Deep Learning

엑셈아카데미 http://exem-academy.com/
온사이트 교육 문의 edu@ex-em.com
엑셈 연구콘텐츠팀 권건우, 이근오, 김정수, 김찬경

엑셈 연구팀 소개 (edu@ex-em.com)

- **권건우 상무**

・ MIS Ph.D.
・ 前 삼성 SDS DBA, TA
・ 現 엑셈 아카데미 상무
・ 現 DB 기술연구소 소장
・ DB Deep Internals 강의
(Oracle, PostgreSQL, MySQL, Aurora)
・ 엑셈아카데미 블록체인 인사이트 강의
・ 엑셈아카데미 인공지능 인사이트 강의
・ < Oracle, PostgreSQL, MySQL Core Architecture 1 > 책 집필
・ < Oracle, PostgreSQL, MySQL Core Architecture 2 > 책 집필

- **이근오 팀장**

・ 삼성카드/삼성생명/ING생명 차세대 PJT 참여
・ 現 DB 기술연구소 팀장
・ 엑셈아카데미 DB Deep Internals 강의
(Oracle, PostgreSQL, MySQL, Aurora)
・ 엑셈아카데미 블록체인 인터널 강의
・ 엑셈아카데미 인공지능 인터널 강의
・ < Oracle, PostgreSQL, MySQL Core Architecture 1 > 책 집필
・ < Oracle, PostgreSQL, MySQL Core Architecture 2 > 책 집필

- **장서연 연구원**

・ 現 엑셈 DB기술연구소 선임 연구원
・ 엑셈아카데미 DB Deep Internals 강의
(Oracle, PostgreSQL, MySQL, Aurora)
・ 엑셈아카데미 오로라 첫걸음 강의
・ 엑셈아카데미 블록체인 강의
・ 엑셈아카데미 왕초보 SQL 강의

- **이대덕 연구원**

・ 現 DB 기술연구소 선임 연구원
・ 엑셈아카데미 DB Deep Internals 강의
(Oracle, PostgreSQL, MySQL, Aurora)
・ 엑셈아카데미 블록체인 인터널 강의
・ < Oracle, PostgreSQL, MySQL Core Architecture 1 > 책 집필
・ < Oracle, PostgreSQL, MySQL Core Architecture 2 > 책 집필

엑셈 연구팀 소개 (edu@ex-em.com)

- **김정수 연구원**

・ 現 엑셈 연구컨텐츠팀 연구원
・ Nike, 세노비스, 스와치그룹 BI프로젝트 참여
・ 엑셈아카데미 머신러닝을 위한 파이썬 라이브러리 살펴보기 강의

- **장호형 연구원**

・ 現 엑셈 DB기술연구소 선임 연구원
・ 코웨이 DB암호화 프로젝트 참여
・ 인천청라지구 U-City 구축사업 참여
・ 마곡 U-City 구축사업 참여
・ 통일부 영상편지 공정관리 시스템 구축사업 참여
・ 환경부 생활환경 안전정보 제공 시스템 구축사업 참여
・ 서울시설공단 도로순찰관리시스템 구축사업 참여
・ 제주 스마트그리드 구축사업 참여

- **문믿음 연구원**

・ 現 엑셈 아카데미 연구콘텐츠팀 객원연구원
・ 통계학 석사과정 재학중
・ 시계열 예측분석 연구실 소속
・ 학부 통계학, 통계학실험 강의조교
・ 엑셈아카데미 R을 활용한 머신러닝 강의

- **권기범 연구원**

・ 現 엑셈아카데미 연구콘텐츠팀 연구원
・ 엑셈 대학생 아카데미 머신러닝 캠프 운영 조교
・ 엑셈 대학생 아카데미 머신러닝 캠프 선형대수 기초 강의

엑셈 연구팀 소개 (edu@ex-em.com)

- **김찬경 연구원**

• 現 엑셈 아카데미 연구콘텐츠팀 연구원
• 엑셈 아카데미
"R찬 세미나, 제대로 R아보자!" 강의
• 엑셈 아카데미
"R을 활용한 데이터 시각화" 강의

목 차

1. Machine Learning의 기초
2. Tensorflow
3. Linear Regression
4. Multi-Variable linear regression
5. Logistic Classifier
6. Softmax Classifier
7. Learning rate, Evaluation, MNIST
8. Tensor Manipulation
9. NN for XOR, TensorBoard, BackProgation

목차

10. NN, ReLu, Xavier, Dropout, and A

11. CNN

12. RNN

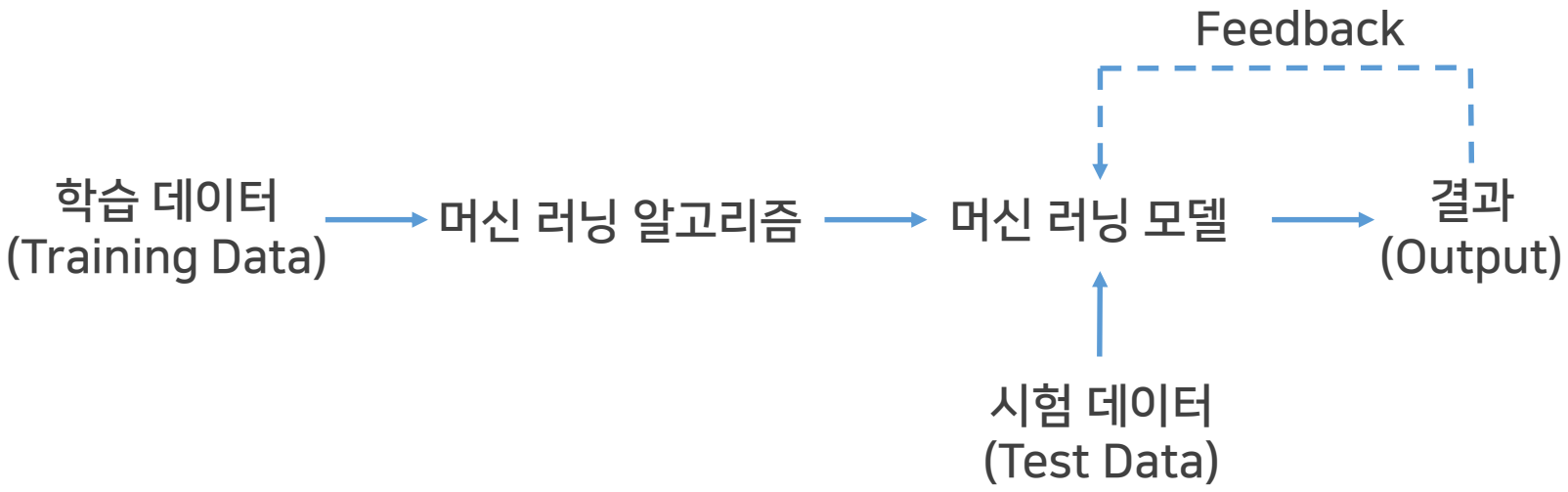
1. Machine Learning의 기초

What is Machine Learning?

머신 러닝의 정의

"컴퓨터에 명시적인 프로그램 없이 배울 수 있는 능력을 부여하는 연구 분야"
- 1959, 아서사무엘(Arthur Samuel) -

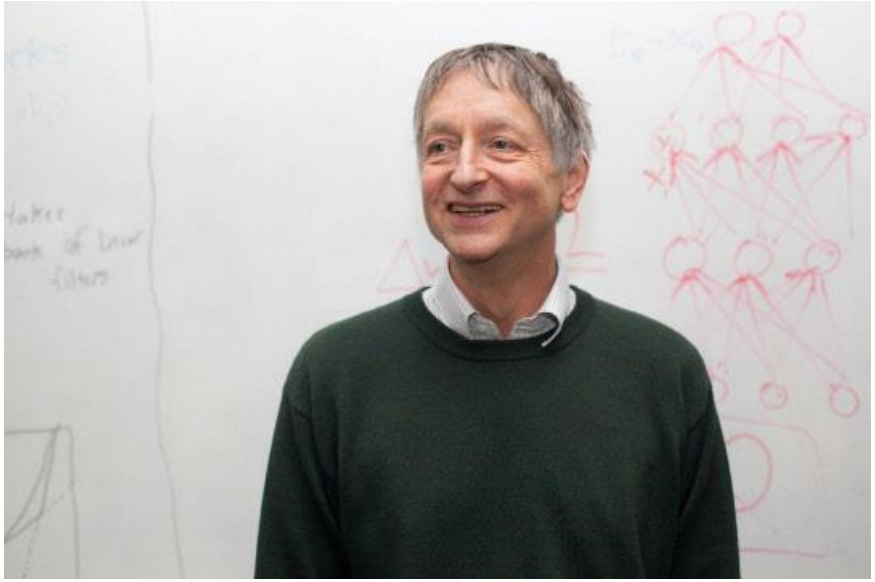
"특정한 작업(Task)을 달성하기 위해 경험(Experience)이 축적될수록 과업 수행의 성능(Performance)이 향상되는 컴퓨터 프로그램 또는 에이전트를 개발하는 것"
- 1997, 톰 미첼(Tom M. Mitchell) -



Source : http://oneclickitsolution.com/2018/07/19/everything-you-need-to-know-about-machine-learning/

머신 러닝이 주목받은 이유

머신 러닝의 역사는 1950년대 부터 시작되어 오래되었지만 2000년대 중반에 들어서 딥러닝(Deep Learning)의 탄생으로 현저한 발전이 이루어졌다.



80~90년대 인공신경망의 암흑기 시절을 끝내게 한 제프리 힌튼 교수

Source : https://news.naver.com/main/read.nhn?oid=029&aid=0002436186

학습 방식에 따른 Learning의 종류

Supervised Learning(지도 학습)

✓ 정답 데이터를 알려주고 학습시키는 방법

Ex) image labeling learning from tagged images

Prediction exam score based on time spent(regression)

E-mail spam filter(binary classification)

Letter grade(ABCDF) based on time spent(multi-label classification)

Unsupervised Learning(비지도 학습)

✓ 정답을 알려주는 과정이 없고 컴퓨터 스스로 수많은 데이터들 속에서 패턴을 발견하는 것

Ex) news grouping, word clustering

Source : https://blog.naver.com/eveoftheday/221301077378

Supervised Learning(지도 학습)의 종류

✓ Regression(회귀)

- 종속변수(y)와 설명변수(x)사이의 관계를 예측하는 분석 기법

Ex) linear regression

Logistic regression(true or false)

Polynomial regression(곡선으로 된 데이터 분포에 알맞은 모델)

Ridge regression

Lasso regression

Elasticnet regression

✓ binary classification(이진 분류)

✓ multi label classification

Source : https://blog.naver.com/eveoftheday/221301077378

2. Tensorflow

Tensorflow란?

Google 내 연구와 제품개발을 위한 목적으로 구글 브레인팀이 만든 오픈소스 라이브러리

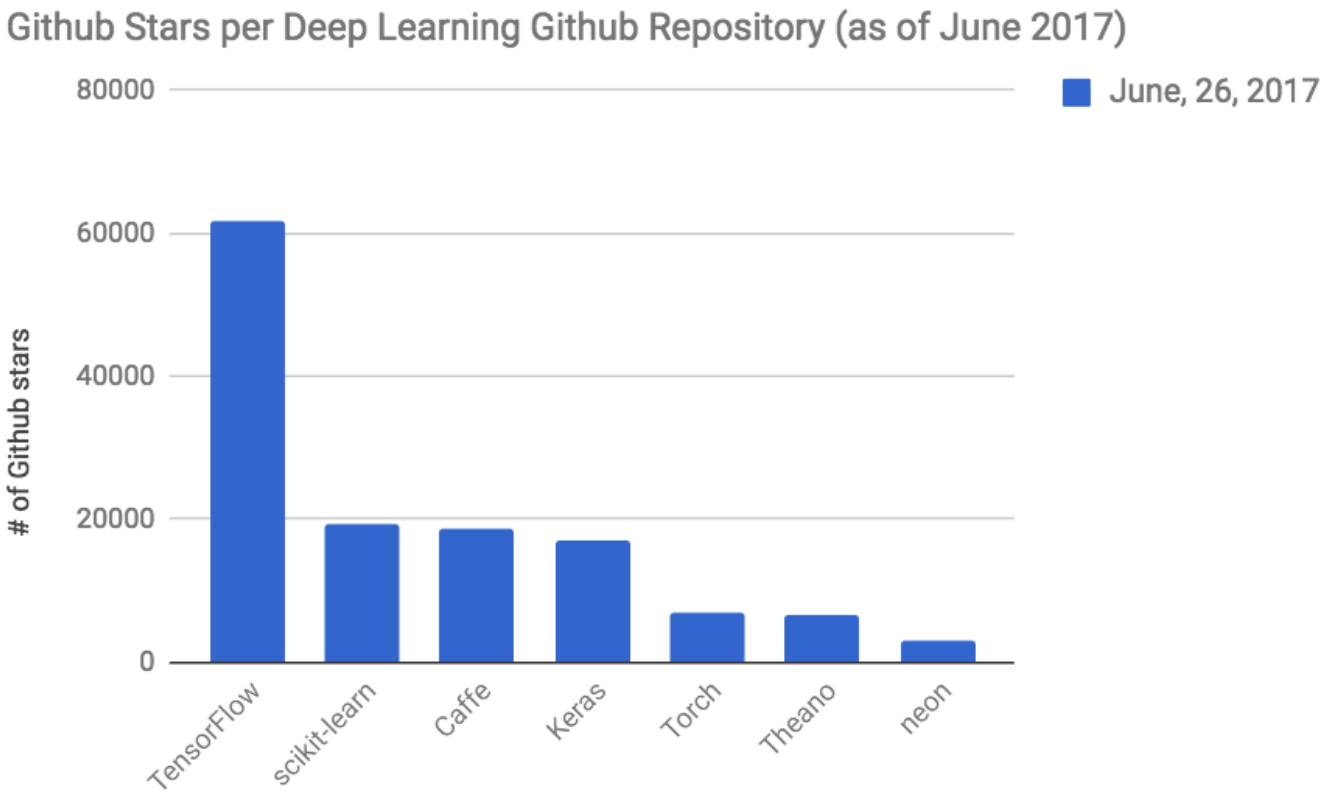
데이터 플로우 프로그래밍과 뉴럴 네트워크같은 기계학습 응용프로그램에도 사용된다.

Source : https://www.tensorflow.org/,
https://ko.wikipedia.org/wiki/%ED%85%90%EC%84%9C%ED%94%8C%EB%A1%9C

Tensorflow 소개

Source : https://www.youtube.com/watch?v=mWl45NkFBOc

왜 Tensorflow를 사용할까?



✓ Github : 웹 기반 분산 소스 관리 시스템, Star는 즐겨찾기를 해두겠다는 의미
다른 라이브러리보다 소스코드가 많다.

Source : http://code-love.com/tag/deep-learning-library/

Data Flow Graph

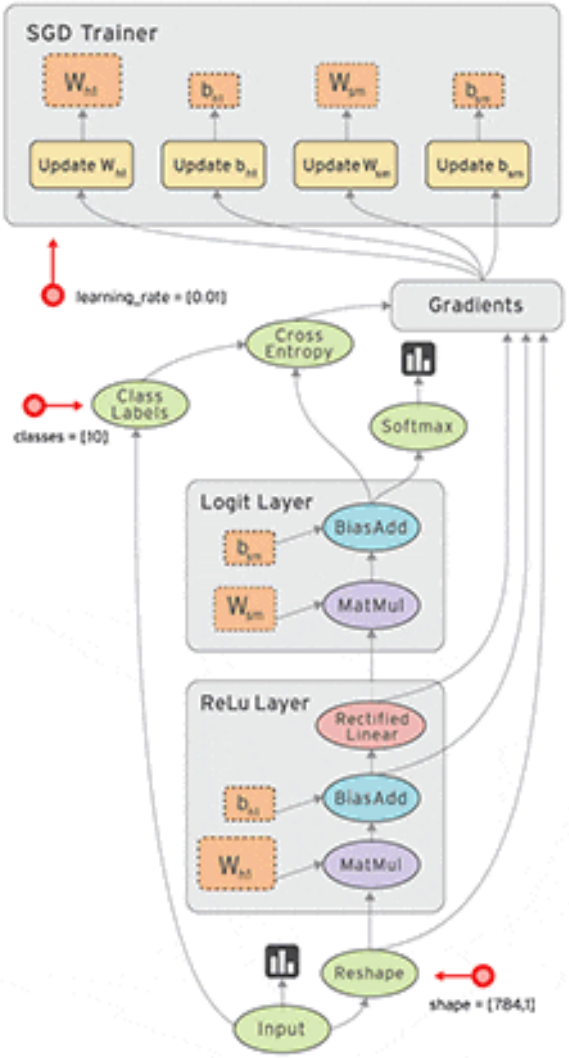
Data Flow Graph란?

- ✓ Tensorflow는 Data Flow Graph 형태로 작동한다.
- ✓ Data Flow Graph 구성요소

  노드(Node): 텐서를 처리하는 연산(Operations)

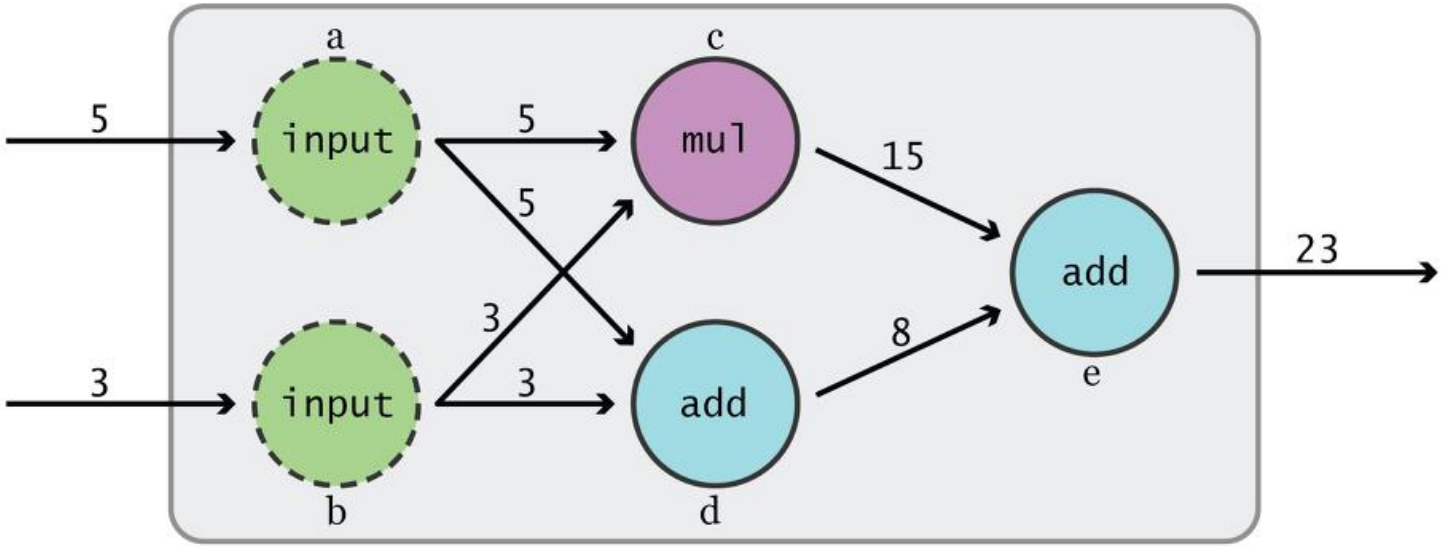
  엣지(Edge): 노드 사이를 이동하는 다차원 데이터 배열(텐서, Tensor)를 나타낸다.
- ✓ 텐서, 즉 데이터가 노드를 지나면서 연산이 이루어진다.

  텐서가 그래프 상에서 이동하여 Tensorflow이다.



Source : https://www.tensorflow.org/

Data Flow Graph Computation



- ✓ Tensor 데이터 구조가 이동하면서 Node에 의해서 연산(Operation)을 하는 방식
- ✓ 위 그래프에서 Tensor는 5나 3같은 숫자, Node는 a, b, c, d, e이다.

Source : http://yamerong.tistory.com/40

What is Tensor?

Tensor란 무엇인가?

✓ Tensor는 배열(array)로서 어떤 값이든 될 수 있다.

✓ Tensor를 설명할 땐 랭크(Rank)와 모양(Shape), 종류(Type)를 사용한다.

Rank	Math entity	Python example
0	Scalar (magnitude only)	`s = 483`
1	Vector (magnitude and direction)	`v = [1.1, 2.2, 3.3]`
2	Matrix (table of numbers)	`m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]`
3	3-Tensor (cube of numbers)	`t = [[[2], [4], [6]], [[8], [10], [12]], [[14], [16], [18]]]`
n	n-Tensor (you get the idea)	`....`

✓ 랭크(Rank)는 Tensor가 몇 차원 배열인지 나타낸다.

Source : http://forensics.tistory.com/5

What is Tensor?

Tensor란 무엇인가?

Rank	Shape	Dimension number	Example
0	[]	0-D	A 0-D tensor. A scalar.
1	[D0]	1-D	A 1-D tensor with shape [5].
2	[D0, D1]	2-D	A 2-D tensor with shape [3, 4].
3	[D0, D1, D2]	3-D	A 3-D tensor with shape [1, 4, 3].
n	[D0, D1, ... Dn-1]	n-D	A tensor with shape [D0, D1, ... Dn-1].

- ✓ 모양(Shape)은 각각의 차원에 몇 개의 원소가 들어있는지를 의미한다.
- ✓ t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]] 이면 shape은 [3, 3]이 된다.

Source : http://forensics.tistory.com/5

Tensor란 무엇인가?

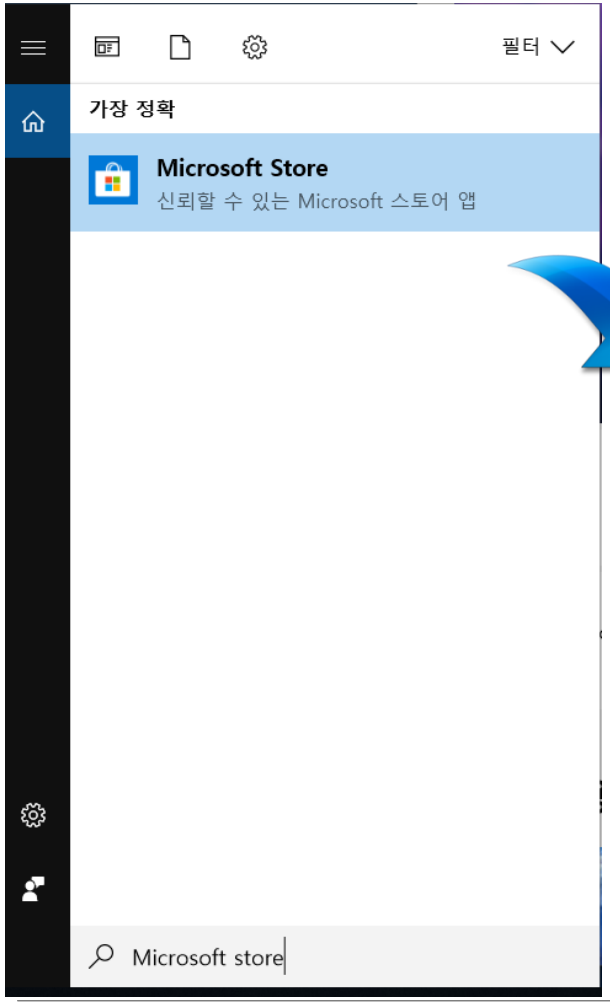
Data type	Python type	Description
DT_FLOAT	tf.float32	32 bits floating point.
DT_DOUBLE	tf.float64	64 bits floating point.
DT_INT8	tf.int8	8 bits signed integer.
DT_INT16	tf.int16	16 bits signed integer.
DT_INT32	tf.int32	32 bits signed integer.
DT_INT64	tf.int64	64 bits signed integer.

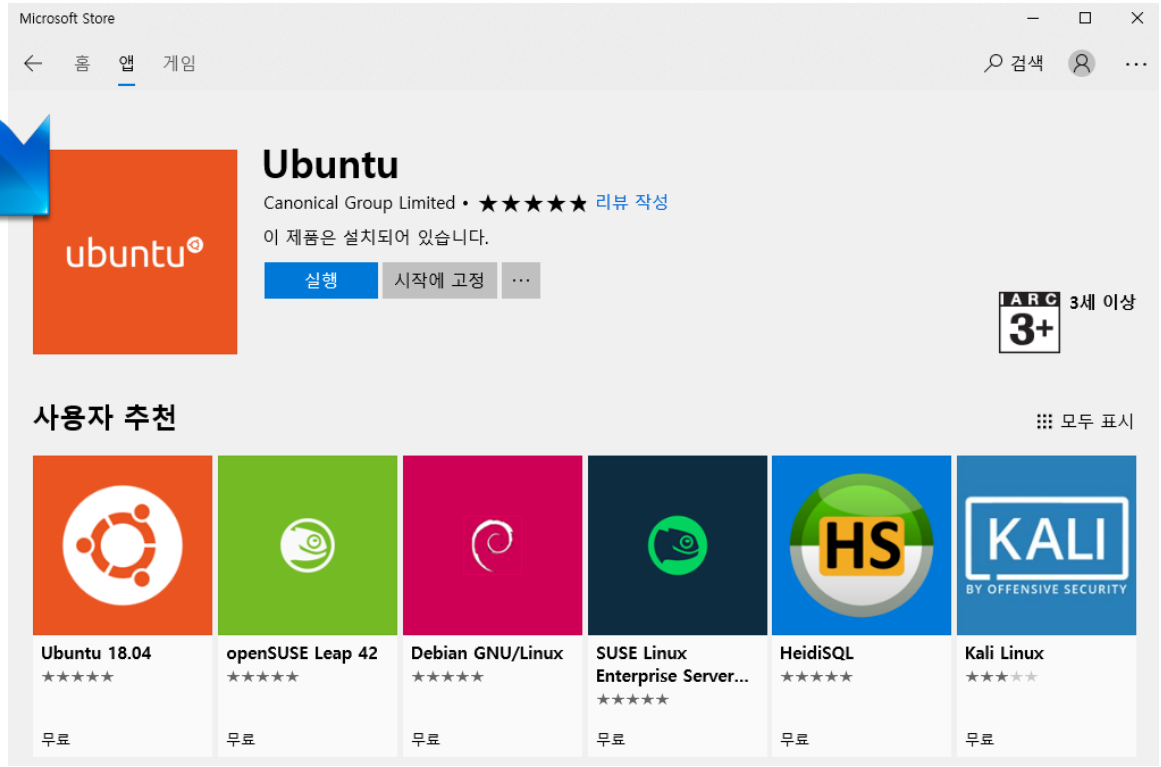
- ✓ 종류(Type)는 데이터의 타입으로 대부분의 경우 float32를 사용한다.
- ✓ Tensor들은 노드(Node)를 지나면서 연산이 이루어지며, 결과값이 나와 변수들이 업데이트 된다.

Source : http://forensics.tistory.com/5

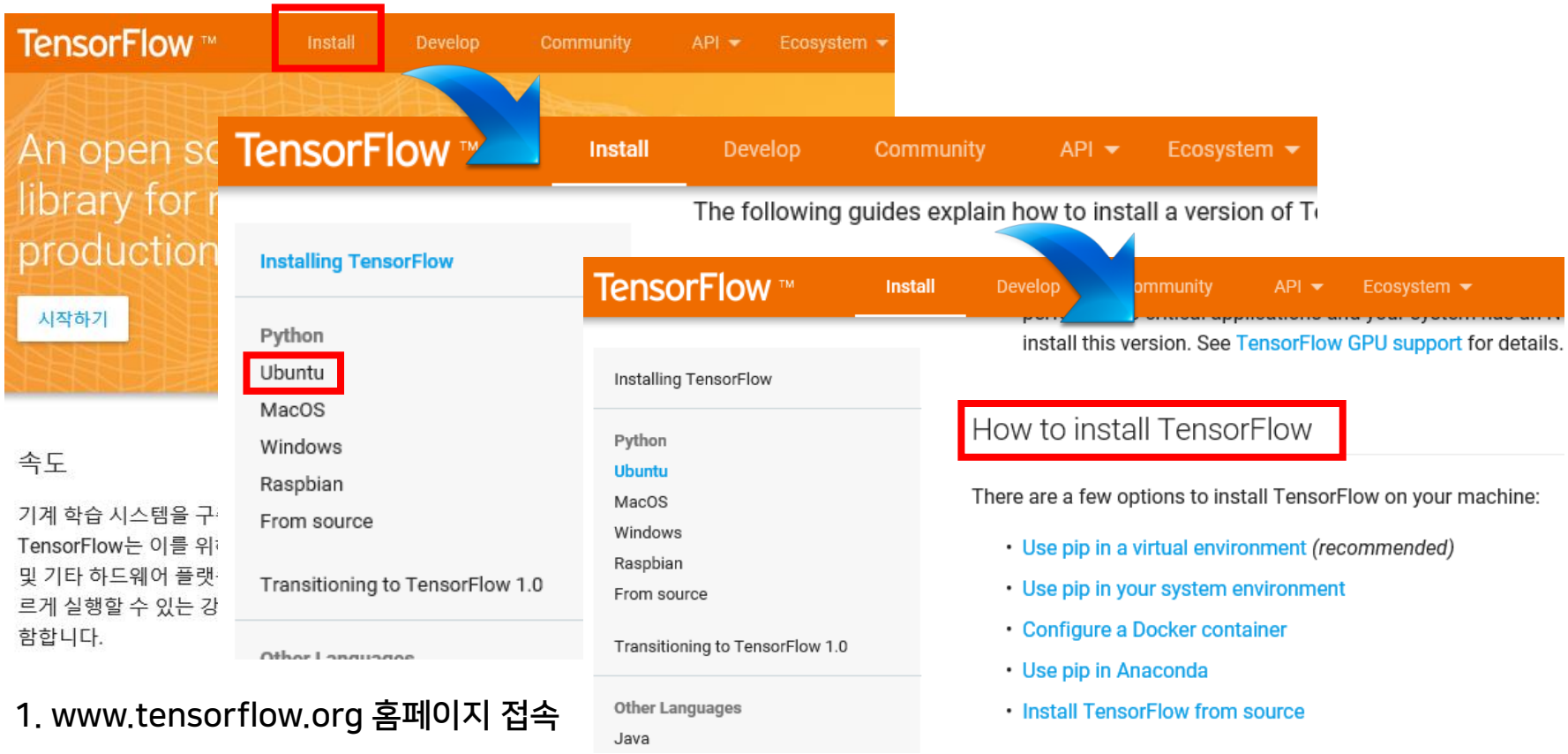
Windows 10에 Ubuntu 설치하기(가상환경 구성)

Microsoft store에서 Ubuntu를 검색하여 설치 후 실행한다.





Tensorflow 설치방법

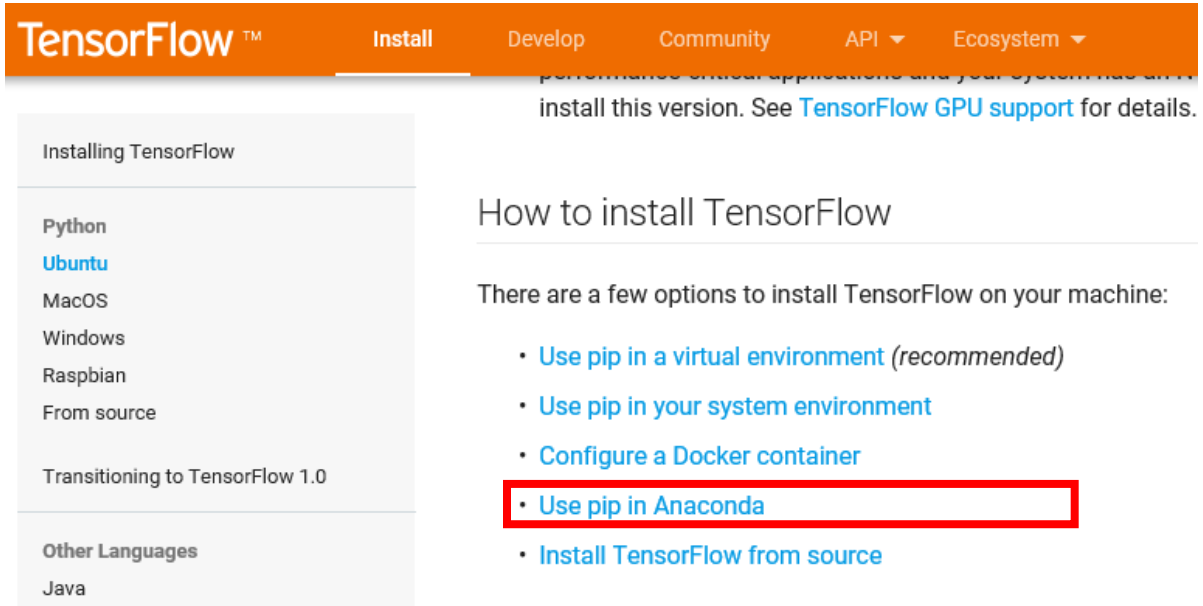


1. www.tensorflow.org 홈페이지 접속
2. Install 탭 클릭
3. Ubuntu 클릭

✓ Tensorflow 실행환경을 Ubuntu로 설정한다, Tensorflow 설치방법은 다양하다.

source : www.tensorflow.org

Tensorflow 설치방법



Tensorflow를 설치하는 매커니즘은 다양하지만 아나콘다 가상환경에서 설치하는 방법(4)을 사용한다.

1. 가상환경에서 pip 사용
2. 시스템 환경에서 pip사용
3. Docker 컨테이너 구성
4. Anaconda에서 pip 사용
5. Source로 부터 설치

source : www.tensorflow.org

Ubuntu 실행

exem@LAPTOP-CIGJQJCV: ~

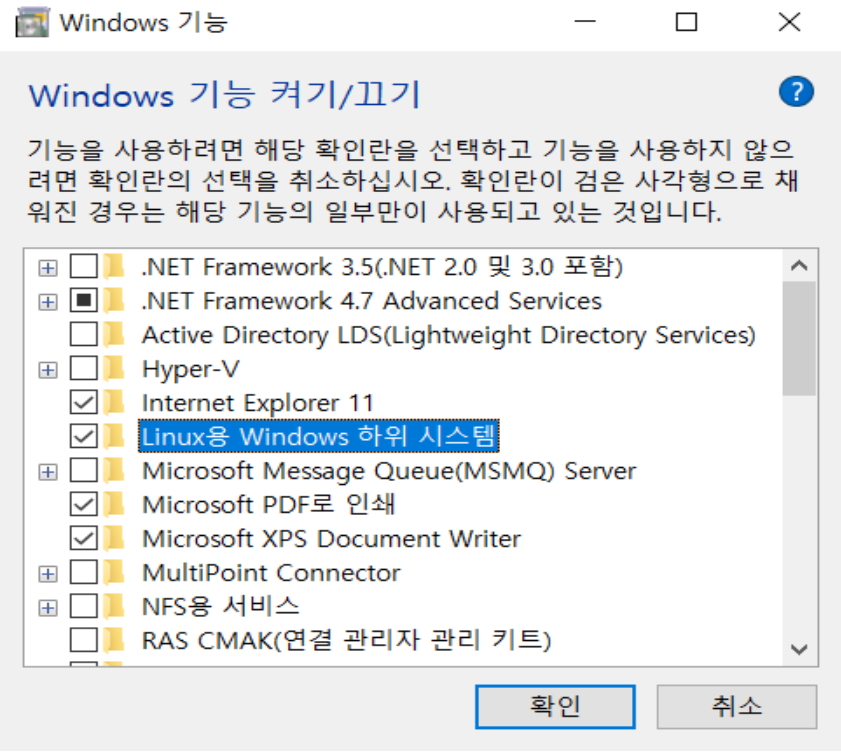
```
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match y
our Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: exem
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

exem@LAPTOP-CIGJQJCV:~$
```

username: exem / password: exem1234

Ubuntu 실행

The Windows Subsystem for Linux optional component is not enabled

Ubuntu 실행 과정에서 만약 위와 같은 메시지가 뜨는 경우 다음과 같이 조치



✓ Windows 기능 켜기/끄기 메뉴에서 'Linux용 Windows 하위 시스템' 체크하여 설치 후 OS 재부팅

원활한 실습을 위한 Ubuntu 업데이트

```
$ sudo apt-get update
```

ck@DESKTOP-EACJR8M: ~

```
ck@DESKTOP-EACJR8M:~$ sudo apt-get update
[sudo] password for ck:
Sorry, try again.
[sudo] password for ck:
Hit:1 http://archive.ubuntu.com/ubuntu xenial InRelease
Get:2 http://security.ubuntu.com/ubuntu xenial-security InRelease [107 kB]
Get:3 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [109 kB]
Get:4 http://archive.ubuntu.com/ubuntu xenial-backports InRelease [107 kB]
Get:5 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 Packages [831 kB]
Get:6 http://archive.ubuntu.com/ubuntu xenial-updates/universe amd64 Packages [680 kB]
Fetched 1,833 kB in 9s (201 kB/s)
Reading package lists... Done
ck@DESKTOP-EACJR8M:~$
```

원활한 실습을 위한 Ubuntu 업데이트

```
$ sudo apt-get upgrade
```

ck@DESKTOP-EACJR8M: ~

```
ck@DESKTOP-EACJR8M:~$ sudo apt-get update
[sudo] password for ck:
Sorry, try again.
[sudo] password for ck:
Hit:1 http://archive.ubuntu.com/ubuntu xenial InRelease
Get:2 http://security.ubuntu.com/ubuntu xenial-security InRelease [107 kB]
Get:3 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [109 kB]
Get:4 http://archive.ubuntu.com/ubuntu xenial-backports InRelease [107 kB]
Get:5 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 Packages [831 kB]
Get:6 http://archive.ubuntu.com/ubuntu xenial-updates/universe amd64 Packages [680 kB]
Fetched 1,833 kB in 9s (201 kB/s)
Reading package lists... Done
ck@DESKTOP-EACJR8M:~$ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages have been kept back:
  open-vm-tools
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
ck@DESKTOP-EACJR8M:~$
```

Python 확인

Ubuntu의 경우 python이 기본적으로 설치되어 있기 때문에 python 설치는 생략하고, 설치되어 있는 버전 확인만 진행한다.

ck@DESKTOP-EACJR8M: ~

```
ck@DESKTOP-EACJR8M:~$ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages have been kept back:
  open-vm-tools
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
ck@DESKTOP-EACJR8M:~$ python3 -V
Python 3.6.5 :: Anaconda, Inc.
ck@DESKTOP-EACJR8M:~$
```

```
$ python3 -V
```

Anaconda 설치

```
$ wget https://repo.anaconda.com/archive/Anaconda3-5.2.0-Linux-x86_64.sh
```

ck@DESKTOP-EACJR8M: ~

```
Python 3.6.5 :: Anaconda, Inc.
ck@DESKTOP-EACJR8M:~$ wget https://repo.anaconda.com/archive/Anaconda3-5.2.0-Linux-x86_64.sh
--2018-08-21 18:05:58--  https://repo.anaconda.com/archive/Anaconda3-5.2.0-Linux-x86_64.sh
Resolving repo.anaconda.com (repo.anaconda.com)... 104.17.107.77, 104.17.109.77, 104.17.110.
77, ...
Connecting to repo.anaconda.com (repo.anaconda.com)|104.17.107.77|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 651745206 (622M) [application/x-sh]
Saving to:  'Anaconda3-5.2.0-Linux-x86_64.sh.1'

Anaconda3-5.2.0-Linux- 100%[===========================>] 621.55M  8.07MB/s    in 81s

2018-08-21 18:07:18 (7.71 MB/s) -  'Anaconda3-5.2.0-Linux-x86_64.sh.1'  saved [651745206/651
745206]

ck@DESKTOP-EACJR8M:~$
```

Anaconda 설치

```
$ sh Anaconda3-5.2.0-Linux-x86_64.sh
```

ck@DESKTOP-EACJR8M: ~

```
ck@DESKTOP-EACJR8M:~$ sh Anaconda3-5.2.0-Linux-x86_64.sh

Welcome to Anaconda3 5.2.0

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
```

Enter키 입력

Anaconda 설치

ck@DESKTOP-EACJR8M: ~

```
openssl
    The OpenSSL Project is a collaborative effort to develop a robust, commercial-grade, ful
l-featured, and Open Source toolkit implementing the Transport Layer Security (TLS) and Secu
re Sockets Layer (SSL) protocols as well as a full-strength general purpose cryptography lib
rary.

pycrypto
    A collection of both secure hash functions (such as SHA256 and RIPEMD160), and various e
ncryption algorithms (AES, DES, RSA, ElGamal, etc.).

pyopenssl
    A thin Python wrapper around (a subset of) the OpenSSL library.

kerberos (krb5, non-Windows platforms)
    A network authentication protocol designed to provide strong authentication for client/s
erver applications by using secret-key cryptography.

cryptography
    A Python library which exposes cryptographic recipes and primitives.


Do you accept the license terms? [yes|no]
[no] >>>
Please answer 'yes' or 'no':'
>>> yes
```

```
$ yes
```

Anaconda 설치

ck@DESKTOP-EACJR8M: ~

```
pyopenssl
    A thin Python wrapper around (a subset of) the OpenSSL library.

kerberos (krb5, non-Windows platforms)
    A network authentication protocol designed to provide strong authentication for client/s
erver applications by using secret-key cryptography.

cryptography
    A Python library which exposes cryptographic recipes and primitives.


Do you accept the license terms? [yes|no]
[no] >>>
Please answer 'yes' or 'no':'
>>> yes

Anaconda3 will now be installed into this location:
/home/ck/anaconda3

  - Press ENTER to confirm the location
  - Press CTRL-C to abort the installation
  - Or specify a different location below

[/home/ck/anaconda3] >>> _
```

Enter키 입력

Anaconda 설치

exem@LAPTOP-CIGJQJCV: ~

```
installing: pytest-doctestplus-0.1.3-py36_0 ...
installing: pywavelets-0.5.2-py36he602eb0_0 ...
installing: scipy-1.1.0-py36hfc37229_0 ...
installing: bkcharts-0.2-py36h735825a_0 ...
installing: dask-0.17.5-py36_0 ...
installing: patsy-0.5.0-py36_0 ...
installing: pytables-3.4.3-py36h02b9ad4_2 ...
installing: pytest-astropy-0.3.0-py36_0 ...
installing: scikit-learn-0.19.1-py36h7aa7ec6_0 ...
installing: astropy-3.0.2-py36h3010b51_1 ...
installing: odo-0.5.1-py36h90ed295_0 ...
installing: scikit-image-0.13.1-py36h14c3975_1 ...
installing: statsmodels-0.9.0-py36h3010b51_0 ...
installing: blaze-0.11.3-py36h4e06776_0 ...
installing: seaborn-0.8.1-py36hfad7ec4_0 ...
installing: anaconda-5.2.0-py36_3 ...
installation finished.
Do you wish the installer to prepend the Anaconda3 install location
to PATH in your /home/exem/.bashrc ? [yes|no]
[no] >>> yes
```

```
$ yes
```

Anaconda 설치

exem@LAPTOP-CIGJQJCV: ~

```
For this change to become active, you have to open a new terminal.

Thank you for installing Anaconda3!

===========================================================================

Anaconda is partnered with Microsoft! Microsoft VSCode is a streamlined
code editor with support for development operations like debugging, task
running and version control.

To install Visual Studio Code, you will need:
  - Administrator Privileges
  - Internet connectivity

Visual Studio Code License: https://code.visualstudio.com/license

Do you wish to proceed with the installation of Microsoft VSCode? [yes|no]
>>> no
```

no (디버깅 기능?, 시간절약을 위해 설치 생략)

Anaconda 가상환경 생성

```
$ conda create –n  tf-py36 pip python=3.6.5
```

```
    ------------------------------------------------------------
                                           Total:        14.7 MB

The following NEW packages will be INSTALLED:

    ca-certificates: 2018.03.07-0
    certifi:         2018.8.13-py36_0
    libedit:         3.1.20170329-h6b74fdf_2
    libffi:          3.2.1-hd88cf55_4
    libgcc-ng:       8.2.0-hdf63c60_0
    libstdcxx-ng:    8.2.0-hdf63c60_0
    ncurses:         6.1-hf484d3e_0
    openssl:         1.0.2p-h14c3975_0
    pip:             10.0.1-py36_0
    python:          3.6.5-hc3d631a_2
    readline:        7.0-ha6073c6_4
    setuptools:      40.0.0-py36_0
    sqlite:          3.24.0-h84994c4_0
    tk:              8.6.7-hc745277_3
    wheel:           0.31.1-py36_0
    xz:              5.2.4-h14c3975_4
    zlib:            1.2.11-ha838bed_2

Proceed ([y]/n)?
```

y 입력

Anaconda 가상환경 생성

ck@DESKTOP-EACJR8M: ~

```
Downloading and Extracting Packages
libstdcxx-ng-8.2.0   |  2.9 MB | ########################################################################## | 100%
certifi-2018.8.13    |  138 KB | ########################################################################## | 100%
setuptools-40.0.0    |  551 KB | ########################################################################## | 100%
openssl-1.0.2p       |  3.5 MB | ########################################################################## | 100%
libgcc-ng-8.2.0      |  7.6 MB | ########################################################################## | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use:
# > source activate tf-py36
#
# To deactivate an active environment, use:
# > source deactivate
#

ck@DESKTOP-EACJR8M:~$
```

생성 완료

Anaconda 가상환경 활성화

```
$ source activate tf-py36
```

ck@DESKTOP-EACJR8M: ~

```
#

ck@DESKTOP-EACJR8M:~$ source activate tf-py36
(tf-py36) ck@DESKTOP-EACJR8M:~$
```

Tensorflow 실행 환경 구축

Anaconda에 jupyter notebook 및 tensorflow 설치

```
(tf-py36) $ pip install jupyter notebook
(tf-py36) $ pip install tensorflow
```

Anaconda에 jupyter notebook 및 tensorflow 설치

```
(tf-py36) $ jupyter-notebook --generate-config
```

ck@DESKTOP-EACJR8M: ~

```
(tf-py36) ck@DESKTOP-EACJR8M:~$ jupyter-notebook --generate-config
Overwrite /home/ck/.jupyter/jupyter_notebook_config.py with default config? [y/N]y
```

y 입력

Anaconda에 jupyter notebook 및 tensorflow 설치

```
(tf-py36) $ jupyter-notebook password
```

```
ck@DESKTOP-EACJR8M: ~
(tf-py36) ck@DESKTOP-EACJR8M:~$ jupyter-notebook --generate-config
Overwrite /home/ck/.jupyter/jupyter_notebook_config.py with default config? [y/N]y
Writing default config to: /home/ck/.jupyter/jupyter_notebook_config.py
(tf-py36) ck@DESKTOP-EACJR8M:~$ jupyter-notebook password
Enter password:
Verify password:
[NotebookPasswordApp] Wrote hashed password to /home/ck/.jupyter/jupyter_notebook_config.jso
n
(tf-py36) ck@DESKTOP-EACJR8M:~$
```

password 입력 ex) oracle

Anaconda에 jupyter notebook 및 tensorflow 설치

```
(tf-py36) $ python -m ipykernel install --user --name [virtualEnv] --display-name "[displayKenrelName]"
```

ck@DESKTOP-EACJR8M: ~

```
(tf-py36) ck@DESKTOP-EACJR8M:~$ python -m ipykernel install --user --name tf-py36 --display-name "[tf-py36]"
Installed kernelspec tf-py36 in /home/ck/.local/share/jupyter/kernels/tf-py36
(tf-py36) ck@DESKTOP-EACJR8M:~$
```

[virtualEnv] : 생성한 가상환경 이름

[displayKernelName] : jupyter notebook에 생성할 kernel 이름

Anaconda에 jupyter notebook 및 tensorflow 설치

```
(tf-py36) $ jupyter kernelspec list
```

ck@DESKTOP-EACJR8M: ~

```
[C 18:30:45.706 NotebookApp] Shutdown confirmed
[I 18:30:45.711 NotebookApp] Shutting down 0 kernels
(tf-py36) ck@DESKTOP-EACJR8M:~$ python -m ipykernel install --user --name tf-py36 --display-
name "[tf-py36]"
Installed kernelspec tf-py36 in /home/ck/.local/share/jupyter/kernels/tf-py36
(tf-py36) ck@DESKTOP-EACJR8M:~$ jupyter kernelspec list
Available kernels:
  tf-py36    /home/ck/.local/share/jupyter/kernels/tf-py36
  python3    /home/ck/anaconda3/envs/tf-py36/share/jupyter/kernels/python3
(tf-py36) ck@DESKTOP-EACJR8M:~$
```

✓ 생성한 kernel 확인

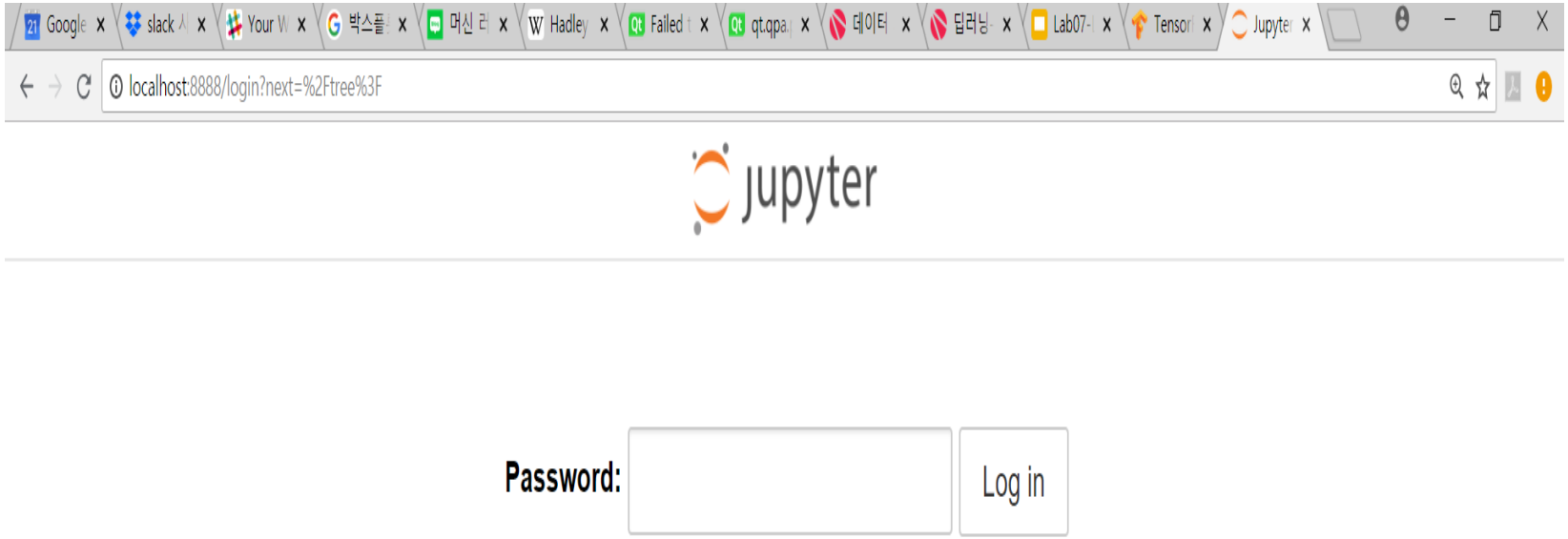
jupyter notebook에서 tensorflow 확인

```
(tf-py36) $ jupyter notebook
```

```
(tf-py36) ck@DESKTOP-EACJR8M:~$ jupyter notebook
[I 18:38:13.805 NotebookApp] Serving notebooks from local directory: /home/ck
[I 18:38:13.806 NotebookApp] The Jupyter Notebook is running at:
[I 18:38:13.809 NotebookApp] http://localhost:8888/
[I 18:38:13.811 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
```

✓ http://localhost:8888/ URL로 브라우저를 이용하여 접속

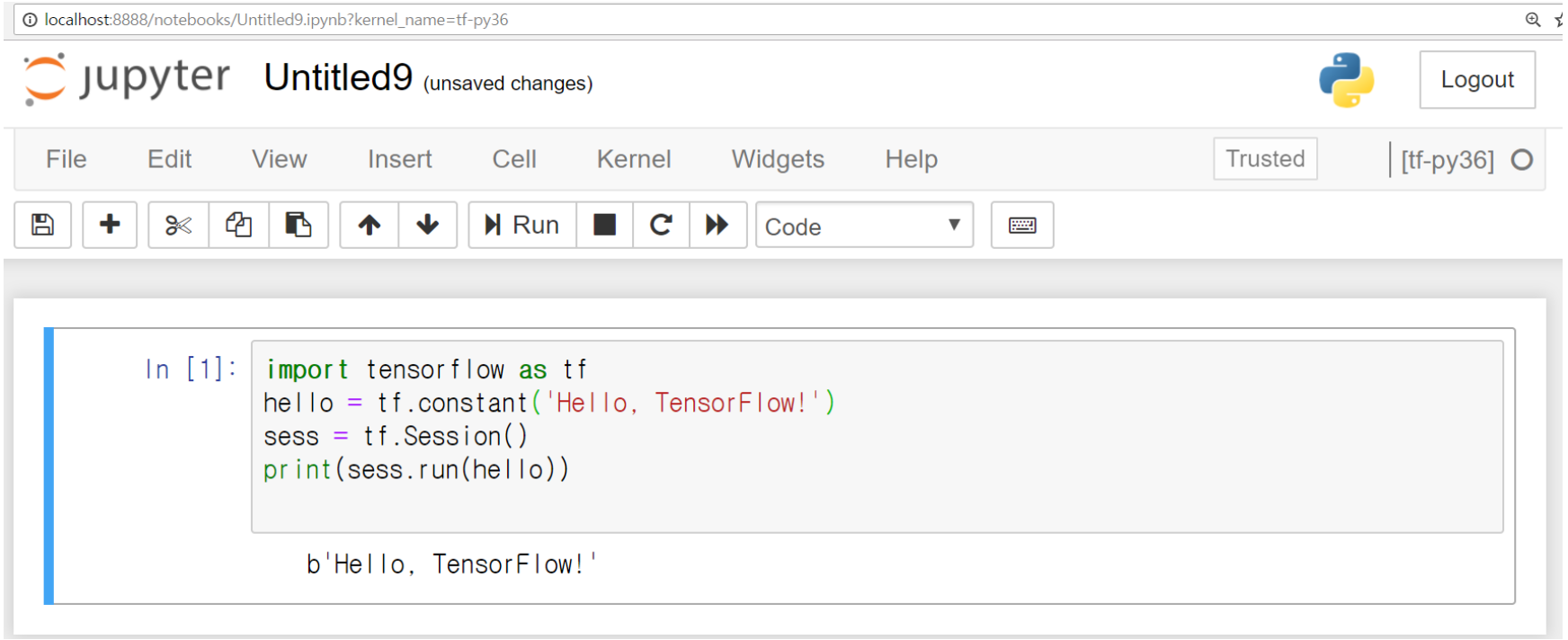
Tensorflow 실행 환경 구축



✓ http://localhost:8888/ URL 접속 후 password 입력

jupyter notebook에서 tensorflow 확인

✓ 우측에 new 버튼 클릭 후 [tf-py36] 선택, 아래의 메시지 입력 후 Run 클릭




```
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
print(sess.run(hello))
```

기타) numpy 등 라이브러리 설치 및 테스트

```
(tf-py36) $ pip install numpy pandas matplotlib

(tf-py36) $ import numpy as np
```

```
(tf-py36) ck@DESKTOP-EACJR8M:~$ import numpy as np
The program 'import' can be found in the following packages:
 * imagemagick
 * graphicsmagick-imagemagick-compat
Try: sudo apt install <selected package>
```

```
(tf-py36) $ jupyter notebook
```

✓ 아나콘다 가상환경에선 실행이 되지 않으므로 jupyter notebook으로 실행한다.

Source : https://www.tensorflow.org/install/install_linux , https://www.tensorflow.org/install/install_linux#the_url_of_the_tensorflow_python_package

기타) numpy 등 라이브러리 설치 및 테스트

In [1]:
```
import numpy as np

list1 = [1, 2, 3, 4]
a = np.array(list1)
print(a.shape) # (4,)

b = np.array([[1, 2, 3], [4, 5, 6]])
print(b.shape) # (2, 3)
print(b[0,0]) # 1
```

```
(4,)
(2, 3)
1
```


```
import numpy as np

list1 = [1, 2, 3, 4]
a = np.array(list1)
print(a.shape) # (4, )

b = np.array([[1, 2, 3], [4, 5, 6]])
print(b.shape) # (2, 3)
print(b[0, 0]) # 1
```


Source : https://www.tensorflow.org/install/install_linux ,
https://www.tensorflow.org/install/install_linux#the_url_of_the_tensorflow_python_package

Computational Graph 예시코드

1 tensorflow library를 사용한 그래프 구축

```
import tensorflow as tf

# X and Y data
node1 = tf.constant(2.0, tf.float32)
node2 = tf.constant(3.0) # also tf.float32 implicitly
node3 = tf.add(node1, node2)
```

2 그래프 실행

3 그래프에 결과값을 반환하고 변수를 업데이트

```
sess = tf.Session()
print("sess.run(node1, node2): ", sess.run([node1, node2]))
print("sess.run(node3): ", sess.run(node3))
```

✓ Tensorflow 함수를 사용하기 위해선 모듈을 import 해야 한다.

Computational Graph 예시코드

1 tensorflow library를 사용한 그래프 구축

```
import tensorflow as tf

# X and Y data
node1 = tf.constant(2.0, tf.float32)
node2 = tf.constant(3.0) # also tf.float32 implicitly
node3 = tf.add(node1, node2)
```

2 그래프 실행

3 그래프에 결과값을 반환하고 변수를 업데이트

```
sess = tf.Session()
print("sess.run(node1, node2): ", sess.run([node1, node2]))
print("sess.run(node3): ", sess.run(node3))
```

✓ 상수를 선언할 땐 tf.constant() 함수를 사용한다. 암묵적으로 float32 타입이 적용된다.

* 변수를 선언할 땐 tf.Variable() 함수를 사용한다.

Computational Graph 예시코드

1 tensorflow library를 사용한 그래프 구축

```
import tensorflow as tf

# X and Y data
node1 = tf.constant(2.0, tf.float32)
node2 = tf.constant(3.0) # also tf.float32 implicitly
node3 = tf.add(node1, node2)
```

2 그래프 실행

3 그래프에 결과값을 반환하고 변수를 업데이트

```
sess = tf.Session()
print("sess.run(node1, node2): ", sess.run([node1, node2]))
print("sess.run(node3): ", sess.run(node3))
```

✓ 그래프를 실행하기 위해선 먼저 tf.Session()으로 세션을 시작한다.

Computational Graph 예시코드

1 tensorflow library를 사용한 그래프 구축

```
import tensorflow as tf

# X and Y data
node1 = tf.constant(2.0, tf.float32)
node2 = tf.constant(3.0) # also tf.float32 implicitly
node3 = tf.add(node1, node2)
```

2 그래프 실행

3 그래프에 결과값을 반환하고 변수를 업데이트

```
sess = tf.Session()
print("sess.run(node1, node2): ", sess.run([node1, node2]))
print("sess.run(node3): ", sess.run(node3))
```

✓ 그래프에 결과값을 반환하기 위해서는 sess.run()으로 세션을 실행한다.

Jupyter notebook 실행결과

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

# X and Y data
node1 = tf.constant(2.0, tf.float32)
node2 = tf.constant(3.0) # also tf.float32 implicitly
node3 = tf.add(node1, node2)

sess = tf.Session()
print("sess.run(node1, node2): ", sess.run([node1, node2]))
print("sess.run(node3): ", sess.run(node3))
```

```
sess.run(node1, node2):  [2.0, 3.0]
sess.run(node3):  5.0
```

Placeholder 예시

```
a = tf.placeholder(tf.float32)
b = tf.placeholder(tf.float32)
adder_node = a + b

print(sess.run(adder_node, feed_dict={a: 3, b: 4.5}))
print(sess.run(adder_node, feed_dict={a: [1, 3], b: [2, 4]}))
```

```
In [5]: a = tf.placeholder(tf.float32)
        b = tf.placeholder(tf.float32)
        adder_node = a + b

        print(sess.run(adder_node, feed_dict={a: 3, b: 4.5}))
        print(sess.run(adder_node, feed_dict={a: [1, 3], b: [2, 4]}))

        7.5
        [3. 7.]
```

✓ Placeholder는 값을 담을 수 있는 그릇으로 이후에 세션을 실행할 때 원하는 값을 넣고 실행 할 수 있다.

Placeholder 예시

```
a = tf.placeholder(tf.float32)
b = tf.placeholder(tf.float32)
adder_node = a + b

print(sess.run(adder_node, feed_dict={a: 3, b: 4.5}))
print(sess.run(adder_node, feed_dict={a: [1, 3], b: [2, 4]}))
```

```
In [5]: a = tf.placeholder(tf.float32)
        b = tf.placeholder(tf.float32)
        adder_node = a + b

        print(sess.run(adder_node, feed_dict={a: 3, b: 4.5}))
        print(sess.run(adder_node, feed_dict={a: [1, 3], b: [2, 4]}))

        7.5
        [3. 7.]
```

✓ Placeholder에 값을 넣을 때는 feed_dict 옵션을 사용한다.

3.Linear Regression

Regression이란?

✓ 그래프 상에 무작위하게 흩어져있는 데이터의 경향을 찾아 선으로 나타내는 것, 선이 직선형태일 경우 linear regression이라 하며 prediction(예측)의 한 방법을 뜻한다.

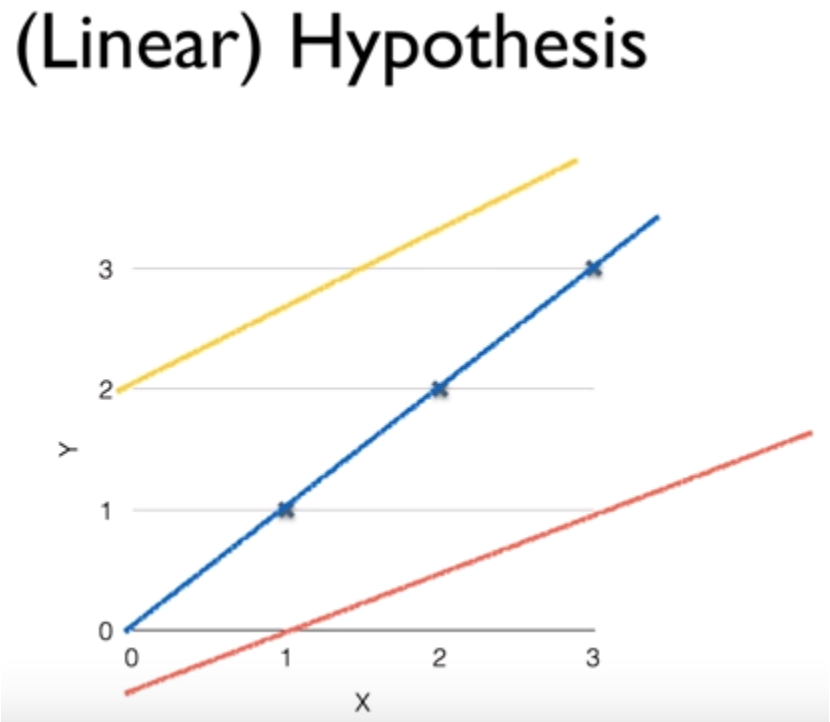
X	Y
1	1
2	2
3	3

X가 4이면 Y의 값은 몇 일까?

source : https://blog.naver.com/dpfkdlt/221247604755

Hypothesis란?

✓ 주어진 데이터를 표현하는 여러 가설(Hypothesis)이 존재한다. 각 직선은 H(x)=Wx+b 로 표현가능하다. W는 Weight(가중치) ,b는 bias(편향)

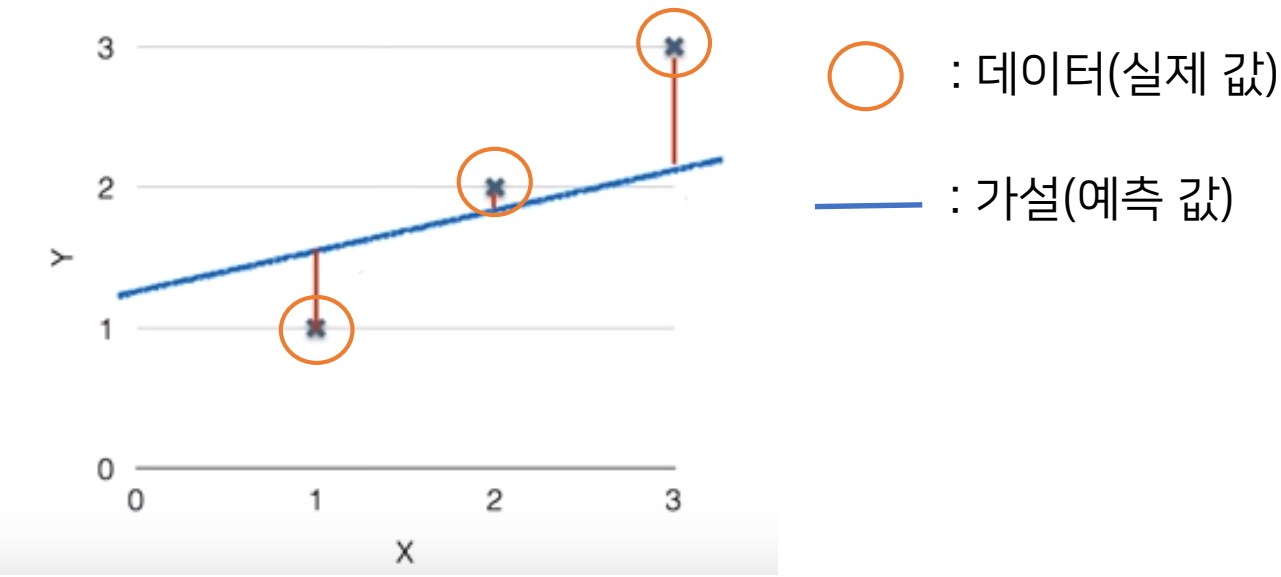


여러 가설 중에 데이터를 가장 잘 표현하는 직선을 찾는 것

source : https://blog.naver.com/dpfkdlt/221247604755

Cost 란?

✓ Hypothesis(예측 값)와 데이터(실제 값)의 차이를 제곱하여 평균 한 것



$$H(x) = Wx + b \qquad cost(W, b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

✓ Cost가 0이면 Hypothesis와 데이터가 정확하게 일치한다

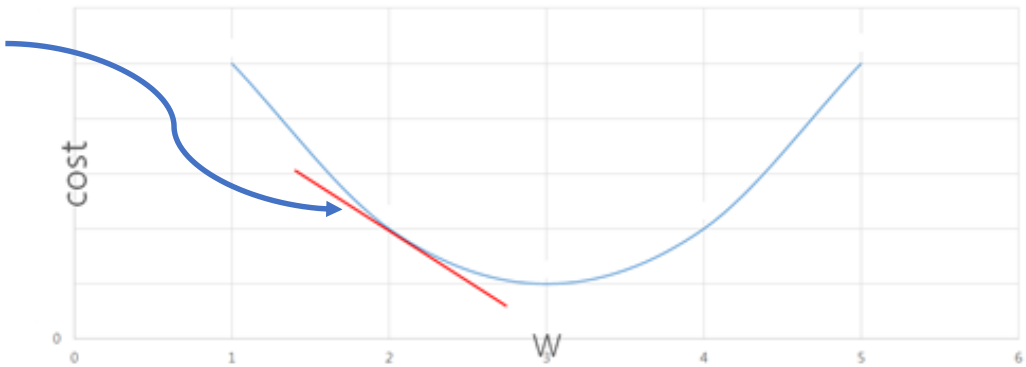
source : https://blog.naver.com/dpfkdlt/221247604755

Linear Regression에서 학습이란?

$$H(x) = Wx + b \qquad cost(W, b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

- ✓ Cost를 최소화하는 W(weight)와 b(bias)를 구하는 것
- ✓ Cost를 최소화하는 W와 b를 구하는 학습방법으론 Gradient Descent algorithm(경사하강법)이 많이 사용된다.
- ✓ 기울기(경사도)는 cost function을 미분하여 구한다.

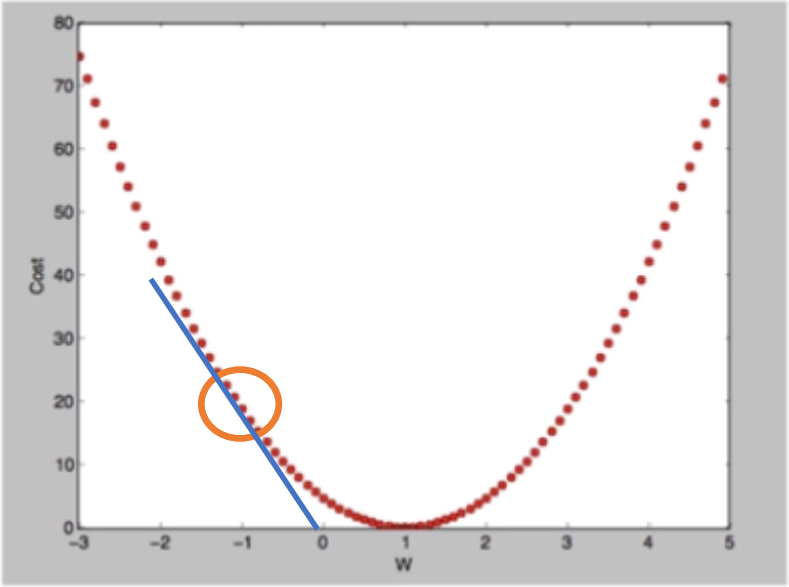
기울기(경사도)



source : https://blog.naver.com/eveoftheday/221301077378

Cost가 0에 가까워지는 점을 어떻게 찾는가?

- ✓ Cost함수의 w에 대한 미분 값(기울기)을 통해 cost의 저점을 찾아간다.



기울기

$$W := W - \alpha \boxed{\frac{\partial}{\partial W} cost(W)}$$

α : learning rate(학습율)

∂ : 미분 기호

- ✓ W에 대해 미분한 값을 기울기라고 하자. W=-1일 때, cost=20이 되고 기울기는 음수(-)이다. cost를 줄이기 위해선 W에서 기울기 값을 빼서 오른쪽으로 이동해야 한다.
- ✓ W에 따라 cost가 변하므로 미분을 통해 W값을 업데이트 한다.

source : https://blog.naver.com/dpfkdlt/221247604755

Gradient Descent(경사하강법)

✓ Cost함수의 w에 대한 미분값을 통해 cost의 저점을 찾아가는 방식

$$W := W - \alpha \frac{\partial}{\partial W} \frac{1}{2m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)})^2$$

: x $\frac{1}{2}$ 은 미분했을 때 값이 깔끔하게 정리되게 하기위함

$$W := W - \alpha \frac{1}{2m} \sum_{i=1}^{m} 2(Wx^{(i)} - y^{(i)})x^{(i)}$$

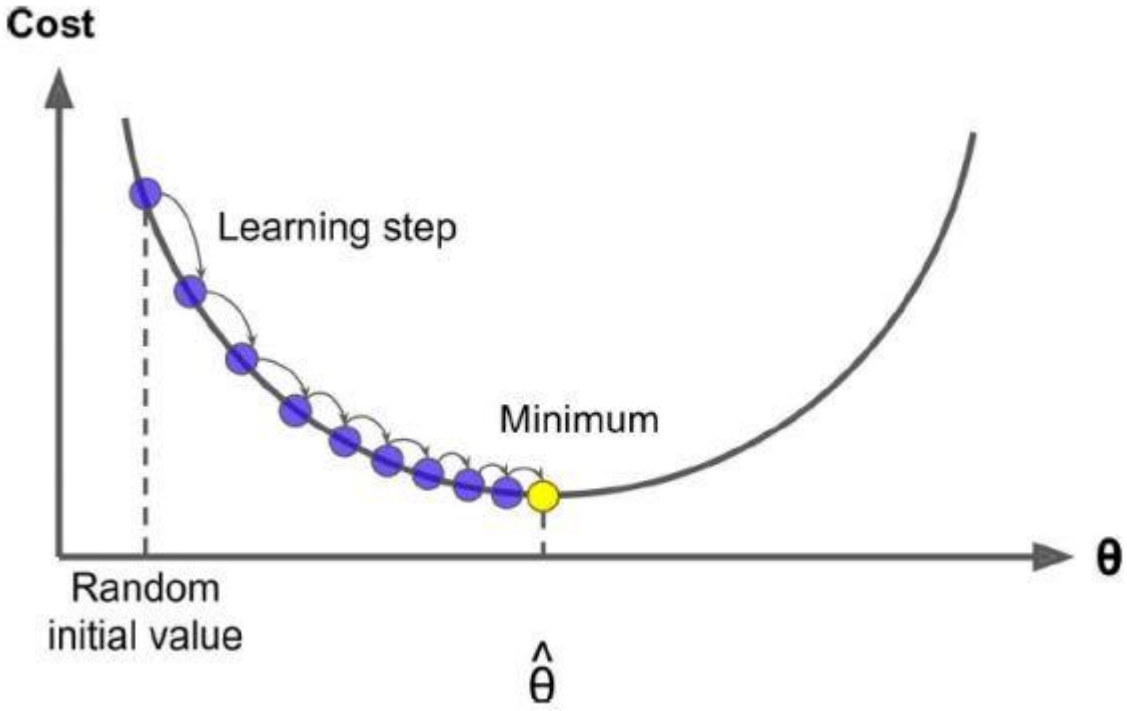
: W에 대해 Cost함수 미분

$$W := W - \alpha \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)})x^{(i)}$$

: Gradient Descent 수식

source : https://blog.naver.com/dpfkdlt/221247604755

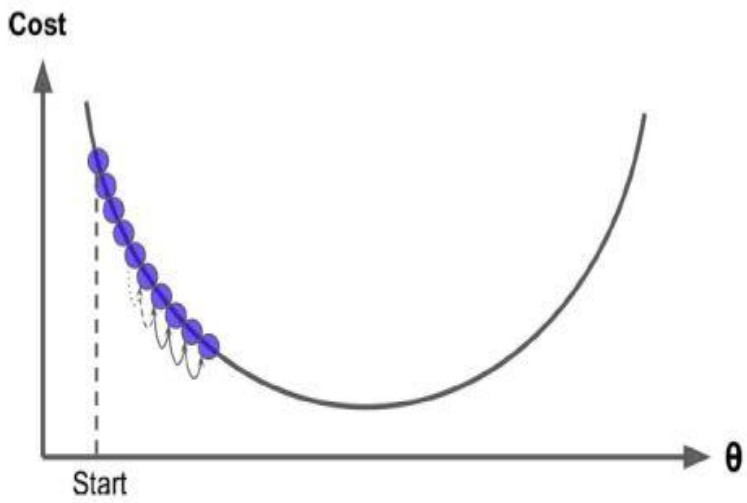
Learning rate란?



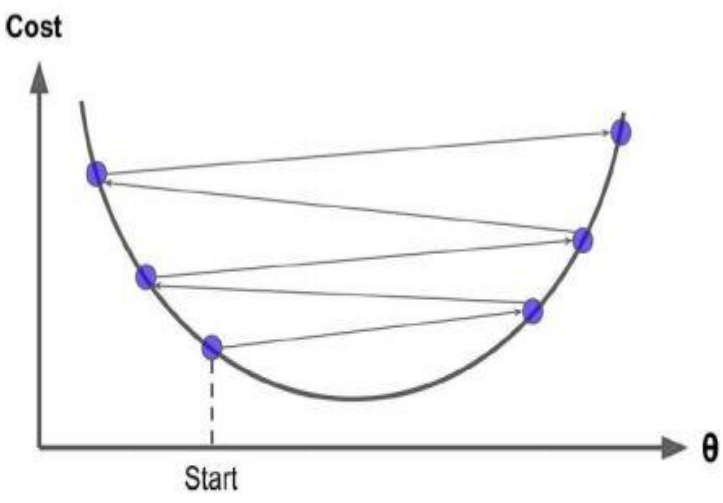
✓ Learning rate란 학습율로 cost함수에서 최소값을 향해 내려가는 step의 크기를 결정한다.

source : http://dokim.tistory.com/category/Machine%20Learning/CNN%20%28ConvNets%29

적절한 Learning rate 값의 선택



Small learning rate

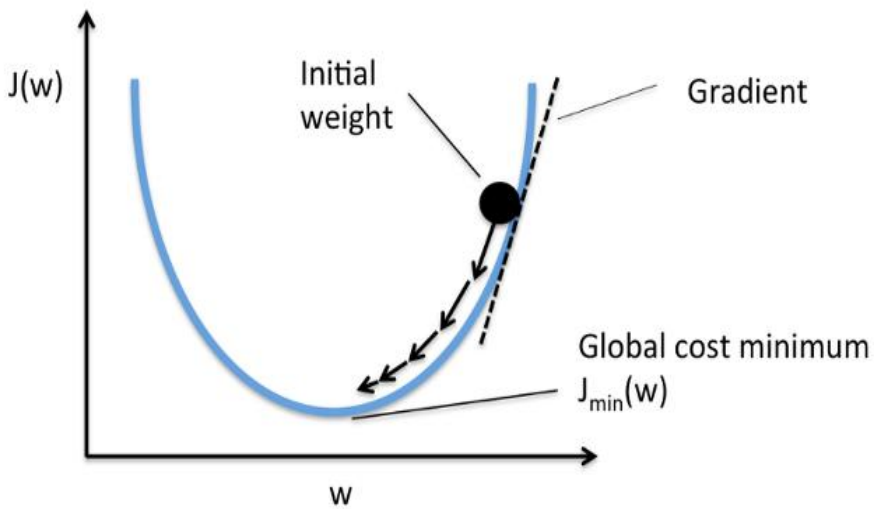


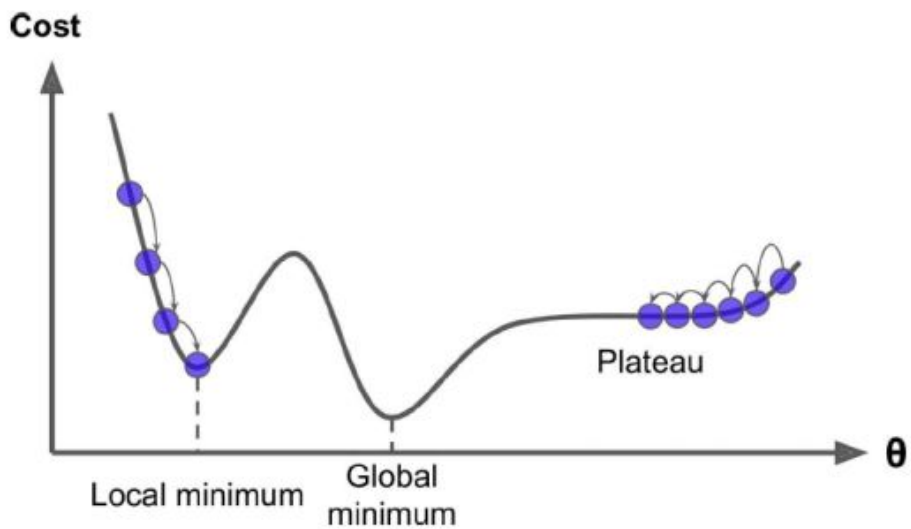
Large learning rate

- ✓ Learning rate가 너무 작은 경우, 학습하는데 오랜 시간이 걸린다. 또한 Cost 함수가 non-convex한 경우 local minimum으로 수렴할 수 있다.
- ✓ Learning rate가 너무 큰 경우, cost의 최저점을 지나 반대편 경사로 이동하며 심할 경우 ∞값으로 갈 수도 있다. 이 경우를 overshooting 현상이라고 한다.

source : http://dokim.tistory.com/category/Machine%20Learning/CNN%20%28ConvNets%29

(참고) convex와 non-convex 함수





- ✓ 왼쪽 cost 함수는 일반적인 convex 함수로 gradient descent를 통해 cost의 저점을 찾아간다.
- ✓ 오른쪽 cost 함수는 non-convex 함수로 gradient descent를 통한 cost의 저점이 local minimum이 될 수도 있다.

source : http://dokim.tistory.com/category/Machine%20Learning/CNN%20%28ConvNets%29, https://www.simplilearn.com/regression-machine-learning-tutorial

tensorflow를 이용한 Linear regression 구현

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

learning_rate = 0.01
training_cnt = 10

train_X = np.array([1, 2, 3])
train_Y = np.array([1, 2, 3])

X = tf.placeholder("float")
Y = tf.placeholder("float")

W = tf.Variable([.0], tf.float32, name = "weight")
b = tf.Variable([.0], tf.float32, name = "bias")

pred = tf.add(tf.multiply(X, W), b)

cost = tf.reduce_mean(tf.pow(pred-Y, 2))
optimizer = tf.train.GradientDescentOptimizer(learning_rate)
op_train = optimizer.minimize(cost)

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

for epoch in range(training_cnt):
    r_cost, r_W, r_b, _ = sess.run([cost, W, b, op_train], feed_dict = {X: train_X, Y: train_Y})
    print("Running count : " '%04d' % (epoch+1), "Training cost =", r_cost, "W =", r_W, "b =", r_b)

print("Optimization Finished!")
print("Running count : " '%04d' % (epoch+1), "Training cost =", r_cost, "W =", r_W, "b =", r_b)
```

https://github.com/aymericdamien/TensorFlow-Examples/blob/master/examples/2_BasicModels/linear_regression.py

모델 구축(Build graph)

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

learning_rate = 0.01
training_cnt = 10

train_X = np.array([1, 2, 3])
train_Y = np.array([1, 2, 3])

X = tf.placeholder("float")
Y = tf.placeholder("float")

W = tf.Variable([.0], tf.float32, name = "weight")
b = tf.Variable([.0], tf.float32, name = "bias")

pred = tf.add(tf.multiply(X, W), b)

cost = tf.reduce_mean(tf.pow(pred-Y, 2))
optimizer = tf.train.GradientDescentOptimizer(learning_rate)
op_train = optimizer.minimize(cost)
```

모델 실행(run/update)

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

for epoch in range(training_cnt):
    r_cost, r_W, r_b, _ = sess.run([cost, W, b, op_train], feed_dict = {X: train_X, Y: train_Y})
    print("Running count : " '%04d' % (epoch+1), "Training cost =", r_cost, "W =", r_W, "b =", r_b)

print("Optimization Finished!")
print("Running count : " '%04d' % (epoch+1), "Training cost =", r_cost, "W =", r_W, "b =", r_b)
```

모델 구축(Build graph)

```
# 필요한 library import
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
```

* 라이브러리 설치

```
:~$ pip install tensorflow
```

모델 구축(Build graph)

```
# 파라메터값 설정
learning_rate = 0.01
training_cnt = 10

# Training Data
train_X = np.array([1, 2, 3])
train_Y = np.array([1, 2, 3])

# tf Graph Input
X = tf.placeholder("float")
Y = tf.placeholder("float")
```

모델 구축(Build graph)

$$H(x) = Wx + b$$

```
# Set model weights
W = tf.Variable([.0], tf.float32, name = "weight")
b = tf.Variable([.0], tf.float32, name = "bias")

# Construct a linear model (Wx+b)
pred = tf.add(tf.multiply(X, W), b)
```

$$cost(W, b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

```
# cost/loss function
cost = tf.reduce_mean(tf.pow(pred-Y, 2))
```

모델 구축(Build graph)

$$cost(W,b) = \boxed{\frac{1}{m}\sum_{i=1}^{m}}\left(H\left(x^{(i)}\right) - y^{(i)}\right)^2$$

- reduce_mean?

```
t = [ 1., 2., 3., 4.]
r_m = tf.reduce_mean(t)
```

```
sess = tf.Session()
sess.run(tf.global_variables_initializer())
sess.run(r_m)
```

2.5

모델 구축(Build graph)

- GradientDecent

cost를 최소하 하기

```
cost = tf.reduce_mean(tf.pow(pred-Y, 2))
optimizer = tf.train.GradientDescentOptimizer(learning_rate)
op_train = optimizer.minimize(cost)
```

모델 실행(run/update)

```
# 세션을 열고 그래프 실행
sess = tf.Session()
# 변수들에 초기값 할당
init = tf.global_variables_initializer()
sess.run(init)
```

```
for epoch in range(training_cnt):
    r_cost, r_W, r_b, _ = sess.run([cost, W, b, op_train], feed_dict = {X: train_X, Y: train_Y})
    print("Running count : " '%04d' % (epoch+1), "Training cost =", r_cost, "W =", r_W, "b =", r_b)

print("Optimization Finished!")
print("Running count : " '%04d' % (epoch+1), "Training cost =", r_cost, "W =", r_W, "b =", r_b)
```

https://github.com/aymericdamien/TensorFlow-Examples/blob/master/examples/2_BasicModels/linear_regression.py

모델 실행(run/update)

```
Running count : 0001 Training cost = 4.6666665 W = [0.09333334] b = [0.04]
Running count : 0002 Training cost = 3.6927407 W = [0.17635557] b = [0.07546666]
Running count : 0003 Training cost = 2.9228852 W = [0.2502104] b = [0.10690311]
Running count : 0004 Training cost = 2.314336 W = [0.31591463] b = [0.13475662]
Running count : 0005 Training cost = 1.8332924 W = [0.37437233] b = [0.1594249]
Running count : 0006 Training cost = 1.4530338 W = [0.42638725] b = [0.18126151]
Running count : 0007 Training cost = 1.1524409 W = [0.47267398] b = [0.20058079]
Running count : 0008 Training cost = 0.9148195 W = [0.51386786] b = [0.21766222]
Running count : 0009 Training cost = 0.72697395 W = [0.5505337] b = [0.23275426]
Running count : 0010 Training cost = 0.5784736 W = [0.58317375] b = [0.24607782]
Optimization Finished!
Running count : 0010 Training cost = 0.5784736 W = [0.58317375] b = [0.24607782]
```

모델 실행(run/update)

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

learning_rate = 0.01
training_cnt = 100        # 100 번 훈련
```

```
for epoch in range(training_cnt):
    r_cost, r_W, r_b, _ = sess.run([cost, W, b, op_train], feed_dict = {X: train_X, Y: train_Y})
    print("Running count : " '%04d' % (epoch+1), "Training cost =", r_cost, "W =", r_W, "b =", r_b)

print("Optimization Finished!")
print("Running count : " '%04d' % (epoch+1), "Training cost =", r_cost, "W =", r_W, "b =", r_b)
```

```
Optimization Finished!
Running count : 0100 Training cost = 0.012100749 W = [0.8725385] b = [0.2897323]
```

def(함수) 사용하여 코딩하기

```
def make_graph() :
    global X, Y, W, b, pred ,cost,optimizer, op_train

    X = tf.placeholder("float")
    Y = tf.placeholder("float")

    W = tf.Variable([.0], tf.float32, name = "weight")
    b = tf.Variable([.0], tf.float32, name = "bias")

    pred = tf.add(tf.multiply(X, W), b)

    cost = tf.reduce_mean(tf.pow(pred-Y, 2))
    optimizer = tf.train.GradientDescentOptimizer(learning_rate)
    op_train = optimizer.minimize(cost)
```

def(함수) 사용하여 코딩하기

```
def run_graph() :

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

    for epoch in range(training_cnt):
        r_cost, r_W, r_b, _ = sess.run([cost, W, b, op_train], feed_dict = {X: train_X, Y: train_Y})
        print("Running count : " '%04d' % (epoch+1), "Training cost =", r_cost, "W =", r_W, "b =", r_b)

    print("Optimization Finished!")
    print("Running count : " '%04d' % (epoch+1), "Training cost =", r_cost, "W =", r_W, "b =", r_b)
```

def(함수) 사용하여 코딩하기

```
make_graph()
```

```
run_graph()
```

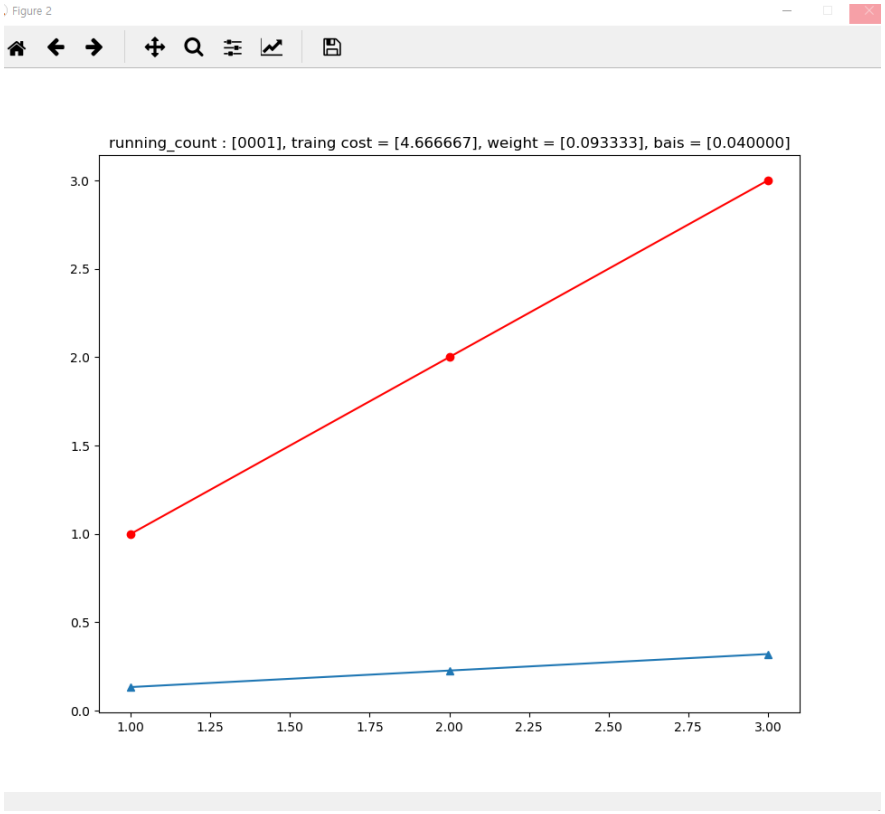
```
Running count : 0001 Training cost = 4.6666665 W = [0.09333334] b = [0.04]
Running count : 0002 Training cost = 3.6927407 W = [0.17635557] b = [0.07546666]
Running count : 0003 Training cost = 2.9228852 W = [0.2502104] b = [0.10690311]
Running count : 0004 Training cost = 2.314336 W = [0.31591463] b = [0.13475662]
Running count : 0005 Training cost = 1.8332924 W = [0.37437233] b = [0.1594249]
Running count : 0006 Training cost = 1.4530338 W = [0.42638725] b = [0.18126151]
Running count : 0007 Training cost = 1.1524409 W = [0.47267398] b = [0.20058079]
Running count : 0008 Training cost = 0.9148195 W = [0.51386786] b = [0.21766222]
Running count : 0009 Training cost = 0.72697395 W = [0.5505337] b = [0.23275426]
Running count : 0010 Training cost = 0.5784736 W = [0.58317375] b = [0.24607782]
Optimization Finished!
Running count : 0010 Training cost = 0.5784736 W = [0.58317375] b = [0.24607782]
```

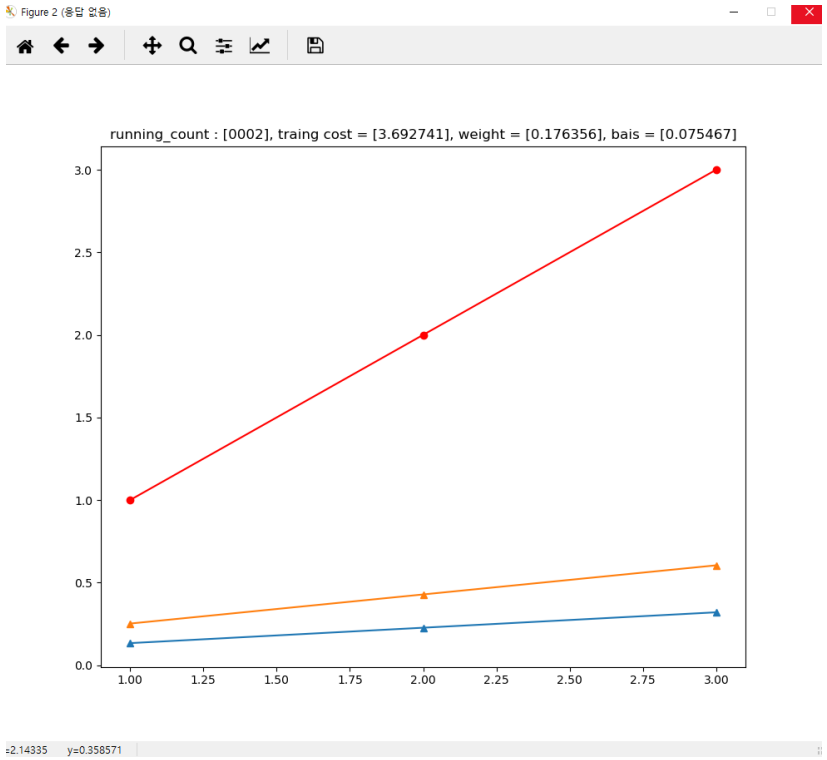
시각화 – 10번 학습

```
# Graphic display
plt.figure(figsize=(10,8))
plt.plot(train_X, train_Y, label='y_data',marker="o",color='r',linewidth=1.5)

pagenation('....')

# 학습( training_cnt 횟수만큼)
for epoch in range(training_cnt):    # training_cnt = 10
    r_cost, r_W, r_b, _ = sess.run([cost, W, b, op_train], feed_dict = {X: train_X, Y: train_Y})

    if (epoch+1) % display_step == 0:  # 나머지가 0 이될 때 결과 출력
        print("Running count : " '%04d' % (epoch+1), "Training cost =", r_cost, "W =", r_W, "b =", r_b)

        # Graphic display
         plt.plot(train_X, sess.run(W) * train_X + sess.run(b), marker = '^')
         plt.title('running_count : [%04d], traing cost = [%.6f], weight = [%.6f], bais = [%.6f]' % (epoch+1, r_cost, r_W, r_b))
         plt.pause(0.1)

         pagenation('....')
print("Optimization Finished!")
print("Running count : " '%04d' % (epoch+1), "Training cost =", r_cost, "W =", r_W, "b =", r_b)
```

https://github.com/aymericdamien/TensorFlow-Examples/blob/master/examples/2_BasicModels/linear_regression.py

시각화

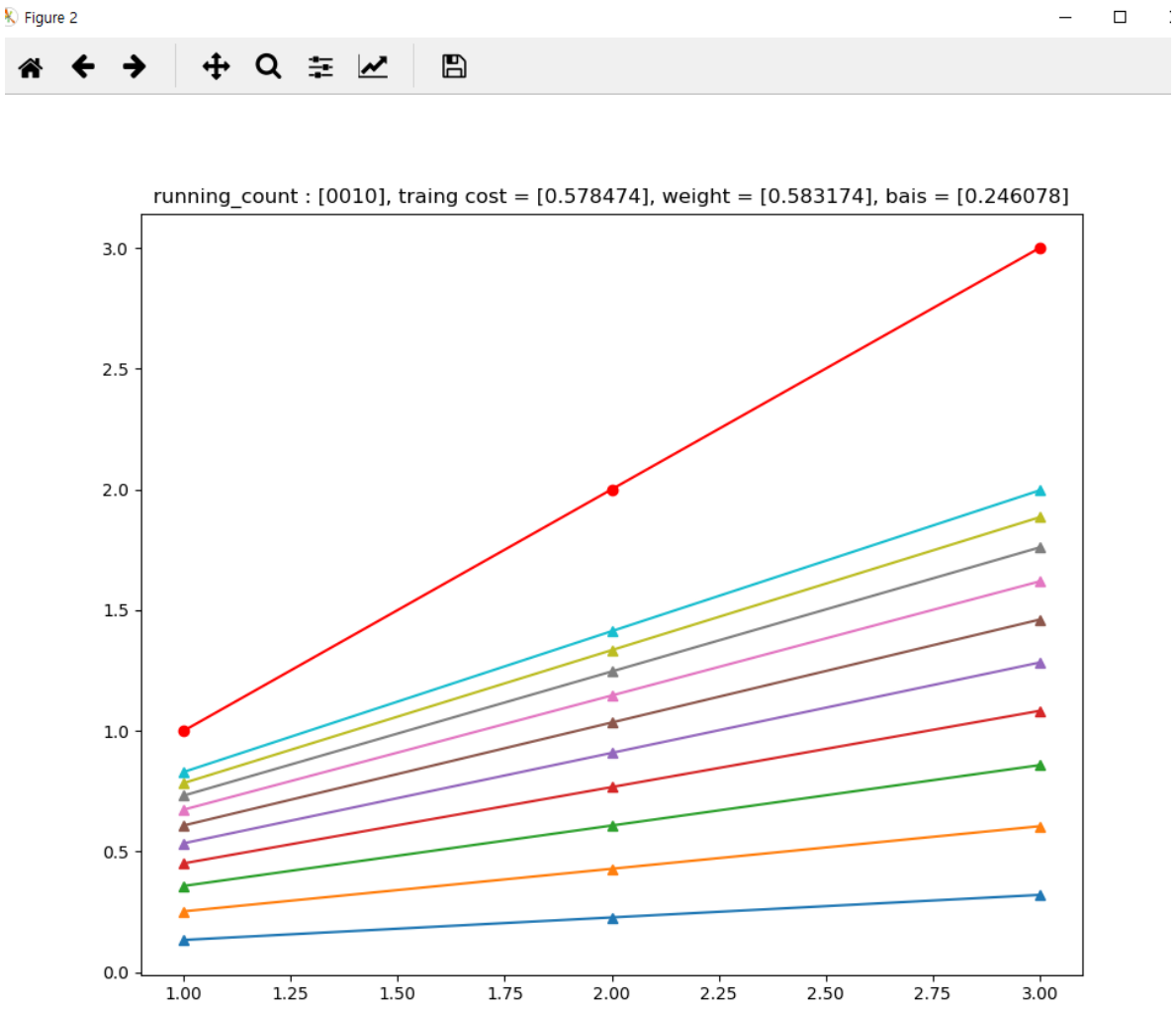




https://github.com/aymericdamien/TensorFlow-Examples/blob/master/examples/2_BasicModels/linear_regression.py

Linear Regression

시각화



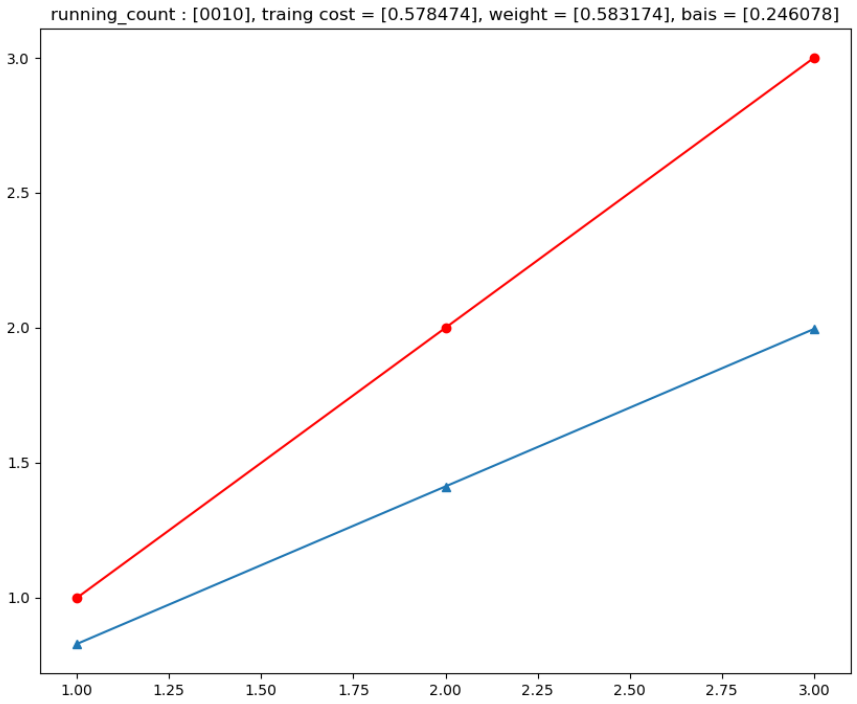
https://github.com/aymericdamien/TensorFlow-Examples/blob/master/examples/2_BasicModels/linear_regression.py

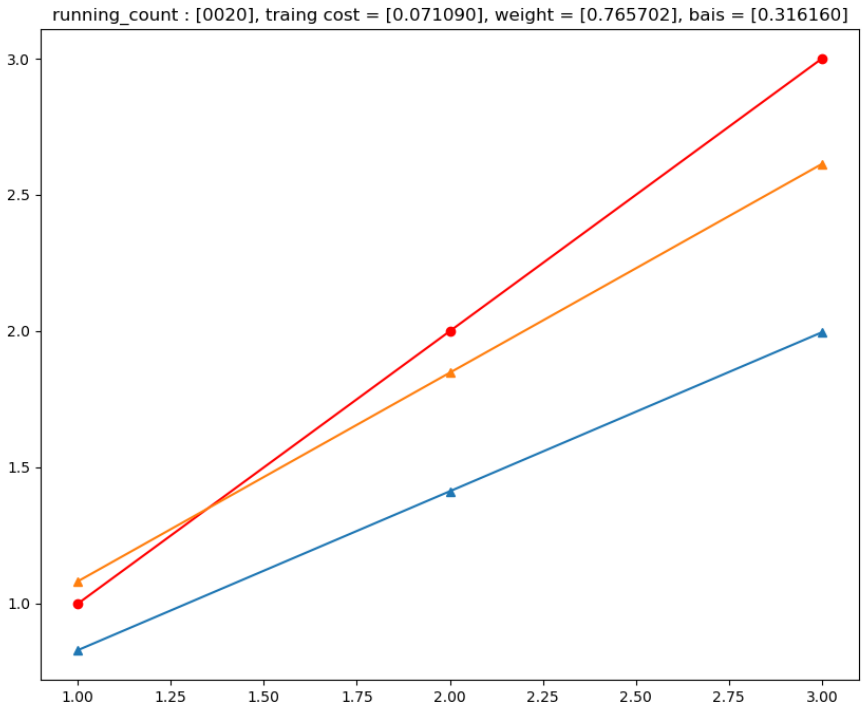
시각화 - 100번 학습

```
# Graphic display
plt.figure(figsize=(10,8))
plt.plot(train_X, train_Y, label='y_data',marker="o",color='r',linewidth=1.5)

pagenation('....')

# 학습( training_cnt 횟수만큼)
for epoch in range(training_cnt):    # training_cnt = 100
    r_cost, r_W, r_b, _ = sess.run([cost, W, b, op_train], feed_dict = {X: train_X, Y: train_Y})

    if (epoch+1) % display_step == 0:  # 나머지가 0 이될 때 결과 출력, display_step= 10 (10 번 마다 출력)
        print("Running count : " '%04d' % (epoch+1), "Training cost =", r_cost, "W =", r_W, "b =", r_b)

        # Graphic display
         plt.plot(train_X, sess.run(W) * train_X + sess.run(b), marker = '^')
         plt.title('running_count : [%04d], traing cost = [%.6f], weight = [%.6f], bais = [%.6f]' % (epoch+1, r_cost, r_W, r_b))
         plt.pause(0.1)

         pagenation('....')
print("Optimization Finished!")
print("Running count : " '%04d' % (epoch+1), "Training cost =", r_cost, "W =", r_W, "b =", r_b)
```

https://github.com/aymericdamien/TensorFlow-Examples/blob/master/examples/2_BasicModels/linear_regression.py

시각화

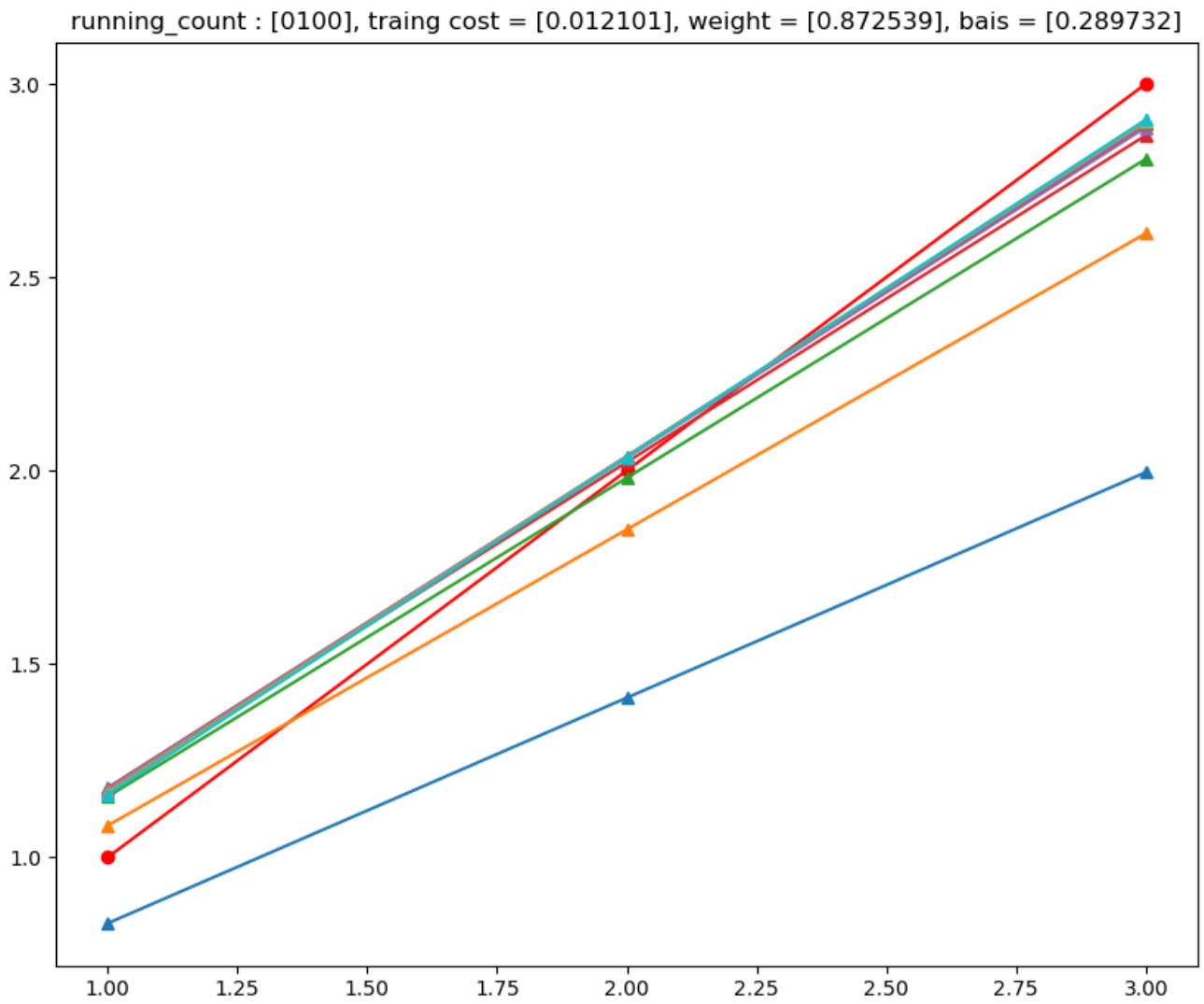




https://github.com/aymericdamien/TensorFlow-Examples/blob/master/examples/2_BasicModels/linear_regression.py

Linear Regression

시각화



https://github.com/aymericdamien/TensorFlow-Examples/blob/master/examples/2_BasicModels/linear_regression.py

4. Multi-Variable Linear Regression

Multi-Variable이란?

✓ 이전까지의 linear regression은 X가 하나 Y가 하나였다.

X	Y
1	1
2	2
3	3

X가 4이면 Y의 값은 몇 일까?

source : https://blog.naver.com/ishkkman/220963300126

Multi-Variable이란?

✓ Multi-Variable linear regression은 X가 여러 개의 입력(Features)인 경우이다.

X_1	X_2	X_3	Y
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

source : https://blog.naver.com/ishkkman/220963300126

Multi-Variable Linear Regression

Hypothesis를 어떻게 설정할까?

- ✓ X가 한 개인 linear regression의 경우

  $H(\boldsymbol{x}) = W\boldsymbol{x} + b$

- ✓ X가 세 개인 Multi-variable linear regression의 경우

  $H(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = W_1\boldsymbol{x}_1 + W_2\,\boldsymbol{x}_2 + W_3\,\boldsymbol{x}_3 + b$

- ✓ X가 n개인 Multi-variable linear regression의 경우

  $H(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \cdots, \boldsymbol{x}_n) = W_1\boldsymbol{x}_1 + W_2\,\boldsymbol{x}_2 + W_3\,\boldsymbol{x}_3 + \cdots + W_n\,\boldsymbol{x}_n + b$

source : https://blog.naver.com/ishkkman/220963300126

Hypothesis를 어떻게 설정할까?

- ✓ X가 3개인 Multi-variable linear regression의 경우

  $H(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = W_1\boldsymbol{x}_1 + W_2\,\boldsymbol{x}_2 + W_3\,\boldsymbol{x}_3 + b$

- ✓ 식이 복잡하므로 행렬(Matrix)을 사용한다, 행렬을 나타낼 땐 대문자로 표기한다.

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1w_1 + x_2w_2 + x_3w_3 \end{pmatrix} \qquad H(X) = XW$$

- ✓ 편의상 b(bias)는 생략한다.

source : https://blog.naver.com/ishkkman/220963300126

행렬(Matrix)의 곱(Multiplication)으로 표현한 Hypothesis

✓ 여러 개의 instance를 한번에 계산할 수 있다.

features : X의 Variable 갯수

instance : 데이터의 수

$$
\begin{bmatrix}
x_{11} & x_{12} & x_{13} \\
x_{21} & x_{22} & x_{23} \\
x_{31} & x_{32} & x_{33} \\
x_{41} & x_{42} & x_{43} \\
x_{51} & x_{52} & x_{53}
\end{bmatrix}
\cdot
\begin{bmatrix}
w_1 \\
w_2 \\
w_3
\end{bmatrix}
=
\begin{bmatrix}
x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\
x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \\
x_{31}w_1 + x_{32}w_2 + x_{33}w_3 \\
x_{41}w_1 + x_{42}w_2 + x_{43}w_3 \\
x_{51}w_1 + x_{52}w_2 + x_{53}w_3
\end{bmatrix}
$$

[5, 3] [3, 1] [5, 1]

source : https://blog.naver.com/ishkkman/220963300126

행렬(Matrix)의 곱(Multiplication)으로 표현한 Hypothesis

✓ instance가 n개인 경우 tensorflow에선 None으로 표현한다.

features : X의 Variable 갯수

instance : 데이터의 수

$$
\begin{bmatrix}
x_{11} & x_{12} & x_{13} \\
x_{21} & x_{22} & x_{23} \\
x_{31} & x_{32} & x_{33} \\
\vdots & \vdots & \vdots \\
x_{n1} & x_{n2} & x_{n3}
\end{bmatrix}
\cdot
\begin{bmatrix}
w_1 \\
w_2 \\
w_3
\end{bmatrix}
=
\begin{bmatrix}
x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\
x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \\
x_{31}w_1 + x_{32}w_2 + x_{33}w_3 \\
\vdots \\
x_{n1}w_1 + x_{n2}w_2 + x_{n3}w_3
\end{bmatrix}
$$

[n, 3] [3, 1] [n, 1]

source : https://blog.naver.com/ishkkman/220963300126

cost를 어떻게 설정할까?

- ✓ X가 1개인 linear regression의 cost function

$$cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

- ✓ X가 3개인 Multi-variable linear regression의 cost function

$$cost(W,b) = \frac{1}{m}\sum_{I=1}^{m}(H(x_1^{(i)}, x_2^{(i)}, x_3^{(i)}) - y^{(i)})^2$$

- ✓ H(x)(예측 값) 부분만 바뀌고 y(실제 값)은 그대로 적용한다.

source : https://blog.naver.com/ishkkman/220963300126

Reference

김성훈 교수, “모두를 위한 딥러닝 강좌 시즌1”

코드참조, https://github.com/hunkim/DeepLearningZeroToAll/

실행 환경 구축, https://www.tensorflow.org/install/install_linux
https://www.tensorflow.org/install/install_linux#the_url_of_the_tensorflow_python_package

Q&A

감사합니다.