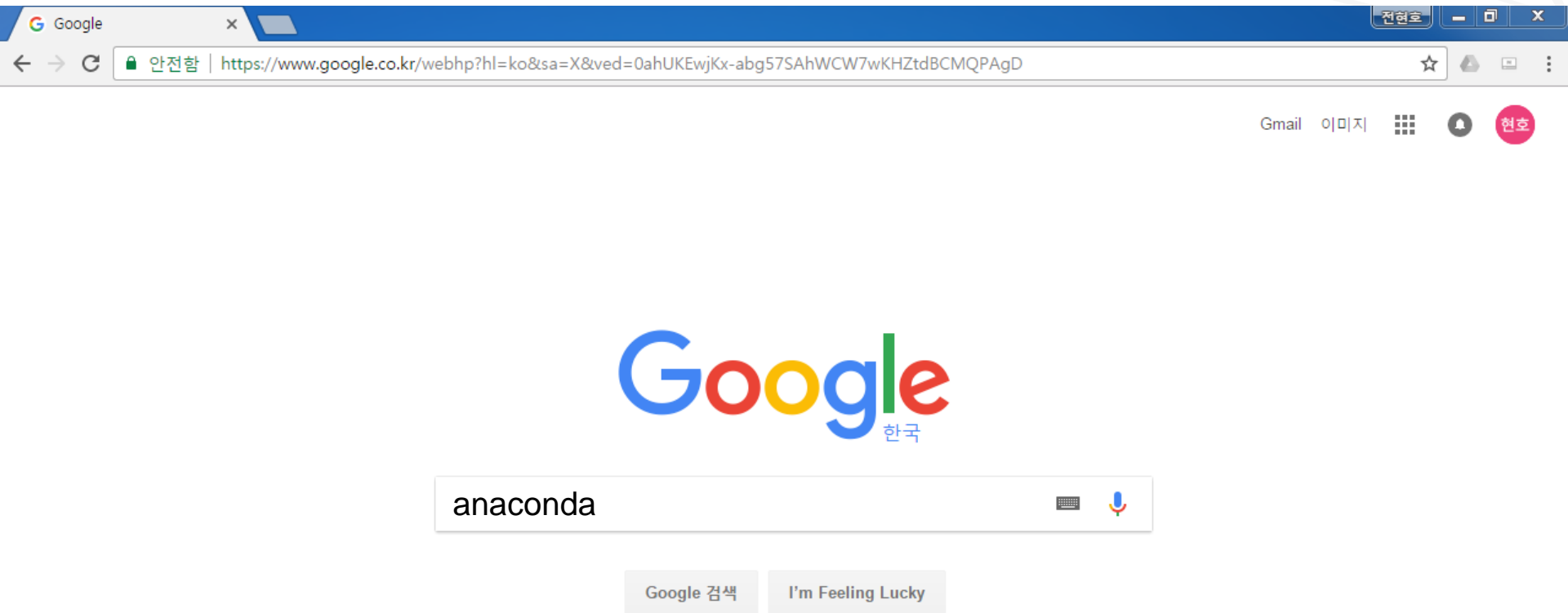
- 텐서플로우(TensorFlow) : 구글에서 개발한 기계 학습 및 딥러닝용 오픈소스 라이브러리
- 왜 텐서플로우를 사용해야 하는가?
 - 구현이 쉬움 (연산, 신경망 구조)
 - 여러 CPU 및 GPU에서 동작이 가능함
 - 데이터 및 구조를 시각적으로 확인할 수 있음 (TensorBoard)
 - 기타 등등 (뒤에서 설명)

TensorFlow Dev Summit - 17.02.16

- 텐서플로우 1.0 공식 버전 릴리즈
- 빨라짐 : 8GPUs + inception v3 모델을 사용할 경우 7.3배 속도 향상
64GPUs + Inception v3 모델을 사용할 경우 58배 속도 향상
- 유연해짐 : 더 고급 수준의 API 추가 및 딥러닝 라이브러리중 하나인 Keras와 완벽한 호환성을 제공하는 새로운 tf.keras 모듈이 추가됨
- 기타 등등
- Python `tf_upgrade.py --infile InputFile --outfile OutputFile`
- `pip install tensorflow`로 대부분 환경에서 바로 업데이트 가능

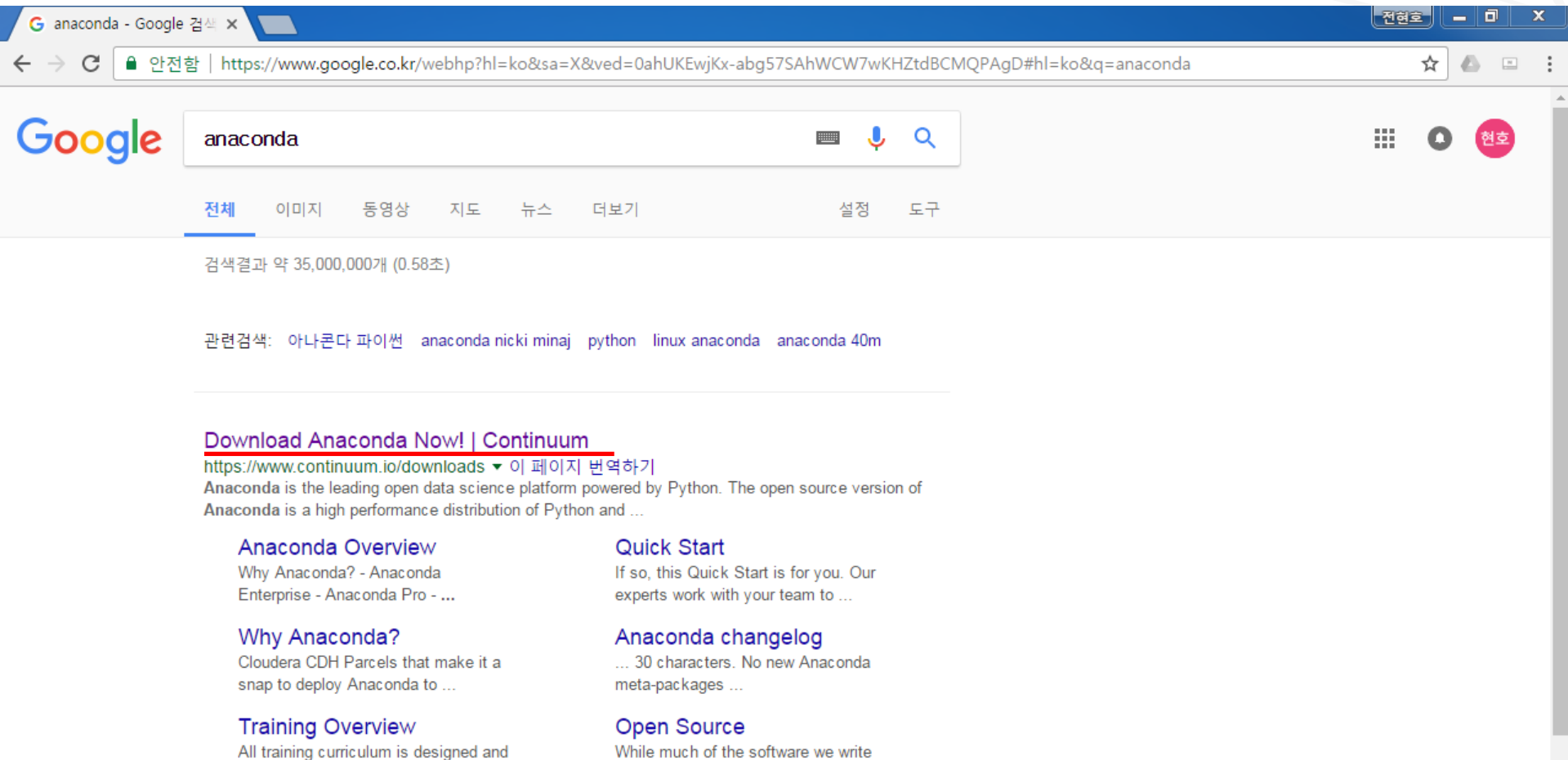
Install TensorFlow

1. Anaconda3 4.2.0 버전 설치 (Python 3.5)



Install TensorFlow

1. Anaconda3 4.2.0 버전 설치 (Python 3.5)



The screenshot shows a Google search for 'anaconda'. The search results include a link to 'Download Anaconda Now! | Continuum' with a URL: <https://www.continuum.io/downloads>. Below the link, there is a description: 'Anaconda is the leading open data science platform powered by Python. The open source version of Anaconda is a high performance distribution of Python and ...'. There are also several related links: 'Anaconda Overview', 'Quick Start', 'Why Anaconda?', 'Anaconda changelog', and 'Training Overview'.

anaconda - Google 검색 x

← → ↻ 안전함 | <https://www.google.co.kr/webhp?hl=ko&sa=X&ved=0ahUKEwjKx-abg57SAhWCW7wKHZtdBCMQPAGD#hl=ko&q=anaconda> ☆

Google anaconda

전체 이미지 동영상 지도 뉴스 더보기 설정 도구

검색결과 약 35,000,000개 (0.58초)

관련검색: [아나콘다 파이썬](#) [anaconda nicki minaj](#) [python](#) [linux anaconda](#) [anaconda 40m](#)

[Download Anaconda Now! | Continuum](https://www.continuum.io/downloads)
<https://www.continuum.io/downloads> ▾ 이 페이지 번역하기

Anaconda is the leading open data science platform powered by Python. The open source version of Anaconda is a high performance distribution of Python and ...

[Anaconda Overview](#)
Why Anaconda? - Anaconda
Enterprise - Anaconda Pro - ...

[Quick Start](#)
If so, this Quick Start is for you. Our experts work with your team to ...

[Why Anaconda?](#)
Cloudera CDH Parcels that make it a snap to deploy Anaconda to ...

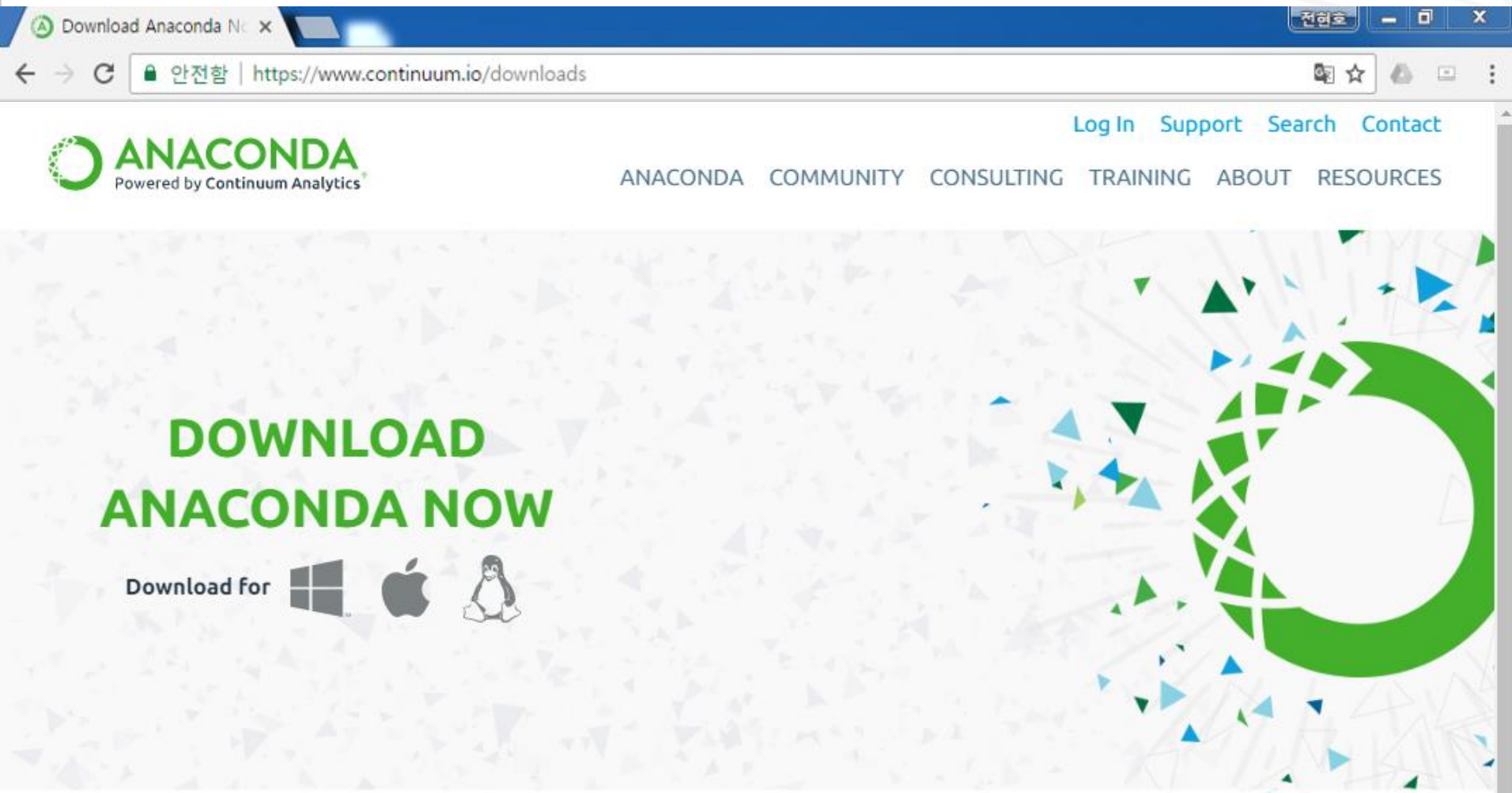
[Anaconda changelog](#)
... 30 characters. No new Anaconda meta-packages ...

[Training Overview](#)
All training curriculum is designed and

[Open Source](#)
While much of the software we write

Install TensorFlow

1. Anaconda3 4.2.0 버전 설치 (Python 3.5)



The image shows a browser window displaying the Anaconda website. The browser's address bar shows the URL <https://www.continuum.io/downloads>. The website header includes the Anaconda logo (a green circle with a white 'C') and the text "ANACONDA Powered by Continuum Analytics". Navigation links for "Log In", "Support", "Search", and "Contact" are visible in the top right. A secondary navigation menu contains "ANACONDA", "COMMUNITY", "CONSULTING", "TRAINING", "ABOUT", and "RESOURCES". The main content area features a large green call to action: "DOWNLOAD ANACONDA NOW". Below this, it says "Download for" followed by icons for Windows, Apple, and Linux (Tux penguin). The background is a light gray with a pattern of small triangles and a large green circular graphic on the right side.

Download Anaconda Now x 전업호




← → ↻ 안전함 | <https://www.continuum.io/downloads> ☆ 🔍 📄 ☰

ANACONDA Powered by Continuum Analytics

[Log In](#) [Support](#) [Search](#) [Contact](#)

[ANACONDA](#) [COMMUNITY](#) [CONSULTING](#) [TRAINING](#) [ABOUT](#) [RESOURCES](#)

DOWNLOAD ANACONDA NOW

Download for   

Install TensorFlow

1. Anaconda3 4.2.0 버전 설치 (Python 3.5)

Download Anaconda No x Anaconda installer archi x

안전함 | https://repo.continuum.io/archive/index.html

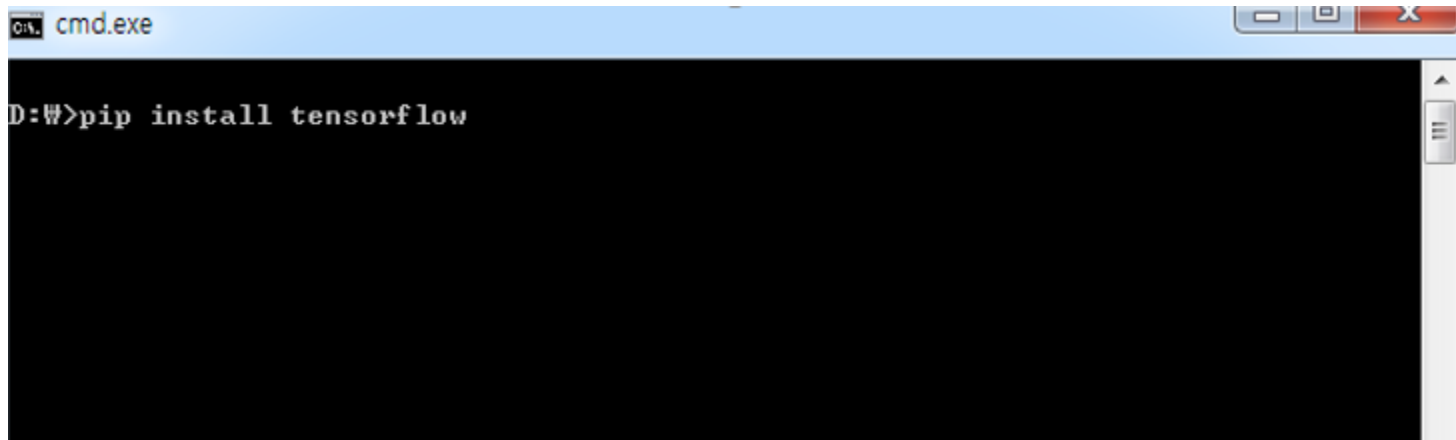
Anaconda installer archive

Filename	Size	Last Modified	MD5
Anaconda2-4.3.0.1-#indows-x86.exe	338.1M	2017-02-02 14:19:05	4bf17044ecf0229a0974ba8429520cad
Anaconda2-4.3.0.1-#indows-x86_64.exe	412.8M	2017-02-02 14:20:08	56b181af1959de40e67fb5ef50612ae2
Anaconda3-4.3.0.1-#indows-x86.exe	347.2M	2017-02-02 14:21:02	5dd0a8b09a5eb6c9d002dc26d6f31492
Anaconda3-4.3.0.1-#indows-x86_64.exe	421.2M	2017-02-02 14:22:10	07ea8c5a2306ac8fabf3902bd6623787
Anaconda2-4.3.0-Linux-x86.sh	386.8M	2017-01-27 14:14:15	65546028c4a48f4bb582c4ee3e43b893
Anaconda2-4.3.0-Linux-x86_64.sh	461.1M	2017-01-27 14:15:08	5f2c3bd60ddb0e213f7a1fc25b88b4
Anaconda2-4.3.0-MacOSX-x86_64.pkg	418.4M	2017-01-27 14:26:23	899e90455db3120d584b2d4961c4eede
Anaconda2-4.3.0-MacOSX-x86_64.sh	357.3M	2017-01-27 14:26:08	80b7958fc805d371d60e133af826752c
Anaconda2-4.3.0-#indows-x86.exe	338.1M	2017-01-27 14:17:06	ffd6296dc4b359684c54ce6f3d10e144
Anaconda2-4.3.0-#indows-x86_64.exe	412.8M	2017-01-27 14:17:59	2c02e21e542d61760c3e19b10b3086fe
Anaconda3-4.3.0-Linux-x86.sh	398.4M	2017-01-27 14:14:29	3f1173aa1ab2c2b6ab3f8a6bd22827fd7
Anaconda3-4.3.0-Linux-x86_64.sh	473.4M	2017-01-27 14:15:21	dbe2e78adeca1923643be2ecaacd6227
Anaconda3-4.3.0-MacOSX-x86_64.pkg	423.1M	2017-01-27 14:26:32	30b108a9c5d215a60187c5de89c459
Anaconda3-4.3.0-MacOSX-x86_64.sh	362.6M	2017-01-27 14:26:15	e080c503c27d5c072d3e324ee1822641
Anaconda3-4.3.0-#indows-x86.exe	347.2M	2017-01-27 14:18:45	ae7ec9752cf81c01983fc10ddfd7cc2
Anaconda3-4.3.0-#indows-x86_64.exe	421.2M	2017-01-27 14:19:41	137043b3f9860519967759fc8ea76514
Anaconda2-4.2.0-MacOSX-x86_64.pkg	409.9M	2016-10-17 19:33:11	cd2ccc991b7f11503335367d80d0317b0
Anaconda3-4.2.0-MacOSX-x86_64.pkg	407.1M	2016-10-17 19:33:47	51ed7f9af7436a1a23068eb00509d6ad
Anaconda2-4.2.0-Linux-x86.sh	365.0M	2016-09-27 15:50:20	e26582abd11d982e18efb2bdf52c5ee6
Anaconda2-4.2.0-Linux-x86_64.sh	446.0M	2016-09-27 15:49:54	a0d1f8e47014b71c6764d76fb403f217
Anaconda2-4.2.0-MacOSX-x86_64.sh	346.4M	2016-09-27 15:50:02	52f8b74e0c462575efc297c8f4e6c1f4
Anaconda2-4.2.0-#indows-x86.exe	324.1M	2016-09-27 15:54:50	f4f12af8811759e56464eef5a484963d
Anaconda2-4.2.0-#indows-x86_64.exe	381.0M	2016-09-27 15:55:47	0a30d509568724dac0ae193e139b9c37
Anaconda3-4.2.0-Linux-x86.sh	373.9M	2016-09-27 15:50:34	7acal0e1ea5b9db0a318b4eed5253747
Anaconda3-4.2.0-Linux-x86_64.sh	455.9M	2016-09-27 15:50:04	4692f716c82deb9fa6b59d78f9f6e85c
Anaconda3-4.2.0-MacOSX-x86_64.sh	349.5M	2016-09-27 15:50:07	7cb61e395eb860e342a5e27236e3f375
Anaconda3-4.2.0-#indows-x86.exe	333.4M	2016-09-27 15:56:30	96e5fe052b22d667da9360fb4edce363
Anaconda3-4.2.0-#indows-x86_64.exe	391.4M	2016-09-27 15:57:21	0ca5ef4dcfe84376aad073bb3f8db00

Install TensorFlow

2. TensorFlow 설치 – CPU 버전

- cmd창에 pip install tensorflow 입력

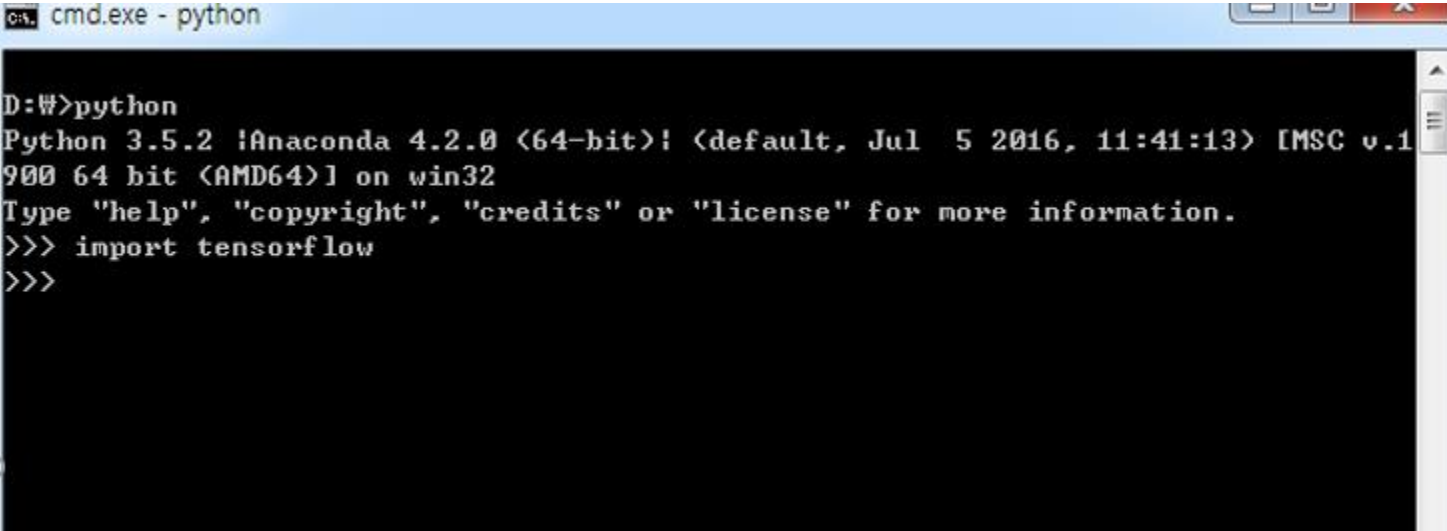


```
cmd.exe
D:\#>pip install tensorflow
```

Install TensorFlow

3. 설치 확인

- Python을 실행시킨 후 import tensorflow를 통해 에러가 나는지 확인



```
cmd.exe - python
D:\>python
Python 3.5.2 [Anaconda 4.2.0 (64-bit): (default, Jul 5 2016, 11:41:13) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow
>>>
```

Install TensorFlow

- GPU 버전 설치 과정

- CUDA 8.0 버전 설치 (<https://developer.nvidia.com/cuda-downloads>)

NVIDIA ACCELERATED COMPUTING Downloads Training Ecosystem Forums Register Now Log in

NVIDIA DGX-1

THE WORLD'S FIRST DEEP LEARNING SUPERCOMPUTER IN A BOX

LEARN MORE

Home > ComputeWorks > CUDA ZONE > Tools & Ecosystem > CUDA Toolkit > CUDA 8.0 Downloads

Learn more about CUDA Toolkit 8.0:

- Read the **Introduction to CUDA C and C++** Parallel Forall Blog Post.
- Read the **CUDA 8 Features Revealed** Parallel Forall Blog Post.
- Review the **What's New in CUDA 8** webinar.
- Review the CUDA 8 Performance Overview **Webinar, Slides**

IBM Power8 Users: Download CUDA 8 from the **CUDA Toolkit for IBM Power 8** Page.

For Linux users upgrading from previous versions of the CUDA Toolkit, click to see instructions in this section before proceeding.

Select Target Platform

Click on the green buttons that describe your target platform. Only supported platforms will be shown.

Operating System **Windows** Linux Mac OS X

Related Links

- CUDA Quick Start Guide
- Release Notes
- EULA

Install TensorFlow

2. cuDNN 5.1 버전 설치 (<https://developer.nvidia.com/cudnn>)
간단한 등록 과정을 요구함

The screenshot shows the NVIDIA Accelerated Computing website for cuDNN. The header includes the NVIDIA logo and navigation links for Downloads, Training, Ecosystem, and Forums. A search bar and links for Register Now and Log in are also present. The main content area features the title 'NVIDIA cuDNN' and the subtitle 'GPU Accelerated Deep Learning'. A breadcrumb trail reads: Home > ComputeWorks > Deep Learning > Software > NVIDIA cuDNN. The main text describes cuDNN as a GPU-accelerated library of primitives for deep neural networks, listing standard routines like convolution, pooling, normalization, and activation layers. It mentions that cuDNN is part of the NVIDIA Deep Learning SDK. A paragraph explains that deep learning researchers and framework developers worldwide rely on cuDNN for high-performance GPU acceleration, allowing them to focus on training neural networks and developing software applications rather than spending time on low-level GPU performance tuning. It lists supported frameworks including Caffe, TensorFlow, Theano, Torch, and CNTK. Below the text are two buttons: 'Register' and 'Download'. A note states that cuDNN is freely available to members of the Accelerated Computing Developer Program. At the bottom, there is a section titled 'What's New in cuDNN 5' with a sub-header 'Data scientists and researchers can take advantage of cuDNN by downloading a Deep Learning frameworks or NVIDIA DIGITS. DIGITS lets you interactively manage data, perform training on multiple GPUs, and export the best performing model for deployment without the need to write code.'

QUICKLINKS

- Accelerated Computing - Training
- CUDA GPUs
- Tools & Ecosystem
- OpenACC: More Science Less Programming
- CUDA FAQ

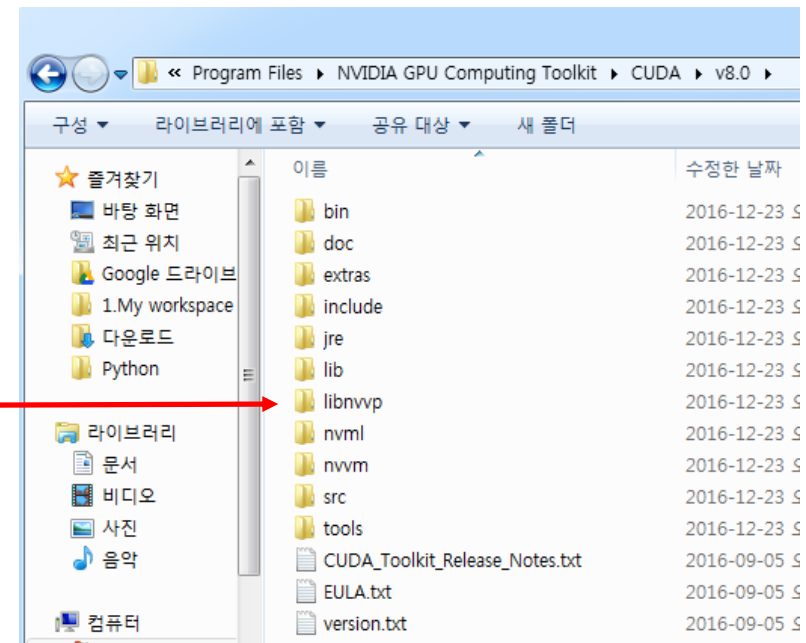
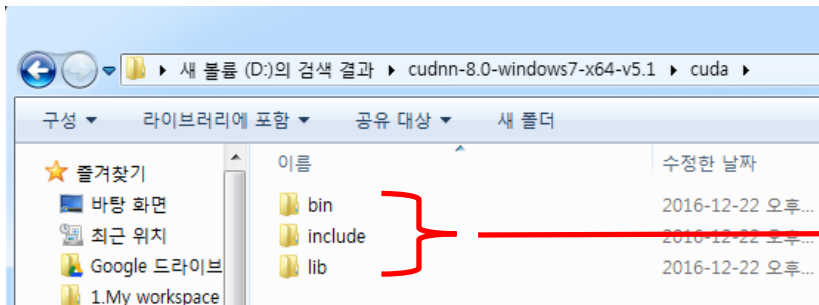
GPU Computing [Follow](#)

CUDA, GPU Computing @GPUComputing

Congrats to the @TensorFlow team on their launch at #TFdevsummit. Learn more about @Google's latest announcement [nvida.ws/2kGGLGuw](https://nvidia.ws/2kGGLGuw)

Install TensorFlow

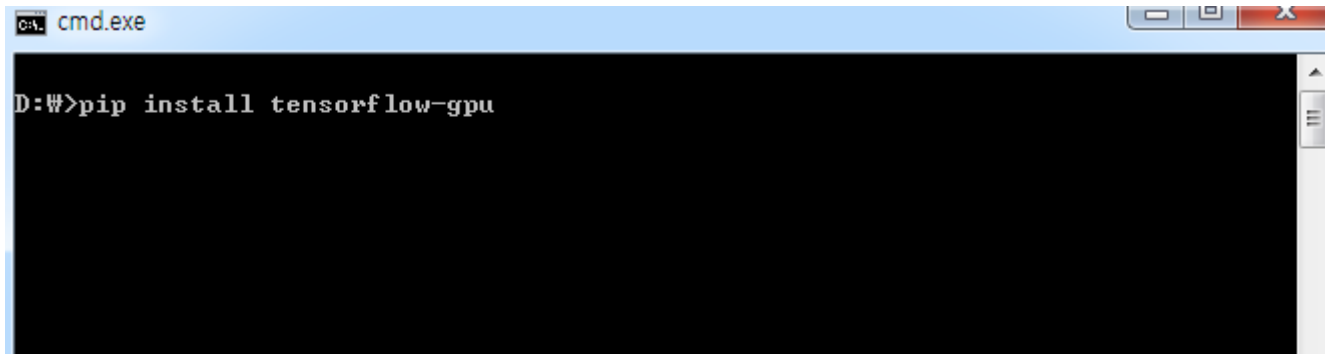
3. cuDNN 파일의 압축풀기
4. 결과로 나오는 bin, include, lib 폴더를 CUDA가 설치된 경로로 접근하여 붙여넣기



Install TensorFlow

5. Tensorflow 설치 - GPU 버전

- cmd창에 pip install tensorflow-gpu 입력
- CPU버전과 마찬가지로 설치가 잘 되었는지 확인



```
cmd.exe
D: #> pip install tensorflow-gpu
```

Install TensorFlow

```
관리자: cmd.exe - python.exe

D:\>cd ProgramFiles

D:\ProgramFiles>cd Python

D:\ProgramFiles\Python>cd Python35

D:\ProgramFiles\Python\Python35>python.exe
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:18:55) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow
I c:\wtf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\stream_executor\dso_loader.cc:128] successfully opened CUDA library cublas64_80.dll locally
I c:\wtf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\stream_executor\dso_loader.cc:128] successfully opened CUDA library cudnn64_5.dll locally
I c:\wtf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\stream_executor\dso_loader.cc:128] successfully opened CUDA library cufft64_80.dll locally
I c:\wtf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\stream_executor\dso_loader.cc:128] successfully opened CUDA library nvcuda.dll locally
I c:\wtf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\stream_executor\dso_loader.cc:128] successfully opened CUDA library curand64_80.dll locally
>>>
```

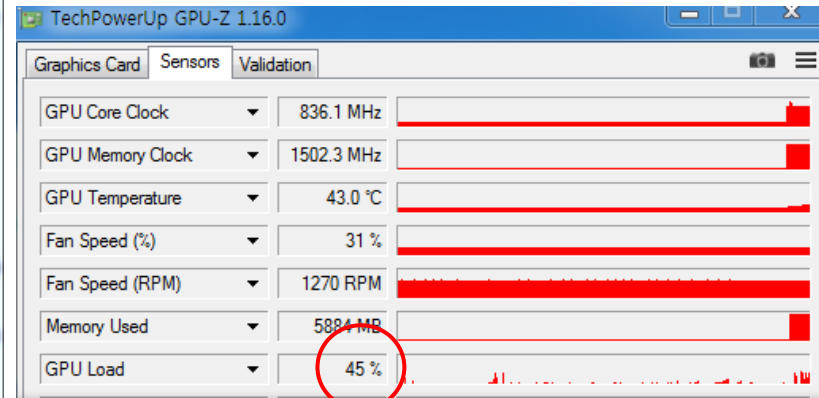
6. quit()을 통해 python 종료

Install TensorFlow

- MNIST 예제를 학습시켜 동작 확인 (3 hidden layer)

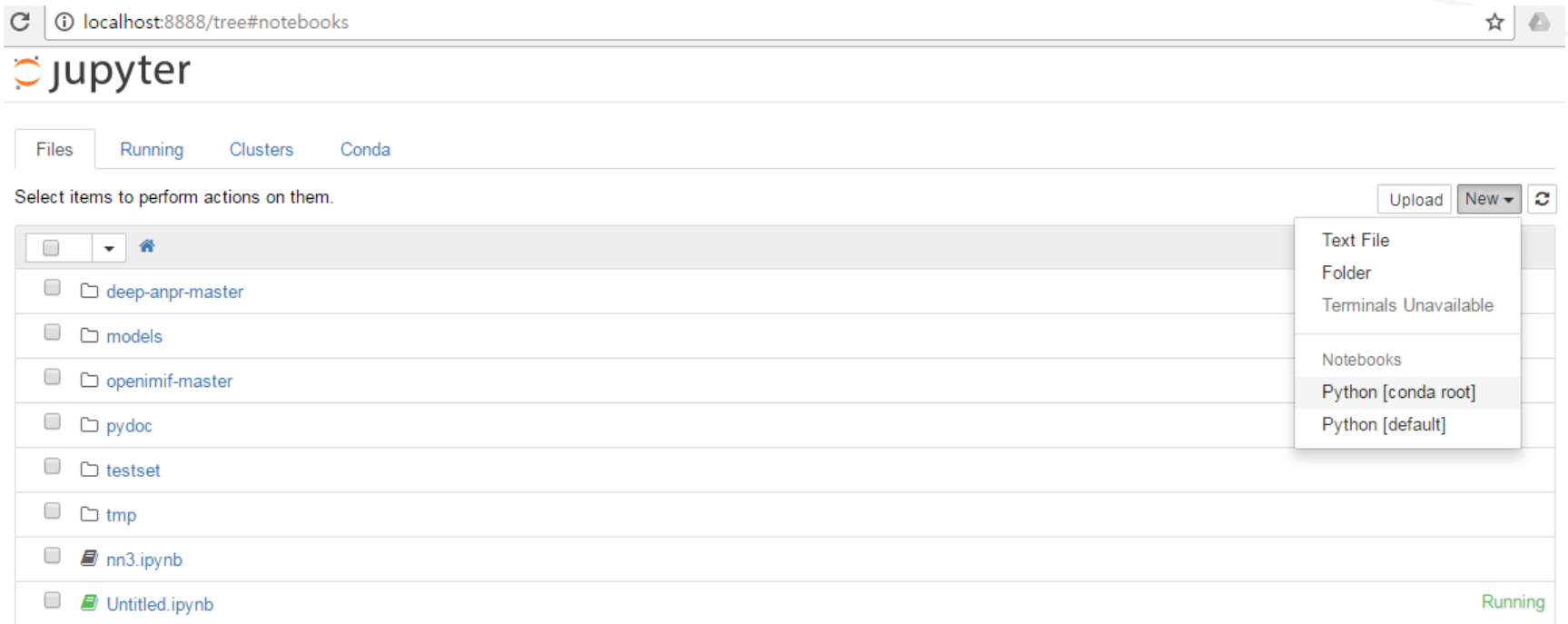
```
c:\ 관리자: cmd.exe
major: 3 minor: 5 memoryClockRate (GHz) 0.8755
pciBusID 0000:01:00.0
Total memory: 6.00GiB
Free memory: 5.70GiB
I c:\Wtf_jenkins\home\workspace\wrelease-win\device\gpu\os\windows\tensorflow\core
\Wcommon_runtime\gpu\gpu_device.cc:906 l DMA: 0
I c:\Wtf_jenkins\home\workspace\wrelease-win\device\gpu\os\windows\tensorflow\core
\Wcommon_runtime\gpu\gpu_device.cc:916 l 0: y
I c:\Wtf_jenkins\home\workspace\wrelease-win\device\gpu\os\windows\tensorflow\core
\Wcommon_runtime\gpu\gpu_device.cc:975 l Creating TensorFlow device (</gpu:0> -> <d
evice: 0, name: GeForce GTX TITAN, pci bus id: 0000:01:00.0)
WARNING:tensorflow:From test.py:49 in train_neural_network.: initialize_all_vari
ables (from tensorflow.python.ops.variables) is deprecated and will be removed a
fter 2017-03-02.
Instructions for updating:
Use `tf.global_variables_initializer` instead.
Epoch 0 completed out of 10 loss: 1847534.27691
Epoch 1 completed out of 10 loss: 386038.85005
Epoch 2 completed out of 10 loss: 208952.075657
Epoch 3 completed out of 10 loss: 120279.905643
Epoch 4 completed out of 10 loss: 70081.6126813
Epoch 5 completed out of 10 loss: 44156.2774823
Epoch 6 completed out of 10 loss: 26902.880532
Epoch 7 completed out of 10 loss: 24727.0853252
Epoch 8 completed out of 10 loss: 19213.9519197
Epoch 9 completed out of 10 loss: 15453.5376992
Accuracy: 0.9516

D:\ProgramFiles\Python>
```



Data type

- cmd - jupyter notebook

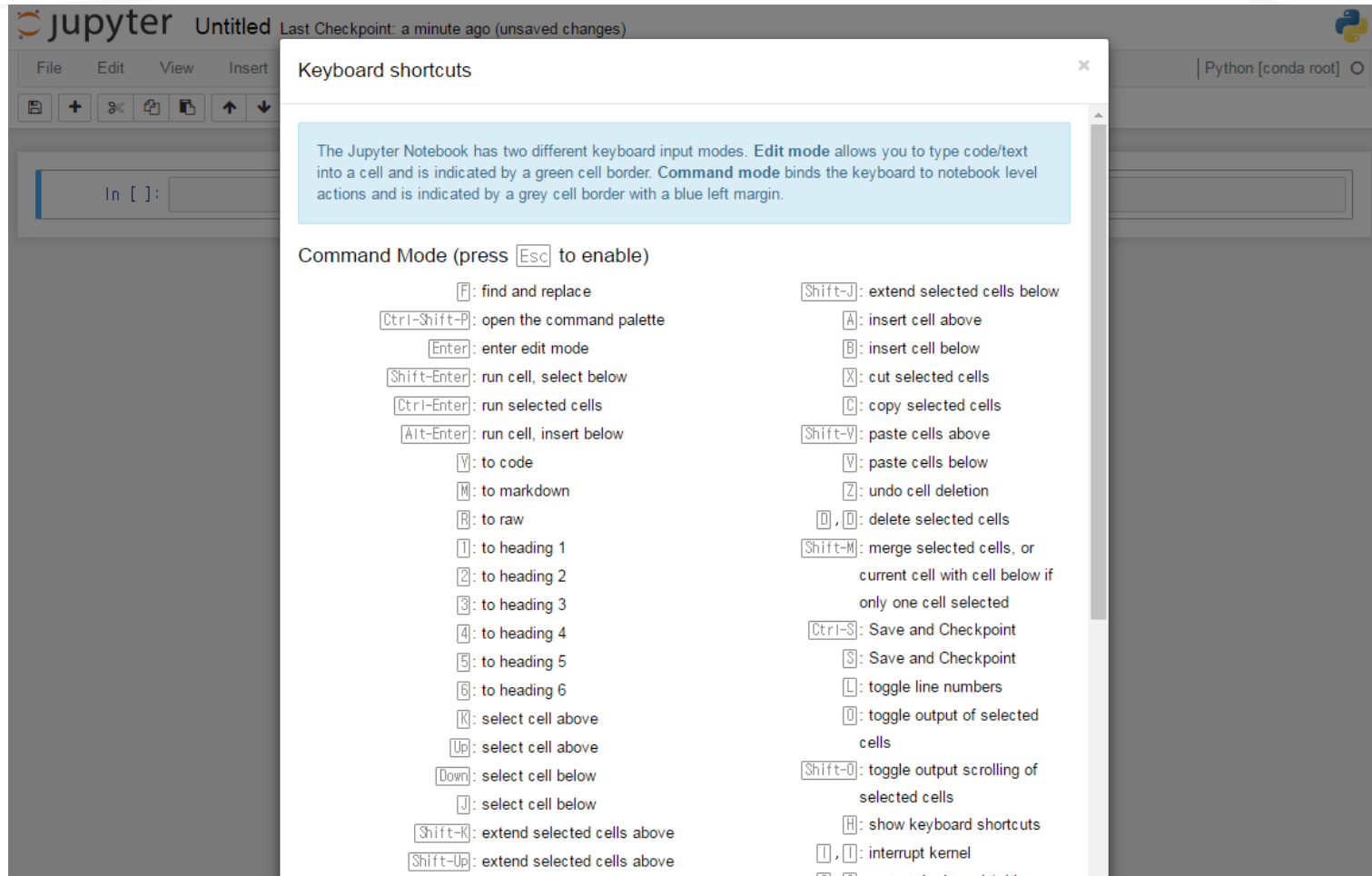


The screenshot shows the Jupyter Notebook web interface in a browser. The address bar displays 'localhost:8888/tree#notebooks'. The page title is 'jupyter'. Below the title, there are tabs for 'Files', 'Running', 'Clusters', and 'Conda'. The 'Files' tab is active, showing a file browser with a list of files and folders: 'deep-anpr-master', 'models', 'openimif-master', 'pydoc', 'testset', 'tmp', 'nn3.ipynb', and 'Untitled.ipynb'. A 'New' dropdown menu is open, showing options: 'Text File', 'Folder', 'Terminals Unavailable', 'Notebooks', 'Python [conda root]', and 'Python [default]'. The 'Python [conda root]' option is highlighted. The 'Untitled.ipynb' file is marked as 'Running'.

※ 주의 : 경로상에 관련 파일이 있지 않을 경우 404 에러 발생

Data type

- jupyter notebook - h



The screenshot shows the Jupyter Notebook interface with a 'Keyboard shortcuts' dialog box open. The dialog box contains the following text:

The Jupyter Notebook has two different keyboard input modes. **Edit mode** allows you to type code/text into a cell and is indicated by a green cell border. **Command mode** binds the keyboard to notebook level actions and is indicated by a grey cell border with a blue left margin.

Command Mode (press `Esc` to enable)

<code>F</code> : find and replace	<code>Shift-J</code> : extend selected cells below
<code>Ctrl-Shift-P</code> : open the command palette	<code>A</code> : insert cell above
<code>Enter</code> : enter edit mode	<code>B</code> : insert cell below
<code>Shift-Enter</code> : run cell, select below	<code>X</code> : cut selected cells
<code>Ctrl-Enter</code> : run selected cells	<code>C</code> : copy selected cells
<code>Alt-Enter</code> : run cell, insert below	<code>Shift-V</code> : paste cells above
<code>V</code> : to code	<code>V</code> : paste cells below
<code>M</code> : to markdown	<code>Z</code> : undo cell deletion
<code>R</code> : to raw	<code>D, D</code> : delete selected cells
<code>1</code> : to heading 1	<code>Shift-M</code> : merge selected cells, or current cell with cell below if only one cell selected
<code>2</code> : to heading 2	<code>Ctrl-S</code> : Save and Checkpoint
<code>3</code> : to heading 3	<code>S</code> : Save and Checkpoint
<code>4</code> : to heading 4	<code>L</code> : toggle line numbers
<code>5</code> : to heading 5	<code>O</code> : toggle output of selected cells
<code>6</code> : to heading 6	<code>Shift-O</code> : toggle output scrolling of selected cells
<code>K</code> : select cell above	<code>H</code> : show keyboard shortcuts
<code>Up</code> : select cell above	<code>I, I</code> : interrupt kernel
<code>Down</code> : select cell below	<code>Ctrl-C</code> : copy selected cells
<code>J</code> : select cell below	<code>Ctrl-V</code> : paste selected cells
<code>Shift-K</code> : extend selected cells above	<code>Shift-C</code> : copy selected cells
<code>Shift-Up</code> : extend selected cells above	<code>Shift-V</code> : paste selected cells

Data type

✂ https://www.tensorflow.org/versions/r0.12/api_docs/python/

Inputs and Readers > Placeholders

Members	
<code>tf.placeholder(dtype, shape=None, name=None)</code>	Inserts a placeholder for a tensor that will be always fed.

Variables > Variables

Members	
<code>class tf.Variable</code>	See the [Variables How To] (../how_tos/variables/index.md) for a high
<code>tf.Variable.__init__(initial_value=None, trainable=True, collections=None, validate_shape=True, caching_device=None, name=None, variable_def=None, dtype=None, expected_shape=None, import_scope=None)</code>	Creates a new variable with value <code>initial_value</code> .

Constants, Sequences, and Random Values > Constant Value Tensors

Members	
<code>tf.zeros(shape, dtype=tf.float32, name=None)</code>	Creates a tensor with all elements set to zero.
<code>tf.zeros_like(tensor, dtype=None, name=None, optimize=True)</code>	Creates a tensor with all elements set to zero.
<code>tf.ones(shape, dtype=tf.float32, name=None)</code>	Creates a tensor with all elements set to 1.
<code>tf.ones_like(tensor, dtype=None, name=None, optimize=True)</code>	Creates a tensor with all elements set to 1.
<code>tf.fill(dims, value, name=None)</code>	Creates a tensor filled with a scalar value.
<code>tf.constant(value, dtype=None, shape=None, name='Const', verify_shape=False)</code>	Creates a constant tensor.

Data type

- Code

```
In [1]: import tensorflow as tf
```

```
In [3]: ph = tf.placeholder(dtype = tf.float32, shape=[3,3])  
var = tf.Variable([1, 2, 3, 4, 5], dtype = tf.float32)  
const = tf.constant([10, 20, 30, 40, 50], dtype = tf.float32)
```

```
In [5]: print(ph)
```

```
Tensor("Placeholder_1:0", shape=(3, 3), dtype=float32)
```

```
In [6]: print(var)
```

```
Tensor("Variable/read:0", shape=(5,), dtype=float32)
```

```
In [7]: print(const)
```

```
Tensor("Const:0", shape=(5,), dtype=float32)
```

```
In [ ]:
```

Data type

- 그래프 생성

$$2(5t+4)+3(1+2)=?$$

Data type

- 그래프 생성

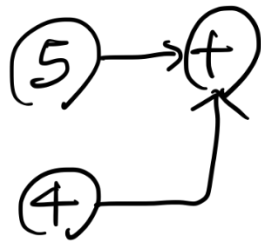
$$2(5+4)+3(1+2)=?$$

$$(5) \rightarrow (+)$$

Data type

- 그래프 생성

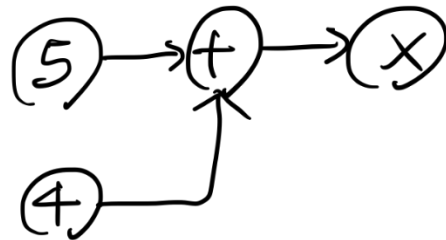
$$2(5+4)+3(1+2)=?$$



Data type

- 그래프 생성

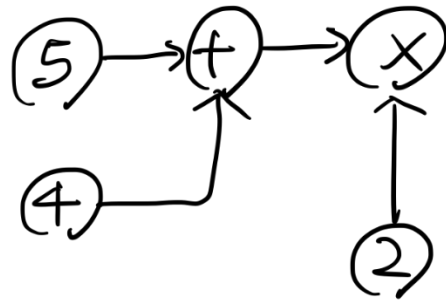
$$2(5+4)+3(1+2)=?$$



Data type

- 그래프 생성

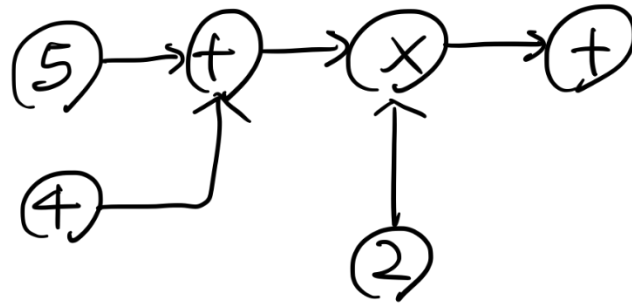
$$2(5+4)+3(1+2)=?$$



Data type

- 그래프 생성

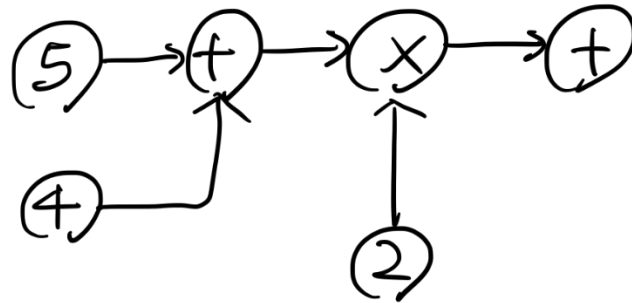
$$2(5+4)+3(1+2)=?$$



Data type

- 그래프 생성

$$2(5+4)+3(1+2)=?$$



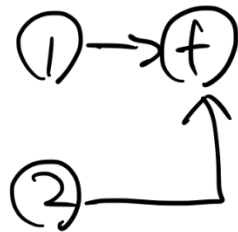
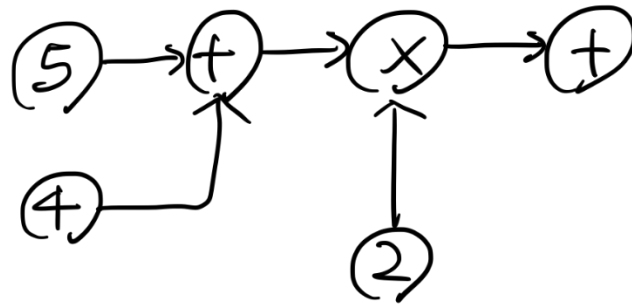
①

②

Data type

- 그래프 생성

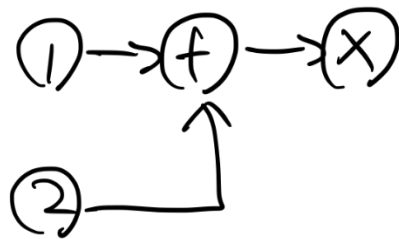
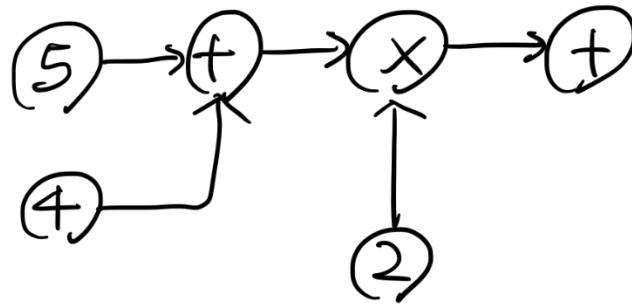
$$2(5+4)+3(1+2)=?$$



Data type

- 그래프 생성

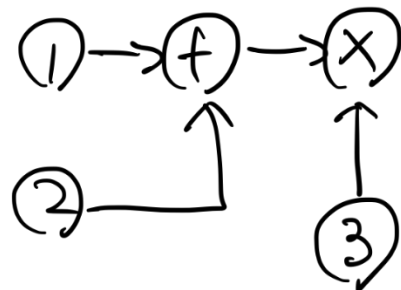
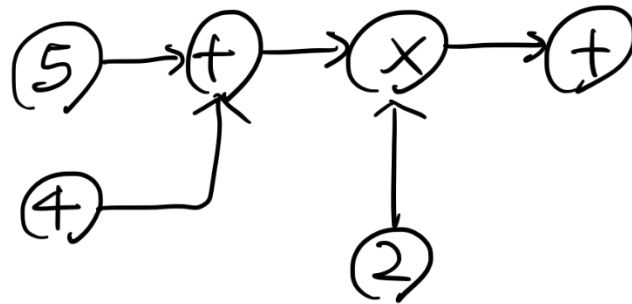
$$2(5+4)+3(1+2)=?$$



Data type

- 그래프 생성

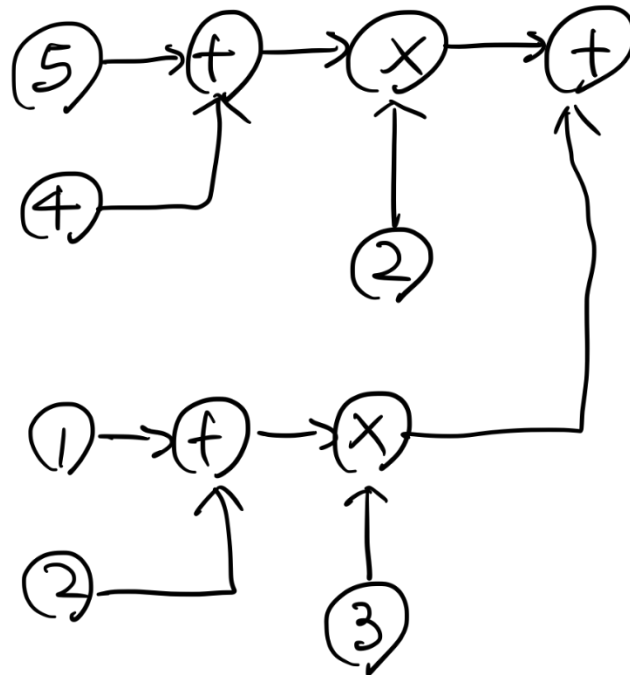
$$2(5+4)+3(1+2)=?$$



Data type

- 그래프 생성

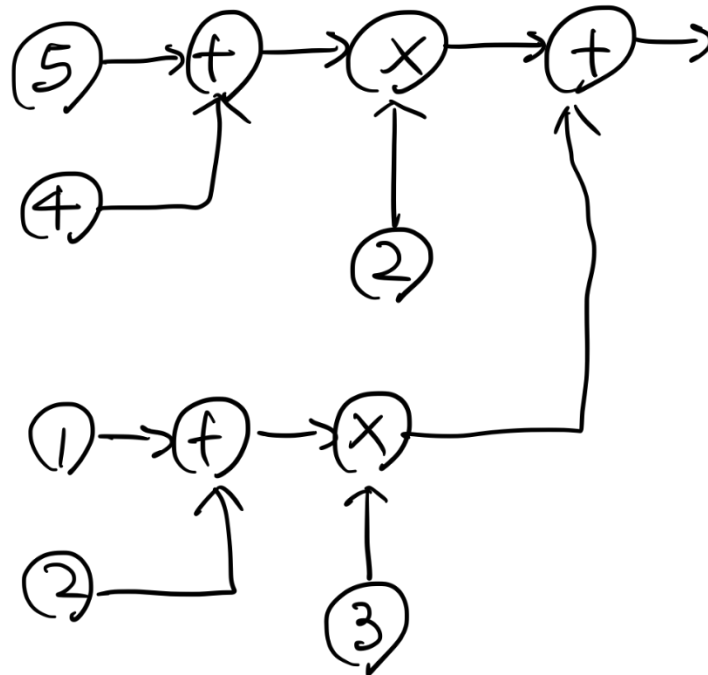
$$2(5+4)+3(1+2)=?$$



Data type

- 그래프 생성

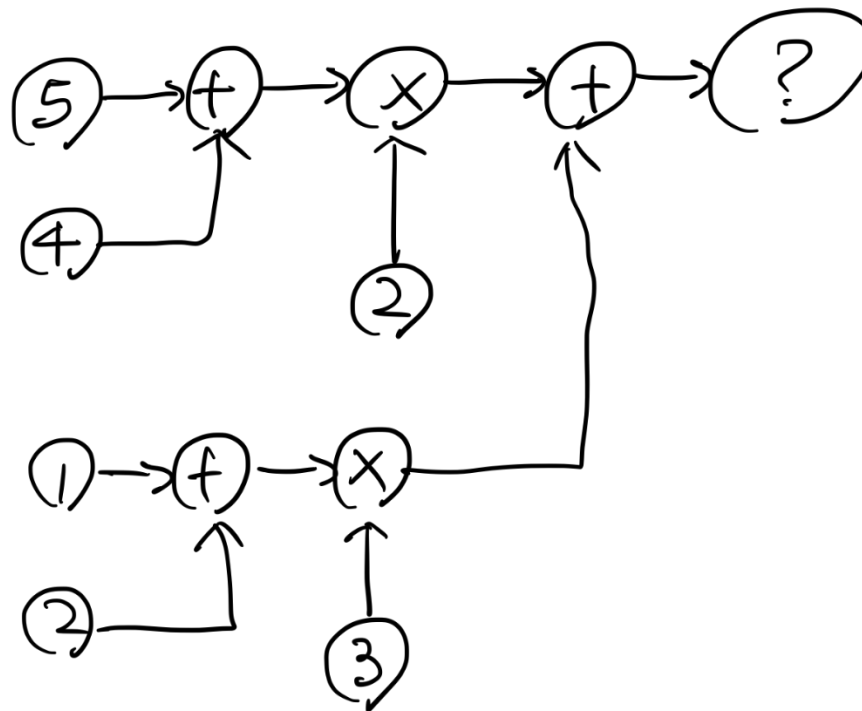
$$2(5+4)+3(1+2)=?$$



Data type

- 그래프 생성

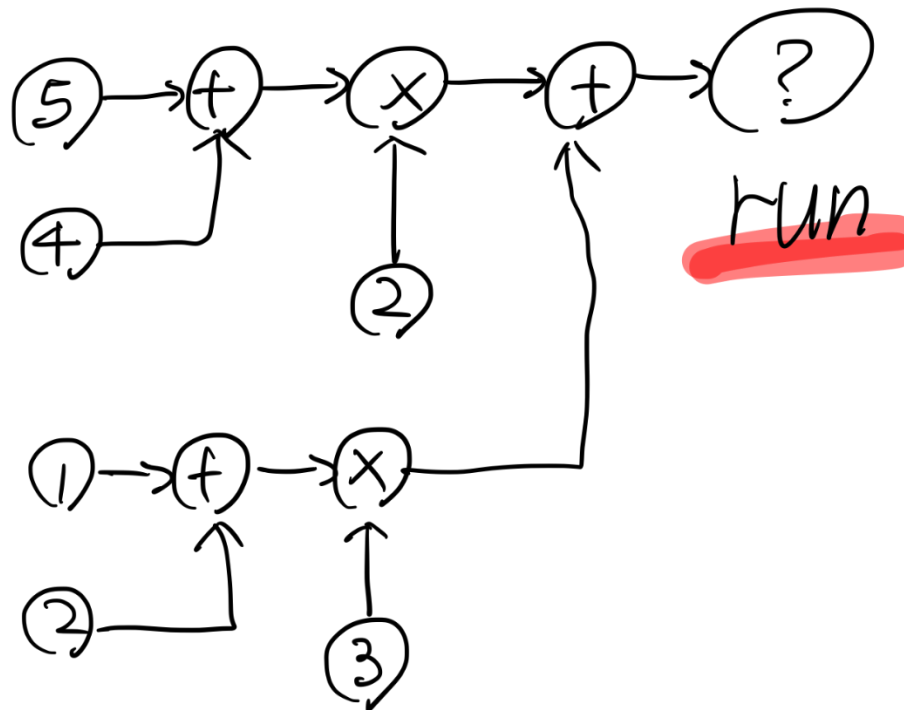
$$2(5+4)+3(1+2)=?$$



Data type

- 그래프 생성

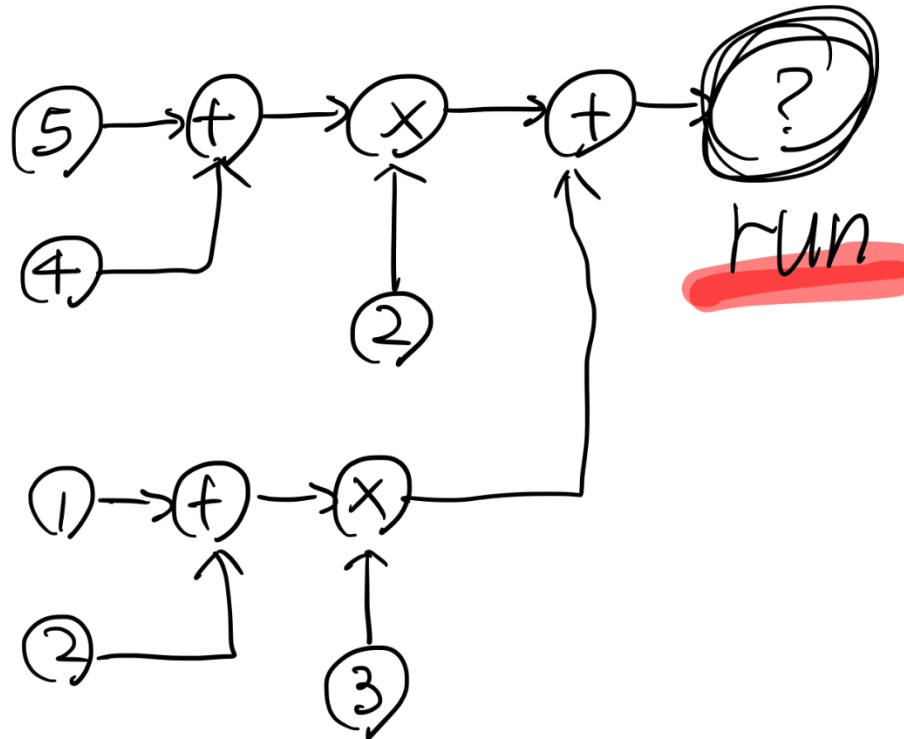
$$2(5+4)+3(1+2)=?$$



Data type

- 그래프 생성

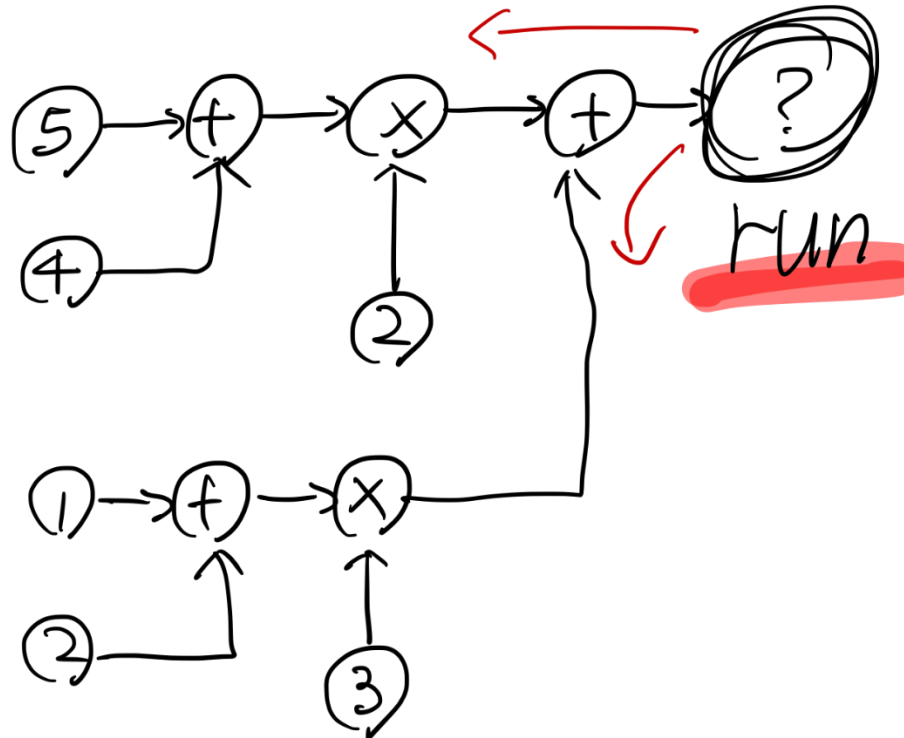
$$2(5+4)+3(1+2)=?$$



Data type

- 그래프 생성

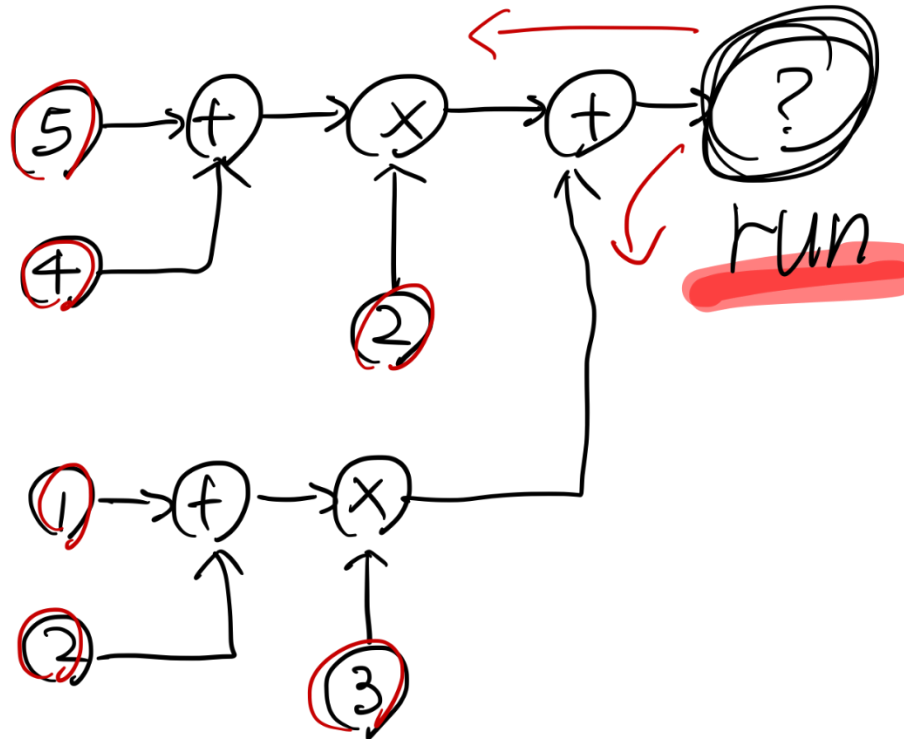
$$2(5+4)+3(1+2)=?$$



Data type

- 그래프 생성

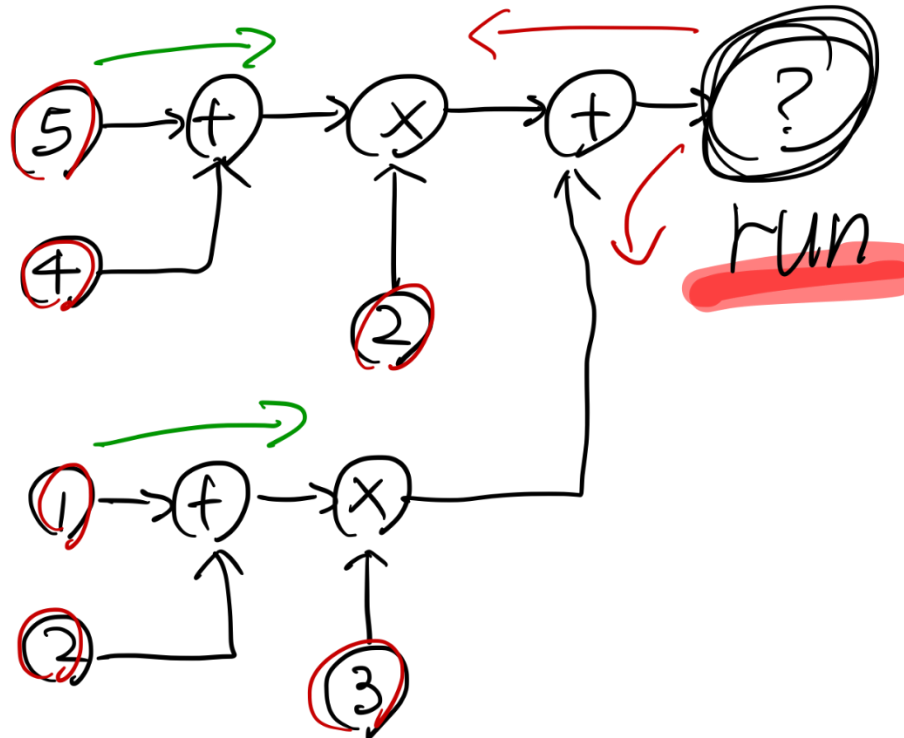
$$2(5+4)+3(1+2)=?$$



Data type

- 그래프 생성

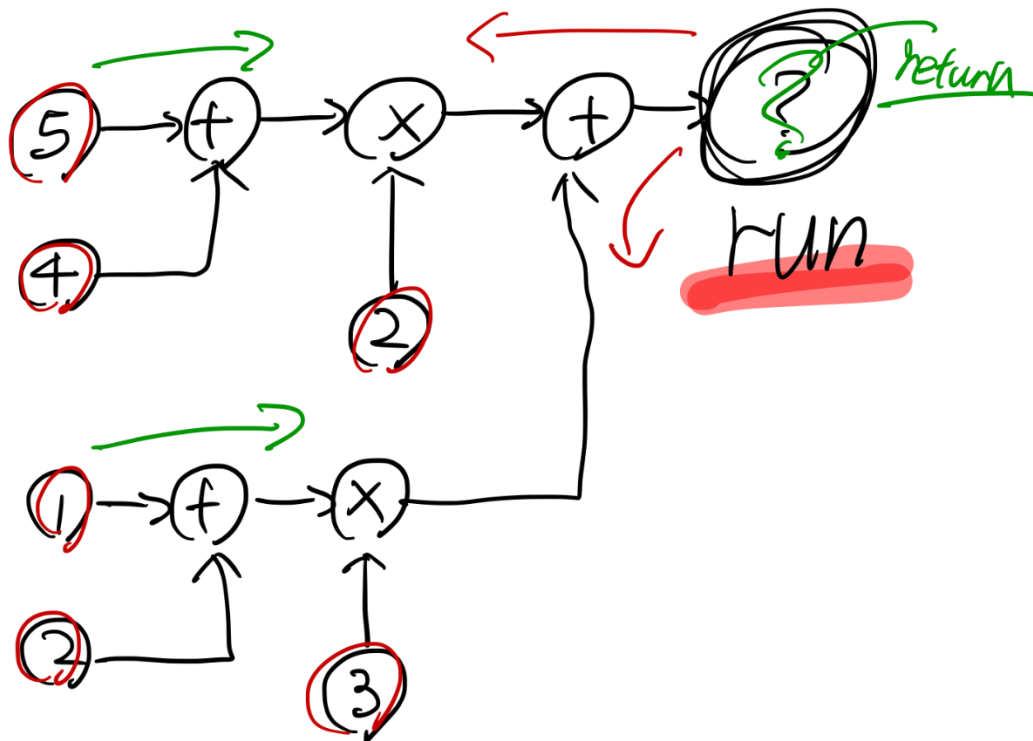
$$2(5+4)+3(1+2)=?$$



Data type

- 그래프 생성

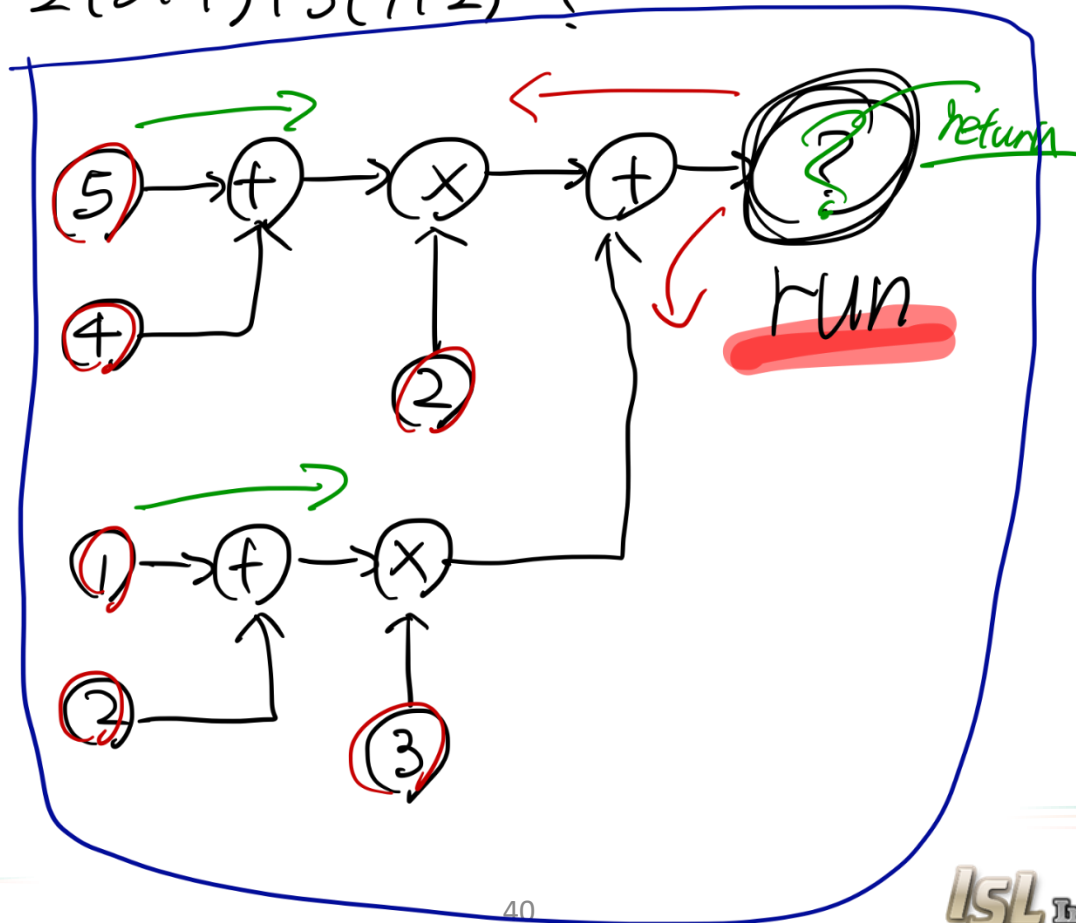
$$2(5+4)+3(1+2)=?$$



Data type

- 그래프 생성

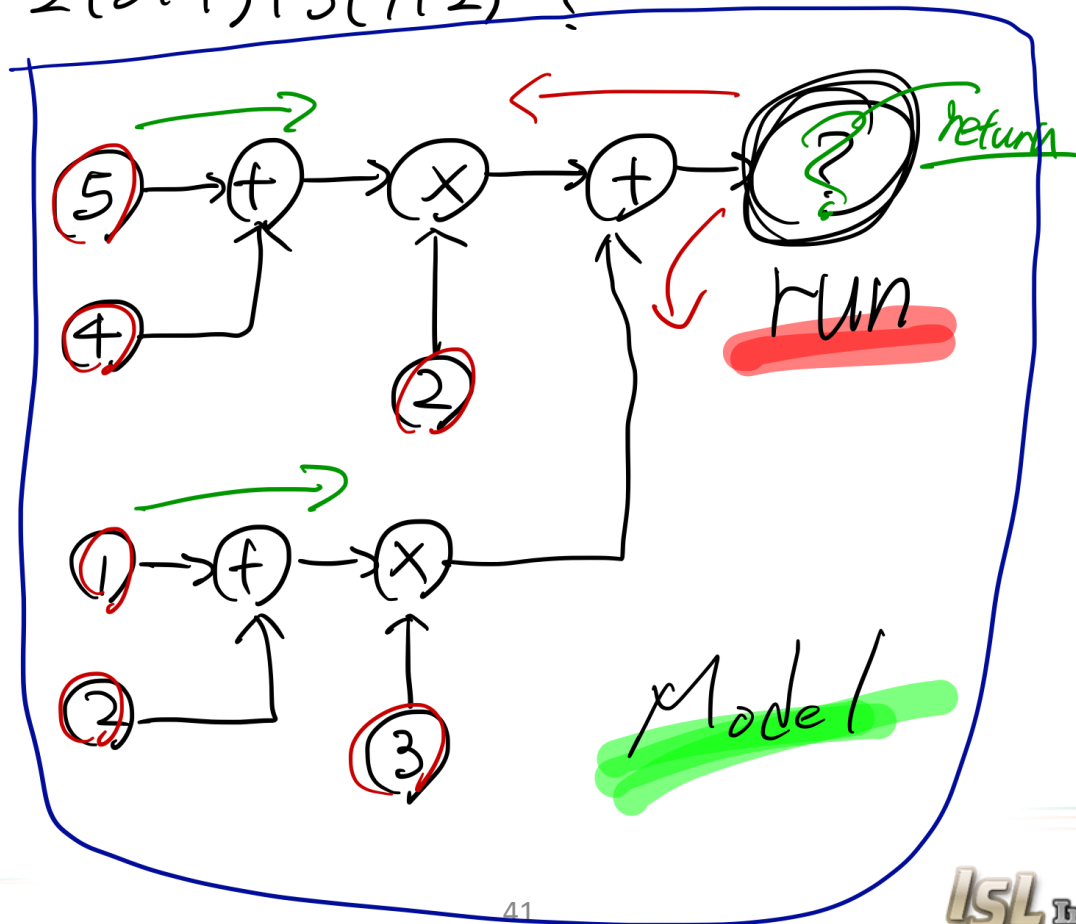
$$2(5+4)+3(1+2)=?$$



Data type

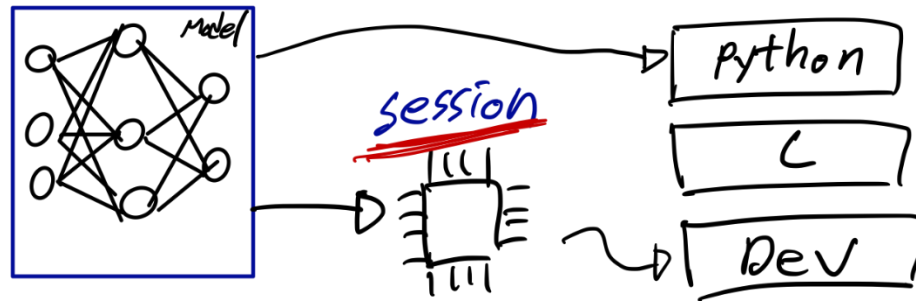
- 그래프 생성

$$2(5+4)+3(1+2)=?$$



Data type

- 세션



- Python이 C기반이기 때문에 상호간의 호환성이 좋음
- Python에서 생성된 모델은 디바이스에서 실제 구조가 만들어짐. 이 과정을 '세션에 보낸다.' 또는 '세션에 올린다.' 고 표현함.
- 하드웨어 단계에서 구현되었기 때문에 **매우 빠른 연산**이 가능해짐 -> TF의 장점

Data type

- 테스트 - constant

```
In [1]: import tensorflow as tf
```

```
In [2]: ph = tf.placeholder(dtype = tf.float32, shape=[3,3])  
var = tf.Variable([1, 2, 3, 4, 5], dtype = tf.float32)  
const = tf.constant([10, 20, 30, 40, 50], dtype = tf.float32)
```

```
In [3]: print(ph)
```

```
Tensor("Placeholder:0", shape=(3, 3), dtype=float32)
```

```
In [4]: print(var)
```

```
Tensor("Variable/read:0", shape=(5,), dtype=float32)
```

```
In [5]: print(const)
```

```
Tensor("Const:0", shape=(5,), dtype=float32)
```

```
In [6]: a = tf.constant([2])  
b = tf.constant([3])  
c = tf.constant([5])  
d = a+b*c  
print(d)
```

```
Tensor("add:0", shape=(1,), dtype=int32)
```

Data type

- 테스트 - constant

```
In [6]: a = tf.constant([2])  
        b = tf.constant([3])  
        c = tf.constant([5])  
        d = a+b+c  
        print(d)  
  
Tensor("add:0", shape=(1,), dtype=int32)
```

```
In [7]: sess = tf.Session()  
        res = sess.run(d)  
        print(res)  
  
[11]
```

```
In [ ]: |
```

Data type

- 테스트 - Variable

```
In [7]: sess = tf.Session()  
res = sess.run(d)  
print(res)
```

[11]

```
In [8]: var_a = tf.Variable([2])  
var_b = tf.Variable([10])  
var_c = tf.Variable([7])  
var_d = var_a * var_b + var_c  
print(var_d)
```

Tensor("add_1:0", shape=(1,), dtype=int32)

Data type

- 테스트 - Variable

```
In [9]: var_res = sess.run(var_d)

-----
FailedPreconditionError                                Traceback (most recent call last)
D:\Anaconda3\lib\site-packages\tensorflow\python\client\session.py in _do_call(self
    1020     try:
-> 1021         return fn(*args)
    1022     except errors.OpError as e:

D:\Anaconda3\lib\site-packages\tensorflow\python\client\session.py in _run_fn(sess
etadata)
    1002         feed_dict, fetch_list, target_list,
-> 1003         status, run_metadata)
    1004

D:\Anaconda3\lib\contextlib.py in __exit__(self, type, value, traceback)
     65     try:
--> 66         next(self.gen)
     67     except StopIteration:

D:\Anaconda3\lib\site-packages\tensorflow\python\framework\errors_impl.py in raise
    160     compat.as_text(numpy.tensorflow.ITF_Message(status))

In []:
```

Variable은 별도의 초기화가 필요함!

Data type

- 테스트 - Variable

```
D:\Anaconda3\lib\site-packages\tensorflow\python\framework\errors_impl.py in raise
160         compat.as_text(rnrn_tensorflow_TF_Message(status))

In [10]: init = tf.global_variables_initializer()
sess.run(init)
var_res = sess.run(var_d)
print(var_res)

[27]

In [ ]:
```

Data type

- 테스트 - placeholder

```
value1 = 10
value2 = 50
value3 = 14

ph_d = ph_a * ph_b + ph_c

tensor_map = {ph_a : value1, ph_b : value2, ph_c : value3}

print(ph_a)
print(ph_b)
print(ph_c)
print(ph_d)

Tensor("Placeholder_1:0", dtype=float32)
Tensor("Placeholder_2:0", dtype=float32)
Tensor("Placeholder_3:0", dtype=float32)
Tensor("add_2:0", dtype=float32)
```

```
In [12]: ph_res = sess.run(ph_d, feed_dict=tensor_map)
print(ph_res)
```

```
514.0
```

```
In [1]:
```

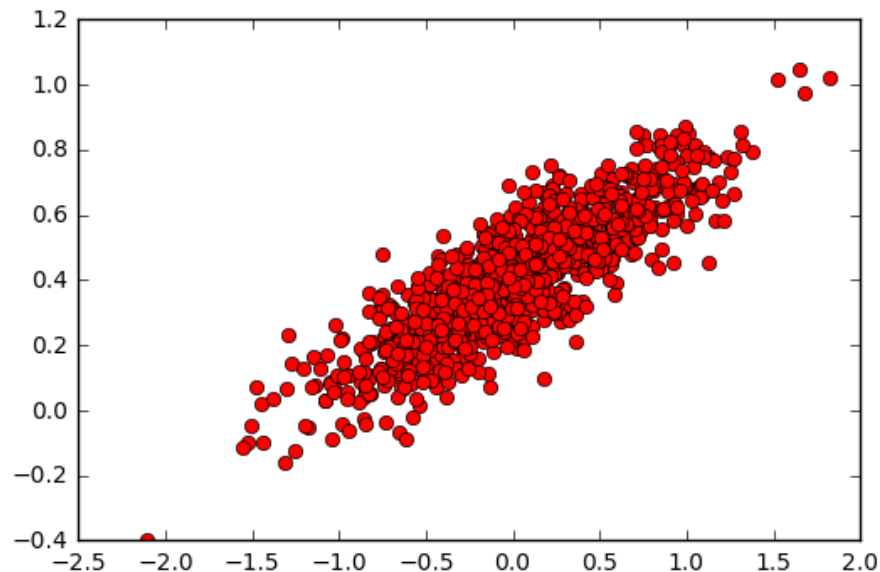

Linear regression + Plot

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
```

```
In [5]: num_point = 1000
vectors_set = []
for i in range(num_point):
    x1 = np.random.normal(0.0, 0.55)
    y1 = x1 * 0.3 + 0.4 + np.random.normal(0.0, 0.1)
    vectors_set.append([x1, y1])

x_data = [v[0] for v in vectors_set]
y_data = [v[1] for v in vectors_set]

plt.plot(x_data, y_data, 'ro')
plt.show()
```



Linear regression + Plot

```
In [6]: W = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
b = tf.Variable(tf.zeros([1]))
y = W * x_data + b

loss = tf.reduce_mean(tf.square(y - y_data)) # cost function
optimizer = tf.train.GradientDescentOptimizer(0.5) # 학습 파라미터
train = optimizer.minimize(loss)
# 모델(그래프) 설계 완료

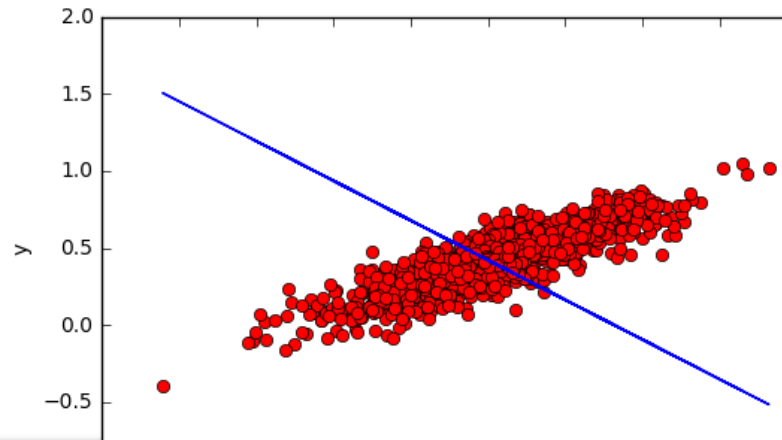
# 변수 초기화 및 세션 열기
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

for step in range(10):
    sess.run(train)
    print(step, sess.run(W), sess.run(b), sess.run(loss))

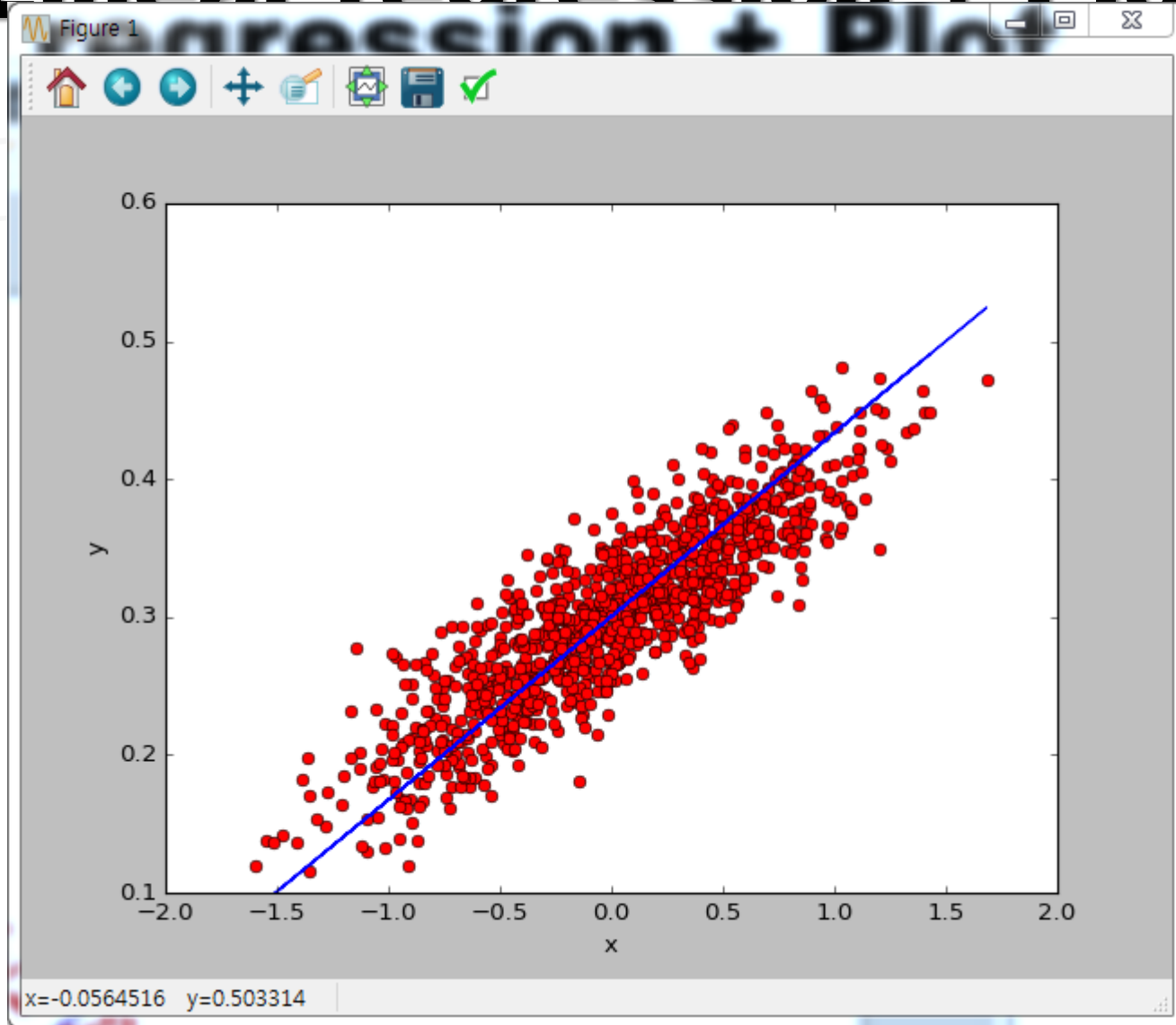
# 데이터 분포
plt.plot(x_data, y_data, 'ro')
# 직선(pred)
plt.plot(x_data, sess.run(W) * x_data + sess.run(b))

plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

0 [-0.51449585] [0.42089418] 0.212177



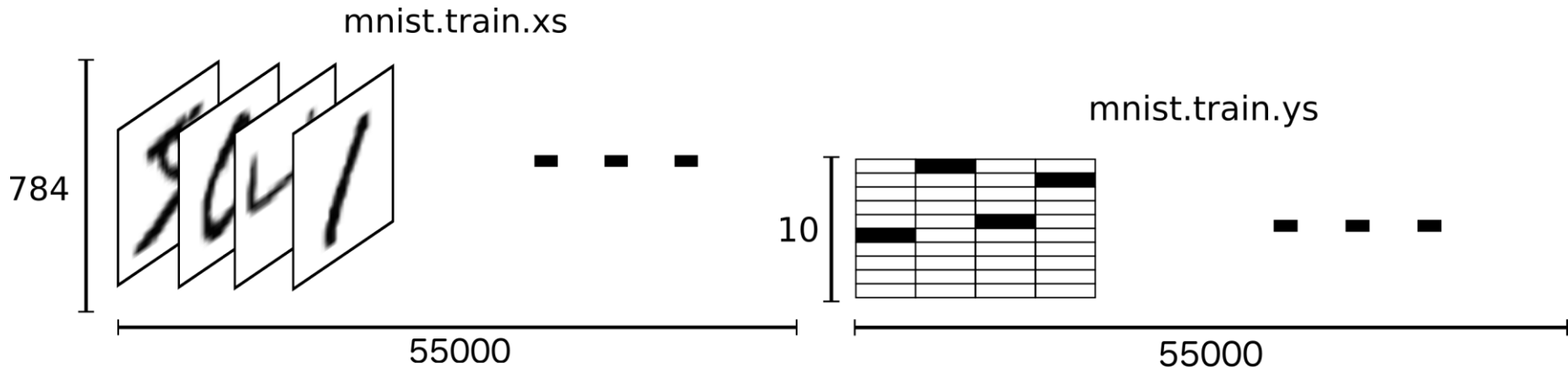
Linear regression + Plot



MNIST

- MNIST

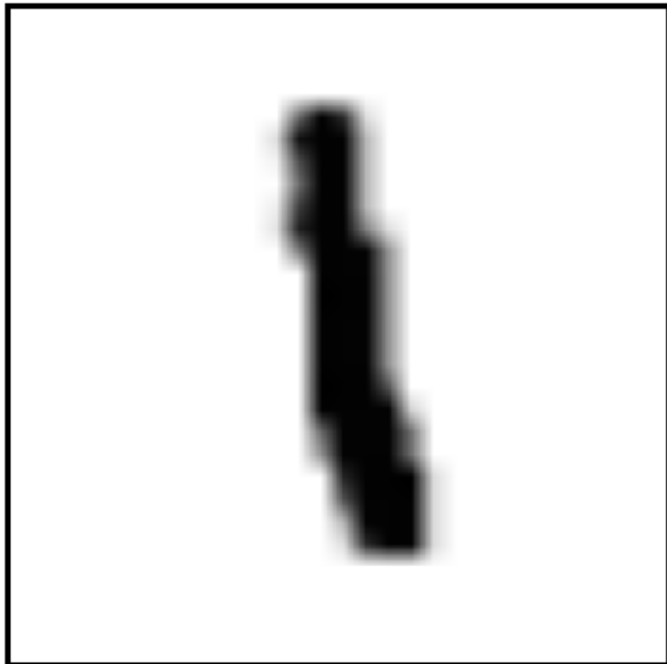
When one learns how to program, there's a tradition that the first thing you do is print "Hello World." Just like programming has Hello World, machine learning has MNIST.



MNIST

- MNIST

When one learns how to program, there's a tradition that the first thing you do is print "Hello World." Just like programming has Hello World, machine learning has MNIST.



12

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	.6	.8	0	0	0	0	0	0
0	0	0	0	0	0	0	.7	1	0	0	0	0	0	0
0	0	0	0	0	0	0	.7	1	0	0	0	0	0	0
0	0	0	0	0	0	0	.5	1	.4	0	0	0	0	0
0	0	0	0	0	0	0	0	1	.4	0	0	0	0	0
0	0	0	0	0	0	0	0	1	.4	0	0	0	0	0
0	0	0	0	0	0	0	0	1	.7	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	.9	1	.1	0	0	0	0
0	0	0	0	0	0	0	0	.3	1	.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

MLP

- Input layer - 3 hidden layer – output layer

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

n_nodes_hl1 = 500
n_nodes_hl2 = 500
n_nodes_hl3 = 500

n_classes = 10
batch_size = 100

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

hidden_1_layer = {'weights':tf.Variable(tf.random_normal([784, n_nodes_hl1])),
                  'biases':tf.Variable(tf.random_normal([n_nodes_hl1]))}

hidden_2_layer = {'weights':tf.Variable(tf.random_normal([n_nodes_hl1, n_nodes_hl2])),
                  'biases':tf.Variable(tf.random_normal([n_nodes_hl2]))}

hidden_3_layer = {'weights':tf.Variable(tf.random_normal([n_nodes_hl2, n_nodes_hl3])),
                  'biases':tf.Variable(tf.random_normal([n_nodes_hl3]))}

output_layer = {'weights':tf.Variable(tf.random_normal([n_nodes_hl3, n_classes])),
                'biases':tf.Variable(tf.random_normal([n_classes]))}

l1 = tf.add(tf.matmul(x,hidden_1_layer['weights']), hidden_1_layer['biases'])
l1 = tf.nn.relu(l1)

l2 = tf.add(tf.matmul(l1,hidden_2_layer['weights']), hidden_2_layer['biases'])
l2 = tf.nn.relu(l2)

l3 = tf.add(tf.matmul(l2,hidden_3_layer['weights']), hidden_3_layer['biases'])
l3 = tf.nn.relu(l3)
```

MLP

- Input layer - 3 hidden layer – output layer

```
output = tf.matmul(l3, output_layer['weights']) + output_layer['biases']

prediction = output
cost = tf.reduce_mean( tf.nn.softmax_cross_entropy_with_logits(prediction,y) )
optimizer = tf.train.AdamOptimizer().minimize(cost)

hm_epochs = 10
sess = tf.Session()
sess.run(tf.global_variables_initializer())

for epoch in range(hm_epochs):
    epoch_loss = 0
    for _ in range(int(mnist.train.num_examples/batch_size)):
        epoch_x, epoch_y = mnist.train.next_batch(batch_size)
        _, c = sess.run([optimizer, cost], feed_dict={x: epoch_x, y: epoch_y})
        epoch_loss += c

    print('Epoch', epoch + 1, 'completed out of', hm_epochs, 'loss:', epoch_loss)

correct = tf.equal(tf.argmax(prediction, 1), tf.argmax(y, 1))

accuracy = tf.reduce_mean(tf.cast(correct, 'float'))
print('Accuracy:', sess.run(accuracy, feed_dict = {x:mnist.test.images, y:mnist.test.labels}))
```

```
Extracting /tmp/data/train-images-idx3-ubyte.gz
Extracting /tmp/data/train-labels-idx1-ubyte.gz
Extracting /tmp/data/t10k-images-idx3-ubyte.gz
Extracting /tmp/data/t10k-labels-idx1-ubyte.gz
Epoch 1 completed out of 10 loss: 1607590.13811
Epoch 2 completed out of 10 loss: 395378.241097
Epoch 3 completed out of 10 loss: 212075.841237
Epoch 4 completed out of 10 loss: 124842.024561
Epoch 5 completed out of 10 loss: 73650.398987
Epoch 6 completed out of 10 loss: 45980.5750535
Epoch 7 completed out of 10 loss: 29941.5561811
Epoch 8 completed out of 10 loss: 23685.460072
Epoch 9 completed out of 10 loss: 19176.7452043
Epoch 10 completed out of 10 loss: 17872.4552318
Accuracy: 0.9494
```

MLP

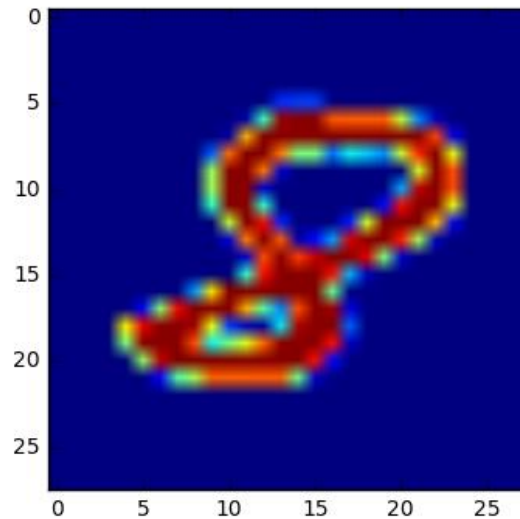
- Input layer - 3 hidden layer – output layer

```
import numpy as np
s = 61
arr = np.array(mnist.test.images[s])
arr.shape = (28, 28)

import matplotlib.pyplot as plt

plt.imshow(arr)
plt.show()

arr.shape = (1, 784)
print('GT : ', np.argmax(mnist.test.labels[s]))
print('Pred : ', np.argmax(sess.run(output, feed_dict={x:arr})))
```



```
GT : 8
Pred : 8
```


MLP

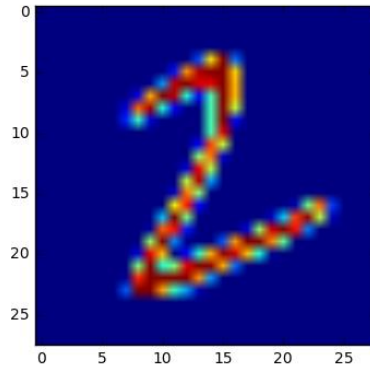
- Failure

```
import numpy as np
s = 33
arr = np.array(mnist.test.images[s])
arr.shape = (28, 28)

import matplotlib.pyplot as plt

plt.imshow(arr)
plt.show()

arr.shape = (1, 784)
print('GT : ', np.argmax(mnist.test.labels[s]))
print('Pred : ', np.argmax(sess.run(output, feed_dict={x:arr})))
```



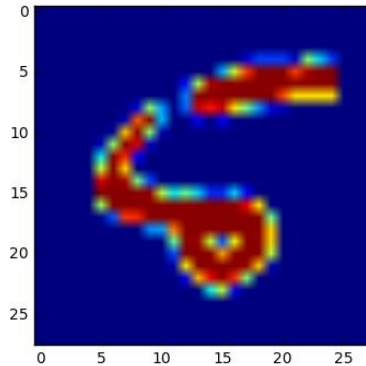
GT : 2
Pred : 1

```
import numpy as np
s = 8
arr = np.array(mnist.test.images[s])
arr.shape = (28, 28)

import matplotlib.pyplot as plt

plt.imshow(arr)
plt.show()

arr.shape = (1, 784)
print('GT : ', np.argmax(mnist.test.labels[s]))
print('Pred : ', np.argmax(sess.run(output, feed_dict={x:arr})))
```



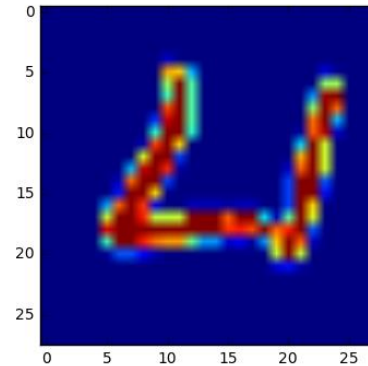
GT : 5
Pred : 6

```
import numpy as np
s = 33
arr = np.array(mnist.test.images[s])
arr.shape = (28, 28)

import matplotlib.pyplot as plt

plt.imshow(arr)
plt.show()

arr.shape = (1, 784)
print('GT : ', np.argmax(mnist.test.labels[s]))
print('Pred : ', np.argmax(sess.run(output, feed_dict={x:arr})))
```



GT : 4
Pred : 0

MLP

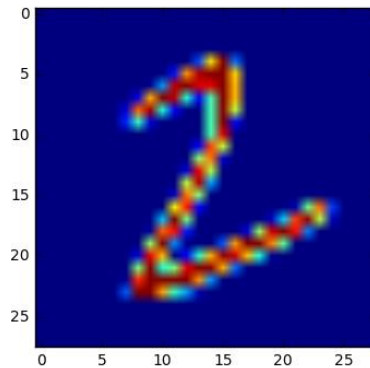
- Failure

```
import numpy as np
s = 33
arr = np.array(mnist.test.images[s])
arr.shape = (28, 28)

import matplotlib.pyplot as plt

plt.imshow(arr)
plt.show()

arr.shape = (1, 784)
print('GT : ', np.argmax(mnist.test.labels[s]))
print('Pred : ', np.argmax(sess.run(output, feed_dict={x:arr})))
```



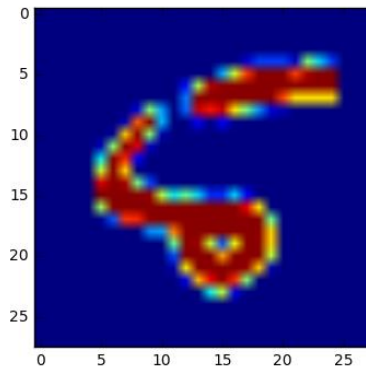
GT : 2
Pred : 1

```
import numpy as np
s = 8
arr = np.array(mnist.test.images[s])
arr.shape = (28, 28)

import matplotlib.pyplot as plt

plt.imshow(arr)
plt.show()

arr.shape = (1, 784)
print('GT : ', np.argmax(mnist.test.labels[s]))
print('Pred : ', np.argmax(sess.run(output, feed_dict={x:arr})))
```



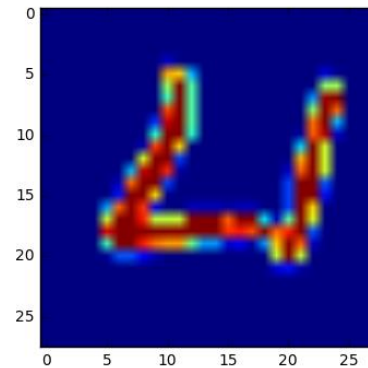
GT : 5
Pred : 6

```
import numpy as np
s = 33
arr = np.array(mnist.test.images[s])
arr.shape = (28, 28)

import matplotlib.pyplot as plt

plt.imshow(arr)
plt.show()

arr.shape = (1, 784)
print('GT : ', np.argmax(mnist.test.labels[s]))
print('Pred : ', np.argmax(sess.run(output, feed_dict={x:arr})))
```



GT : 4
Pred : 0

CNN (ConvNet)

- 버그

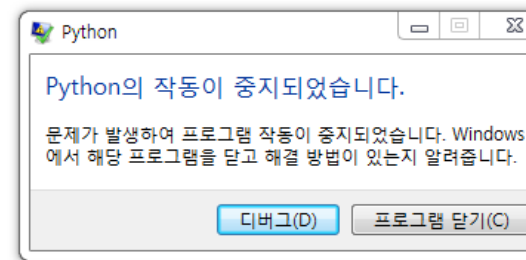
```
In [13]: sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

```
In [+]: for i in range(20000):#20000
batch = mnist.train.next_batch(50)
if i % 1000 == 0:
train_accuracy = sess.run(accuracy, feed_dict={
x:batch[0], y_: batch[1], keep_prob: 1.0})
print("step %d, training accuracy %g"%(i, train_accuracy))
sess.run(train_step,feed_dict={x: batch[0], y_: batch[1], keep_prob: 0.5})
```

```
step 0, training accuracy 0.12
step 1000, training accuracy 0.94
step 2000, training accuracy 0.96
step 3000, training accuracy 1
step 4000, training accuracy 0.98
step 5000, training accuracy 1
step 6000, training accuracy 0.96
step 7000, training accuracy 1
step 8000, training accuracy 0.98
step 9000, training accuracy 0.96
step 10000, training accuracy 0.98
step 11000, training accuracy 1
step 12000, training accuracy 1
step 13000, training accuracy 1
step 14000, training accuracy 0.98
step 15000, training accuracy 1
step 16000, training accuracy 1
step 17000, training accuracy 1
step 18000, training accuracy 1
step 19000, training accuracy 1
```

```
In [+]: print("test accuracy %g"% sess.run(
accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels, keep_prob: 1.0}))
```

CPU버전 삭제로 해결



CNN (ConvNet)

```
In [12]: cross_entropy = -tf.reduce_sum(y_*tf.log(y_conv))
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
```

```
In [13]: sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

```
In [17]: for i in range(20000):#20000
        batch = mnist.train.next_batch(50)
        if i % 1000 == 0:
```

```
In [19]: print("Accuracy : ", sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels, keep_prob: 1.0}))
```

Accuracy : 0.9927

```
step 0, training accuracy 0.94
step 1000, training accuracy 1
step 2000, training accuracy 1
step 3000, training accuracy 1
step 4000, training accuracy 1
step 5000, training accuracy 0.98
step 6000, training accuracy 1
step 7000, training accuracy 1
step 8000, training accuracy 1
step 9000, training accuracy 1
step 10000, training accuracy 1
step 11000, training accuracy 1
step 12000, training accuracy 1
step 13000, training accuracy 1
step 14000, training accuracy 1
step 15000, training accuracy 1
step 16000, training accuracy 1
step 17000, training accuracy 1
step 18000, training accuracy 1
step 19000, training accuracy 1
```

Q & A

example



True: 7



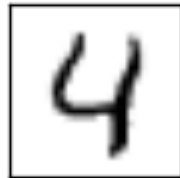
True: 2



True: 1



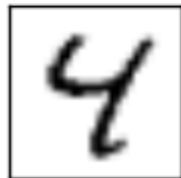
True: 0



True: 4



True: 1



True: 4



True: 9



True: 5

Before any opt

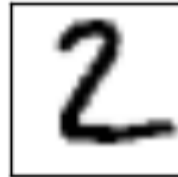
```
print_accuracy()
```

Accuracy on test-set: 9.8%

```
plot_example_errors()
```



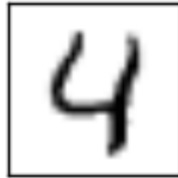
True: 7, Pred: 0



True: 2, Pred: 0



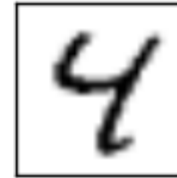
True: 1, Pred: 0



True: 4, Pred: 0



True: 1, Pred: 0



True: 4, Pred: 0



True: 9, Pred: 0



True: 5, Pred: 0



True: 9, Pred: 0

After 1 opt iter

```
optimize(num_iterations=1)
```

```
print_accuracy()
```

Accuracy on test-set: 40.7%

```
plot_example_errors()
```



True: 2, Pred: 6



True: 4, Pred: 0



True: 9, Pred: 6



True: 5, Pred: 6



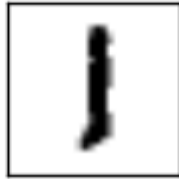
True: 9, Pred: 6



True: 6, Pred: 0



True: 9, Pred: 6

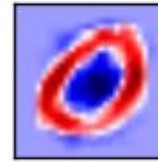


True: 1, Pred: 6

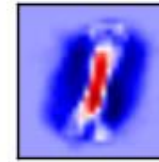


True: 5, Pred: 0

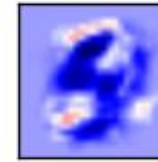
```
plot_weights()
```



Weights: 0



Weights: 1



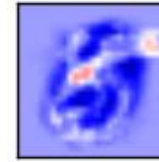
Weights: 2



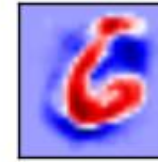
Weights: 3



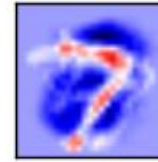
Weights: 4



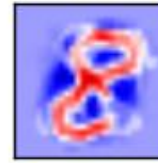
Weights: 5



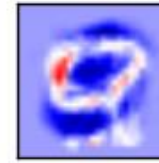
Weights: 6



Weights: 7



Weights: 8



Weights: 9



After 10 opt iter

```
# We have already performed 1 iteration,  
optimize(num_iterations=9)
```

```
print_accuracy()
```

Accuracy on test-set: 78.2%

```
plot_example_errors()
```



True: 9, Pred: 4



True: 6, Pred: 2



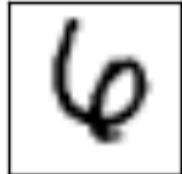
True: 9, Pred: 7



True: 5, Pred: 2



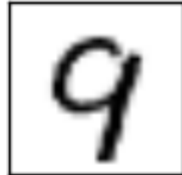
True: 5, Pred: 3



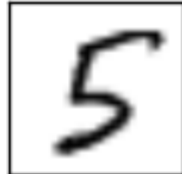
True: 6, Pred: 4



True: 9, Pred: 7

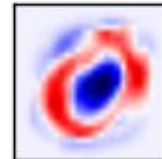


True: 9, Pred: 4

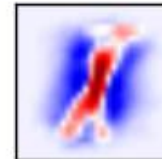


True: 5, Pred: 8

```
plot_weights()
```



Weights: 0



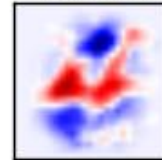
Weights: 1



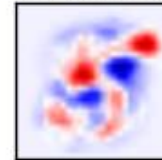
Weights: 2



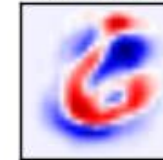
Weights: 3



Weights: 4



Weights: 5



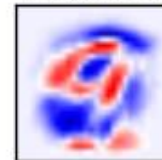
Weights: 6



Weights: 7



Weights: 8



Weights: 9



After 1000 opt iter

```
# We have already performed 10 iterations.  
optimize(num_iterations=990)
```

```
print_accuracy()
```

Accuracy on test-set: 91.7%

```
plot_example_errors()
```



True: 5, Pred: 6



True: 9, Pred: 7



True: 7, Pred: 4



True: 4, Pred: 6



True: 2, Pred: 7



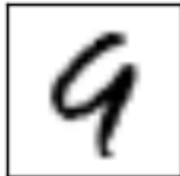
True: 2, Pred: 9



True: 3, Pred: 2

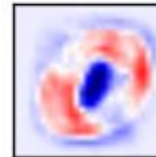


True: 9, Pred: 4

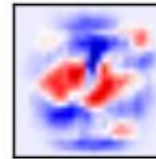


True: 9, Pred: 4

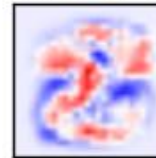
```
plot_weights()
```



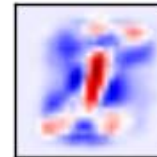
Weights: 0



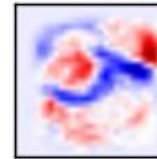
Weights: 4



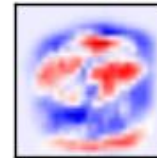
Weights: 8



Weights: 1



Weights: 5



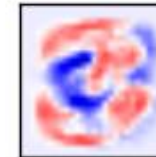
Weights: 9



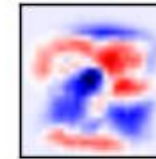
Weights: 2



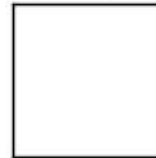
Weights: 6



Weights: 3



Weights: 7



After 1000 opt iter

```
print_confusion_matrix()
```

```
[[ 957  0  3  2  0  5  11  1  1  0]
 [  0 1108  2  2  1  2  4  2  14  0]
 [  4  9  914  19  15  5  13  14  35  4]
 [  1  0  16  928  0  28  2  14  13  8]
 [  1  1  3  2  939  0  10  2  6  18]
 [ 10  3  3  33  10  784  17  6  19  7]
 [  8  3  3  2  11  14  915  1  1  0]
 [  3  9  21  9  7  1  0  959  2  17]
 [  8  8  8  38  11  40  14  18  825  4]
 [ 11  7  1  13  75  13  1  39  4  845]]
```

