

이것이 우리에게 딱 맞는 딥러닝 with 텐서플로

박재용 (jaeyong1@naver.com)
2018.11.10-11

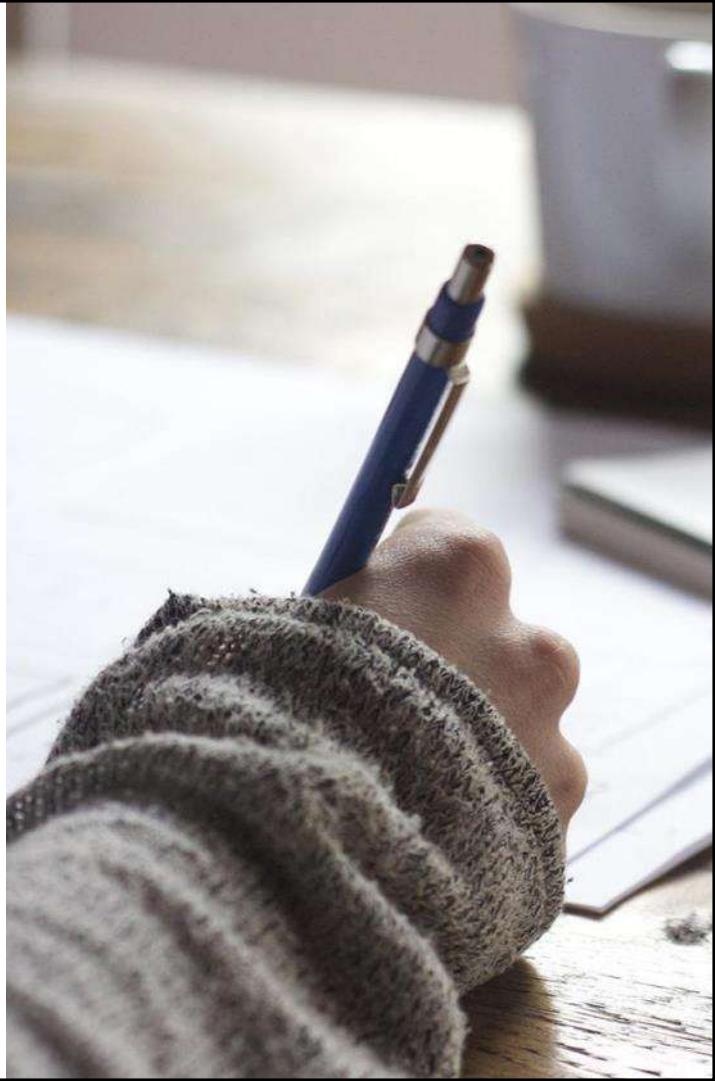


HELLO!

박재용

LG이노텍 전장사업부 차량통신SW개발팀. 선임연구원 . 2016-
LG전자 MC연구소 프로토콜팀. 2011-
컴퓨터공학과 네트워크전공 석사. 2009-2011

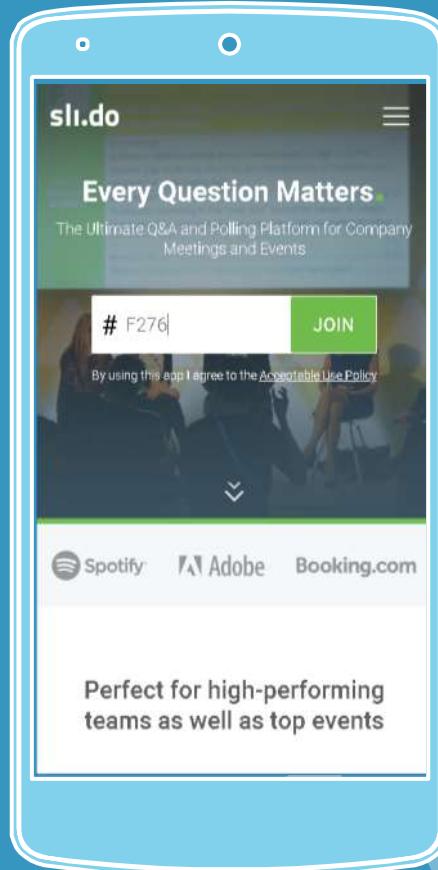
- ✉ jaeyong1@naver.com
- 🌐 YongsLab.com
- 🌐 Github.com/jaeyong1



sli.do

#

자유롭게 질문을 남겨주세요



구글 로그인을 준비해 주세요

<https://colab.research.google.com>



Windows에서 Tensorflow 설치하기

- » Windows에서 Tensorflow를 사용하기 위해서는 아나콘다(Anaconda)를 기반을 권장
- » 아나콘다 : 필요한 라이브러리 패키지 관리 및 환경 설정 등을 쉽게 해주는 도구
 - ◊ 한 대의 PC에 목적에 따라 여러 개의 독립적인 개발환경을 구성
 - ◊ 개발환경에 필요한 다양한 라이브러리들을 한 줄의 명령으로 손쉽게 다운로드 및 설치 가능
 - ◊ www.anaconda.com/download/
 - ◊ 윈도우 아이콘을 눌러서 윈도우용 아나콘다를 다운받아 설치



» <https://github.com/jaeyong1/deeplearningforus>

The screenshot shows a GitHub repository page for 'jaeyong1 / deeplearningforus'. The top navigation bar includes 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the repository name, it shows 'forked from tensorflow/deeplearning'. The main menu has tabs for 'Code', 'Pull requests', 'Projects', 'Wiki' (which is highlighted with a red oval), 'Insights', and 'Settings'. On the right, there are buttons for 'Unwatch', 'Star' (0), 'Fork' (3), 'Edit', and 'New Page'. The 'Links' section contains a list of resources: '발표자료(PDF)', '실습자료(Colab)' (with a link to <https://colab.research.google.com>), 'Chap 2. Hello world in 텐서플로우', and 'Colaboratory에서 GPU로 실행하기' (with a link to <http://goo.gl/M488k4>). To the right, a sidebar titled 'Pages' lists 'Home', 'Links' (which is highlighted with a red oval), and 'Market'.

〈자료출처 안내〉

본 발표자료는 “이것이 우리에게 딱 맞는 딥러닝 with 텐서플로”의 내용을 기반으로 하고 있습니다.
또한, 내용의 이해를 돋기 위해서 선행연구자들이 인터넷에 공개해 놓은 자료를 차용하고 있습니다.

이러한 내용들은 많은 분들이 함께 협업을 통해 만들어 나가고 있는 자료들이며,
다음 사이트에서 원문을 찾아보실 수 있습니다.

» 모두를 위한 머신러닝/딥러닝 강의 (김성훈교수님)

<https://hunkim.github.io/ml/>

» 머신러닝을 배우려는 실무자를 위한 강의 (구글)

<https://developers.google.com/machine-learning/crash-course/introduction-to-neural-networks/>

» Deep Learning and Tensorflow Basic (이도엽님)

<https://github.com/LeeDoYup/Deep-Learning-Tensorflow-Basic>

» 텐서플로우 공식사이트

www.tensorflow.org

» Coursera 강의

Machine Learning(Andrew Ng.)

Neural Network for Machine Learning (Jeff. Hinton)

» Stanford Course

cs231n (Convolutional Neural Networks for Visual Recognition)

» 그 외 슬라이드에 표기



1.

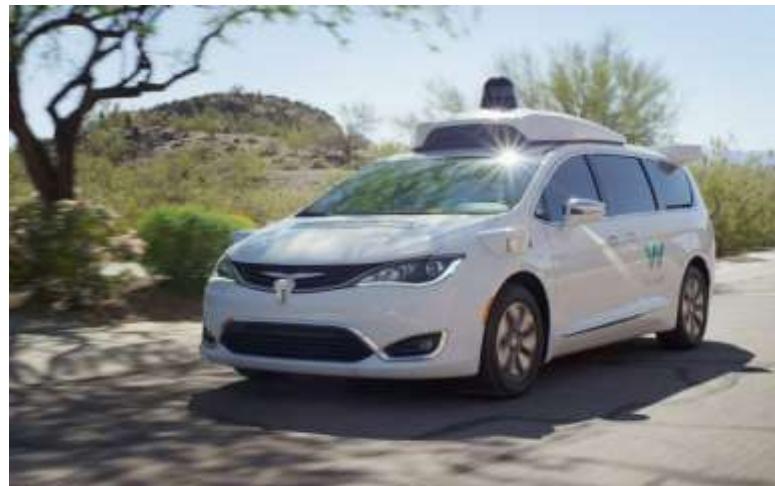
인공지능에 대하여

우리주위에서 인공지능
딥러닝과 뉴런, 뉴럴 네트워크
인공지능과 머신러닝, 딥러닝

우리주위에서 인공지능

» 웨이모(Waymo)

- ◊ 구글의 자율주행차
- ◊ 누적 주행기록이 500만 마일을 갱신
- ◊ 지구 200바퀴에 해당하는 거리
- ◊ 미국 성인 한 사람의 일 년간 주행거리를 단 하루 만에 주행



<https://youtu.be/aaOB-ErYq6Y>

» 알파고(AlphaGo)

- ◊ ‘알파(α)’ : 구글의 자주회사 이름인 알파벳, 그리스 문자의 첫 번째 글자로 최고를 의미
- ◊ ‘Go’ : ‘碁(바둑)’의 일본어 발음을 차용한 영어단어
- ◊ 통산 전적은 73승 1패



알파고와 이세돌의 바둑대결

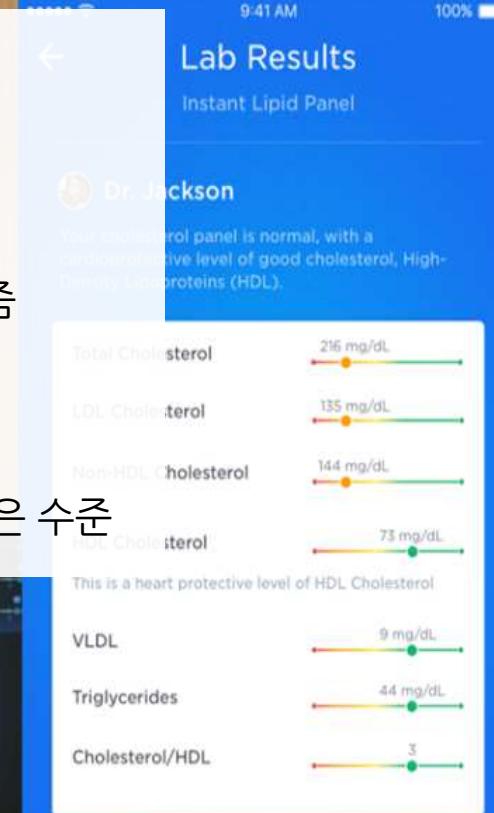


» IBM Watson Health

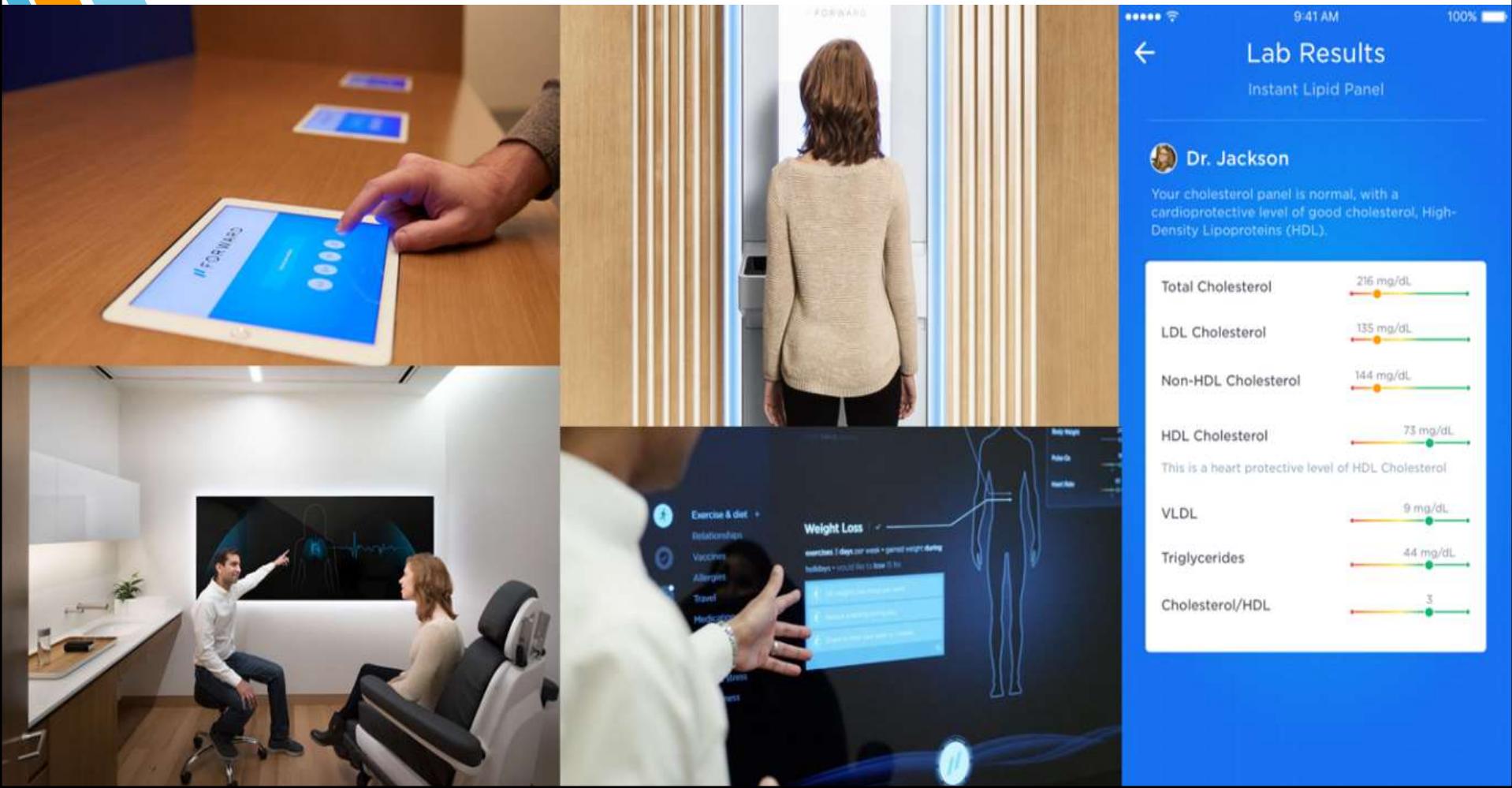
- 개개인에게 필요한 맞춤형 의료경험을 제공하는 인공지능 프로그램
 - 미국 메모리얼 슬론 캐터링(MSK, 세계 3대 암 센터)의 의학자료로 훈련
→ 300개 이상의 의학 학술지, 200개 이상의 의학 교과서, 1500만 페이지 분량 의료정보 데이터
 - 암환자에게 맞춤형 진료 기회를 제공
 - 기준보다 빠르고 정밀한 유전자 분석 결과를 제공
 - 의료진은 더 효과적인 치료법을 결정하는데 도움을 받음

» 포워드(Forward)

- ❖ 스타트업 회사 . 인공지능을 의학적으로 활용
- ❖ 센서와 인공지능을 활용한 맞춤형 헬스-케어 멤버쉽 서비스를 제공
- ❖ 사내 의사가 있으며 추가 치료가 필요한 환자를 외부 병원으로 연결시켜 줌
- ❖ 환자는 모바일 앱을 통해 언제든지 검사결과를 보거나 의사에게 질문가능
- ❖ 월정액 \$149(약17만원)으로 24시간 건강 검진 및 무제한 상담
 - 미국의 고가 의료비를 생각했을 때 이 정도의 1차 진료비용은 비싸지 않은 수준



포워드(Forward)



로보어드바이저

» '로보어드바이저' 서비스

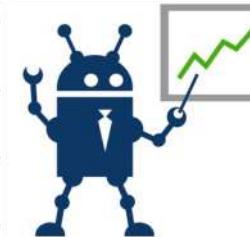
= 로봇을 뜻하는 로보(Robo) + 자문가를 뜻하는 어드바이저(Advisor)

- » 금융 산업에서 인공지능 기술 활용도는 의료 산업에 이어 두 번째가 될 것으로 기대
- » 알고리즘이 투자의 중심이 되는 로봇 기반의 인공지능 투자 플랫폼

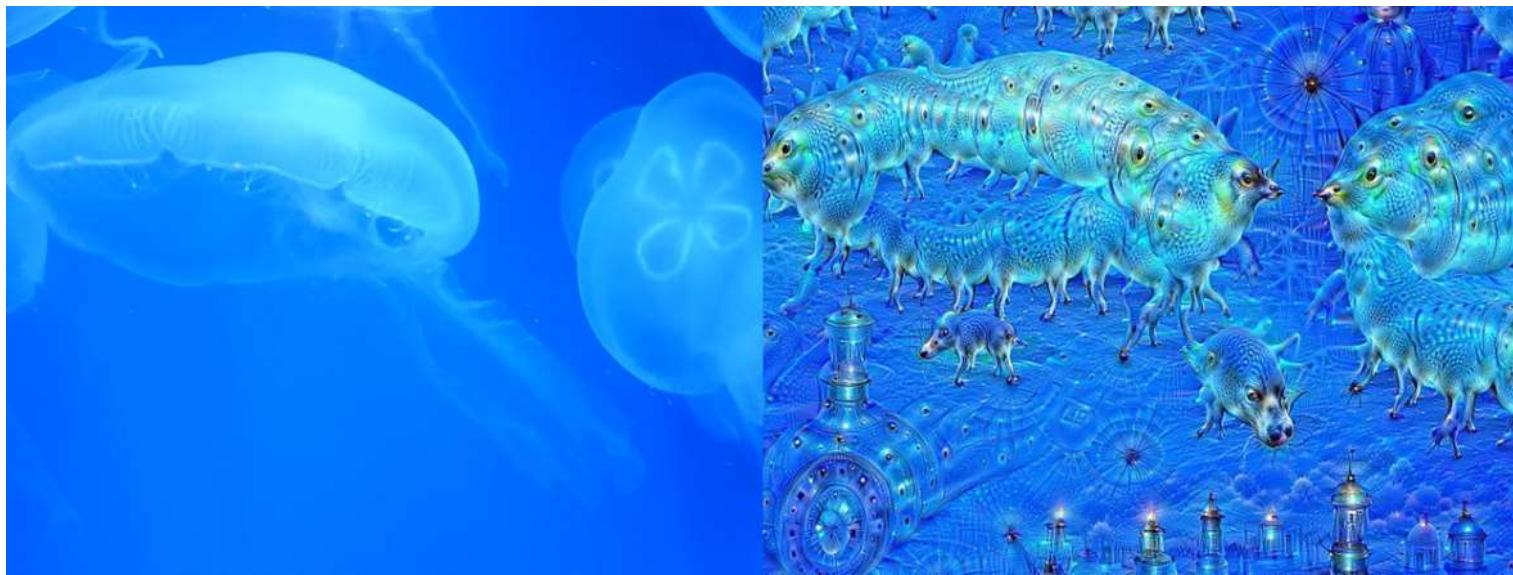
투자자의 위험 감수 성향과 목표 수익률, 자금성격 등 정보를 토대로 맞춤형 포트폴리오 제공

금융권 로보어드바이저 도입 현황

삼성증권	자체 로보어드바이저 플랫폼 개발
현대증권	로보어드바이저 자문형 랩어카운트 출시
동부증권	로보어드바이저 서비스 시행
NH투자증권	QV(큐브)로보어카운트 출시
KB국민은행	로보어드바이저 자문형 신탁상품 '쿼터백 R-1' 출시
KEB하나은행	로보어드바이저 '사이버(Cyber) PB' 자체 개발



» Deep Dream: 구글의 엔지니어인 Alexander Mordvintsev가 만든 컴퓨터 비전 프로그램
이미지 패턴을 찾고 향상시키는 과정을 통하여 의도적으로 과도하게 처리된 이미지를 만들어 냄
=> 마치 꿈에서 본 듯한 멋진 이미지 결과물



원본 이미지(왼쪽)과 Deep Dream 알고리즘을 10번 반복(오른쪽)



» 딥드림 프로젝트의 대표 그림

- » 인공지능(AI)이 그린 그림
- » 2018. 10. 25. 미국 뉴욕 크리스티 경매에서 43만2500달러 (약4억9132만원)에 판매

- » AI '화가'의 이름은 '오비어스(Obvious)'
- » 14~20세기에 그려진 초상화 1만5000여점을 토대로 AI가 새로운 작품을 창작



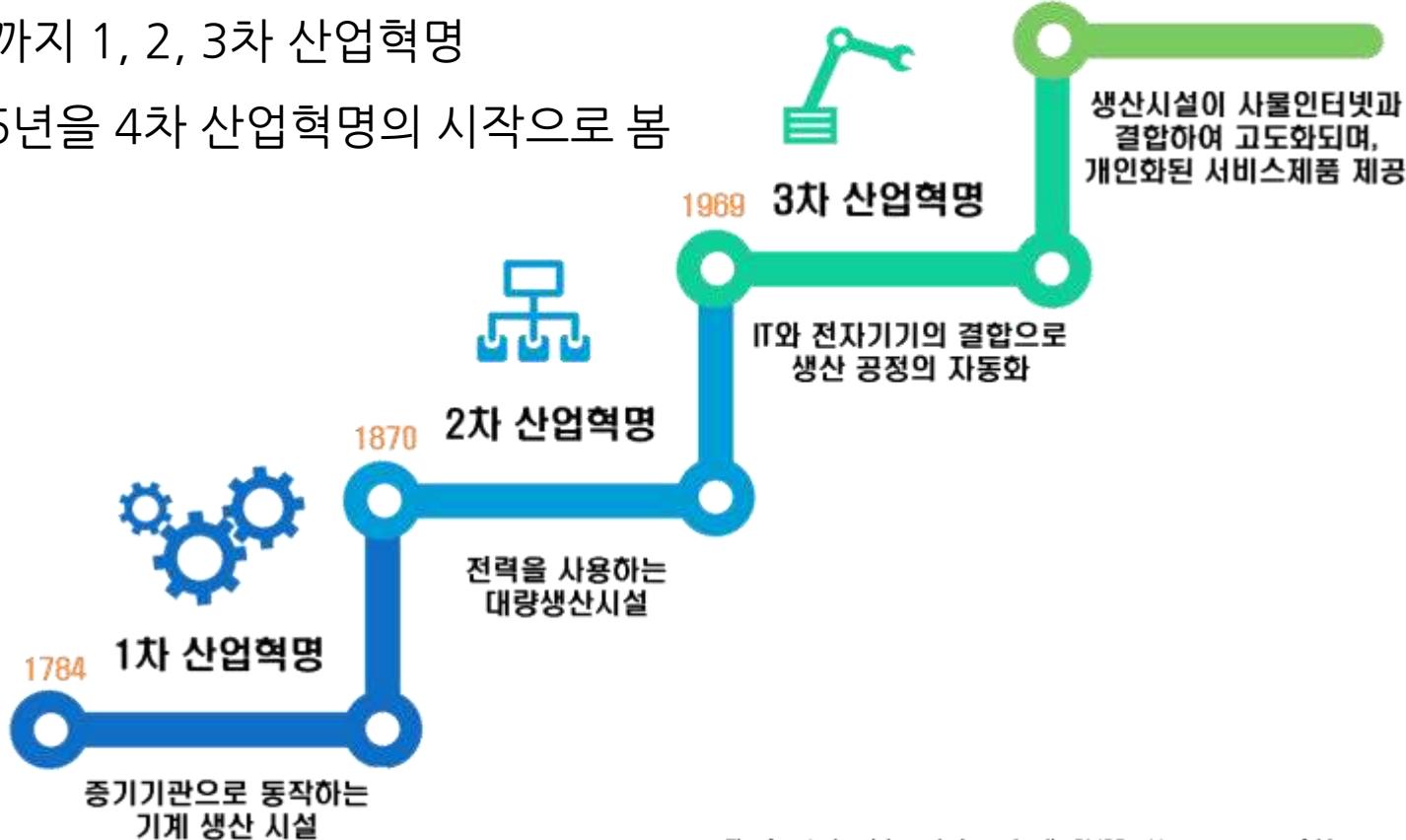
AI가 그린 '에드몽 드 벨라미'(Edmond de Belamy)

산업혁명과 인공지능



4차 산업혁명

- » 지금까지 1, 2, 3차 산업혁명
- » 2015년을 4차 산업혁명의 시작으로 봄



인공지능의 사전적 의미?

» **인공** : 사람의 힘으로 가공하거나 어떠한 작용을 한 것

산업적으로는 인위적인 제조공정을 거친 것

» **지능** : 새로운 대상이나 상황에 부딪혀 그 의미를 이해하고

합리적인 적응 방법을 알아내는 지적 활동의 능력

» **인공지능**이란?

사람이 인위적으로 만들어낸 것으로, 새로운 상황에서 합리적인 판단을 내리는 것

» Artificial Narrow Intelligence(ANI)

- 한 분야에 특화된 인공지능으로 주어진 문제에서 답을 잘 찾는 수준

» 수표나 우편봉투의 필기체 인식, 검색엔진의 이미지 분류기, 퀴즈대회참여, 의료분야에 사용되는 IBM의 왓슨, 바둑으로 인간을 뛰어넘은 알파고, 음성대화가 가능한 아마존(Amazon)의 알렉사(Alexa), 애플(Apple)의 시리(Siri) 등이 해당



» Artificial General Intelligence(AGI)

- 학습을 바탕으로 추론적 사고가 가능, 문제를 스스로 정의할 수 있음
- 인간과 유사한 방식으로 지속적인 학습이 가능한 단계

» 글로벌 기업들의 AI 엔진들은 이 수준 도달을 목표로 연구 중



» Artificial Super Intelligence(ASI)

- 일반지식, 사회적 능력, 과학기술 등 제 분야의 모든 능력에서 인간보다 더 우수한 단계

» 아직까지는 추상적인 수준

» 현 시대의 인공지능은 1단계 문제는 잘 해결하며, 2단계로의 연구가 진행 중,

프로토타입 같은 3단계가 일부 제안되고 있음



- » 머신러닝(Machine Learning, ML)은 인공지능의 한 분야
- » 아서 사무엘(1950년대 IBM근무. 스탠포드대학교수)

“머신러닝이란 컴퓨터에게 명시적으로 프로그래밍을 하지 않고도 컴퓨터가 학습할 수 있는 분야를 말한다.”

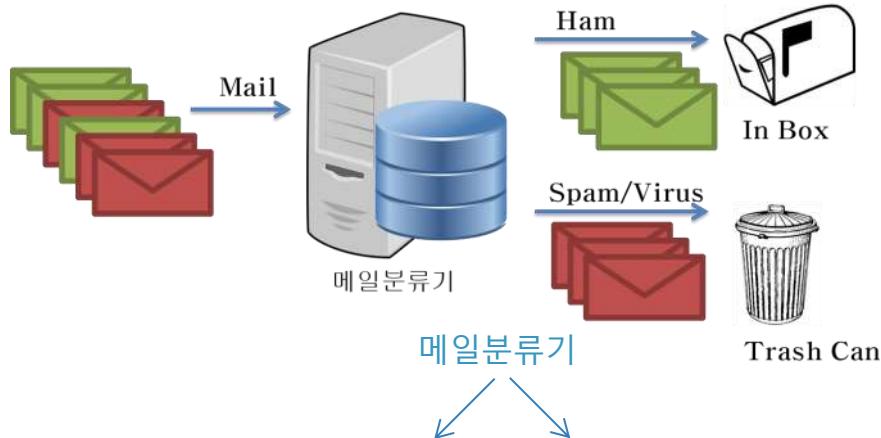
The field of study that gives computers the ability to learn without being explicitly programmed.

» 톰 미첼(Tom Michael Mitchell) 교수

“태스크(T)에 대해 꾸준한 경험(E)을 통해 T 에 대한 성능(P)을 높이는 것을
머신러닝이라고 한다.”

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T , as measured by P , improves with experience E .

머신러닝 적용 예



1. 인간이 메일의 분류조건을 프로그래밍



분류조건
프로그래밍



메일분류기

2. 결과를 바탕으로 기준을 반복학습



메일분류기

머신러닝 학습방법에 따른 분류

- 1) Supervised Learning (지도학습)
- 2) Unsupervised Learning (비지도학습)
- 3) Reinforcement Learning (강화학습)

- Labeled data
- Direct feedback
- Predict outcome/future

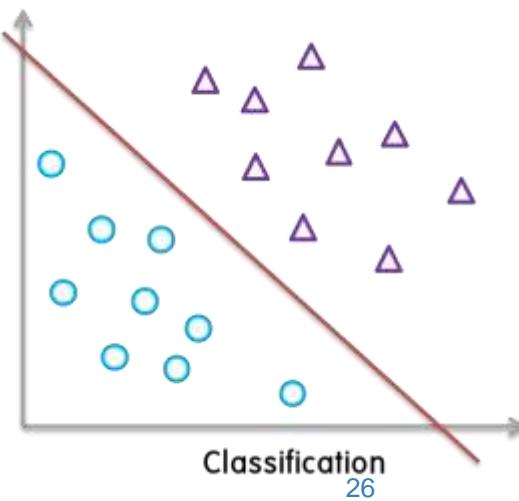
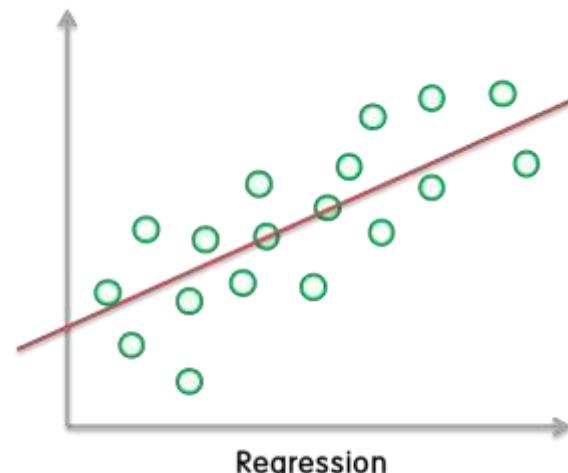


- No labels
- No feedback
- “Find hidden structure”

- Decision process
- Reward system
- Learn series of actions

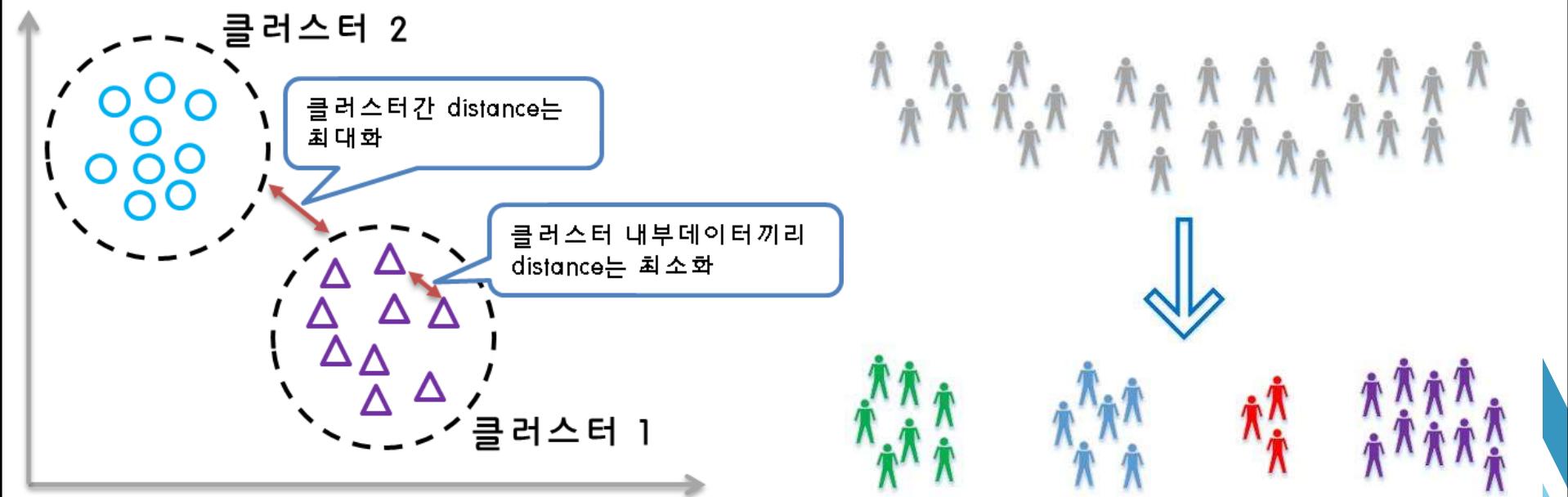
Supervised Learning(지도학습)

- » 학습할 데이터와 그 정답을 컴퓨터에게 알려주고 적당한 예측공식을 찾게 하는 것
- » Labeled Data라는 정답지가 있음
- » 입력데이터와 출력데이터를 가장 잘 나타낼 수 있는 예측공식을 만드는 것이 목표
- » 얼마나 예측이 정확한지 오차를 수치로 표현
- » $y=ax+b$ 에서 a 와 b 가 컴퓨터가 예측한 값이며, 이 파라미터를 찾는 과정이 머신러닝의 학습과정



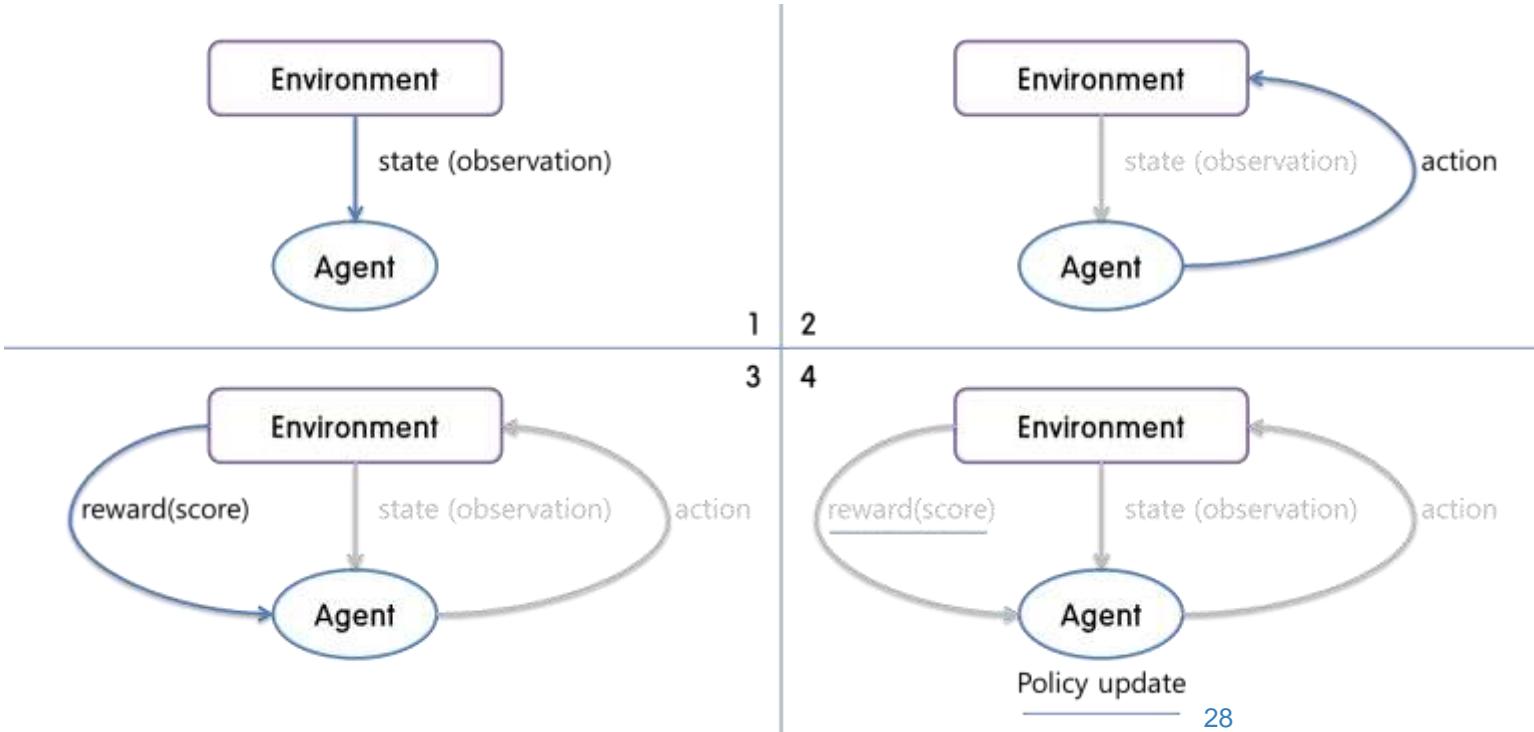
Unsupervised Learning(비지도학습)

- » 학습할 데이터는 있는데 정답지가 없는 학습방법
- » 입력데이터를 클러스터(cluster)라는 단위로 묶는 것을 목표
- » 데이터의 특징들을 비교하여 합리적으로 클러스터를 잘 나누는 것



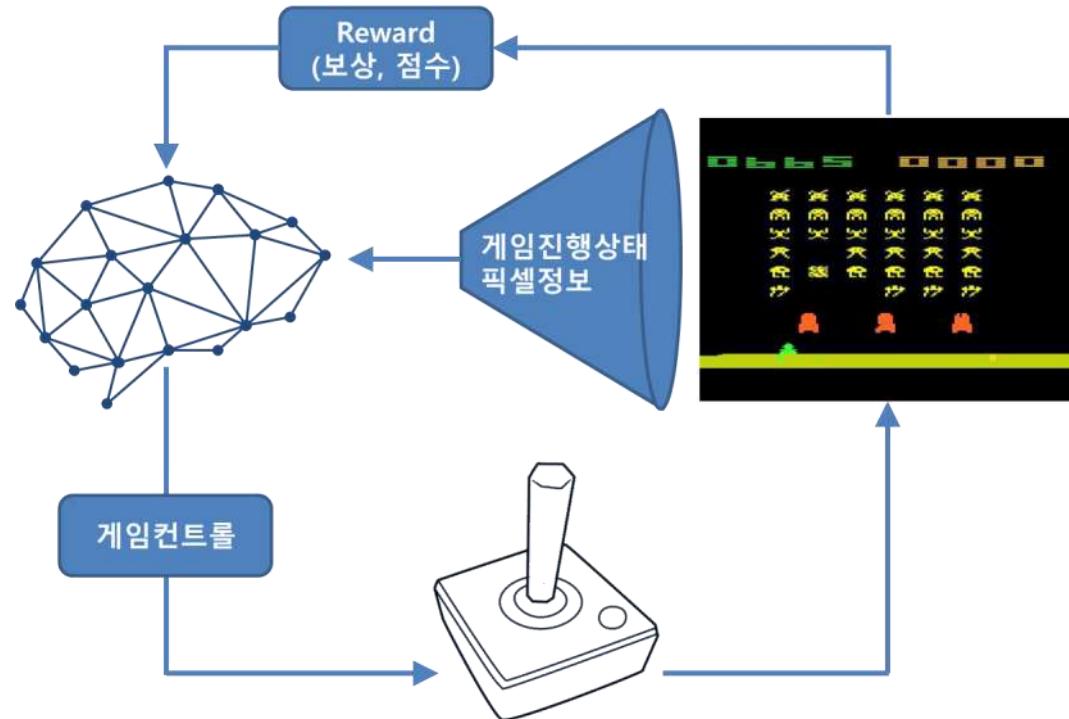
Reinforcement Learning(강화학습)

- » 학습 주체인 Agent가, 주어진 상황 Environment에서,
- » 미래의 기대보상인 Reward가 최대가 되는 행동 Action을 결정하도록,
- » 행동방침(Policy)을 계속해서 학습해가는 것

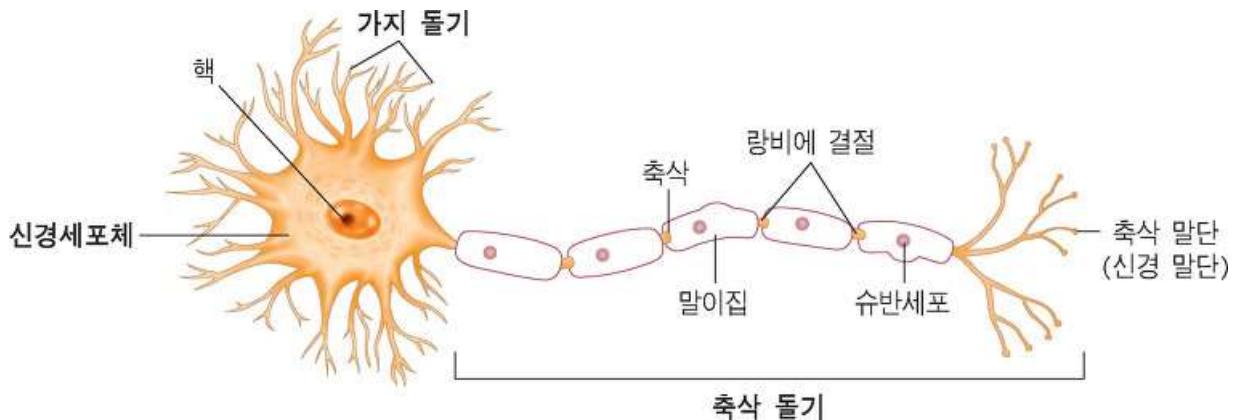


Reinforcement Learning(강화학습)

» 처음 학습을 시작할 때에는 행동을 잘 하지 못하지만,
학습이 반복되면서 점차 아주 잘 하는 실력을 갖게 됨



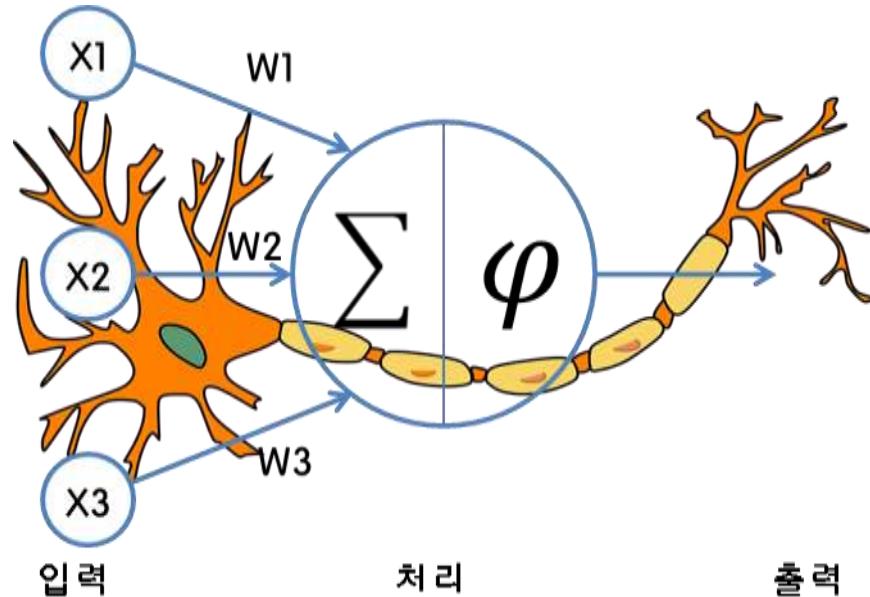
딥러닝과 뉴런, 뉴럴 네트워크



- » 인간처럼 생각하는 기계를 만들기 위해 → 인간의 뇌를 모방하기로 함
- » **뉴런(Neuron)** : 인간의 뇌의 기본단위, 아주 단순한 구조
수많은 뉴런들이 서로 연결되어 아주 복잡한 계산을 효과적으로 해내고 있음
- » **가지 돌기** : 다른 세포로부터 자극 또는 신호를 받음
- » **축삭 돌기** : 다른 세포에 신호를 전달

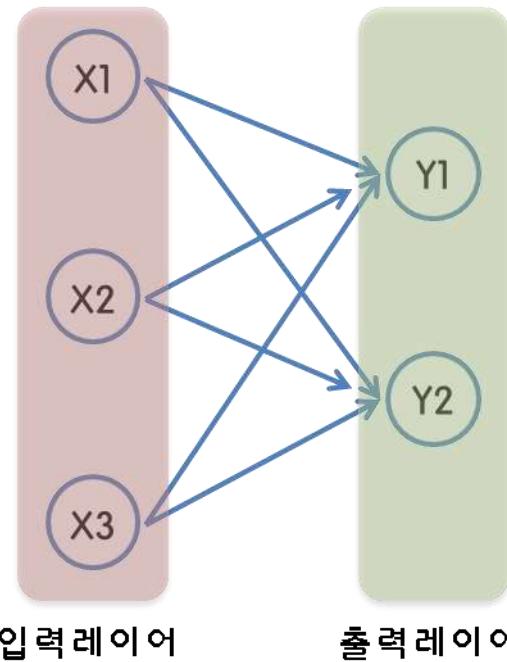
생명체의 뉴런과 AI의 뉴런

- » 딥러닝 과정에서 공부할 뉴런의 형태의 예
(3개를 입력 -- 2개의 연산 -- 1개를 출력)



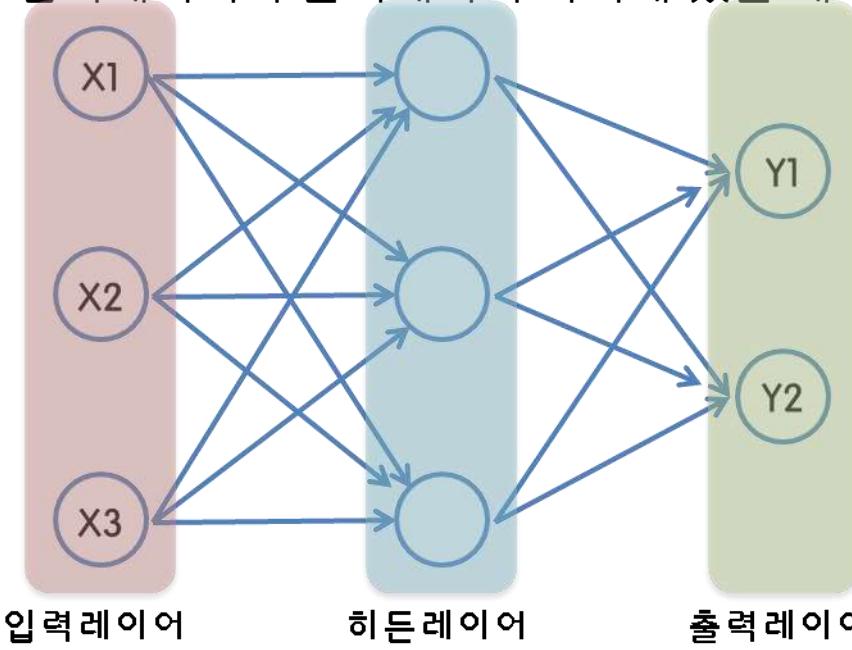
Single-Layer Perceptron

- » X_1, X_2, X_3 로 한 개의 입력 레이어를 구성
- » Y_1, Y_2 로 한 개의 출력레이어로 구성



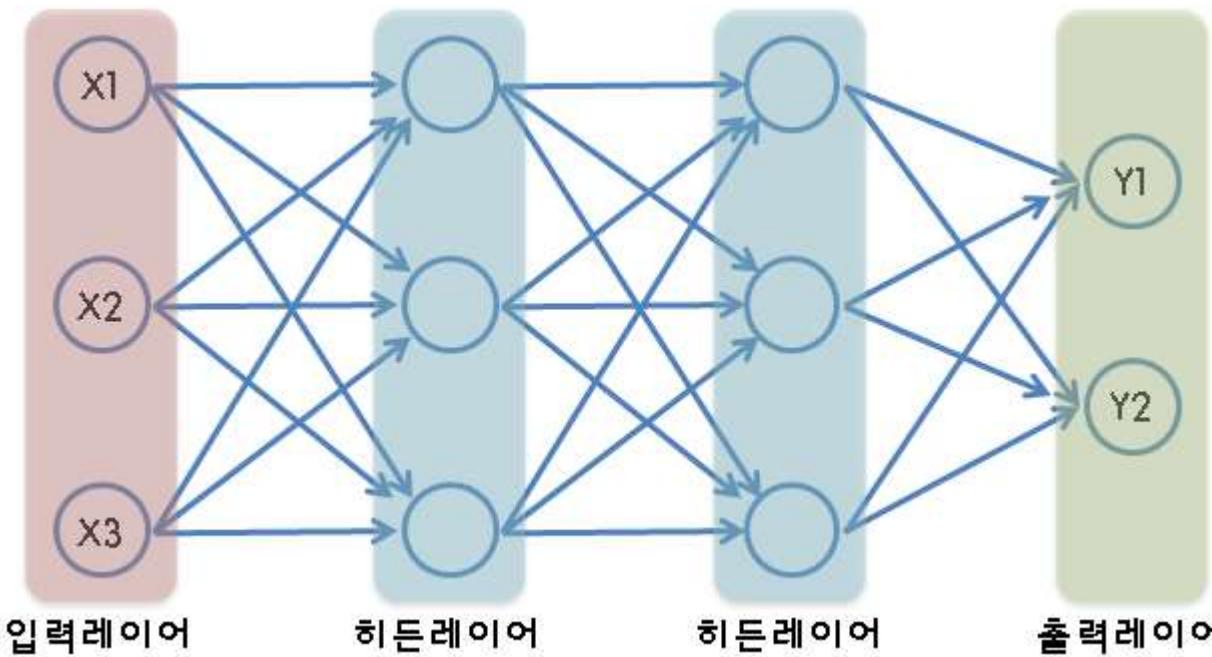
Perceptron with Hidden Layer

- » X_1, X_2, X_3 로 한 개의 입력 레이어를 구성
- » Y_1, Y_2 로 한 개의 출력레이어로 구성
- » **히든레이어** : 입력레이어와 출력레이어 사이에 있는 레이어



2-Hidden-Layer Neural Net

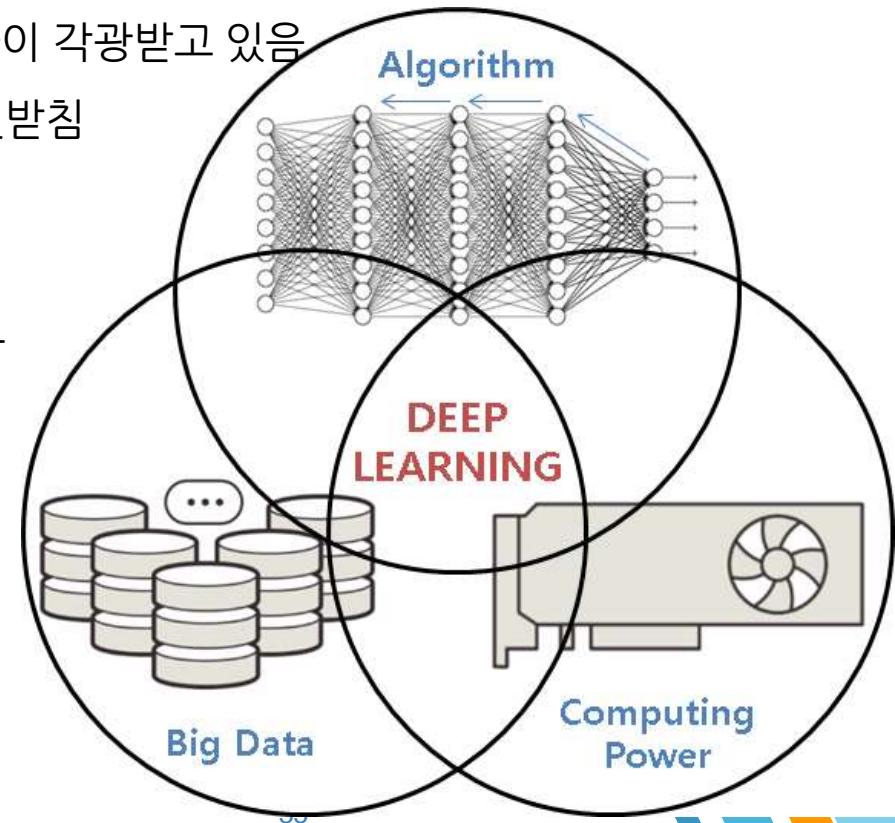
» 'Deep'하다? : 히든레이어가 여러 층으로 이루어져 있을 때



딥러닝은 최신 알고리즘인가???

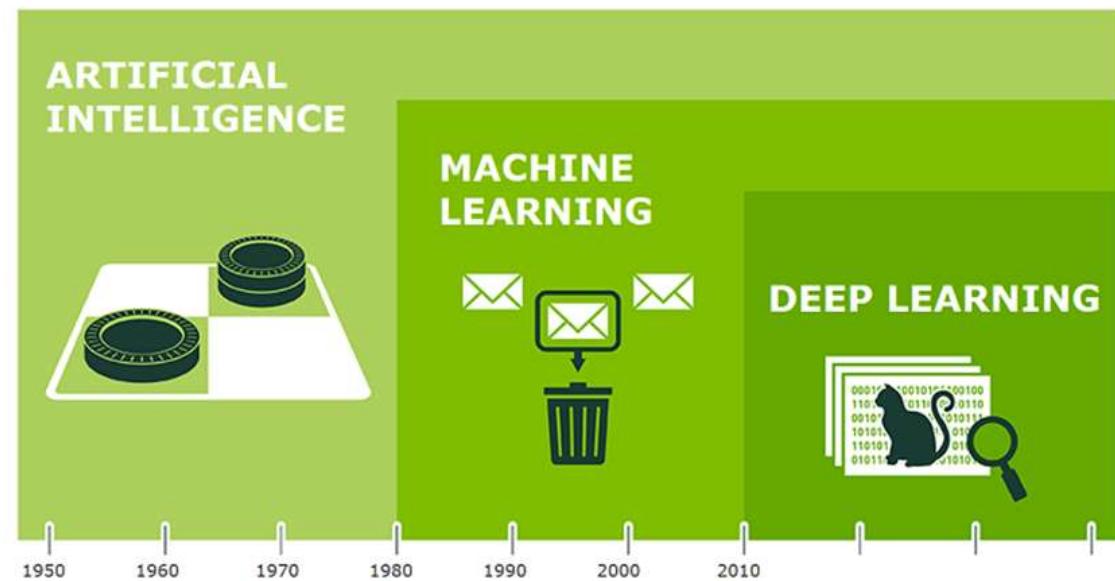
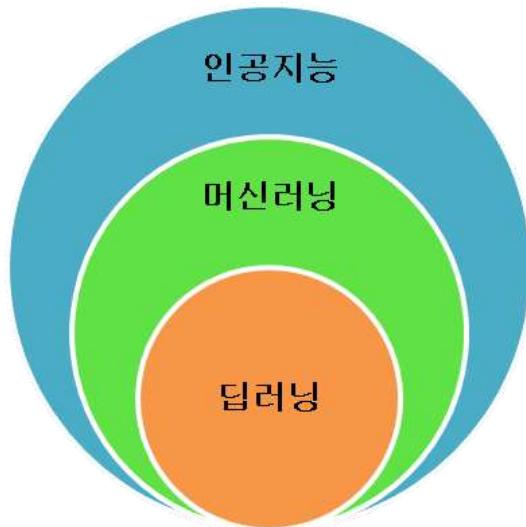
» **NO**

- » 오랜 기간 동안 주목받지 못하다가 최근 몇 년 사이 각광받고 있음
- » 기존 알고리즘의 한계를 극복한 연구사례들이 뒷받침
- » 대용량 정보를 체계적으로 저장하고
분석하는 빅데이터가 성장한 덕분
- » 과거보다 뛰어난 계산성능을 가진 하드웨어 등장



인공지능, 머신러닝, 딥러닝의 관계

- » 시간의 흐름으로 볼 수도 있고, 포함관계로 볼 수도 있음
- » 인공지능을 구현하는 기술 중에 기계학습을 통해 구현하는 머신러닝
- » 머신러닝의 한 형태로 뇌의 신경세포 뉴런을 흉내 낸 인공신경망에서 발전한 딥러닝
- » 수많은 기업들이 머신러닝/딥러닝의 기술 확보를 위해 노력 중



전통적인 프로그래밍



머신러닝 프로그래밍



2.

Hello world in 텐서플로우

Windows에서 Tensorflow 설치하기

구글클라우드 Colaboratory 사용하기

MNIST의 소개

» 알고리즘을 이해하더라도?

실제 코드로 구현해서 실행을 시켜보고, 본인의 분야에서 응용해보아야 자신의 것!!

» 머신러닝 알고리즘을 처음부터 구현하기는 쉽지 않음

» 머신러닝 구현을 위해서 Caffe, Keras, MatConvNet, Torch, OpenDeep 등등

많은 라이브러리가 있음

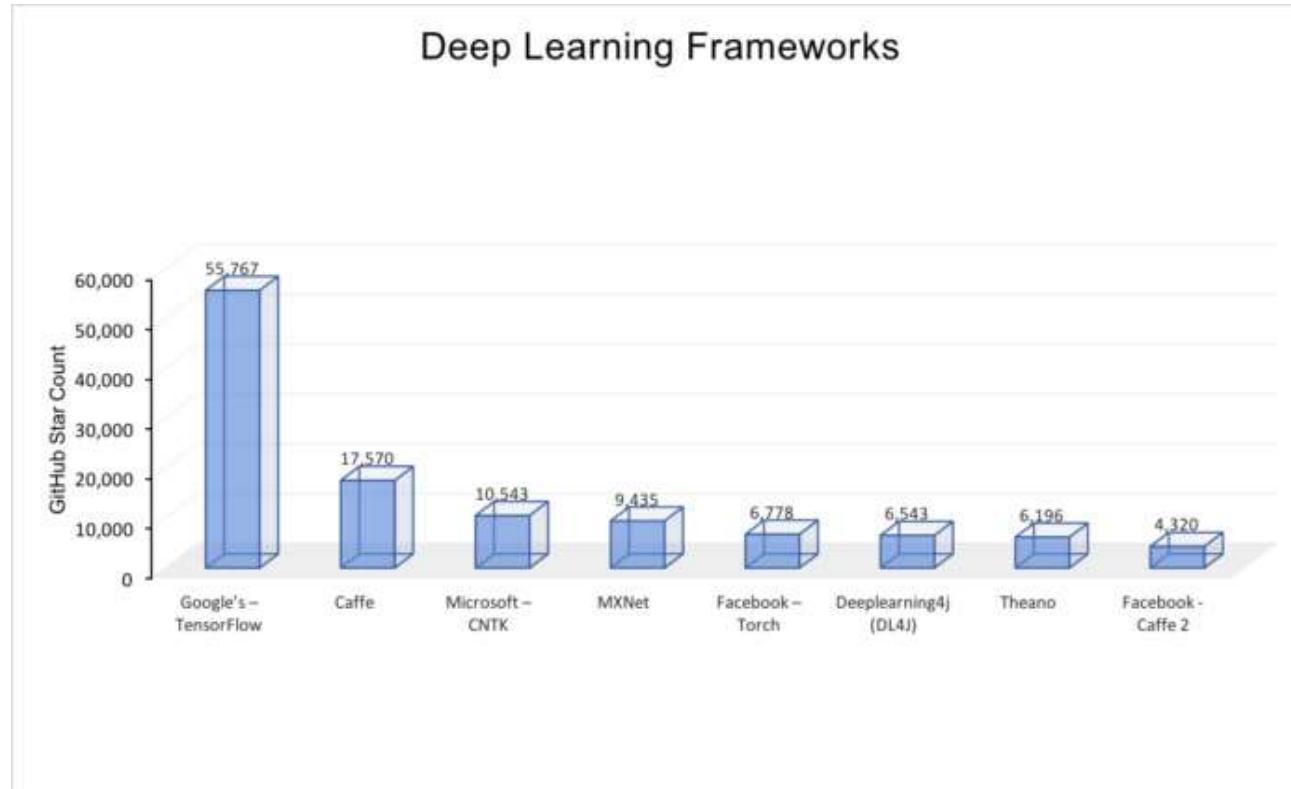
» 본 과정에서는 텐서플로(TensorFlow)를 사용



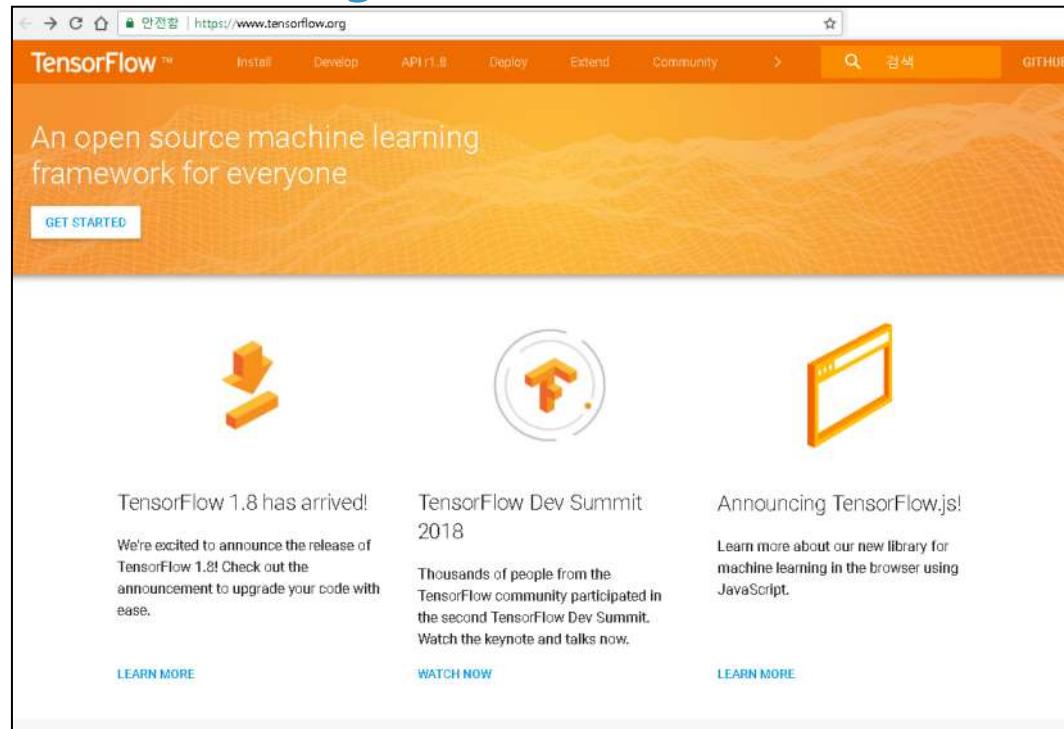
- » 오픈소스 소프트웨어 라이브러리
- » 데이터 플로우 그래프(Data flow graph)를 사용한 수치연산
- » 그래프의 노드(Node)는 수치 연산을 나타내고 엣지(edge)는 노드 사이를 이동하는
다차원 데이터 배열(텐서, tensor)를 나타냄.
- » 유연한 아키텍처로 구성되어 있어 한번 작성시
데스크탑, 서버 혹은 모바일 디바이스에서 CPU나 GPU를 사용하여 연산가능

Why 텐서플로?

- » GitHub에서 인기 있는 딥러닝 프레임워크들이 받은 별(Star)의 수



- » (다른 머신러닝 라이브러리와 비교해볼 때 비교적 뒤늦게 발표되었으나..)
- » 구글이 개발하고 공개함
- » Python API
- » Large community
- » 다양한 디바이스에서 구동 가능함
- » 1개 이상의 CPU 또는 GPU에서 병렬적으로 동작 가능
- » 다음과 같은 편리한 도구를 제공
 - ◊ TensorBoard: 네트워크 모델링 및 성능을 위한 매우 잘 설계된 시각화 도구.
 - ◊ TensorFlow Serving: 새로운 알고리즘과 실험을 쉽게 배포할 수 있음.
다른 유형의 모델 및 데이터를 제공하도록 쉽게 확장될 수 있음.

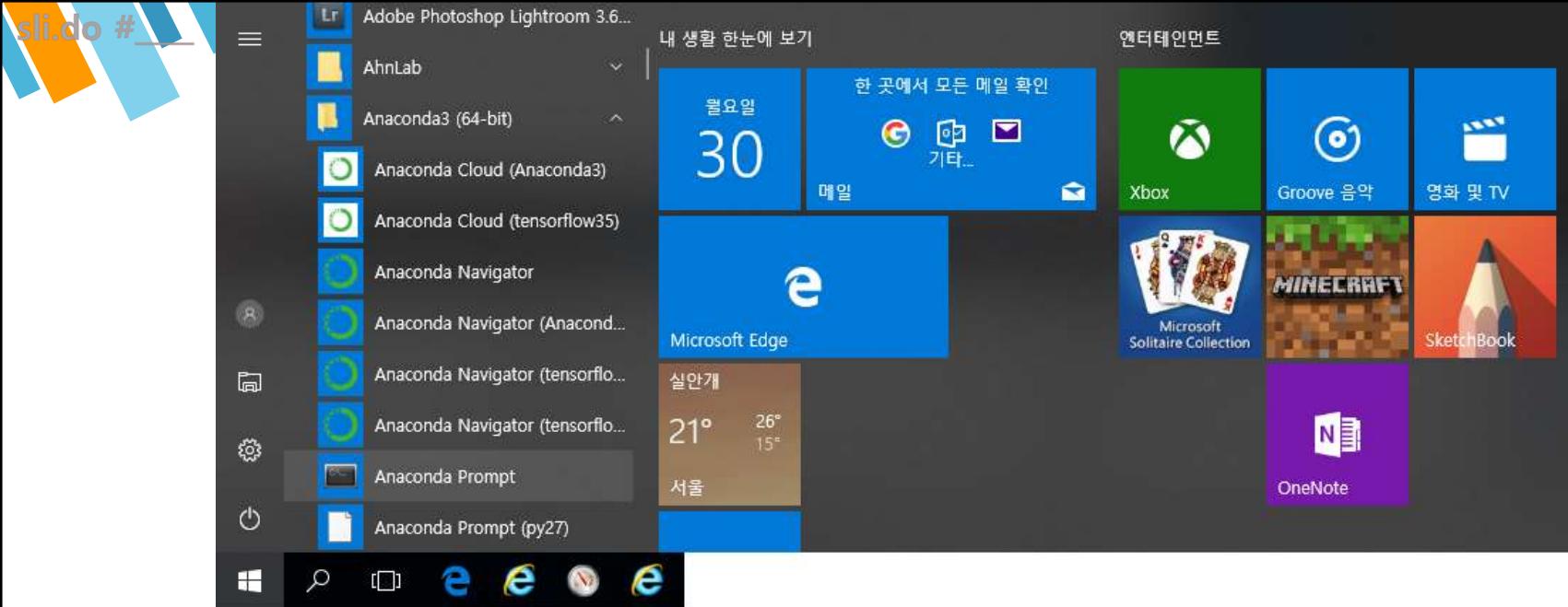


- » TensorFlow 설치와 매뉴얼에 관한 내용이 있음
- » 파이썬 라이브러리는 버전별로 의존성이 큰 편
- » 파이썬 동작환경은 가상환경으로 만들어 쓰는 것이 일반적

Windows에서 Tensorflow 설치하기

- » Windows에서 Tensorflow를 사용하기 위해서는 아나콘다(Anaconda)를 기반을 권장
- » 아나콘다 : 필요한 라이브러리 패키지 관리 및 환경 설정 등을 쉽게 해주는 도구
 - ◊ 한 대의 PC에 목적에 따라 여러 개의 독립적인 개발환경을 구성
 - ◊ 개발환경에 필요한 다양한 라이브러리들을 한 줄의 명령으로 손쉽게 다운로드 및 설치 가능
 - ◊ www.anaconda.com/download/
 - ◊ 윈도우 아이콘을 눌러서 윈도우용 아나콘다를 다운받아 설치





- » 시작 프로그램에 설치된 아나콘다 확인
- » 'Anaconda Prompt'를 통해 임의의 'tensorflowtest'라는 프로젝트를 만들고
→ 거기에 텐서플로와 필요한 패키지를 설치 + 실행

conda create -n tensorflowtest python=3.5

```
Anaconda Prompt  
(C:\Anaconda3) C:\Users\JY>  
(C:\Anaconda3) C:\Users\JY>conda create -n tensorflowtest python=3.5  
Fetching package metadata: ..  
Solving package specifications: .  
  
Package plan for installation in environment C:\Anaconda3\envs\tensorflowtest:  
  
The following NEW packages will be INSTALLED:  
  
certifi: 2018.4.16-py35_0  
pip: 9.0.3-py35_0  
python: 3.5.5-h0c2934d_2  
setuptools: 39.0.1-py35_0  
vc: 14-h0510ff6_3  
vs2015_runtime: 14.0.25123-3  
wheel: 0.31.0-py35_0  
wincertstore: 0.2-py35hfebbdb8_0  
  
Proceed ([y]/n)? y  
  
vs2015_runtime 100% [########################################] Time: 0:00:00 10.96 MB/s  
vc-14-h0510ff6 100% [########################################] Time: 0:00:00 548.05 kB/s  
python-3.5.5-h 100% [########################################] Time: 0:00:01 11.82 MB/s  
certifi-2018.4 100% [########################################] Time: 0:00:00 9.77 MB/s  
wincertstore-0 100% [########################################] Time: 0:00:00 2.30 MB/s  
setuptools-39. 100% [########################################] Time: 0:00:00 11.22 MB/s  
wheel-0.31.0-p 100% [########################################] Time: 0:00:00 8.30 MB/s  
pip-9.0.3-py35 100% [########################################] Time: 0:00:00 11.70 MB/s  
#  
# To activate this environment, use:  
# > activate tensorflowtest  
#  
# To deactivate an active environment, use:  
# > deactivate  
#  
# * for power-users using bash, you must source  
#  
(C:\Anaconda3) C:\Users\JY>
```

activate tensorflowtest

» activate <프로젝트명>을 입력하면 해당 프로젝트 활성화

pip install --upgrade tensorflow

```
Anaconda Prompt - pip install --upgrade tensorflow
(tensorflowtest) C:\Users\JY>pip install --upgrade tensorflow
Collecting tensorflow
  Downloading https://files.pythonhosted.org/packages/3f/bb/dd01844cf88d15264d92e12a8b89526e1d805c082b8e945b632d4a1989a
/tensorflow-1.8.0-cp35-cp35m-win_amd64.whl (34.4MB)
    100% |#####| 34.4MB 38kB/s
Collecting gast>=0.2.0 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/5c/78/ff794fcae2ce8aa6323e789d1f8b3b7765f601e7702726f430e814822b9
/gast-0.2.0.tar.gz
Collecting tensorboard<1.9.0,>=1.8.0 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/59/a6/0ae6092b7542cfedba6b2a1c9b8dceaf278238c39484f3ba03b03f07803
/tensorboard-1.8.0-py3-none-any.whl (3.1MB)
    100% |#####| 3.1MB 415kB/s
```

[에러발생시!!]

**python -m pip install --upgrade pip
pip install --ignore-installed --upgrade tensorflow**

» 버전업과 기존 패키지를 무시하고 진행

```
>>> import tensorflow as tf  
>>> tf.__version__
```

언더바(_)가 두 개

1.프로젝트이름확인 2.파이썬실행

```
Anaconda prompt - python  
(tensorflowtest) C:\Users\JY>python  
Python 3.5.5 |Anaconda, Inc.| (default, Apr  7 2018, 04:52:34) [MSC v.1900 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import tensorflow as tf  
>>> tf.__version__  
'1.8.0'  
>>>
```

3.텐서플로우 버전조회

Anaconda Navigator

- » 윈도우 ‘시작’에서 ‘Anaconda Navigator’를 실행
- » ‘Application on’에서 방금 만든 “tensorflowtest”를 선택
- » 실습코드들은 Jupyter Notebook 또는 Spyder를 통해 실행



구글 클라우드 Colaboratory 사용

- » 개인 PC에 별다른 프로그램 설치 없이 구글에서 제공하는 Colaboratory에 접속하여 실습
- » 구글 가입과 로그인 과정의 설명은 생략

<https://colab.research.google.com>

Colaboratory 화면구성

<https://colab.research.google.com>

The screenshot shows the Google Colaboratory interface. A green arrow points from the text "파일이름" to the file name input field at the top. Another green arrow points from the text "메뉴" to the menu icon. A green bracket underlines the "코드" and "텍스트" buttons in the toolbar, with the text "코드나 텍스트창 추가" above them. A green arrow points from the text "코드나 텍스트창 추가" to the "Code" tab. A green arrow points from the text "텍스트를 추가할 수 있습니다" to the text area. A green arrow points from the text "큰 제목은 ## Colab.. ## 식으로 씁니다" to the text "Colaboratory에 오신 것을 환영합니다!". A green arrow points from the text "코드를 작성합니다" to the code cell containing the command `print('Hello, Colaboratory!')`. A green arrow points from the text "실행결과가 나옵니다" to the output cell showing the result "Hello, Colaboratory!". A green arrow points from the text "실행버튼" to the play button icon in the code cell.

파일이름

메뉴

코드나 텍스트창 추가

텍스트를 추가할 수 있습니다

큰 제목은 ## Colab.. ## 식으로 씁니다

Colaboratory에 오신 것을 환영합니다!

코드를 작성합니다

실행결과가 나옵니다

실행버튼

» 텐서플로는 1.12.0 버전이 설치되어 있음

```
!cat /proc/cpuinfo
```

```
'1.12.0-rc2'
```

» CPU 확인하기

```
!cat /proc/cpuinfo
```

```
processor      : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 79
model name    : Intel(R) Xeon(R) CPU @ 2.20GHz
stepping       : 0
```

» 메모리 확인하기

```
!cat /proc/meminfo
```

```
MemTotal:      13341832 kB
MemFree:        680672 kB
MemAvailable:   12286892 kB
Buffers:        148876 kB
Cached:        11095592 kB
```

◆ CPU는 인텔 Xeon 2.20GHz이며, 메모리도 13기가바이트나 제공

goo.gl/M4BBk4

실습코드 : <https://github.com/jaeyong1> → repositories에서 deeplearningforus선택 → chap02 선택
1_Colab_GPU찾기.txt

실습코드 : <https://github.com/jaeyong1> → repositories에서 deeplearningforus선택 → chap02 선택
2_GPU로_실행.txt

```
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
    raise SystemError('GPU장치를 찾지못했습니다.')
print('GPU를 찾았습니다. 장치 : {}'.format(device_name))
```

```
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
    raise SystemError('GPU장치를 찾지못했습니다.')
print('GPU를 찾았습니다. 장치 : {}'.format(device_name))

GPU를 찾았습니다. 장치 : /device:GPU:0
```

```
SystemError:Traceback (most recent call last)
<ipython-input-5-0e9a91f5221f> in <module>()
      2 device_name = tf.test.gpu_device_name()
      3 if device_name != '/device:GPU:0':
----> 4     raise SystemError('GPU장치를 찾지못했습니다.')
      5 print('GPU를 찾았습니다. 장치 : {}'.format(device_name))

SystemError: GPU장치를 찾지못했습니다.
```

SEARCH STACK OVERFLOW

파일 수정 보기 삽입 런타임 도구 도움말

코드 실행취소 Ctrl+Shift+Z

다시 실행 Ctrl+Shift+Y

모두 선택 Ctrl+Shift+A

선택항목 잘라내기

선택항목 복사

붙여넣기

선택된 셀 삭제 Ctrl+M D

찾기/바꾸기 Ctrl+H

다음 찾기 Ctrl+G

이전 찾기 Ctrl+Shift+G

노트 설정

코드 표시/숨기기

모든 출력 지우기

이브로 복사

Confirm TensorFlow can see the GPU

Simply select "GPU" in the Accelerator drop-down in Notebook Settings (either through the cmd/ctrl-shift-P).

The screenshot shows the 'Notebook Settings' dialog in Google Colab. It includes fields for '런타임 유형' (Runtime Type) set to 'Python 2', '하드웨어 가속기' (Hardware Accelerator) set to 'GPU', and a checkbox for '이 노트를 저장할 때 GPU 출력 선택' (Select GPU output when saving notebook). The 'GPU' option in the dropdown is circled in red.

실습코드 : <https://github.com/jaeyong1> → repositories에서 deeplearningfromScratch선택 → chap02 선택
1_Colab_GPU찾기.txt



Time (s) to convolve 32x7x7x3 filter over random 100x100x100x3 images (batch x height x width x channel). Sum of ten runs.

CPU로 계산시간 (초):

8.22801208496

GPU로 계산시간 (초):

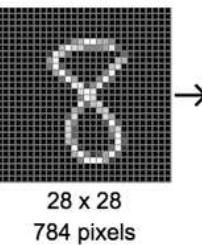
0.18327999115

GPU 속도는 CPU보다 44x 배 빠릅니다.

- » 테스트한 시점에는 GPU가 CPU보다 약 44배 빠른 결과를 보임
- » 구글클라우드는 많은 사람들과 같이 쓰는 공유자원이므로 그때그때 성능차이가 다를 수 있음.
- » 사용량이 많은 경우에는 사용하지 못할 수도 있음(잠시 기다렸다가 다시 시작가능)
- » 현재는 무료로 누구나 사용 가능하지만 과금정책은 차후에 변경될 수 있음.

- » 손으로 쓴 숫자를 이미지화 한 데이터 셋
 - » 실제로는 숫자 하나하나씩 나뉘어져 저장되어 있음
 - » 약 60,000여 개의 train set
 - » 약 10,000여 개의 test set
 - » 데이터의 크기는 일정하며, 숫자정보는 이미지 가운데
 - » 각각의 픽셀정보를 정수 값으로 표현한 것

- » 손쉽게 사용할 수 있기 때문에
머신러닝을 처음 배우는 사람부터,
알고리즘을 전문적으로 연구하는 사람까지
다양한 수준의 사람들이 사용중



000000000000000000
111111111111111111
222222222222222222
333333333333333333
444444444444444444
555555555555555555
666666666666666666
777777777777777777
888888888888888888
999999999999999999

goo.gl/UJ9GdT

실습코드 : <https://github.com/jaeyong1> → repositories에서 deeplearningforus선택 → chap02 선택
3_MNIST_다운받기.txt

실습코드 : <https://github.com/jaeyong1> → repositories에서 deeplearningforus선택 → chap02 선택
4_MNIST_확인하기.txt

```
from __future__ import division
from __future__ import print_function
import os.path
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
tf.logging.set_verbosity(tf.logging.ERROR)
from tensorflow.examples.tutorials.mnist import input_data
```

```
#MNIST를 인터넷에서 받아서 mnist_data 폴더에 저장
mnist = input_data.read_data_sets('mnist_data', one_hot=True)
```



Extracting mnist_data/train-images-idx3-ubyte.gz
Extracting mnist_data/train-labels-idx1-ubyte.gz
Extracting mnist_data/t10k-images-idx3-ubyte.gz
Extracting mnist_data/t10k-labels-idx1-ubyte.gz

- » 코드에서 지정한 “mnist_data” 폴더가 생겨있고, 안에 4개 파일이 다운로드 됨



```
!ls -al  
!ls mnist_data -al
```



```
total 20  
drwxr-xr-x 1 root root 4096 Nov  4 13:58 .  
drwxr-xr-x 1 root root 4096 Nov  4 13:21 ..  
drwxr-xr-x 4 root root 4096 Nov  1 16:29 .config  
drwxr-xr-x 2 root root 4096 Nov  4 13:58 mnist_data  
drwxr-xr-x 2 root root 4096 Nov  1 16:42 sample_data  
total 11344  
drwxr-xr-x 2 root root    4096 Nov  4 13:58 .  
drwxr-xr-x 1 root root    4096 Nov  4 13:58 ..  
-rw-r--r-- 1 root root 1648877 Nov  4 13:58 t10k-images-idx3-ubyte.gz  
-rw-r--r-- 1 root root   4542 Nov  4 13:58 t10k-labels-idx1-ubyte.gz  
-rw-r--r-- 1 root root 9912422 Nov  4 13:58 train-images-idx3-ubyte.gz  
-rw-r--r-- 1 root root 28881 Nov  4 13:58 train-labels-idx1-ubyte.gz
```

» one-hot 인코딩은 결과를 1과 0을 사용하여 값을 표현하는 방법

» 만약 세 가지 값이 존재한다면?

- ◆ 100, 010, 001 가능

- ◆ 한 개만 1로 표시하고 나머지를 모두 0으로 표시하는 방법

[0.01, [0,
0.04, → 0,
0.95] 1]

» MNIST의 정답레이블은 0에서 9까지 중 하나를 표시하므로 열 자리 숫자로 표현

- ◆ 열 개 중 한 개는 1이며 나머지는 0

- ◆ 정답레이블이 7이라면 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]

- ◆ 코드에서 one_hot을 True로 했기 때문에 이렇게 저장됨

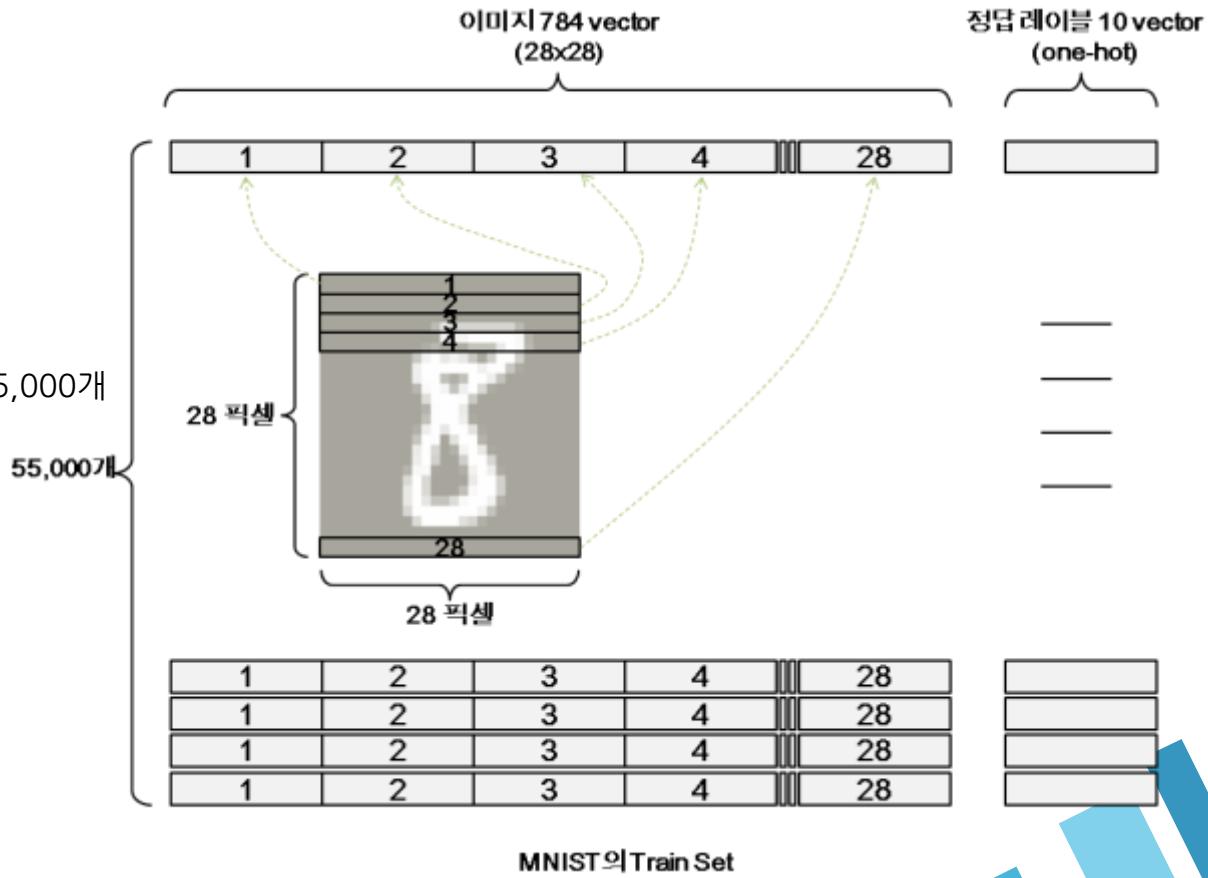
» Train set

- ❖ 학습을 위한 데이터셋
- ❖ image의 shape는 (55000, 784)

= 숫자이미지 픽셀정보

= 784는 가로 세로가 28*28인 픽셀 값을
한 줄로 이어놓을 때의 벡터값

= 784개로 이루어진 숫자 이미지 정보가 55,000개
❖ labels의 shape는 (55000, 10)
= 정답레이블



» Test set

- ◊ image의 shape는 (10000, 784),
- ◊ labels는 (10000, 10)
- ◊ 총 10,000장의 이미지를 가지고 있음

» Validation set

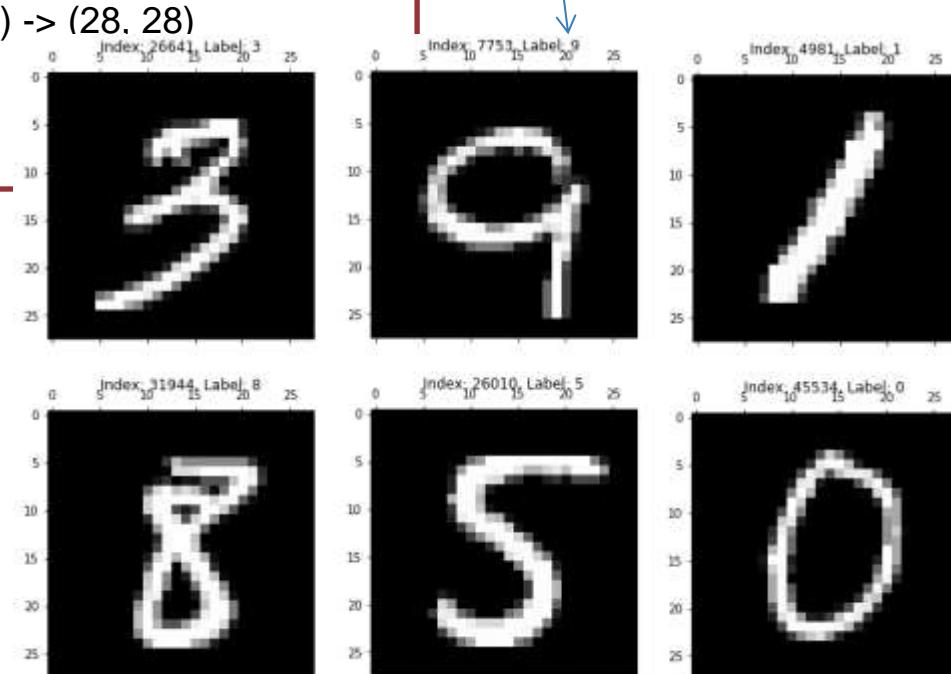
- ◊ image의 shape (5000, 784)
- ◊ labels의 shape (5000, 10)
- ◊ 5,000장의 이미지 정보를 가지고 있음

```
#show 6 images from random index
for i in np.random.randint(55000, size=6):
    imgvec = mnist.train.images[i, :]
    labelvec = mnist.train.labels[i, :]
    imgmatrix = np.reshape(imgvec, (28, 28)) # (784,) -> (28, 28)
    label = np.argmax(labelvec) #[0 0 1..] -> 2
    plt.matshow(imgmatrix, cmap=plt.get_cmap('gray'))
    plt.title("Index: %d, Label: %d" % (i, label))
```

- » 55000개를 모두 확인하기에는 양이 많으니
6개만 랜덤으로 뽑아서 화면에 출력
- » argmax는 one hot 인코딩 역변환

$$[0, 0, 1, 0, 0] \rightarrow 2$$

Argmax로 타이틀에 정답 출력

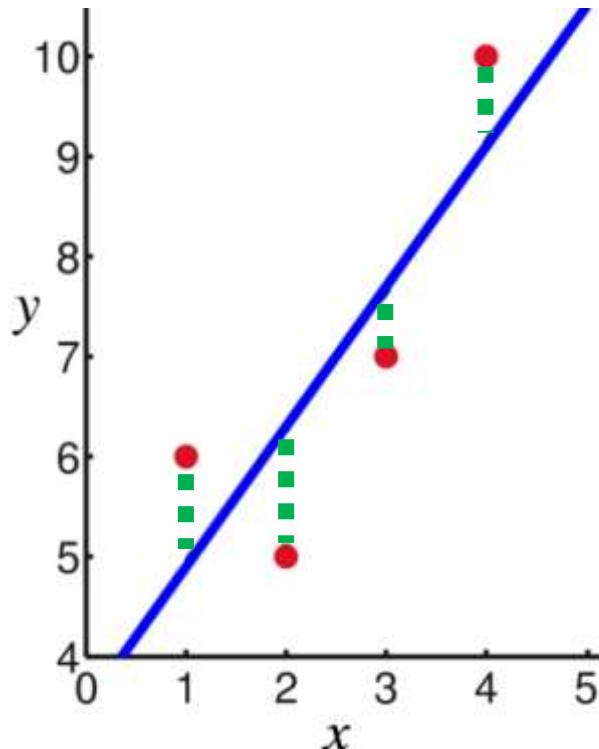


3.

머신러닝 알고리즘

1. 학습하고자 하는 가설 hypothesis를 수식으로 나타낸다.
2. 가설의 성능을 측정할 수 있는 비용함수(cost function)을 정의한다.
3. cost function을 최소화 할 수 있는 학습 알고리즘을 설계한다.

» 선형회귀. 하나이상의 독립변수 X와 종속변수 Y의 관계를 모델링방법



점 : 실제 데이터(x, y)를 2차원 평면에 표시

직선 : 점들을 가장 잘 표현하는 $y=ax+b$ 그래프

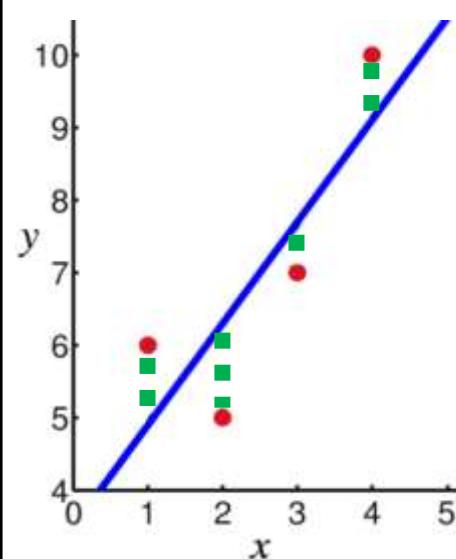
단, 모든 점 데이터가 완벽히 $y=ax+b$ 로 표현될 수 없다면
가상의 직선과 실제 값 사이에는 오차가 존재

점과 직선 사이의 편차가 가장 작은 가상의 $y=ax+b$ 식을,
→ Hypothesis, 줄여서 h 또는 가설

Linear Regression의 Cost function(비용함수)의 이해

- » 실제 점 데이터와 가설의 그래프간 오차의 비교를 위함
- » 정확한 크기 비교를 위해서는 정형화된 수식이 필요
- » 예제데이터 좌표 : (1,6) (2,5) (3,7) (4,10)
- » 예제데이터는 4개이므로 $m=4$

$$\frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$



인덱스 i	실제점데이터X	실제점데이터Y	예측그래프Y	두 Y의 오차
1	1	6	4.9	1.1
2	2	5	6.3	-1.3
3	3	7	7.7	-0.7
4	4	10	9.1	0.9

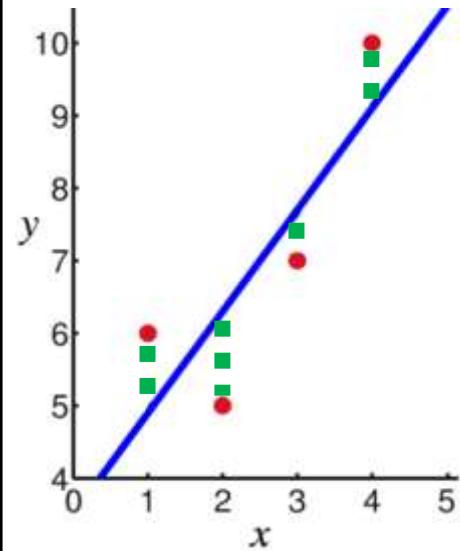
(예측그래프: $y = 1.4x + 3.5$)

Linear Regression의 Cost function(비용함수)의 이해

» 예제를 직접 계산해 봅시다.

$$\begin{aligned} \text{Cost} &= \frac{1}{8} \times \{(1.1)^2 + (-1.3)^2 + (-0.7)^2 + (0.9)^2\} \\ &= \frac{1}{8} \times 4.2 = 0.525 \end{aligned}$$

$$\frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

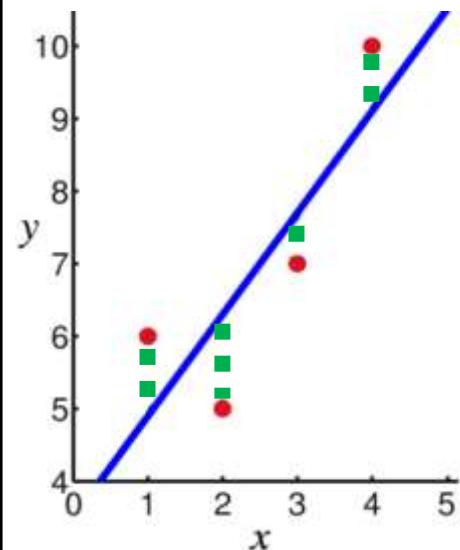


i	$x^{(i)}$	$y^{(i)}$	$h_\theta(x^{(i)})$	$h_\theta(x^{(i)}) - y^{(i)}$
1	1	$y^{(1)} = 6$	$h_\theta(x^{(1)}) = h_\theta(1) = 4.9$	1.1
2	2	$y^{(2)} = 5$	$h_\theta(x^{(2)}) = h_\theta(2) = 6.3$	-1.3
3	3	$y^{(3)} = 7$	$h_\theta(x^{(3)}) = h_\theta(3) = 7.7$	-0.7
4	4	$y^{(4)} = 10$	$h_\theta(x^{(4)}) = h_\theta(4) = 9.1$	0.9

한번에 $y=ax+b$ 를 찾을 수 있을까?

» 가설(예측직선)을 임의로 그렸을 때, 단번에 정확한 수식을 알아맞히는 것은 쉽지 않음.

» 직선의 형태이므로 $y=wx+b$ 라는 형태라는 것을 알고 있을 때,

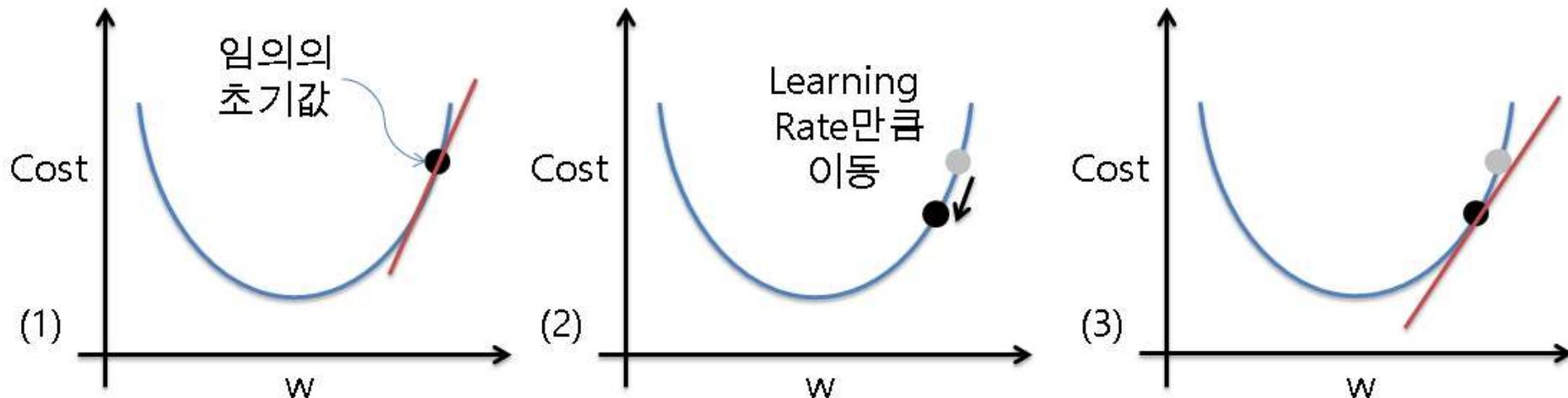


» 조금씩 w 와 b 를 바꾸어 보면서 cost값이 최소값을 찾는 과정을 상상해 봅시다 

» cost가 가장 작은 지점을 찾았다면 분명 $w=1.40$ 이고 $b=3.5$ 일 때

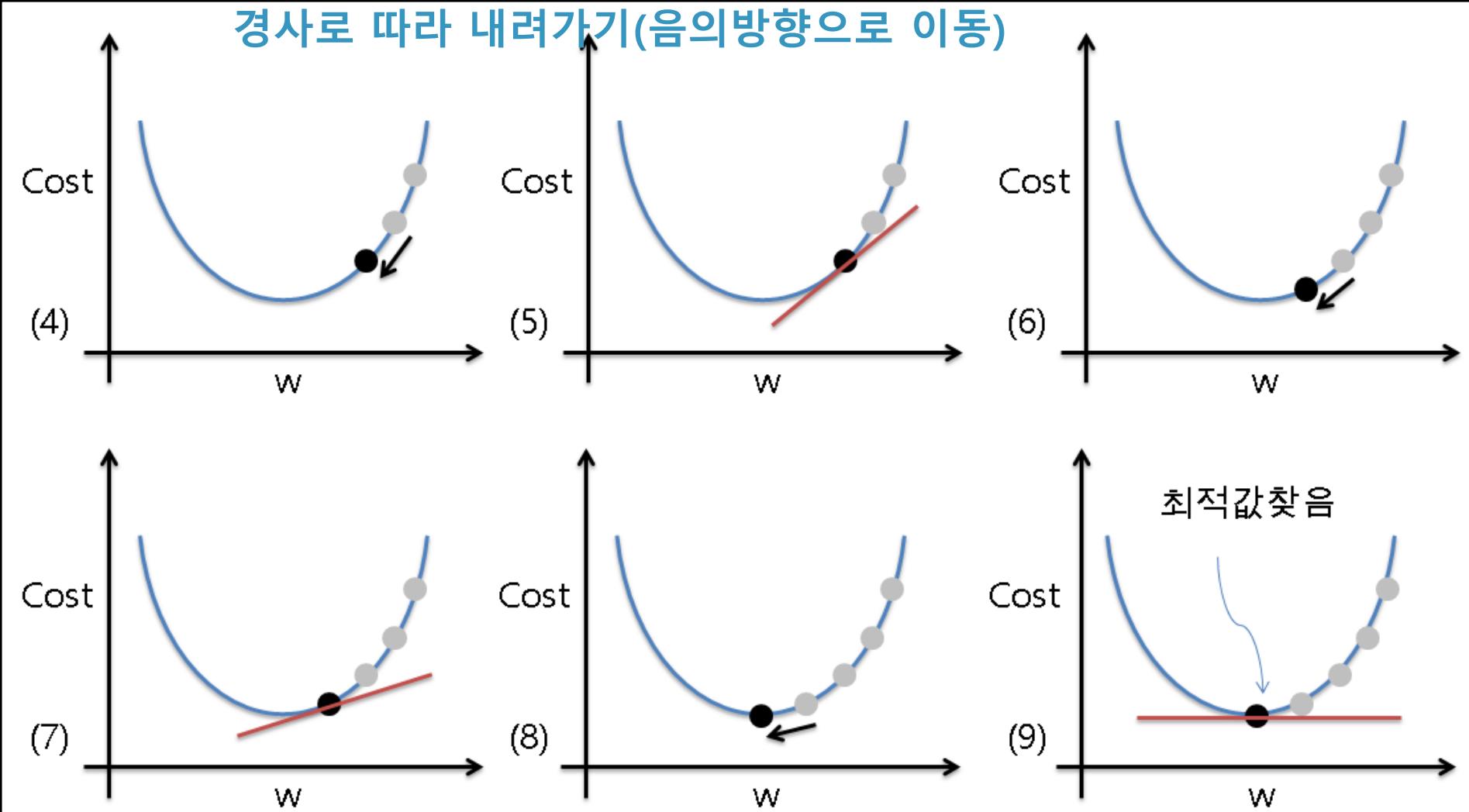
- » Optimizer : 최적화 함수를 반복적으로 실행하여 가장 좋은 값을 찾아가는 일을 함
 - ◊ 다양한 Optimizer가 발표되어 있음.
 - ◊ “경사면 따라 내려가기” 인 Gradient Descent 알고리즘을 소개
- » 개념적 이해를 위해 단순화하여, $y=wx+b$ 대신 $y=wx$ 그래프로 설명함

경사로 따라 내려가기(음의방향으로 이동)



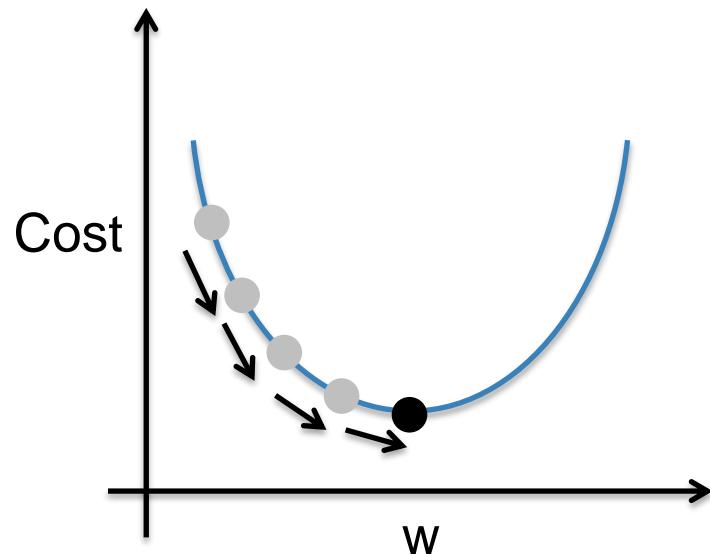
- » w 는 랜덤초기값으로 설정하고 최적값을 찾아가는 과정을 시작
- » 현재 값에서 미분을 하여 기울기를 구하여 기울기가 양수라면 음의 방향으로 이동
- » 과정을 기울기가 0(또는 끝나는 조건)이 될 때까지 반복
- » 이 이동하는 정도를 Learning rate라 함

경사로 따라 내려가기(음의방향으로 이동)



경사로 따라 내려가기(양의방향으로 이동)

- » 만약 초기값을 최적값보다 작은 값으로 시작했다면?
- » 미분하면 기울기는 음수.
- » 점점 양의 방향으로 이동하면서 최적값을 찾아나갈 수 있음.



부적절한 Learning Rate 경우

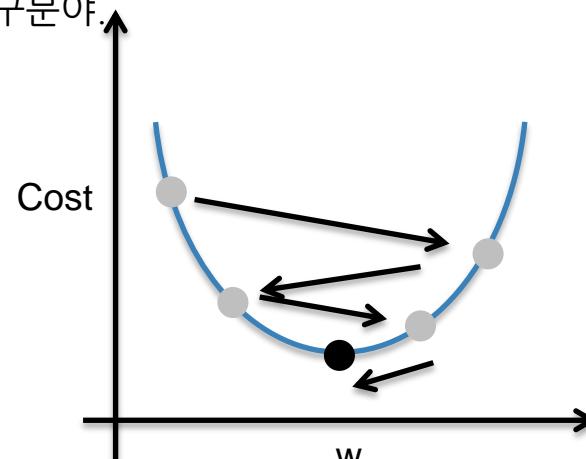
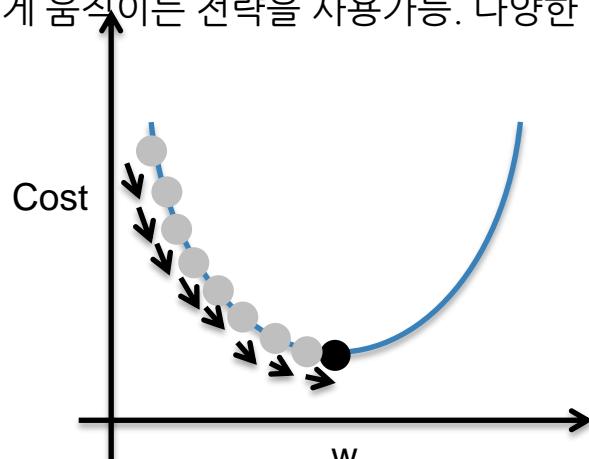
» Learning Rate가 너무 작다면?

- ◊ 최적값까지 도달하는데 필요이상으로 많은 횟수의 계산필요.
- ◊ 시간이 오래 걸림

» Learning Rate가 너무 크다면?

- ◊ 성큼성큼 이동하다가 최적값을 지나쳐버림

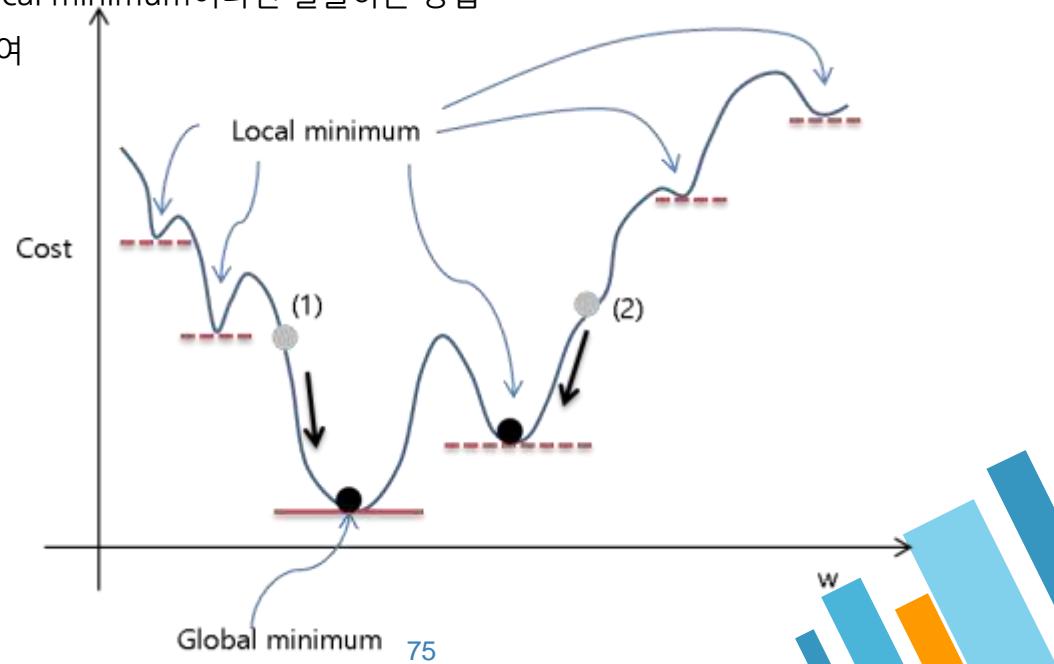
» 이를 보완하기 위하여 처음에는 좀 성큼성큼 움직이다가 이후에는 최적값 근처에 왔을 수 있기 때문에 촘촘하게 움직이는 전략을 사용가능. 다양한 연구분야.



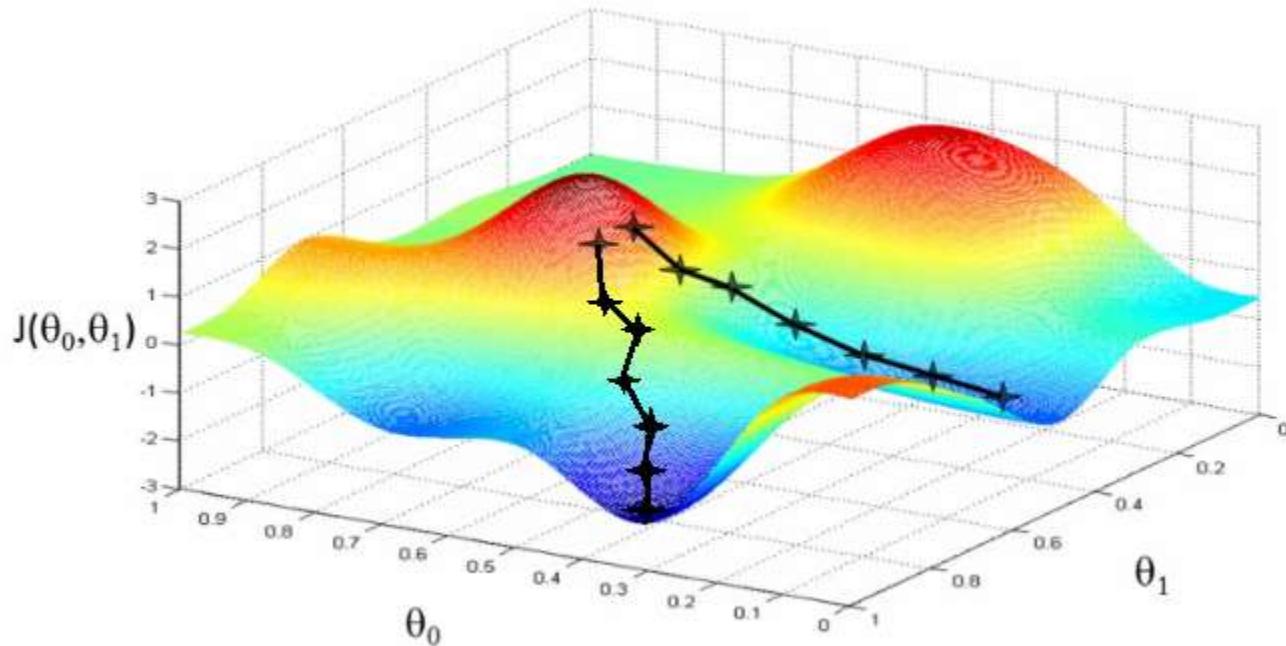
Global minimum과 Local minimum

- » 그래프를 통틀어서 가장 최적화된 점을 global minimum
- » 현재지점에서 가까운 지역적인 최적값을 local minimum
- » Local Optima에 빠졌다면?

- ◊ 내리막에서 속도가 붙는 것처럼 가중치를 사용하여 local minimum이라면 탈출하는 방법
- ◊ optimizer를 여러 개를 만들고 시작지점을 다르게 하여 각자가 local minimum에 빠졌더라도 그 중에서 제일 좋은 값을 찾는 방법
- ◊ 그밖에 다양한 제안들...



3차원그래프에서 경사로 따라 내려가기



- » 3차원그래프(두 개의 변수)에서 출발점이 약간만 달라지더라도, 서로 완전히 다른 경사로를 타고 내려가서 다른 최적점에 도달할 수 있음

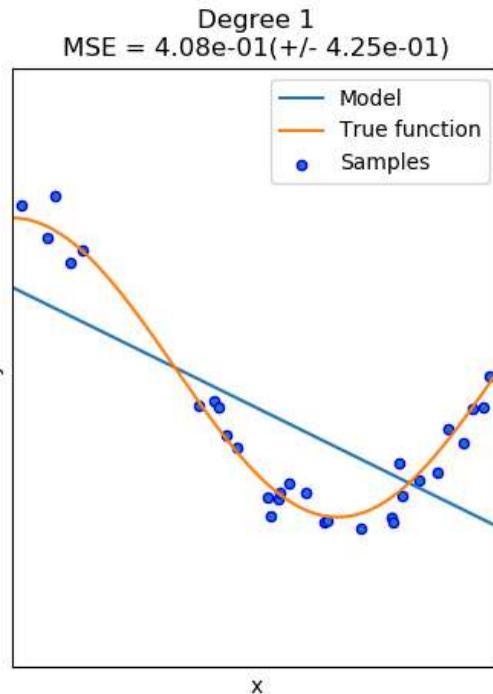
오버피팅(Overfitting)

» 머신러닝의 학습이 잘 안 되는 원인으로 가장 많이 언급

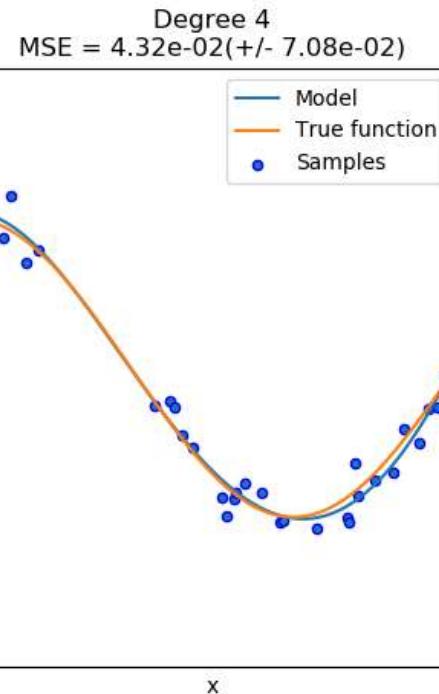
점 : 실제 데이터

주황색 : 정답그래프(True function)

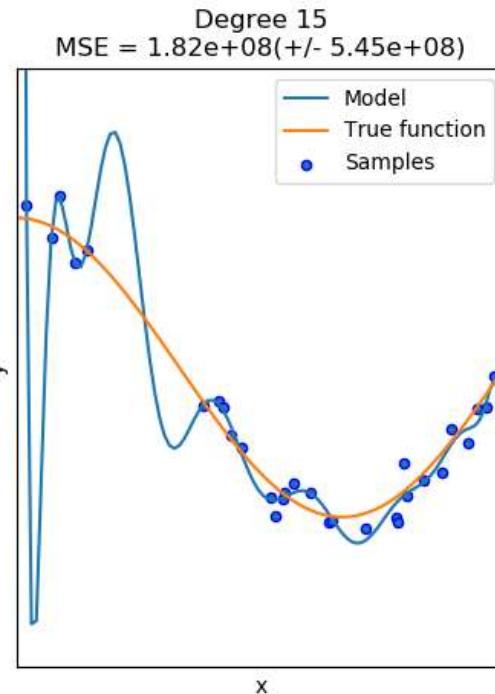
파란색 : 예측모델



1차함수:너무 오차가 큼



4차함수:적절해보임.
하지만 완벽하진 않음



15차함수:완벽히 모든점 지남

» 머신러닝의 학습

15차 함수가 가장 적절한 표현식일까요?

제 데이터

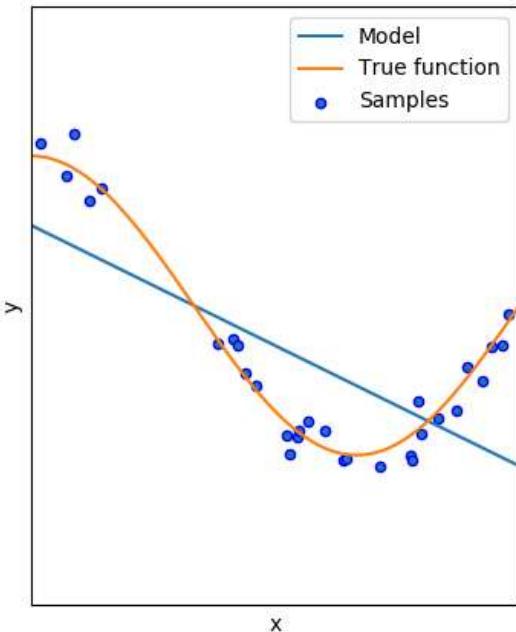
: 정답그래프(True function)

: 예측모델

Degree 1

MSE = 4.08e-01 (+/- 4.25e-01)

Model
True function
Samples

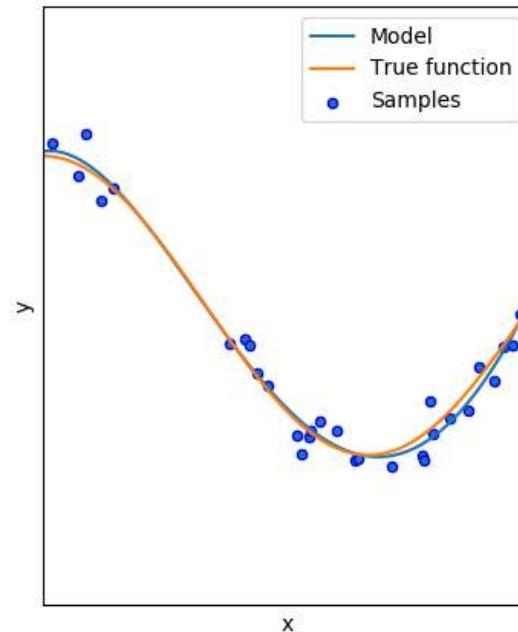


1차함수: 너무 오차가 큼

Degree 4

MSE = 4.32e-02 (+/- 7.08e-02)

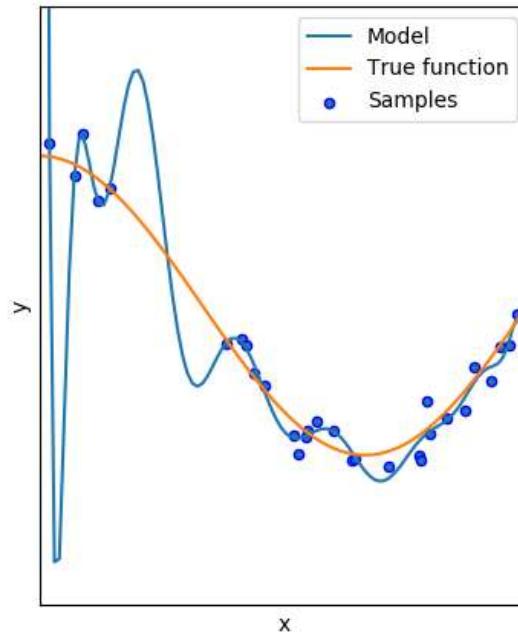
Model
True function
Samples

4차함수: 적절해보임.
하지만 완벽하진 않음

Degree 15

MSE = 1.82e+08 (+/- 5.45e+08)

Model
True function
Samples

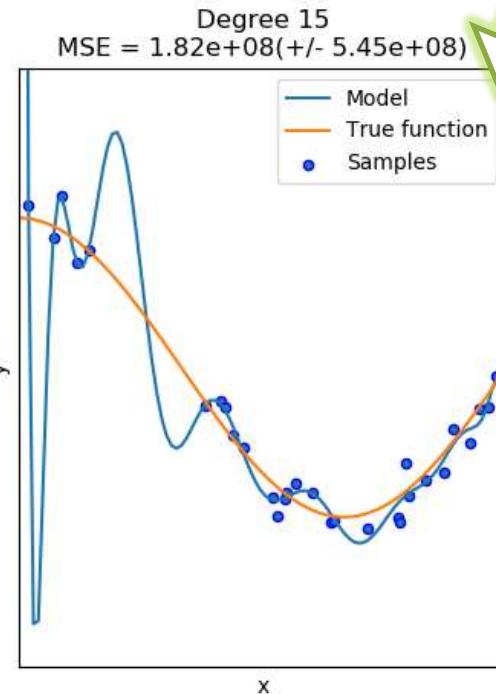
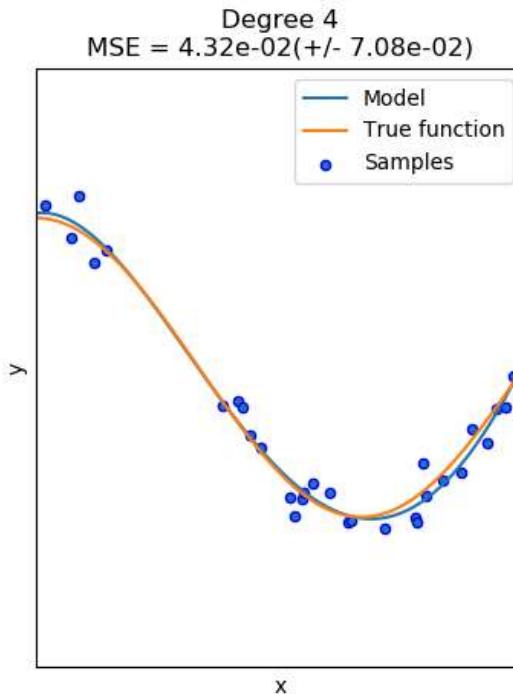
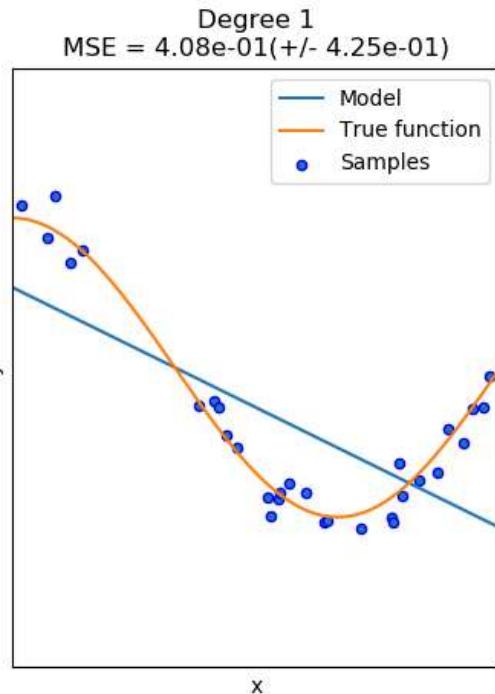


15차함수: 완벽히 모든점 지남

오버피팅(Overfitting)

» 머신러닝의 학습이 잘 안 되는 원인으로 가

불필요한 데이터의 노이즈까지
과하게 학습된 것으로 판단



그러면 언제까지??

sli.do # Train Set과 Validation Set 그리고 Test Set

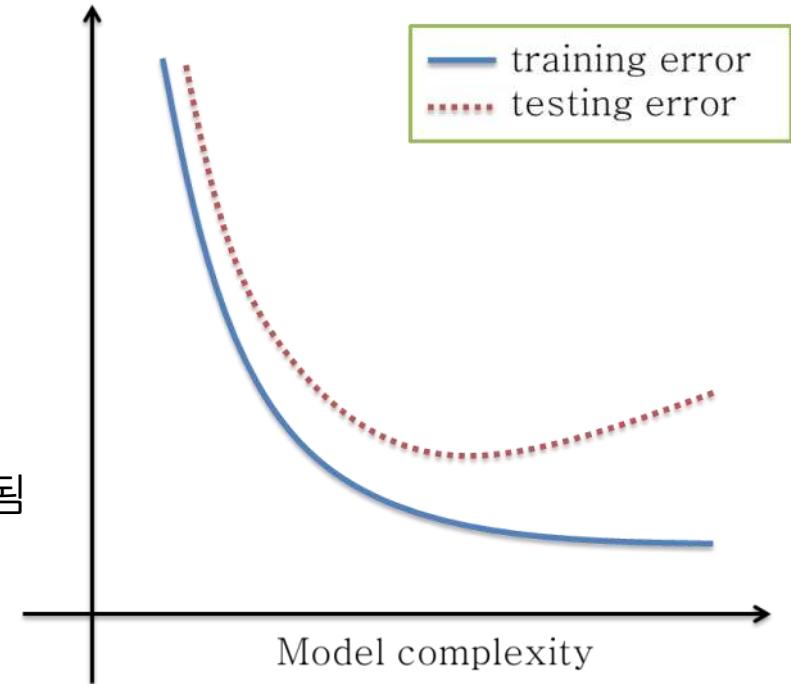
- » 오버피팅문제 : 언제 가장 적절한 식을 스스로 만들었다고 결정하기 어려움
- » 주어진 데이터를 적절히 나누어서 학습에 사용하지 않고 남겨둠



- » Train set : 실제 학습에 사용할 데이터를
- » Validation set : 옳은 방향으로 업데이트가 되고 있는지 관찰목적.
트레이닝에는 사용되지 않음.
- » Test set : 예측식의 최종성능을 도출.
트레이닝에는 사용되지 않음.

Training set과 Test set의 에러율

- » 학습을 계속 할 수록..
 - ◊ 수식의 복잡도, 차원이 증가
 - ➔ 학습데이터에 대한 오차는 점점 감소
- » 에러율은 어느 시점을 지나면 다시 증가
- » 데이터셋은 어느 한쪽으로 편중된 데이터여서는 안됨



» [가상의 데이터셋] 1분마다 트래픽 데이터를 일 년 동안 수집

→ 50만개

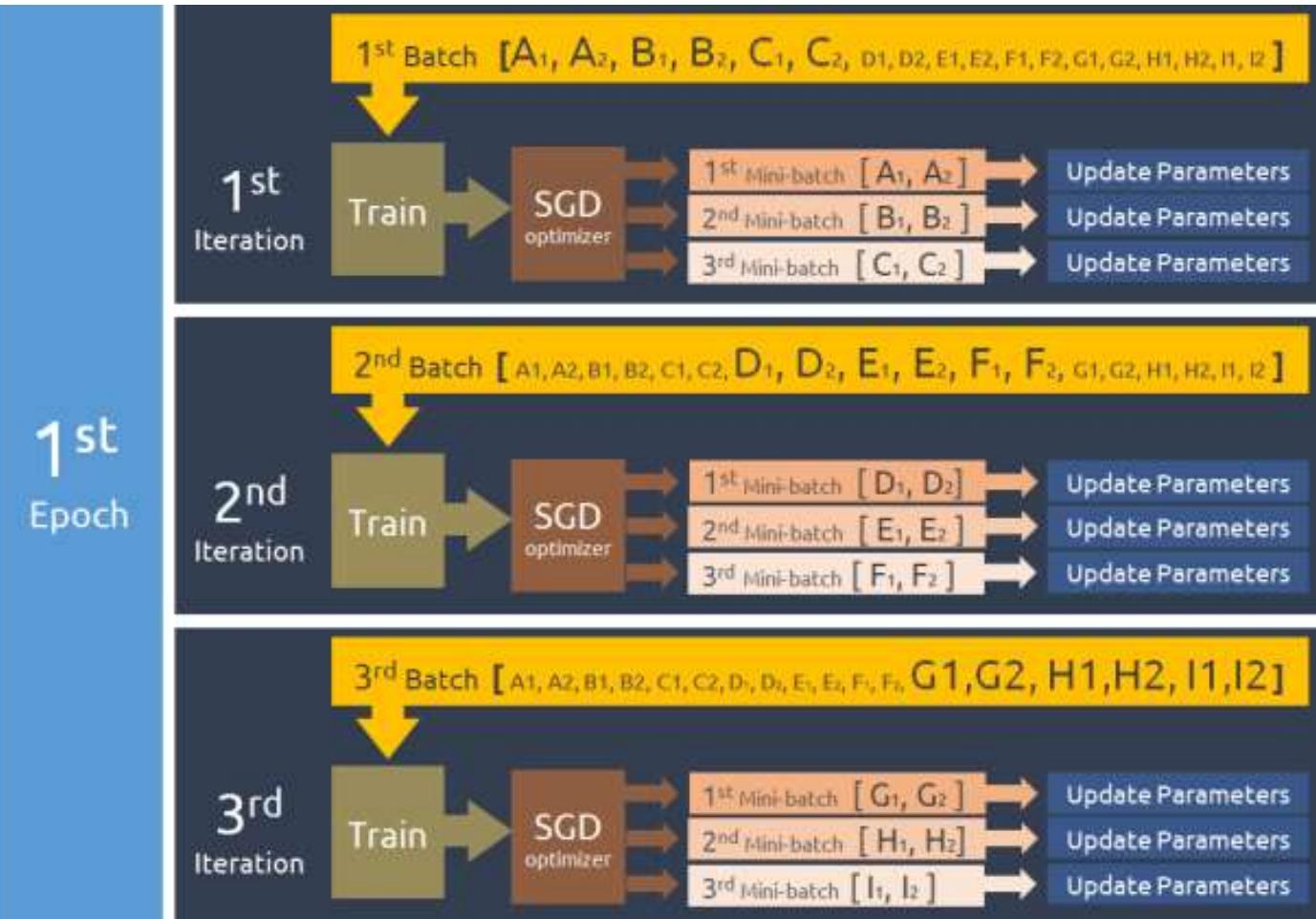
» 앞 써 사용한 방법대로, Optimizer가 최적화하려면

- ◊ 모든 데이터의 COST계산
- ◊ 업데이트 방향결정
- ◊ Weight , Bias 업데이트
- ◊ 반복..

» 자주 업데이트 하도록(속도향상)으로 데이터셋 크기를 효율적으로 줄여보자

↔ 데이터가 많을 수록 예측 성능이 좋아짐

Batch, Mini batch 그리고 Epoch



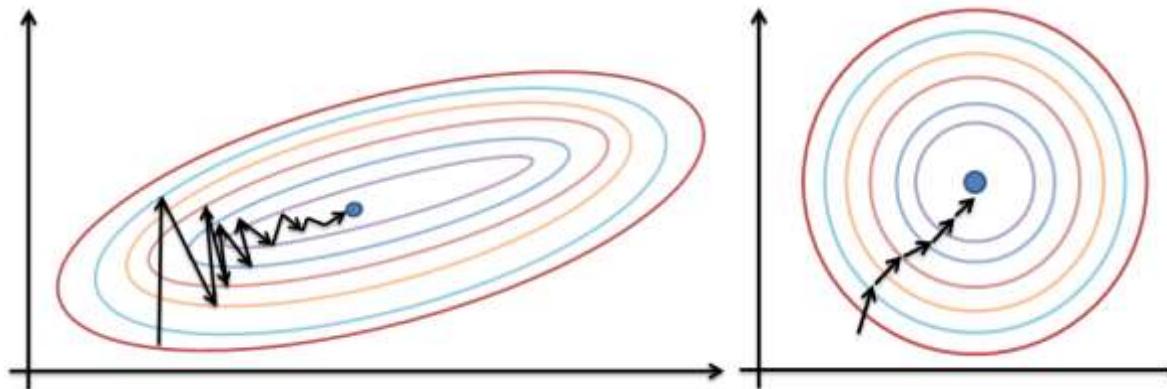
» 통계적으로 정규화는 다양한 의미

- ◊ (단순하게) 평준화의 의미.
- ◊ 평균이 다른 집단간 일반적인 기준으로 조정하는 것을 의미
- ◊ 확률분포를 조정

» 데이터의 정규화

→ 서로 다른 데이터셋 간의 데이터의 범위를 조정하거나 분포를 맞춤

- » 더 빨리 학습하고 Local Optima(국소적인 최적점)에 빠질 가능성을 줄여줌



정규화 적용전후 Gradient Descent Algorithm(경사로 따라 내려가기)의 동작

4.

가벼운 파이썬문법과 TF API 리뷰

파이썬 기본명령어

NUMPY 소개

TensorFlow API 소개

sli.do #_ 파이썬 기본명령어(print, for, if, 함수)



```
x = 123:  
print ("정수출력: %d" % x)
```



정수출력: 123



```
for x in range(0, 3):  
    print ("for연습 : %d" % (x))
```



for연습 : 0

for연습 : 1

for연습 : 2



```
animals = ["개", "고양이", "곰", "사자", "두더지"]  
for x in animals:  
    print ("[%s]" % x)
```



[개]

[고양이]

[곰]

[사자]

[두더지]



```
a=10  
if (a>10):  
    print("값은 10보다 큽니다.")  
elif (a==10):  
    print("값은 10입니다.")  
else :  
    print("값은 10보다 작습니다.")
```



값은 10입니다.



```
def sum_diff(a, b):  
    return a+b, a-b
```



함수 여러값 리턴

a = 100

b = 200

c, d = sum_diff(a, b)

print ("c : %d + %d = %d" % (a, b, c))

print ("d : %d - %d = %d" % (a, b, d))



c : 100 + 200 = 300

d : 100 - 200 = -100

파이썬 기본명령어(List)



```
# 리스트 길이
animals = ["개", "고양이", "곰", "사자", "두더지"]
print(animals)
print ("아이템수 = ", len(animals))
print (''개'의 수 = ", animals.count("개"))
```



['개', '고양이', '곰', '사자', '두더지']
 아이템수 = 5
 '개'의 수 = 1



```
animals = ["개", "고양이", "곰", "사자", "두더지"].
# 리스트 잘라내기
print("잘라내기[0:2] = ", animals[0:2])
print("잘라내기[:] = ", animals[:])
print("잘라내기[:-1] = ", animals[:-1])
```

끝에서 첫 번째 라는 의미 -1



잘라내기[0:2] = ['개', '고양이']
 잘라내기[:] = ['개', '고양이', '곰', '사자', '두더지']
 잘라내기[:-1] = ['개', '고양이', '곰', '사자']

파이썬 기본명령어(Dictionary)



dictionary 사용 예시

```
dict = {'이름': '바둑이', '견종': '푸들', '나이': 5}  
print ("dict['이름'] = ", dict['이름'])
```



dict['이름'] = 바둑이

- » 배열, 수치연산을 도와주는 파이썬의 범용적인 라이브러리
- » 텐서플로우가 제공하는 API들은 완벽히 numpy의 입력, 출력과 일치

```
import numpy as np
```

- » 많은 예제코드에서 numpy를 np라고 줄여서 사용

» 데이터와 함께 넘겨진 정수형(np.int), 문자열(np.string), 실수형(np.float)에 맞추어서 저장



```
import numpy as np
lst = [1, 2, 3, 4]
arr1 = np.array(lst, np.int)
print(arr1)
arr2 = np.array(lst, np.str)
print(arr2)
arr3 = np.array(lst, np.float)
print(arr3)
```



```
[1 2 3 4]
['1' '2' '3' '4']
[1. 2. 3. 4.]
```

- » 배열의 차원
- » 파이썬에서는 차원의 수를 RANK라고도 함

```
import numpy as np  
arr = np.array(5)  
print(arr)  
print(arr.ndim)  
# 실행결과 - 0차원
```

5
0

```
import numpy as np  
lst = [1, 2, 3, 4]  
arr = np.array(lst)  
print(arr)  
print(arr.ndim)  
# 실행결과 - 1차원
```

[1 2 3 4]
1

```
import numpy as np  
lst = [[1, 2], [3, 4]]  
arr = np.array(lst)  
print(arr)  
print(arr.ndim)  
# 실행결과 - 2차원
```

[[1 2]
 [3 4]]
2

- » 머신러닝을 경험하다 보면 꼭 사용하게 되는 함수
- » 배열이 가진 데이터를 유지하면서 배열의 모양만 바꿈



```
import numpy as np
arr = np.array([1, 2, 3, 4])
print("origin:\n", arr)
arr2 = arr.reshape(2, 2)
print("after:\n", arr2)
```



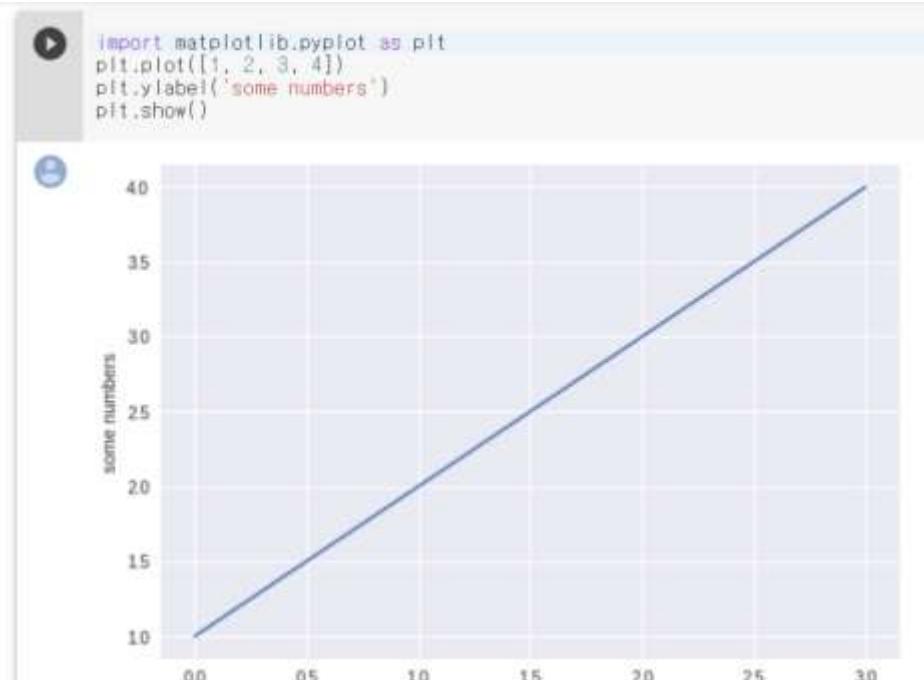
```
origin:
[1 2 3 4]
after:
[[1 2]
 [3 4]]
```

- » Not a Number의 의미
- » 외부파일을 읽어오거나 전처리 과정에서 발생할 수 있음
- » isnan()함수를 사용하여 nan여부를 검사할 수 있음

```
import numpy  
x=float('nan')  
print(numpy.isnan(x))
```

import matplotlib.pyplot as plt

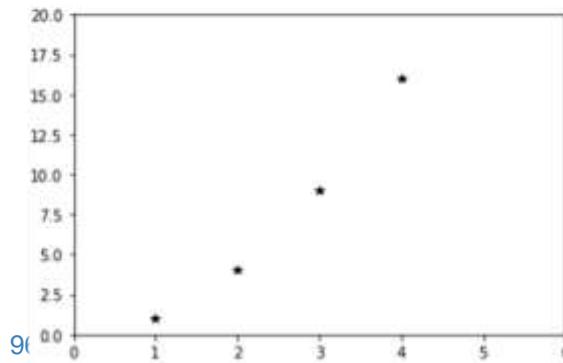
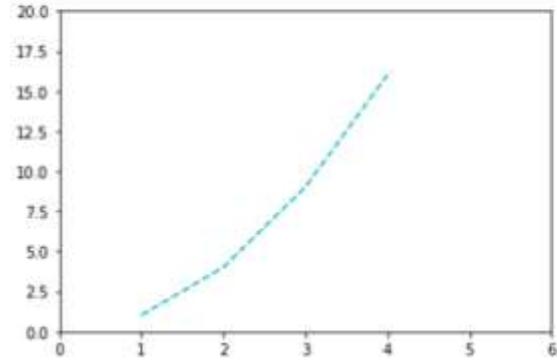
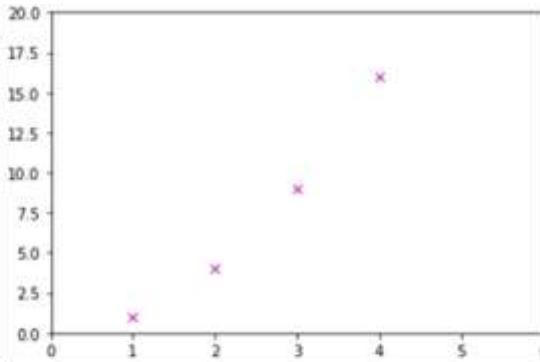
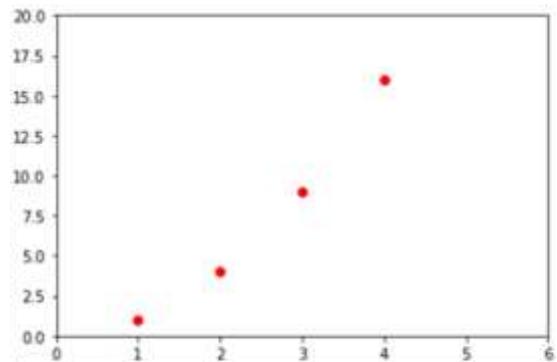
- » 시각화 패키지로 차트(chart)나 플롯(plot)으로 정보를 표시할 수 있는 기능을 제공
- » <http://matplotlib.org/> 에서 쉽게 확인 가능





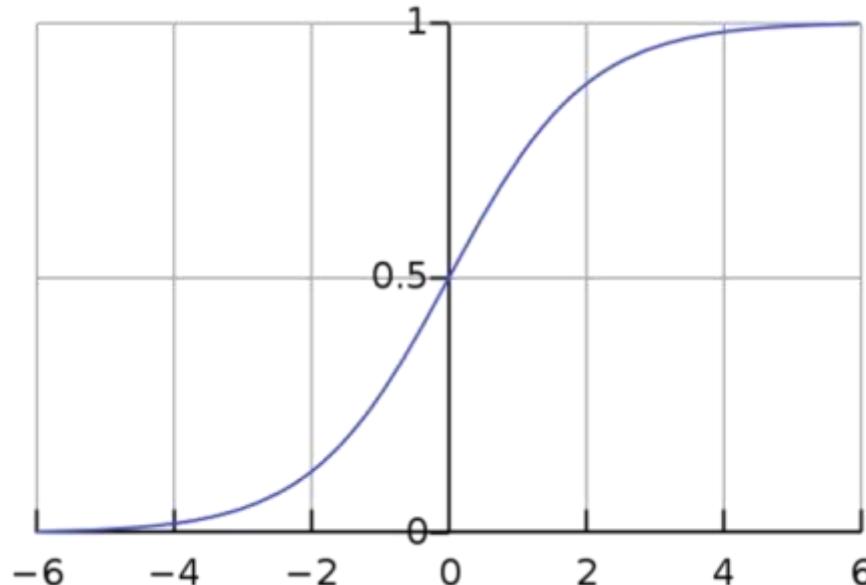
```
import matplotlib.pyplot as plt  
  
# red circle 그림  
plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'ro')  
plt.axis([0, 6, 0, 20])  
plt.show()
```

- » 동일코드, 표현방식차이
- » ro, mx, c--, k* 그림



Sigmoid 함수

- » 참과 거짓, Yes/No, True/False 등 양분화 되는 결과를 예측하는 경우 많음
- » 가장 큰 특징 :
모든 실수 입력 값 X 에 대해서도 Y 값의 범위가 0보다 크고 1보다 작은 값으로 표현
- » 계산결과를 시그모이드 함수에 통과시킴 ➔ 0~1 값으로 바뀜. 0.5를 중심으로 참/거짓 판단



$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

Sigmoid 함수

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

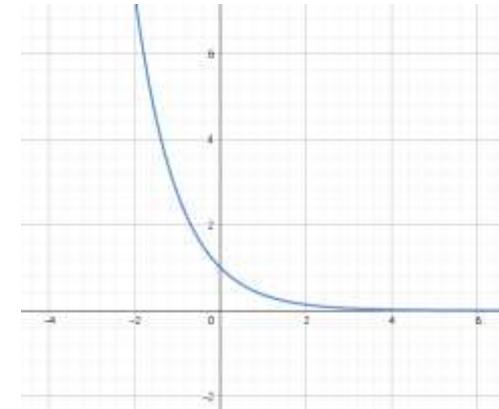
» Sigmoid를 줄여서 σ 로 표기

$a \rightarrow -\infty$ 일때, $\sigma(a) \rightarrow 0$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \approx \frac{1}{1 + \infty} \approx 0$$

$a \rightarrow +\infty$ 일때, $\sigma(a) \rightarrow 1$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \approx \frac{1}{1 + 0} \approx 1$$

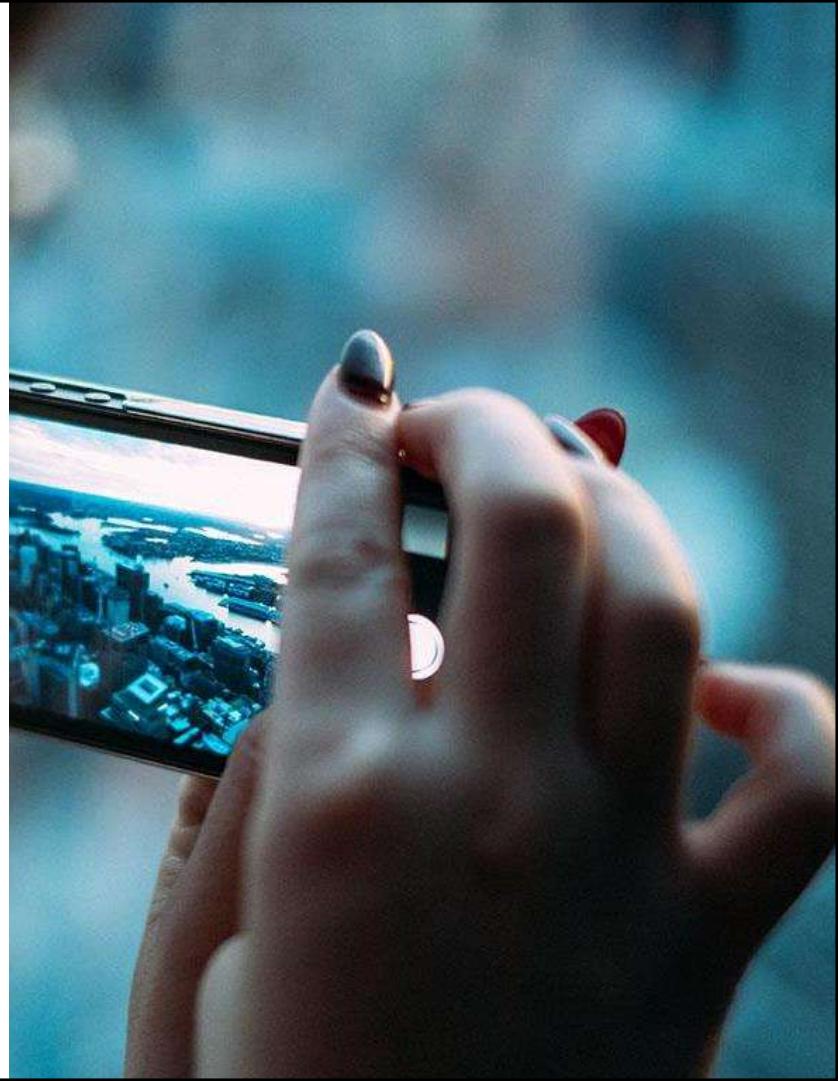


e 의 $-x$ 승 그래프

Linear Regression 실습

김성훈 교수님의 “모두를 위한 머신러닝/딥러닝 강의”

<https://hunkim.github.io/ml>



전통적인 프로그래밍

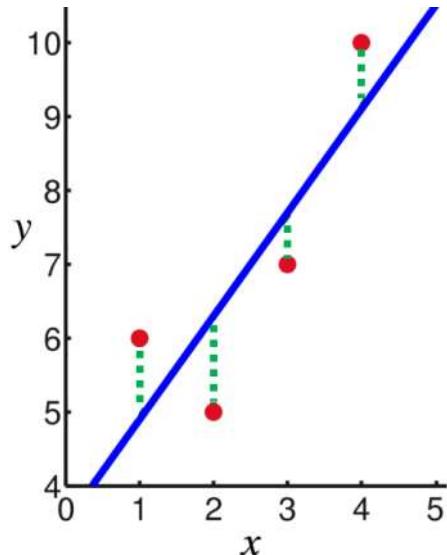


머신러닝 프로그래밍



Linear Regression 실습

» 주어진 데이터 4개. 이를 가장 잘 표현할 수 있는 직선그래프를 구하기



i(index)	$x^{(i)}$	$y^{(i)}$
1	1	$y^{(1)} = 6$
2	2	$y^{(2)} = 5$
3	3	$y^{(3)} = 7$
4	4	$y^{(4)} = 10$

» 정답

가장 잘 나타내는 식(cost값이 최소가 되는 회귀식)

$$y = 1.4x + 3.5$$

» 일단 실행해서 결과를 보고 공부합시다

<https://github.com/jaeyong1/deeplearningforus/>

→ Chap 04

→ 08_linear_regression.py

Colab 실습 : goo.gl/dcFygA

Linear Regression 실습

```

1 import tensorflow as tf
2
3 # 학습데이터 (X, Y)
4 x_train = [1, 2, 3, 4]
5 y_train = [6, 5, 7, 10]
6
7 # 변수 선언
8 W = tf.Variable(tf.random_normal([1]), name='weight')
9 b = tf.Variable(tf.random_normal([1]), name='bias')
10
11 # 가설식 정의 H(x) = Wx+b
12 hypothesis = x_train * W + b
13
14 # cost/loss function
15 cost = tf.reduce_mean(tf.square(hypothesis - y_train))
16
17 # Minimize, 최적화 함수
18 optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
19 train = optimizer.minimize(cost)
20

```

```

21 # Launch the graph in a session.
22 sess = tf.Session()
23
24 # Initializes global variables in the graph.
25 sess.run(tf.global_variables_initializer())
26
27 # 최적값 찾기 반복
28 for step in range(2000):
29     sess.run(train)
30     print(step, sess.run(cost), sess.run(W), sess.run(b))

```

1997	1.00000040	[1.4017715]	[3.4947740]
1998	1.0500042	[1.4017719]	[3.4947906]
1999	1.0500042	[1.4017665]	[3.494806]

Linear Regression 실습

```

1 import tensorflow as tf
2
3 # 학습데이터 (X, Y)
4 x_train = [1, 2, 3, 4]
5 y_train = [6, 5, 7, 10]
6
7 # 변수 선언
8 W = tf.Variable(tf.random_normal([1]), name='weight')
9 b = tf.Variable(tf.random_normal([1]), name='bias')
10
11 # 가설식 정의 H(x) = Wx+b
12 hypothesis = x_train * W + b
13
14 # cost/loss function
15 cost = tf.reduce_mean(tf.square(hypothesis - y_train))
16
17 # Minimize, 최적화 함수
18 optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
19 train = optimizer.minimize(cost)
20

```

그래프 구성(construction) 단계

```

21 # Launch the graph in a session.
22 sess = tf.Session()
23
24 # Initializes global variables in the graph.
25 sess.run(tf.global_variables_initializer())
26
27 # 최적값 찾기 반복
28 for step in range(2000):
29     sess.run(train)
30     print(step, sess.run(cost), sess.run(W), sess.run(b))

```

그래프 실행(execution) 단계

```

1 import tensorflow as tf
2
3 # 학습데이터 (X, Y)
4 x_train = [1, 2, 3, 4]
5 y_train = [6, 5, 7, 10]
6
7 # 변수 선언
8 W = tf.Variable(tf.random_normal([1]), name='weight')
9 b = tf.Variable(tf.random_normal([1]), name='bias')
10
11 # 가설식 정의 H(x) = Wx+b
12 hypothesis = x_train * W + b
13
14 # cost/loss function
15 cost = tf.reduce_mean(tf.square(hypothesis - y_train))
16
17 # Minimize, 최적화 함수
18 optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
19 train = optimizer.minimize(cost)
20

```

- » 텐서플로우 라이브러리를 import 함
- » 학습데이터(x, y) 입력

```

26
27 # 최적값 찾기 반복
28 for step in range(2000):
29     sess.run(train)
30     print(step, sess.run(cost), sess.run(W), sess.run(b))

```

```

1997 1.00000040 [1.4017715] [3.4947906]
1998 1.0500042 [1.4017719] [3.4947906]
1999 1.0500042 [1.4017665] [3.494806]

```

```

1 import tensorflow as tf
2
3 # 학습데이터 (X, Y)
4 x_train = [1, 2, 3, 4]
5 y_train = [6, 5, 7, 10]
6
7 # 변수 선언
8 W = tf.Variable(tf.random_normal([1]), name='weight')
9 b = tf.Variable(tf.random_normal([1]), name='bias')
10
11 # 가설식 정의 H(x) = Wx+b
12 hypothesis = x_train * W + b
13
14 # cost/loss function
15 cost = tf.reduce_mean(tf.square(hypothesis - y_train))
16
17 # Minimize, 최적화 함수
18 optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
19 train = optimizer.minimize(cost)
20

```

```

21 # Launch the graph in a session.
22 sess = tf.Session()
23
24 # Initializes global variables in the graph.
25 sess.run(tf.global_variables_initializer())

```

» Hypothesis의 정의.

- ◊ 가설식은 $H(x)=Wx+b$ 형태로 미리 정해 줌
- ◊ 4개의 점 (X, Y) 데이터를 입력
- ◊ W와 b를 컴퓨터가 최적값을 찾도록 변수(Variable)로 선언

1998	1.0500042	[1.4017719]	[3.4947906]
1999	1.0500042	[1.4017665]	[3.494806]

```

1 import tensorflow as tf
2
3 # 학습데이터 (X, Y)
4 x_train = [1, 2, 3, 4]
5 y_train = [6, 5, 7, 10]
6
7 # 변수 선언
8 W = tf.Variable(tf.random_normal([1]), name='weight')
9 b = tf.Variable(tf.random_normal([1]), name='bias')
10
11 # 가설식 정의 H(x) = Wx+b
12 hypothesis = x_train * W + b
13
14 # cost/loss function
15 cost = tf.reduce_mean(tf.square(hypothesis - y_train))
16
17 # Minimize, 최적화 함수
18 optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
19 train = optimizer.minimize(cost)
20

```

```

21 # Launch the graph in a session.
22 sess = tf.Session()
23
24 # Initializes global variables in the graph.
25 sess.run(tf.global_variables_initializer())
26
27 # 최적값 찾기 반복
28 for step in range(2000):
    ...

```

» 비용함수 (Cost Function) 정의

- ◊ 가설식과 실제데이터간의 거리를 평균
- ◊ `reduce_mean()`은 입력된 데이터들의 평균을 취함

```

1 import tensorflow as tf
2
3 # 학습데이터 (X, Y)
4 x_train = [1, 2, 3, 4]
5 y_train = [6, 5, 7, 10]
6
7 # 변수 선언
8 W = tf.Variable(tf.random_normal([1]), name='weight')
9 b = tf.Variable(tf.random_normal([1]), name='bias')
10
11 # 가설식 정의 H(x) = Wx+b
12 hypothesis = x_train * W + b
13
14 # cost/loss function
15 cost = tf.reduce_mean(tf.square(hypothesis - y_train))
16
17 # Minimize, 최적화 함수
18 optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
19 train = optimizer.minimize(cost)
20

```

```

21 # Launch the graph in a session.
22 sess = tf.Session()
23
24 # Initializes global variables in the graph.
25 sess.run(tf.global_variables_initializer())

```

» 최적화 함수(Optimizer)의 정의.

- ◊ 경사로 따라 내려가기 기법을 적용
- ◊ learning_rate = 0.01 (경험치)

» train은 그래프가 무엇을 계산할지 정의

➔ Optimizer가 경사로 따라 내려가기 기법을 사용하며,
cost가 점점 줄어드는 방향으로 학습을 진행

```

1 import tensorflow as tf
2
3 # 학습데이터 (X, Y)
4 x_train = [1, 2, 3, 4]
5 y_train = [6, 5, 7, 10]
6
7 # 변수 선언
8 W = tf.Variable(tf.random_normal([1]), name='weight')
9 b = tf.Variable(tf.random_normal([1]), name='bias')
10
11 # 가설식 정의 H(x) = Wx+b
12 hypothesis = x_train * W + b
13
14 # cost/loss function
15 cost = tf.reduce_mean(tf.square(hypothesis - y_train))
16
17 # Minimize, 최적화 함수
18 optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
19 train = optimizer.minimize(cost)
20

```

```

21 # Launch the graph in a session.
22 sess = tf.Session()
23
24 # Initializes global variables in the graph.
25 sess.run(tf.global_variables_initializer())
26
27 # 최적값 찾기 반복
28 for step in range(2000):
29     sess.run(train)

```

변수(Variable)초기화



- » 그래프를 실행하기 위해 Session을 만듦
- » 실행과정에서 컴퓨터는 주어진 “tf.Variable(변수)”를 적절히 변경해야 하므로…
- ◊ `global_variables_initializer()` 를 통해 ‘변수’를 초기화

```

1 import tensorflow as tf
2
3 # 학습데이터 (X, Y)
4 x_train = [1, 2, 3, 4]
5 y_train = [6, 5, 7, 10]

```

» 초기화된 변수 W와 b를

경사로 따라 내려가기 방법으로 cost를 최소화 하도록

W와 b를 적절한 값으로 2000번 갱신함.

(Cost값을 계산해보고 늘릴지 줄일지 결정.)

```

13
14 # cost/loss function
15 cost = tf.reduce_mean(tf.square(hypothesis - y_train))
16
17 # Minimize, 최적화 함수
18 optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
19 train = optimizer.minimize(cost)
20

```

```

21 # Launch the graph in a session.
22 sess = tf.Session()
23
24 # Initializes global variables in the graph.
25 sess.run(tf.global_variables_initializer())
26
27 # 최적값 찾기 반복
28 for step in range(2000):
29     sess.run(train)
30     print(step, sess.run(cost), sess.run(W), sess.run(b))

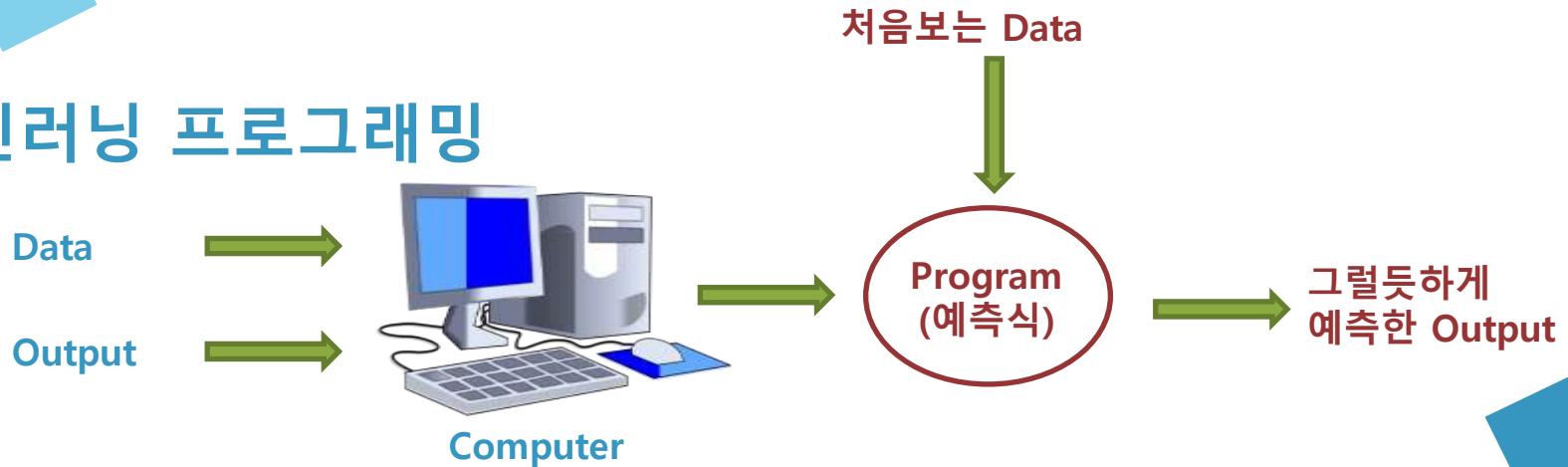
```

```

1997 1.00000040 [1.4017710] [3.4947940]
1998 1.0500042 [1.4017719] [3.4947906]
1999 1.0500042 [1.4017665] [3.494806]

```

머신러닝 프로그래밍



텐서플로우에서 사용하는 자료형

» 상수(Constant)

- ❖ 고정값.
- ❖ 예) $x_train = [1, 2, 3, 4]$

» 변수(Variable)

- ❖ 앞의 W 와 b 는 학습을 통해서 컴퓨터가 찾아내야 하는 값
- ❖ 이러한 W 와 b 를 변수(variable)라 함

» 플레이스 홀더 (Placeholder)

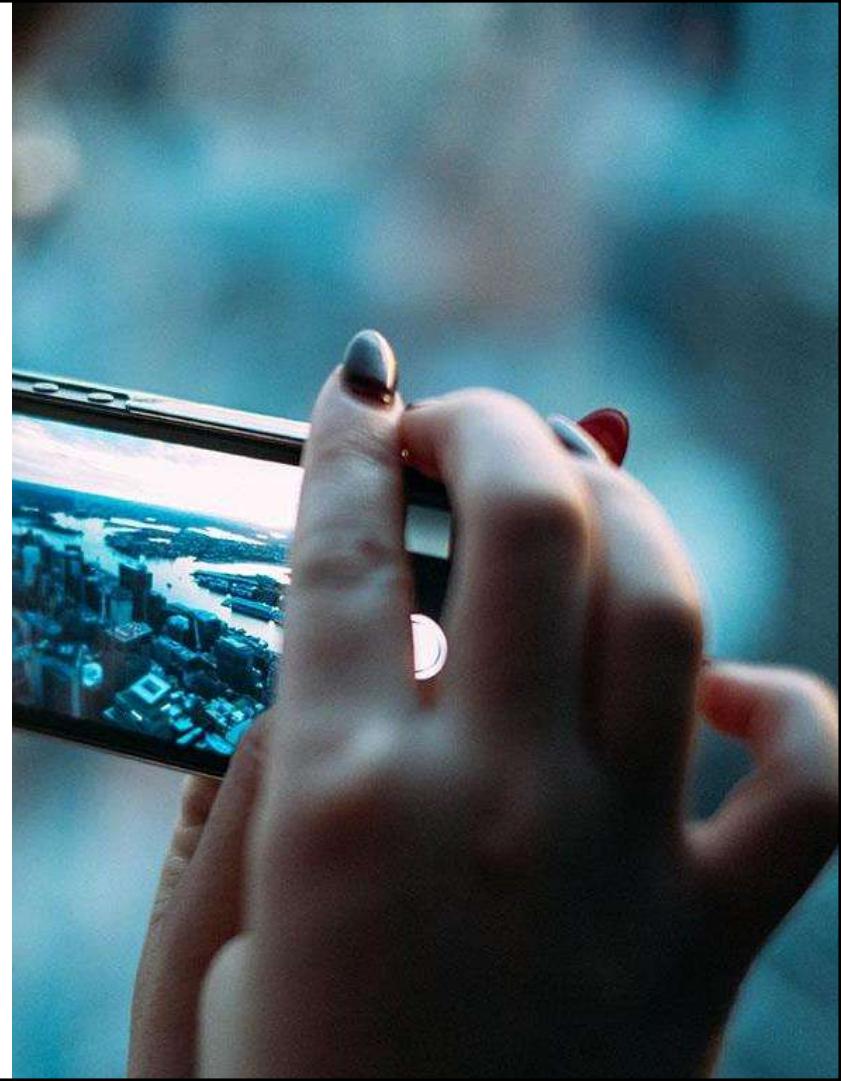
- ❖ 그래프를 실행할 때 데이터를 입력하는 방법 (실행시점에 외부에서 읽어옴)

} 앞의 실습에서 경험

MNIST 실습

Ritchie Ng. Deep Learning Theory.

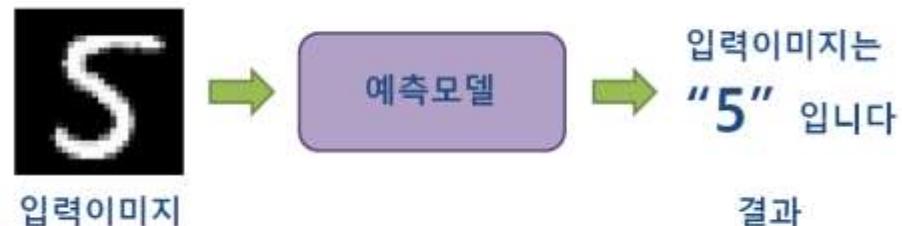
<http://www.ritchieng.com/machine-learning/deep-learning/neural-nets/>



MNIST 숫자이미지 분류기 실습

- » 목표 : 손으로 그린 숫자 이미지를 모아놓은 MNIST를 활용하여 숫자를 인식해보기
- » 1단계 : MNIST 데이터셋을 우리가 만든 예측모델을 학습시키기 위해 사용
 - ◊ 예측모델은 경사로 따라 내려가기 기법을 사용하여 입력이미지를 점점 정확하게 알아볼 수 있게 됨
- » 2단계 : 처음 보는 테스트 데이터셋을 입력 받았을 때 이것을 얼마나 잘 알아맞히는지 확인

트레이닝데이터셋
{이미지, 정답}



» 일단 실행해서 결과를 보고 공부합시다

<https://github.com/jaeyong1/deeplearningforus/>

→ Chap 04

→ 13_mnist.py

goo.gl/RNCw2j

```
2 import tensorflow as tf
3
4 #불필요안내멘트 출력제한
5 tf.logging.set_verbosity(tf.logging.ERROR)
6
7 # MNIST 데이터를 다운로드 한다.
8 from tensorflow.examples.tutorials.mnist import input_data
9 mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
10
11 # 변수들을 설정한다.
12 X = tf.placeholder(tf.float32, [None, 784]) # mnist 이미지데이터 형태는 28 * 28 = 784
13 Y = tf.placeholder(tf.float32, [None, 10]) # 0-9 숫자분류 = > 10 classes
14
15 #Logistic Classifier (Linear Classifier)
16 W = tf.Variable(tf.zeros([784, 10]))
17 b = tf.Variable(tf.zeros([10]))
18 logit_y = tf.matmul(X, W) + b
```

```
2 import tensorflow as tf
3
4 #불필요안내멘트 출력제한
5 tf.logging.set_verbosity(tf.logging.ERROR)
6
7 # MNIST 데이터를 다운로드 한다.
8 from tensorflow.examples.tutorials.mnist import input_data
9 mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
10
11 # 변수들을 설정한다.
12 X = tf.placeholder(tf.float32, [None,
13 Y = tf.placeholder(tf.float32, [None,
14
15 #Logistic Classifier (Linear Classifier)
16 W = tf.Variable(tf.zeros([784, 10]))
17 b = tf.Variable(tf.zeros([10]))
18 logit_y = tf.matmul(X, W) + b
```

» 인터넷에서 MNIST 데이터를 다운로드

- ◊ MNIST_data라는 하위폴더에 저장
- ◊ 정답레이블은 one-hot 인코딩을 사용

```
2 import tensorflow as tf
3
4 #불필요안내멘트 출력제한
5 tf.logging.set_verbosity(tf.logging.ERROR)
6
7 # MNIST 데이터를 다운로드 한다.
8 from tensorflow.examples.tutorials.mnist import input_data
9 mnist = input_data.read_data_sets("MNIST_data/")
10
11 # 변수들을 설정한다.
12 X = tf.placeholder(tf.float32, [None, 784]) # mnist 이미지데이터 형태는 28 * 28 = 784
13 Y = tf.placeholder(tf.float32, [None, 10]) # 0-9 숫자분류 = > 10 classes
14
15 #Logistic Classifier (Linear Classifier)
16 W = tf.Variable(tf.zeros([784, 10]))
17 b = tf.Variable(tf.zeros([10]))
18 logit_y = tf.matmul(X, W) + b
```

» Input Output 데이터 Placeholder로 정의

- ❖ 그래프 실행중에 입력받음
- ❖ None : 개수 미지정(동적으로 늘림)
- ❖ 784 : 이미지데이터 28 픽셀 x28 픽셀 =784
- ❖ 10 : 0 ~ 9 정답레이블

```
2 import tensorflow as tf
3
4 #불필요안내멘트 출력제한
5 tf.logging.set_verbosity(tf.logging.ERROR)
6
7 # MNIST 데이터를 다운로드 한다.
8 from tensorflow.examples.tutorials.mnist import input_data
9 mnist = input_data.read_data_sets("MNIST_data/")
10
11 # 변수들을 설정한다.
12 X = tf.placeholder(tf.float32, [None, 784])
13 Y = tf.placeholder(tf.float32, [None, 10])
14
15 #Logistic Classifier (Linear Classifier)
16 W = tf.Variable(tf.zeros([784, 10]))
17 b = tf.Variable(tf.zeros([10]))
18 logit_y = tf.matmul(X, W) + b
```

» 가설식 설정

- ◆ tf.zeros() : 입력된 배열만큼 0으로 채움
- ◆ tf.matmul() : 행렬곱셈연산
- ◆ Variable() : 그래프 실행중에 컴퓨터가 값을 바꿀 대상
- ◆ W에서 [784, 10] : 입력이 784개값, 출력이 10개값
- ◆ b의 [10] : W의 출력10개의 bias조정값

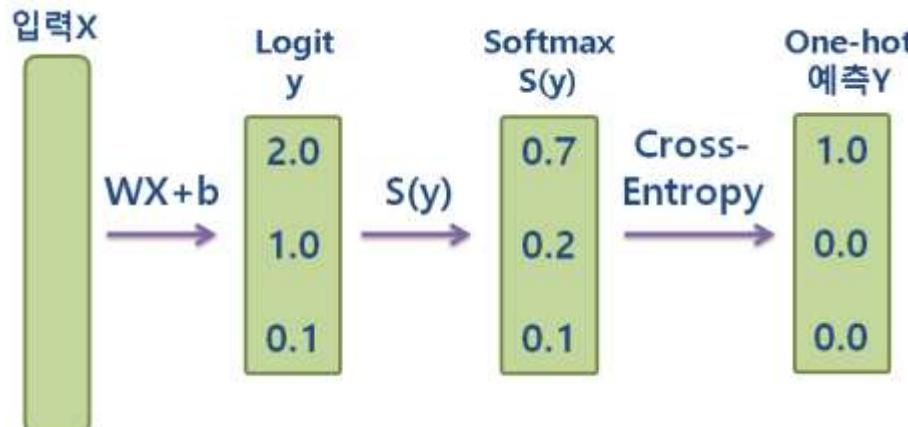
```

20 # softmax와 cross-entropy 모델을 설정한다.
21 softmax_y = tf.nn.softmax(logit_y)
22 cross_entropy = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(softmax_y), reduction_indices=[1]))
23 train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
24
25 # 경사하강법으로 초기화
26 init = tf.global_variables_initializer()
27 sess = tf.Session()
28 sess.run(init)
29 for i in range(1000):
30     batch_xs, batch_ys = mnist.train.next_batch(100)
31     sess.run(train_step, feed_dict={X: batch_xs, Y: batch_ys})
32
33 # 학습된 모델이 얼마나 정확한지 출력
34 correct_predictions = tf.equal(tf.argmax(softmax_y, 1), tf.argmax(Y, 1))
35 accuracy = tf.reduce_mean(tf.cast(correct_predictions, "float"))
36 print("정확도 : ", sess.run(accuracy, feed_dict={X: mnist.test.images, Y: mnist.test.labels}))

```

» Cost함수(비용함수) 및 예측모델 구성

- ❖ logit_y를 Softmax를 취한 후, Cross Entropy를 계산.
- ❖ cross-entropy : 예측한 Y의 확률분포와 실제 Y의 분포를 비교하여 예측이 참값을 표현하는데 얼마나 비효율적인지를 측정하는 함수



```

20 # softmax와 cross-entropy 모델을 설정한다.
21 softmax_y = tf.nn.softmax(logit_y)
22 cross_entropy = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(softmax_y), reduction_indices=[1]))
23 train_step = tf.train.GradientDescentOptimizer(0.1).minimize(cross_entropy)
24
25 # 경사하강법으로 모델을 학습한다.
26 init = tf.global_variables_initializer()
27 sess = tf.Session()
28 sess.run(init)
29 for i in range(100):
30     batch_xs, batch_ys = mnist.train.next_batch(100)
31     sess.run(train_step, feed_dict={X: batch_xs, Y: batch_ys})
32
33 # 학습된 모델이 얼마나 정확한지를 계산하고 출력한다.
34 correct_prediction = tf.equal(tf.argmax(softmax_y, 1), tf.argmax(Y, 1))
35 accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
36 print("정확도 : ", sess.run(accuracy, feed_dict={X: mnist.test.images, Y: mnist.test.labels}))

```

» 학습은 경사로 따라 내려가기 기법(Gradient Descent) 사용

- ◆ learning rate는 0.1로 설정
- ◆ cross_entropy를 감소시키는 방향으로 진행
- ◆ 학습을 train이 아닌 train_step이라고 이름 붙인것은,
데이터전체를 한번에 입력받지 않고 나눠서 받을 예정이므로



```
20 # softmax와 cross-entropy 모델을 설정한다.  
21 softmax_y = tf.nn.softmax(logit_y)  
22 cross_entropy = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(softmax_y), reduction_indices=[1]))  
23 train_step = tf.train.GradientDescentOptimizer(0.1).minimize(cross_entropy)  
24  
25 # 경사하강법으로 모델을 학습한다.  
26 init = tf.global_variables_initializer()  
27 sess = tf.Session()  
28 sess.run(init)  
29 for i in range(100)  
30     batch_xs, batch_ » Variable(변수)초기화 및 텐서플로우 세션을 오픈  
31         sess.run(train_s  
32  
33 # 학습된 모델이 얼마나 정확한지를 계산하고 출력한다.  
34 correct_prediction = tf.equal(tf.argmax(softmax_y,1), tf.argmax(Y,1))  
35 accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))  
36 print("정확도 : ", sess.run(accuracy, feed_dict={X: mnist.test.images, Y: mnist.test.labels}))
```

```

20 # softmax와 cross-entropy 모델을 설정한다.
21 softmax_y = tf.nn.softmax(logit_y)
22 cross_entropy = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(softmax_y), reduction_indices=[1]))
23 train_step = tf.
24
25 # 경사하강법으로
26 init = tf.global_variables_initializer()
27 sess = tf.Session()
28 sess.run(init)

29 for i in range(1000):
30     batch_xs, batch_ys = mnist.train.next_batch(100) #배치크기는 100
31     sess.run(train_step, feed_dict={X: batch_xs, Y: batch_ys})

32
33 # 학습된 모델이 얼마나 정확한지를 계산하고 출력한다.
34 correct_prediction = tf.equal(tf.argmax(softmax_y, 1), tf.argmax(Y, 1))
35 accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
36 print("정확도 : ", sess.run(accuracy, feed_dict={X: mnist.test.images, Y: mnist.test.labels}))

```



```

20 # softmax와 cross-entropy 모델을 설정한다.
21 softmax_y = tf.nn.softmax(logit_y)
22 cross_entropy = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(softmax_y), reduction_indices=[1]))
23 train_step = tf.train.GradientDescentOptimizer(0.1).minimize(cross_entropy)
24
25 # 경사하강법으로 모델을 학습한다.
26 init = tf.global_variables_initializer()
27 sess = tf.Session()
28 sess.run(init)
29 for i in range(1000):
30     batch_xs, batch_ys = mnist.train.next_batch(100)
31     sess.run(train_step, feed_dict={X: batch_xs, Y: batch_ys})
32
33 # 학습된 모델이 얼마나 정확한지를 계산하고 출력한다.
34 correct_prediction = tf.equal(tf.argmax(softmax_y, 1), tf.argmax(Y, 1))
35 accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
36 print("정확도 : ", sess.run(accuracy, feed_dict={X: mnist.test.images, Y: mnist.test.labels}))

```

» train 결과 확인

- ◆ correct_prediction : 예측결과(0~9)와 실제정답(0~9)가 일치여부를 리스트로 만듬
- ◆ tf.cast() : True=1. 으로, False=0. 으로 출력
- ◆ accuracy : correct_prediction 의 평균값을 취함 → 정답률



» Test Set에 있는 10개 이미지중 9개를 맞춤

(의미 : 아주 짧은 학습시간, 가장 기본 알고리즘으로 분류기 작성)



```
Extracting MNIST_data/train-images-idx3-ubyte.gz  
Extracting MNIST_data/train-labels-idx1-ubyte.gz  
Extracting MNIST_data/t10k-images-idx3-ubyte.gz  
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz  
정확도 : 0.9093
```

» 실질적인 데이터를 사용하여 연습해 보기 위한 사이트 : 캐글(Kaggle)

- ◊ 2010년 설립된 예측모델 및 분석 대회 플랫폼
- ◊ 기업 및 단체에서 과제와 함께 데이터셋을 공유하면,
- ◊ 데이터 과학자들이 이를 해결하는 모델을 개발하기 위해 경쟁
- ◊ 상금을 놓고 경쟁하지만, 공부목적으로도 충분히 좋은 데이터를 얻을 수 있음

www.kaggle.com/c/titanic



타이타닉호의 침몰사고는 역사상 가장 끔찍했던 사건입니다. 타이타닉호는 최초항해도중 **1912년 4월 15일** 빙산과 충돌한 후 침몰하여 **2224명**의 승객과 승무원 중 **1502명**이 사망했습니다. 이 끔찍한 비극으로 국제사회는 큰 충격을 받았으며, 선박안정규정을 개선하게 되었습니다.

침몰사고로 많은 생명을 잃은 이유 중 하나는 승객과 승무원을 위한 구명정이 충분하지 않았기 때문입니다. 침몰사고에서 살아남은 사람은 운도 따랐겠지만 전체적으로 여성과 어린이, 상류층 사람들의 생존율이 좀더 높았습니다.

이 과제에서 우리는 어떤 종류의 사람들이 생존할지 예측모델을 생성하고, 그 예측모델을 바탕으로 각 승객이 비극에서 살아남을 확률을 예측해봅시다.

» 실습코드

goo.gl/Wmu9T6

타이타닉 생존자 예측모델 (소개)

The screenshot shows a web browser window with the URL <https://www.kaggle.com/c/titanic>. The search bar contains the text "titanic". The main content area displays several competition entries related to the Titanic dataset:

- Titanic: Machine Learning from Disaster**
competition - Start here! Predict survival on the Titanic and ge...
- Titanic Insofe**
competition - Titanic data set to check the functionality of Ka...
- Titanic Survival**
competition - Predict who survived the titanic disaster
- Titanic@Insofe**
competition - Trial hackathon to check out features of Kaggle ...
- Titanic competition**
competition - prediction using women and children model
- Humana Titanic**
competition - Explore Titanic Problem
- Titanic HSKA - Intelligent Systems**
competition - Titanic Dataset Challange - Intelligent Systems ...
- Data Science Dojo Capstone Project: Titanic Model**
competition - Predict survival on the Titanic dataset
- Titanic Classification MIT FDP**
competition - Titanic Classification Competition during 5 days...

A red arrow points to the first competition entry: "Titanic: Machine Learning from Disaster".

타이타닉 생존자 예측모델 (소개)

- » 목표 : 각 승객이 타이타닉 침몰에서 살아남을지 여부를 예측하는 것
- » 3개의 csv파일 : 트레이닝 데이터셋, 테스트 데이터셋(문제와 정답이 구분)

ID	탑승자정보 성별/나이/…	생존여부	
1	성별/나이/…	N	
.	.	Y	
.	.	.	
891	.	.	

Train set

892	성별/나이/…	892	Y
.	.	.	N
.	.	.	.
1309	.	1309	.

Test set

타이타닉 생존자 예측모델 (소개)

Titanic 데이터 일부

	A	B	C	D	E	F	G	H	I	J	K	L
1	Passenger	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	1	0	3	Braund, M	male	22	1	0	A/5 21171	7.25		S
3	2	1	1	Cumings, female		38	1	0	PC 17599	71.2833	C85	C
4	3	1	3	Heikkinen	female	26	0	0	STON/O2.	7.925		S
5	4	1	1	Futrelle, M	female	35	1	0	113803	53.1	C123	S
6	5	0	3	Allen, Mr.	male	35	0	0	373450	8.05		S
7	6	0	3	Moran, M	male		0	0	330877	8.4583		Q
8	7	0	1	McCarthy, male		54	0	0	17463	51.8625	E46	S

변수	정의	키값
survival	생존유무	0 = No, 1 = Yes
pclass	Ticket class(사회경제적 지위)	1 = 1st, 2 = 2nd, 3 = 3rd
sex	성별	
Age	나이	
sibsp	타이타닉에 탑승한 형제/배우자 수	
parch	타이타닉에 탑승한 부모/자녀 수	
ticket	Ticket number	
fare	운임요금	
cabin	승무원 번호	
embarked	승선항	C = Cherbourg, Q = Queenstown, S = Southampton

5.

뉴럴 네트워크

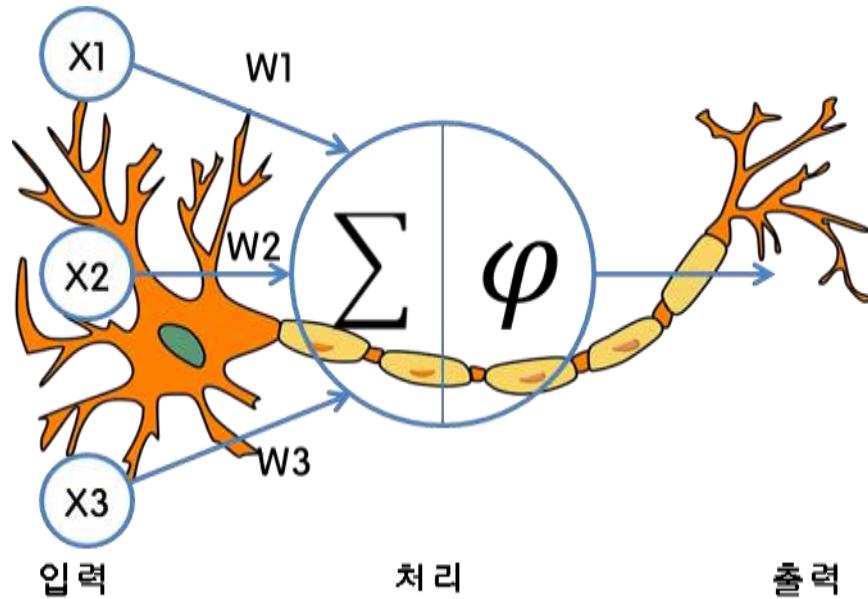
신경망 소개

Activation Function

역전파 알고리즘(Backpropagation)

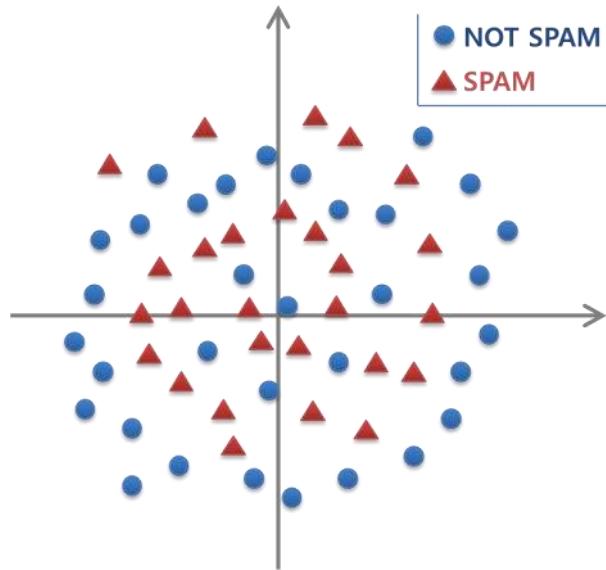
Drop Out

Fully Connected Network



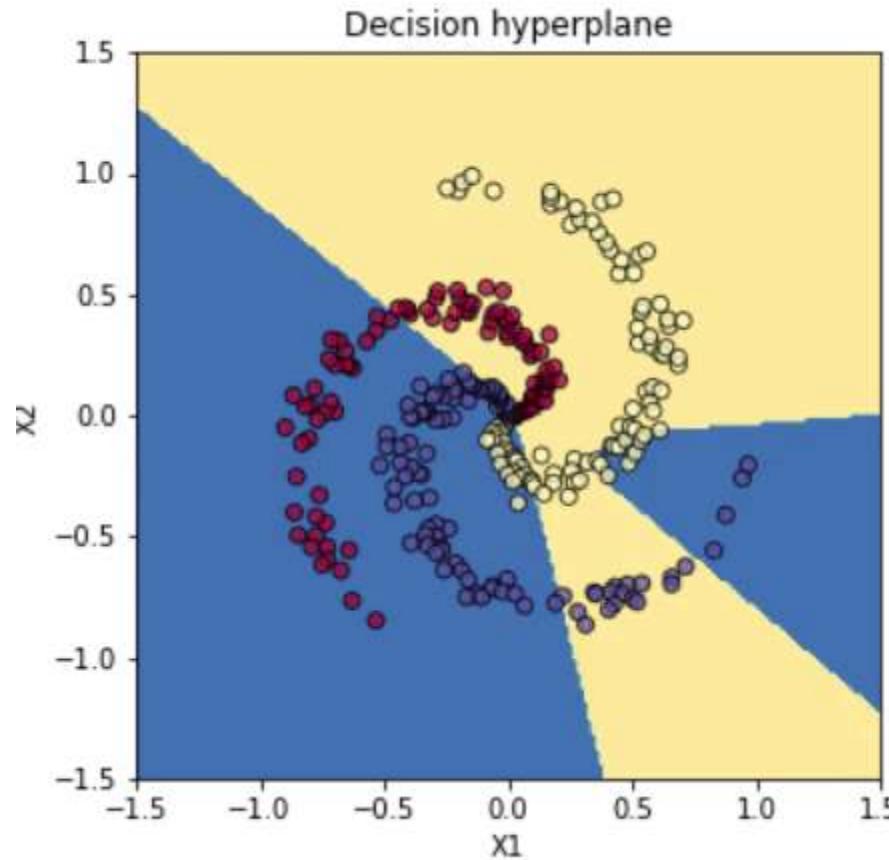
» 생명체의 뉴런과 유사하게 인공지능에서 뉴런은 입력, 연산, 출력을 수행하는 간단한 뉴런을 모델링

» 스팸메일 분류기에서 직선을 그어서 분류할 수 없는 이러한 분포를 가지는 것이라면 어떻게 할까요?



- » 그림과 같이 서로 상호작용하는 나선형 관계가 있을 수도 있음.
- » ‘비선형’ 의미 : $y = w_1x_1 + w_2w_2 + b$ 같은 식으로 예측식을 세울 수 없음

» 신경망을 이용하면
이런문제도



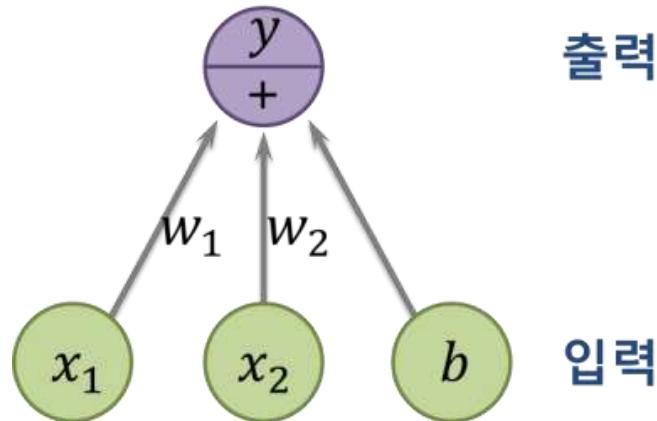
- » 우리는 더 이상 식의 구조가 어떻게 생겼을지 일일이 지정해 주기 어려움
- » 일일이 알려주지 않아도 스스로 비선형적인 식의 특성도 학습할 수 있게 하고 싶음

→ **심층신경망**

- » 이미지, 오디오, 동영상 등 복잡한 데이터 처리에 우수한 성능을 낸다고 알려져 있음

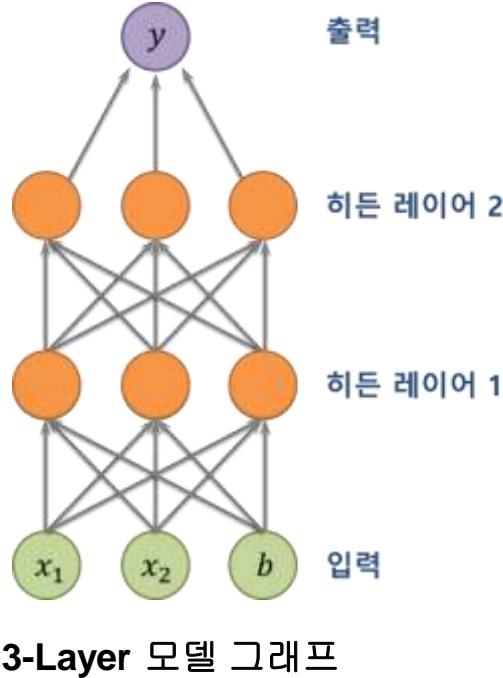
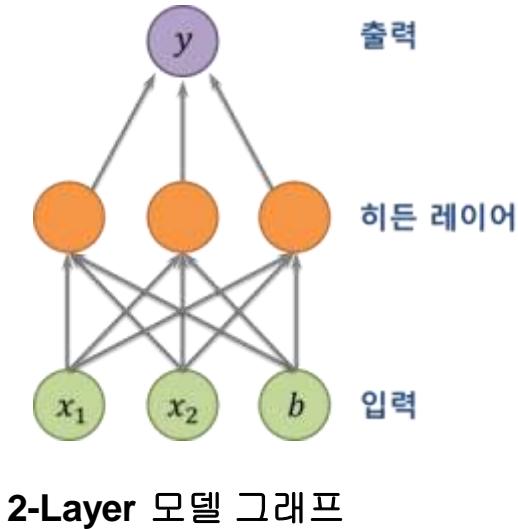
선형그래프 도식화

- » 선형그래프인 $y = w_1x_1 + w_2x_2 + b$ 그래프를 도식화



- » 화살표는 가중치이며 입력 값과 곱셈 연산을 수행
- » 그렇다면, 비선형 그래프는 어떻게?

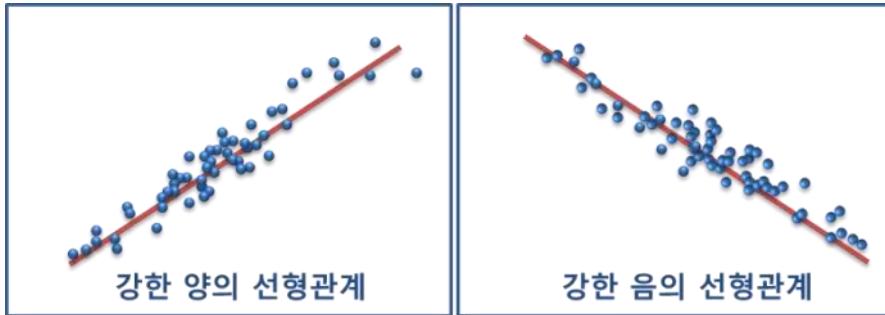
히든레이어를 추가하는 경우



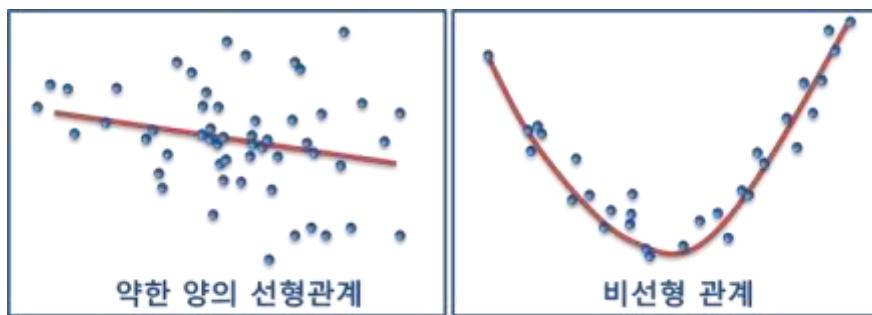
» 선형함수를 여러 개 쌓는다 하더라도 이는 하나의 히든레이어로 요약될 수 있음.

→ 비선형 표현불가

[참고] 선형? 비선형?

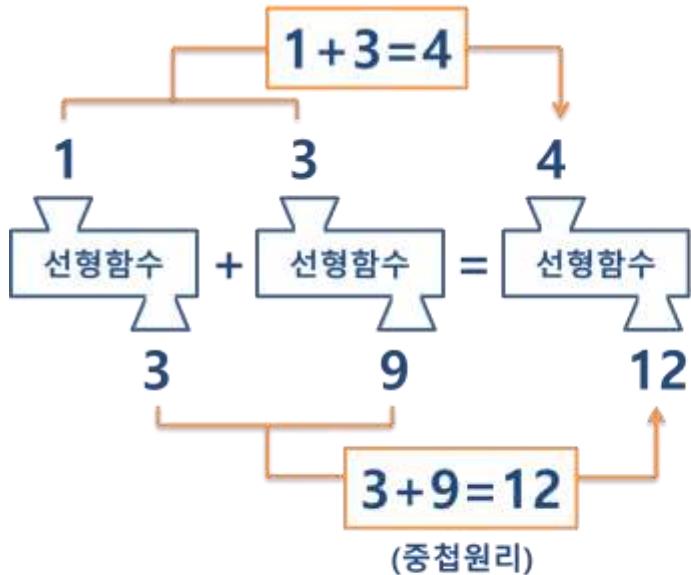


강한 선형관계 그래프

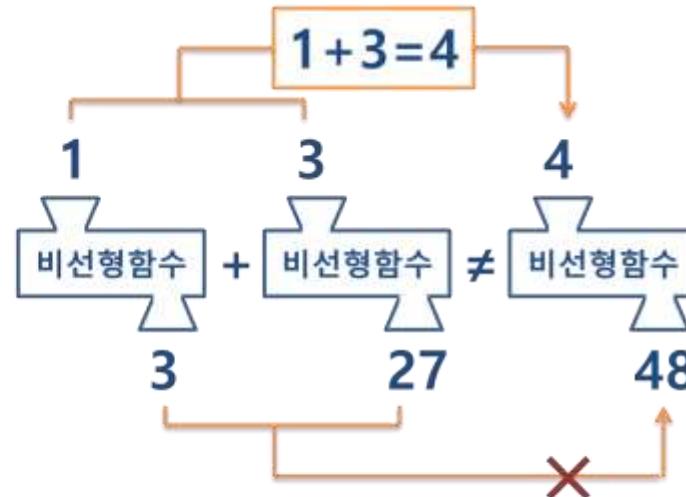


약한 선형관계와 비선형관계 그래프

[참고] 선형함수와 비선형함수 중첩계산 예

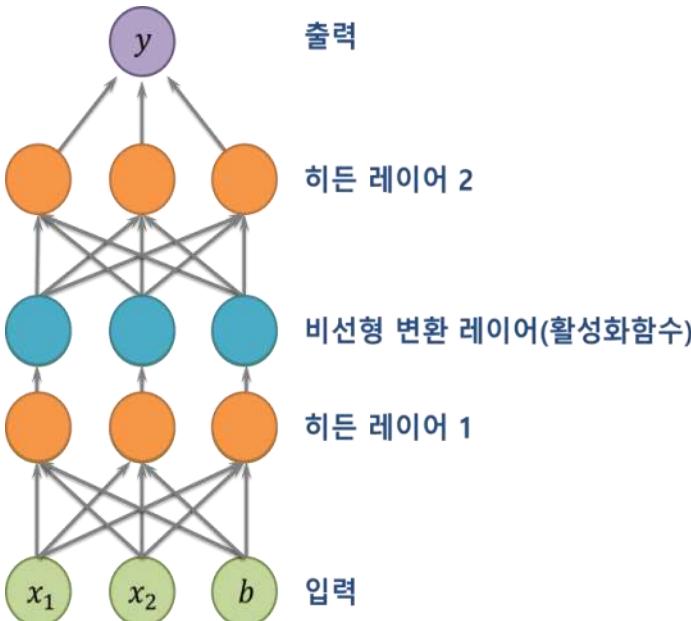


$$y = 3x$$

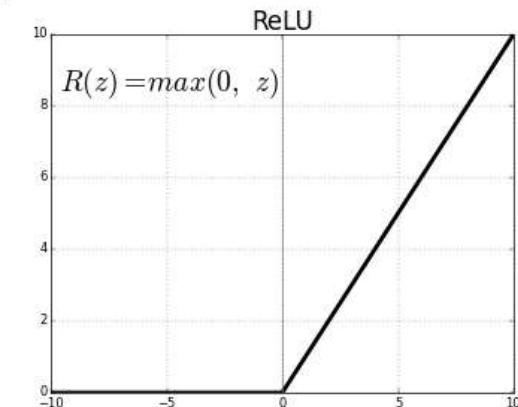
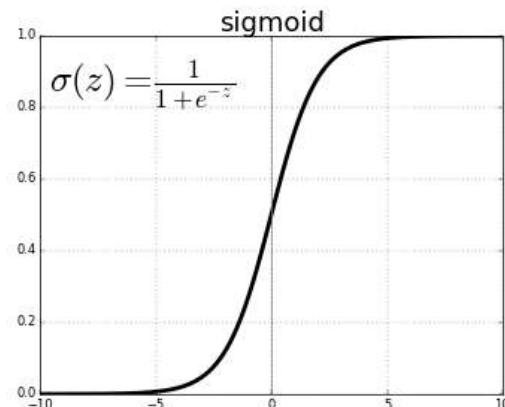


$$y = 3x^2$$

Activation Function

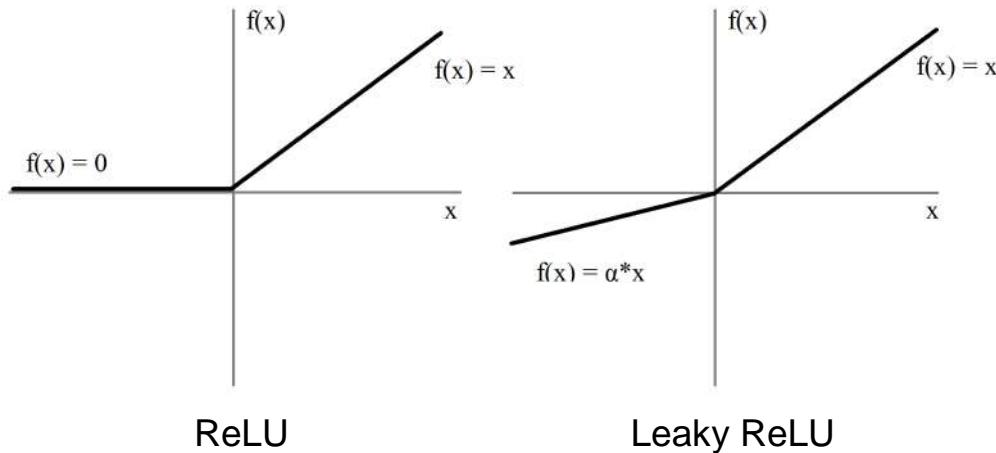


» 활성화 함수라고 부르는 비선형함수 레이어를
히든레이어에 추가하면 비선형 문제를 모델링 할 수
있음!!



» 텐서플로우는 다양한 활성화 함수를 제공

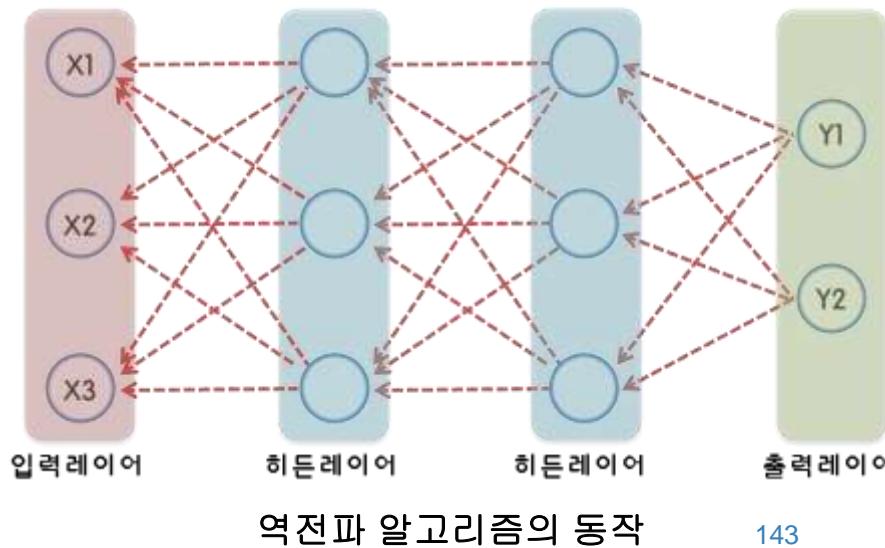
Activation Function



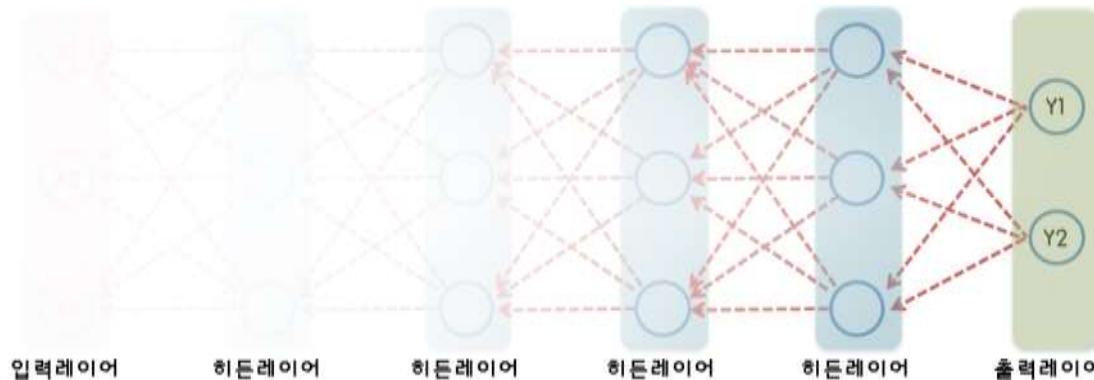
- » ReLU 입력 0이하 : 0을 리턴
- » Leaky ReLU 입력 0이하: 약간의 음의 값 리턴

역전파 알고리즘(Backpropagation)

- » 신경망을 사용한 그래프를 구성했다면 이제 학습을 진행
- » 입력 값과 출력 값의 조합을 알려줌 → 신경망 내부에 있는 가중치를 반복 업데이트
- » 가중치를 갱신하는 방향은 신경망의 처리 방향과는 반대
 - ◆ 정답--출력값 오차 존재 → 오차에 비례하여 출력레이어에서 입력레이어 방향으로 가중치를 바꿔나감



그레디언트 소실 문제(Vanishing Gradient Problem)



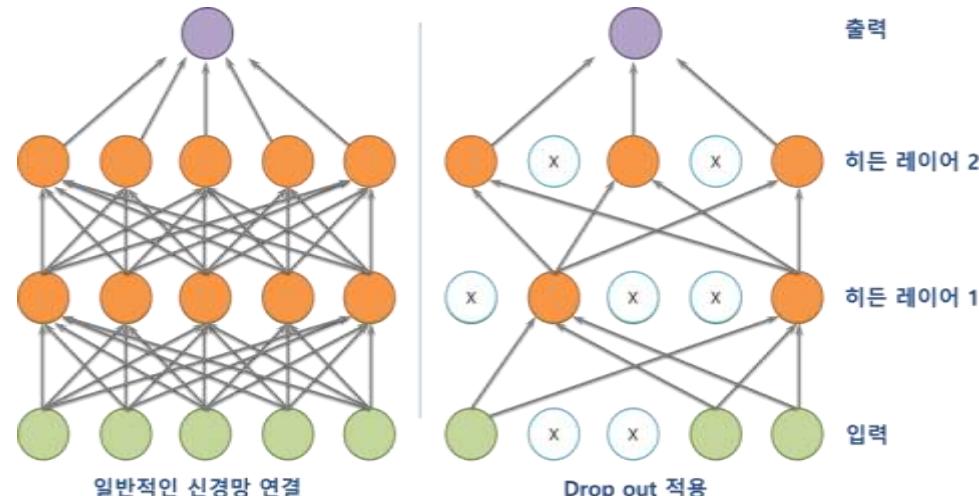
그레디언트 소실 문제(Gradient Vanishing Problem)를 형상화

» 출력레이어에서 먼 히든레이어의 가중치들은 학습이 잘 진행되지 못함

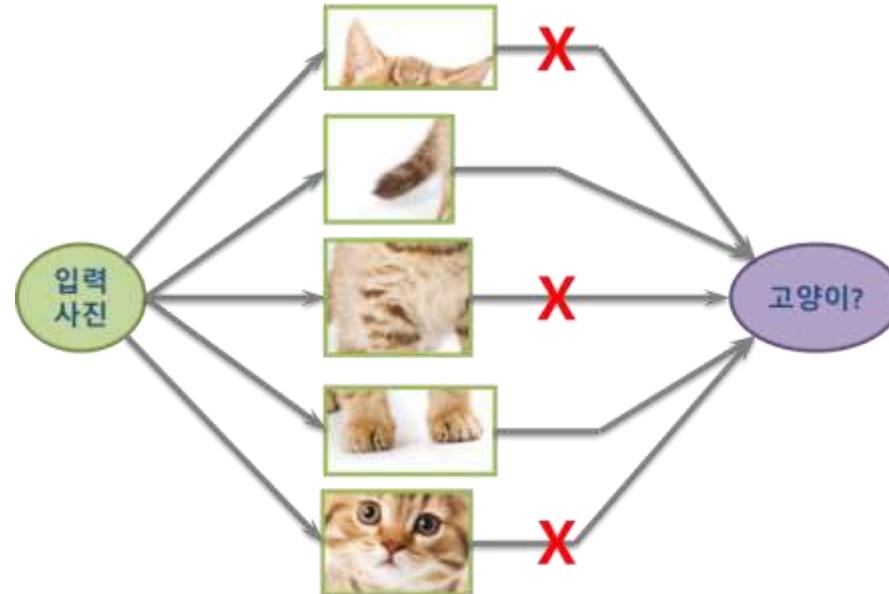
◊ Hinton교수가 2006년에 잘못된 Non-Linearity 방법(sigmoid) 때문에 발생한다는 것을 찾아냄

» 대표적인 해결법은 노드들의 활성화 함수를 ReLU 함수로 교체

- » 신경망의 크기가 불필요하게 커질 경우
 - ◊ overfitting에 빠질 가능성이 높아짐
 - ◊ 학습시간도 오래 걸리게 됨
- » overfitting을 피할 수 있는 방법 중 하나가 drop out 기법



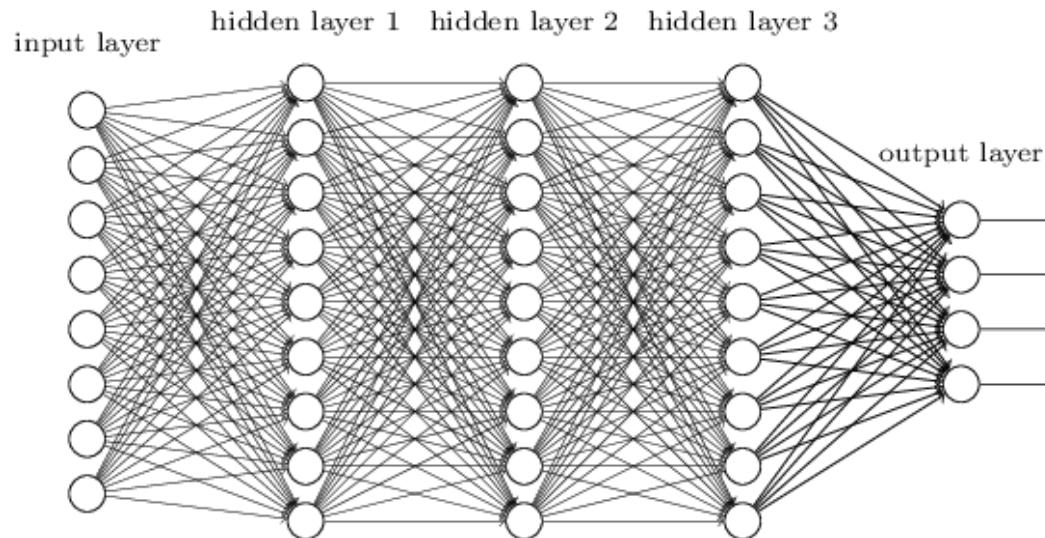
- » Drop out 동작이해를 돋기 위한 상징적인 예시
- » 고양이의 일부 특징들을 배제하고도 고양이 인지 판별할 수 있도록 학습을 진행



Drop out을 적용한 고양이 판별기의 학습과정

Fully Connected Network

- » 입력레이어로부터 출력레이어까지 이르는 모든 히든레이어의 노드들은 이전 노드와 Fully 연결이 되어 있음
- » 같은 레어어의 뉴런들 사이에는 연결이 존재하지 않고 서로 독립적인 상태
- » 모든 히든레이어는 반드시 같은 수의 노드를 가져야 한다는 뜻은 아님



Fully connected Neural Network

6.

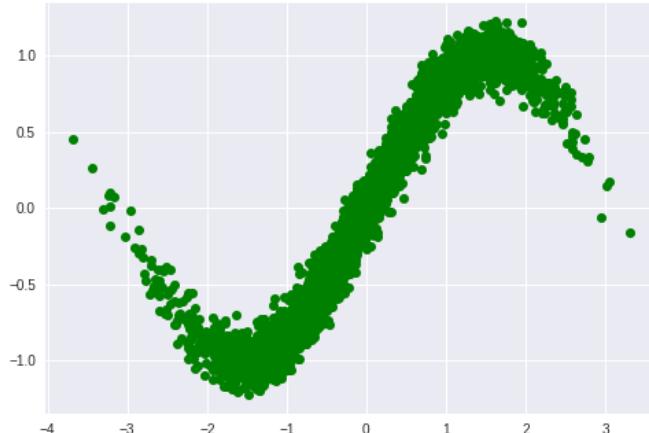
뉴럴네트워크의 구현 실습

Sin 그래프 예측모델

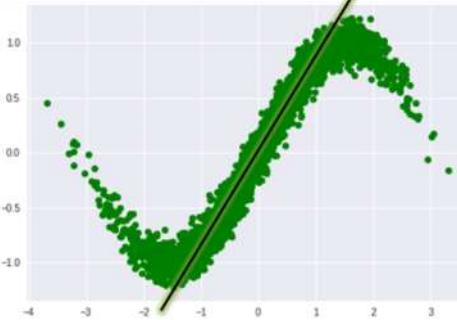
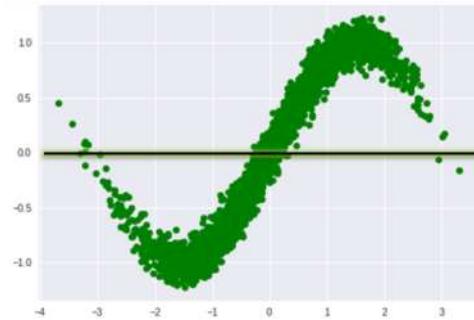
비행기 이륙거리 예측모델

» 기존실습

- ◊ $Y = WX + B$ 와 같은 가설식의 형태를 알려줌
- ◊ 수많은 X (입력)와 Y (출력)를 제시
- ◊ 경사로 따라 내려가기 방법으로 W 와 B 를 찾음
- ◊ 새로운 입력 값 → 예측한 값과 정답과 비교



Sin함수에 Random noise를 추가한 데이터

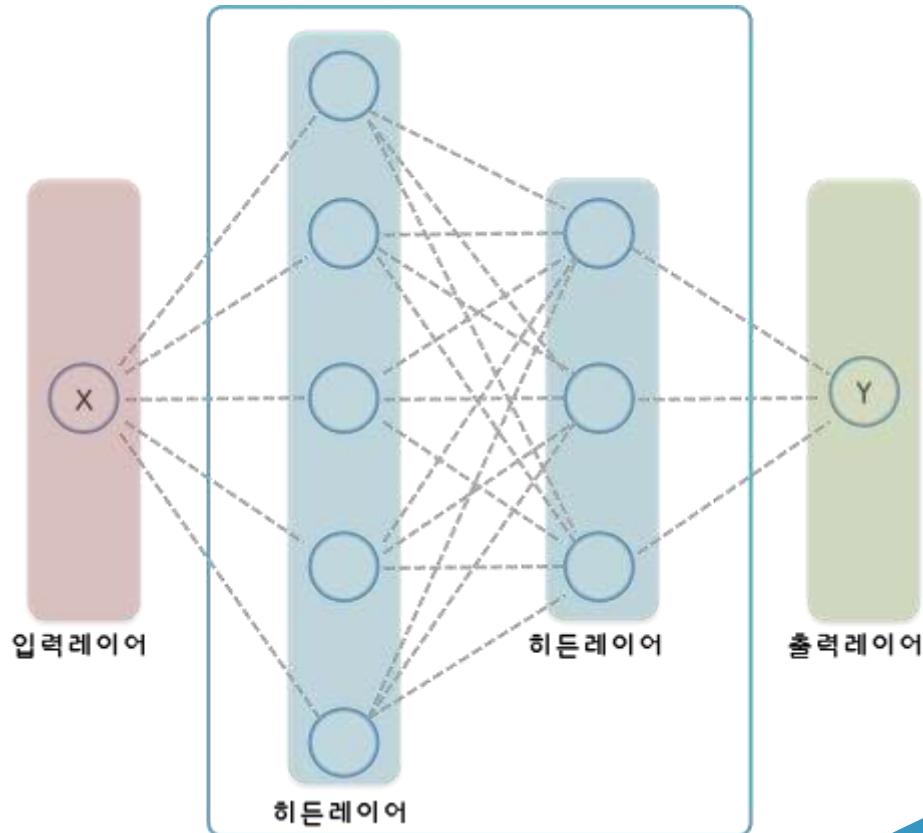


Sin함수를 선형식으로 예측한 경우 예측모델

» 입력노드(1개)

- 5개의 뉴런으로 된 히든레이어를 연결
- 3개의 뉴런으로 된 히든레이어를 연결
- 1개 노드로 결과출력

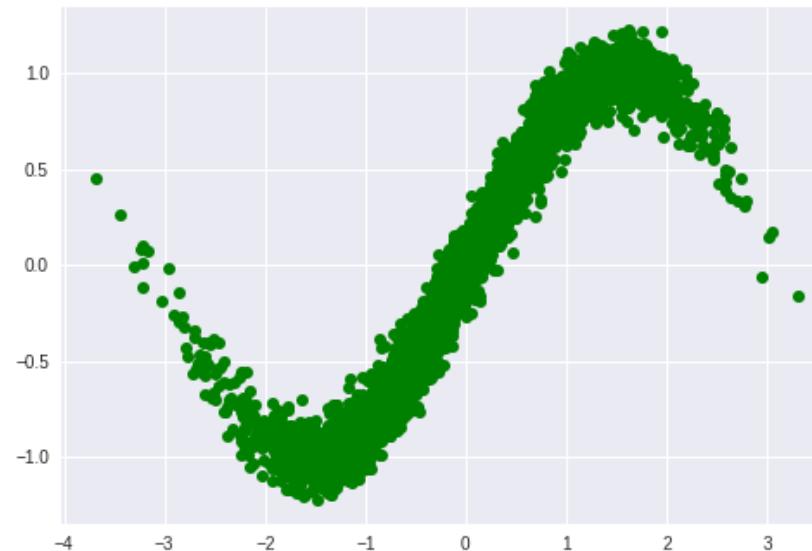
goo.gl/7vBBUu



학습을 위한 뉴럴네트워크 정의

» 만든 데이터셋 확인

```
#green색(g)에 둥근점(o)로 시각화  
plt.plot(x_data, y_data, 'go')  
plt.legend()  
plt.show()
```



» 배치실행을 적용

- » train set 5000개 전체를 고려하여 cost를 계산한 후
뉴럴네트워크 내부의 값(가중치)을 업데이트 하는 것이 아니라,
- » 100개 단위로 쪼개어서 cost를 계산한 후 뉴럴네트워크의 가중치를 업데이트

#배치 수행단위

BATCH_SIZE = 100

BATCH_NUM = int(len(x_data)/BATCH_SIZE)

#데이터를 세로로(한개씩)나열한 형태로 reshape

x_data = np.reshape(x_data, [len(x_data), 1])

y_data = np.reshape(y_data, [len(y_data), 1])

#총 갯수는 정해지지않았고 1개씩 들어가는 Placeholder 생성

input_data = tf.placeholder(tf.float32, shape=[None, 1])

output_data = tf.placeholder(tf.float32, shape=[None, 1])

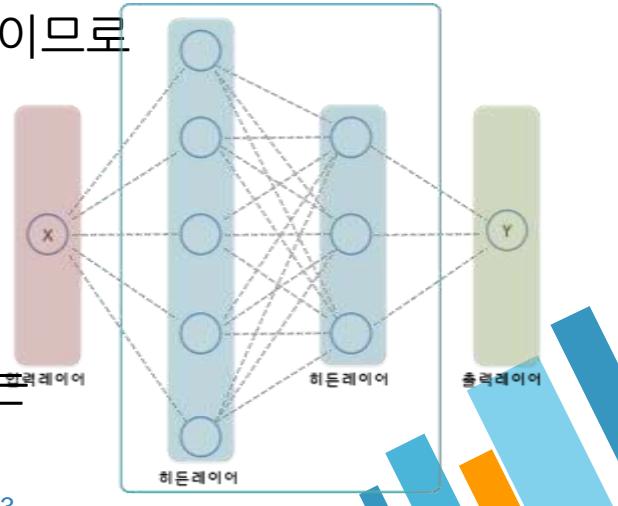
#레이어간 Weight 정의후 랜덤값으로 초기화. 그럼에서는 선으로 표시.

```
W1 = tf.Variable(tf.random_uniform([1,5], -1.0, 1.0))
W2 = tf.Variable(tf.random_uniform([5,3], -1.0, 1.0))
W_out = tf.Variable(tf.random_uniform([3,1], -1.0, 1.0))
```

#레이어의 노드가 하는 계산. 이전노드와 현재노드의 곱셈. 비선형함수로 sigmoid 추가.

```
hidden1 = tf.nn.sigmoid(tf.matmul(input_data,W1))
hidden2 = tf.nn.sigmoid(tf.matmul(hidden1,W2))
output = tf.matmul(hidden2, W_out)
```

- » 입력레이어에는 노드가 1개, 히든레이어1에는 노드가 5개이므로
첫 번째 Weight(W1)는 [1,5] 형태
- » 히든레이어2에는 노드가 3개이므로
두 번째 Weight(W2)는 [5,3] 형태
- » 출력레이어에는 노드가 1개이므로 히든레이어2와 연결하는
W_out은 [3,1] 형태



» 이륙조건에 따라 필요한 이륙거리를 예측하는 모델을 만들어 보기

◆ 목표 : 누적된 정보를 학습하여 임의의 속도와 무게를 입력했을 때 필요한 활주거리를 예측



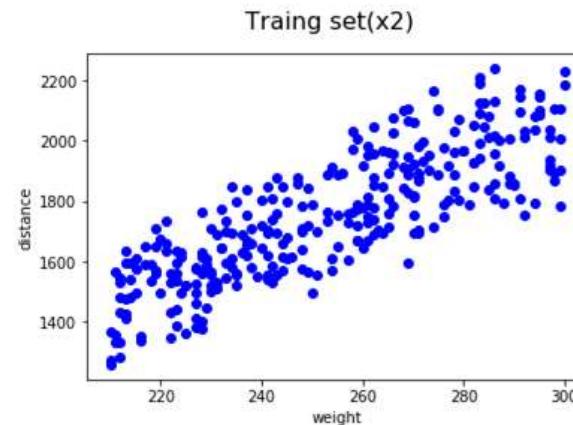
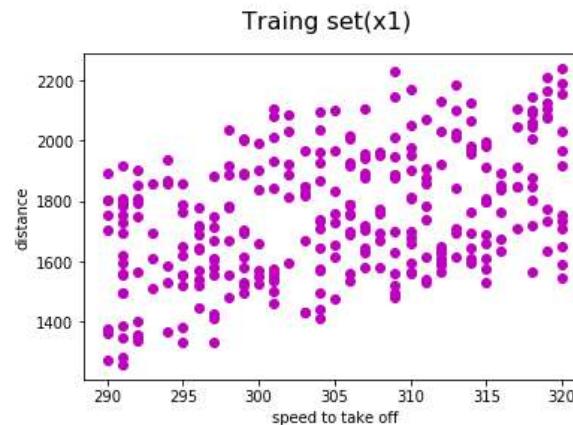
	A	B	C
1	#Takeoff_Speed(km/h)	Weight(ton)	Distance(m)
2	310	261	1814
3	312	221	1632
4	314	261	1981
5	319	274	2163
6	319	221	1733
7	310	228	1567
8	310	228	1567

비행기이륙거리 실습 Training Set

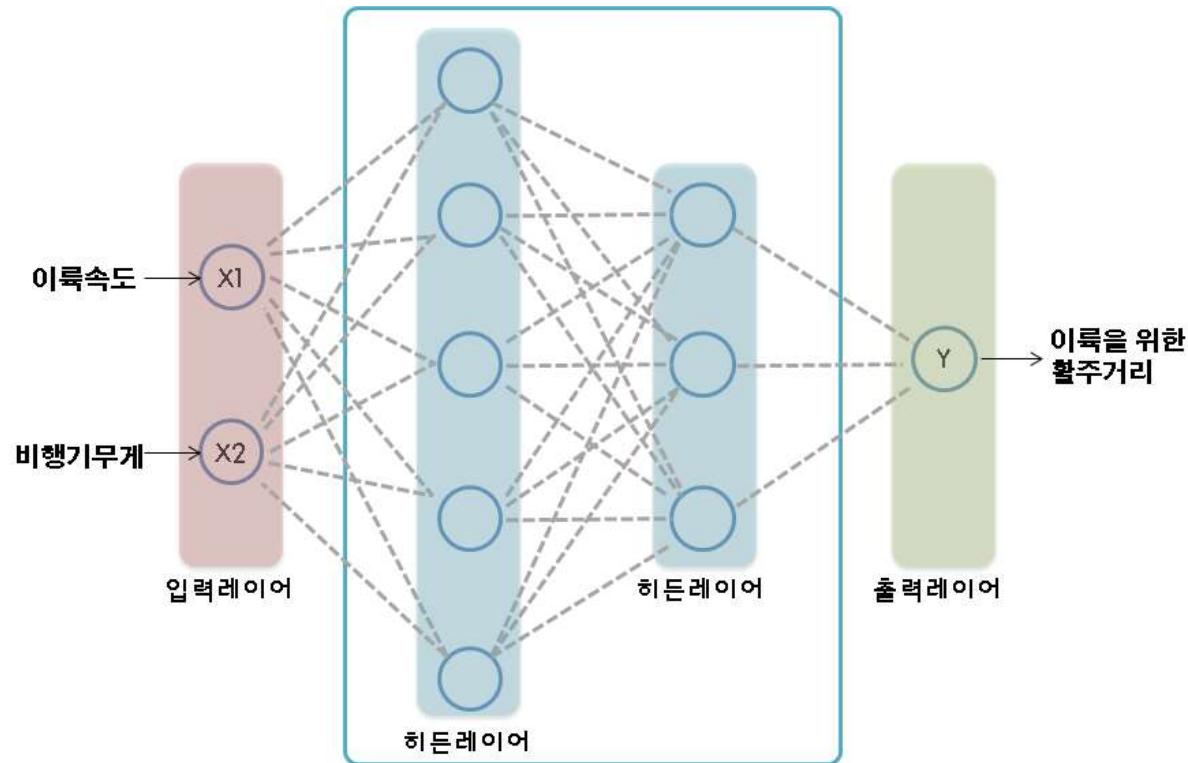
	A	B	C
1	#Takeoff_Speed(km/h)	Weight(ton)	Distance(m)
2	319	298	2257
3	316	298	2215
4	317	293	2192
5	314	295	2165
6	318	286	2153
7	309	300	2132
8	318	283	2120

비행기이륙거리 실습 Test Set

- » 두 변수(속도, 무게) 각각은 이륙거리와 선형적인 관계를 가짐
- » 무게가 조금 더 강한 선형관계



속도와 이륙거리, 무게와 이륙거리 시각화

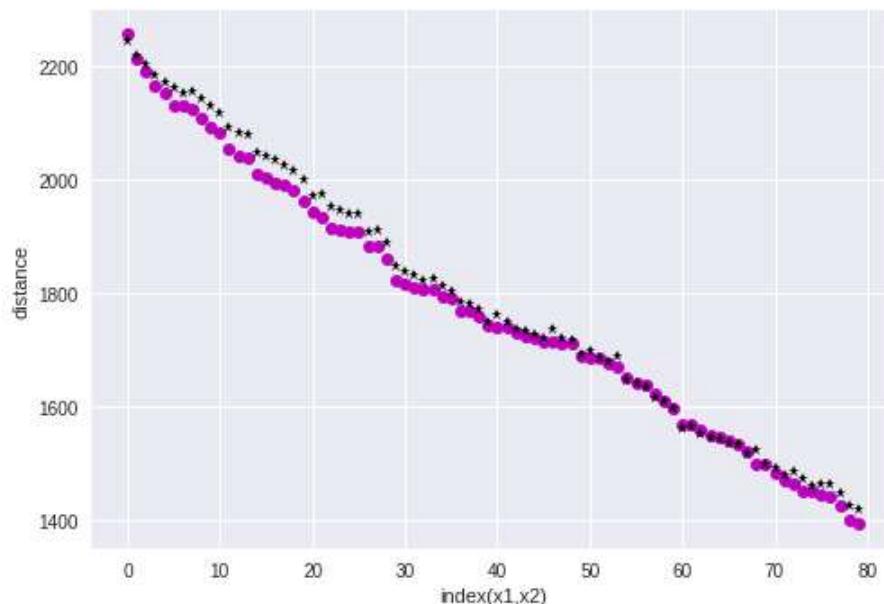


이륙거리 예측모델 뉴럴네트워크 구성

- » 컴퓨터는 각 입력변수와 결과간에 각각 선형적인 관계임을 알지못함
→ 간단히, 무게와 이륙거리, 속도와 이륙거리간의 관계를 따로따로 확인 어려움
- » 테스트 셋의 입력값(속도, 무게)은 랜덤값이지만 그 결과는 우하향 그래프가 그려지게 준비함

Test Set 검증결과 그래프

Test Result



7.

Convolutional Neural Network

CNN에 대한 소개

Stride

Zero padding과 출력 이미지의 크기 계산

LeNet과 Alex Net

- » Convolution Neural Network,

CNN은 컴퓨터 비전 분야에서 가장 영향력 있는 신경망 형태

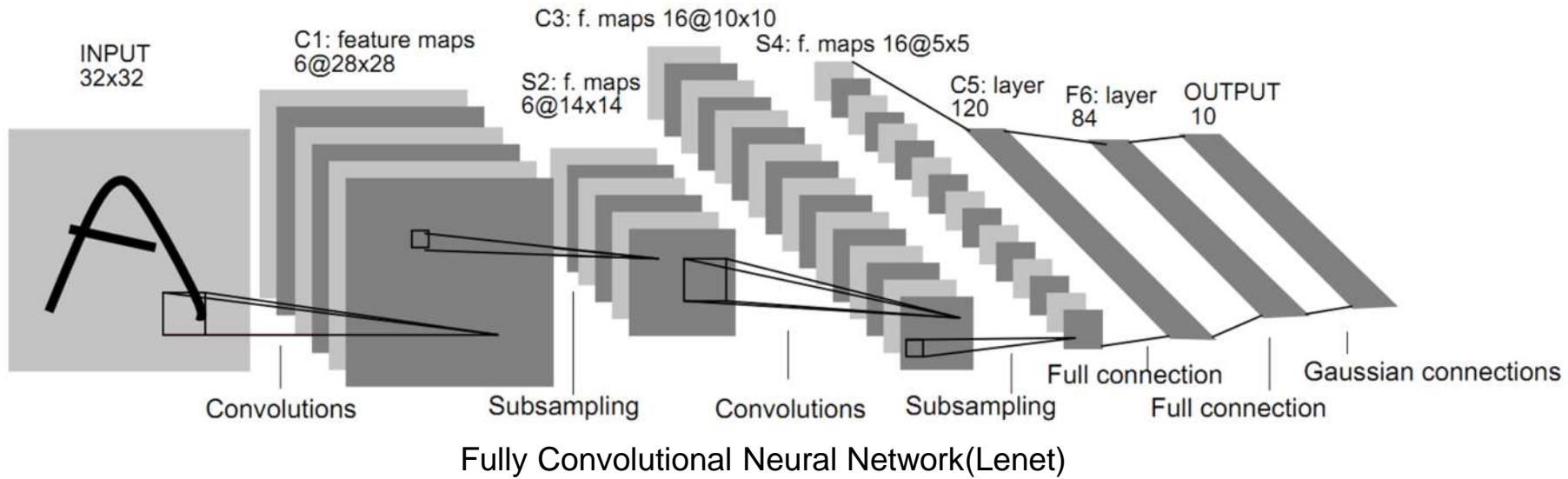
- » ImageNet 대회

- ◊ 2005년부터 진행
 - ◊ 1000개의 항목 중에서 주어진 사진이 무엇인지를 맞추는 대회

- » ImageNet 대회에서..

- ◊ 2012년 Alex Krizhevsky가 CNN을 사용하여 에러율을 기존 26%에서 15%로 떨어뜨리며 획기적인 개선
 - ◊ 인간이 사진을 분류했을 때 발생하는 에러율 3%보다 기계분류를 통해 더 낮은 에러율을 보임
 - ◊ 이제 이미지 분류는 머신러닝이 정복했다고 보임

CNN에 대한 소개



- » Convolution + Subsampling + Convolution + Subsampling 이 반복되다가 마지막에 Full Connected Layer로 이루어진 네트워크
- » (이 구조에 대해 알아봅시다..)



©Warren Photographic

인간은..

- » 강아지의 눈,코,귀,꼬리와 고양이의 눈,코,귀,꼬리 모양을 알고 있음
- » 이미지 전체를 보는 것이 아니라 눈,코,귀,꼬리 같은 특정 부분을 보고 이미지를 구분
- ➔ 일부분만을 찍은 사진으로도 이 둘을 구분하는데 전혀 어려움이 없음

인간이 보는 고양이 사진과 컴퓨터가 보는 고양이 사진



```
DC 5B 7B 4F 1B D9 15 FF BF 52 BF C3 C8  
E5 61 63 20 B1 02 51 C8 6E AA AA E9 6E  
C0 C4 1E 82 B5 7E C9 9E B0 89 AA 4A 0E  
78 17 B3 B1 C1 9B D8 D4 69 08 31 29 DB  
41 85 ED F7 99 7B E7 3B F4 77 EE CC 38  
0E 0A 2A 42 E0 C7 CC 99 F3 F8 9D 73 CF  
4B 97 EF 46 23 CA 82 96 4C 85 E3 B1 29  
D4 A3 68 B1 60 3C 14 8E DD 9E F2 7C FB  
0B 1E 25 A5 AB B1 90 1A 89 C7 B4 29 CF  
B9 3C FD FB DF 5D 4A 04 52 91 90 82 BB  
3A E5 99 D7 F5 44 60 64 24 15 9C D7 A2  
9E D0 62 F8 6E 2E 9E 8C AA 3A DE 26 6F  
EA F7 A0 1A 8D 8C F8 46 47 27 46 A2 6A  
EF 4F 76 73 7F 7C 6E 2E 1C D4 BE 88 07  
98 6E 11 49 6A 11 55 07 E7 A9 F9 70 22  
74 43 2D 91 D4 52 20 23 EE 3E C2 D2 34  
46 42 F4 3F 95 F8 26 A9 69 F4 2A B6 F0  
36 71 23 29 BE FE 6A E1 46 52 09 87 A0
```

- » 컴퓨터에게 고양이 사진은 수많은 데이터의 나열
- » Red, Green, Blue 세 가지 색의 데이터로 나눌 수 있음. 한 개의 픽셀정보로 분해됨.



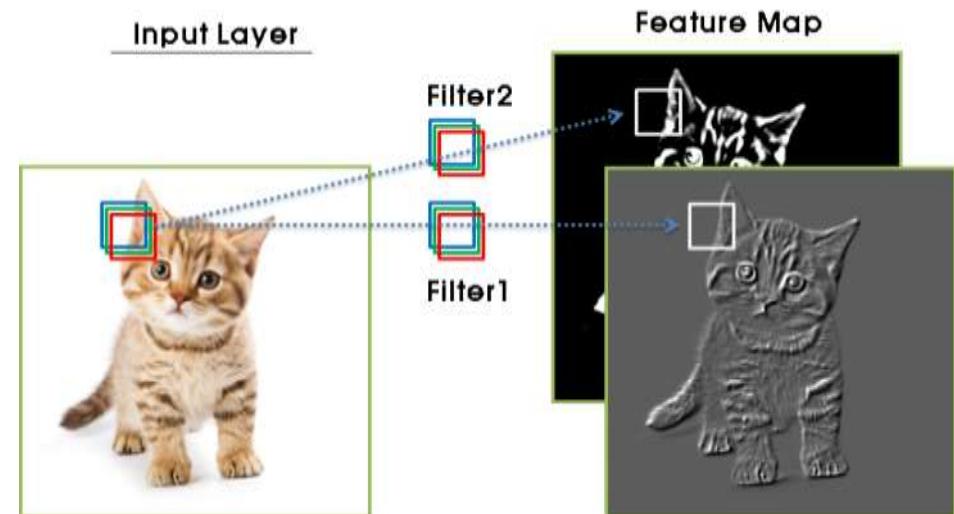
sli.do # 완전한 Neural Network로 이미지를 학습하려면?

- » 작은 $32 \times 32 \times 3$ (가로세로 32픽셀, 3개 색상채널) 크기의 이미지 학습을 위해
 - ◊ 첫 번째 히든레이어에는 $32 \times 32 \times 3 = 3072$ 개의 Weight가 필요함
 - ◊ 레이어를 쌓을수록 가중치 개수는 레이어 수에 비례하여 증가함
- » 가로세로 300픽셀 정도되는 중간사이즈의 이미지를 학습하려면
 - ◊ 첫 번째 레이어에만 $300 \times 300 \times 3 = 270,000$ 개의 Weight를 학습필요
 - ◊ 히든레이어가 더 추가된다면,
이 많은 변수들이 제대로 학습되기 힘들뿐더러 오버피팅(overfitting) 문제도 발생
- » [결론] 일반신경망은 이미지를 다루는 데에는 좋지 않음

Feature Extraction

- » Feature Extraction : 이미지의 특징을 잘 뽑아내는 과정
 - Convolution 와 Subsampling 과정을 반복해서 수행
- » 각각에 대해 알아봅니다.

- » 일정한 필터를 거쳐서 Featuremap이라는 것을 생성
- » 이 필터를 ‘커널’이라고 부름
- » 필터에 따라서 Featuremap이 다양한 특징을 감지해 낼 수 있게 됨
- » 학습하는 과정 중에 필터는 점점 더 고양이를 잘 구분할 수 있도록 적합한 필터값으로 업데이트 됨



고양이사진에서 필터를 적용하여 **Feature map**생성

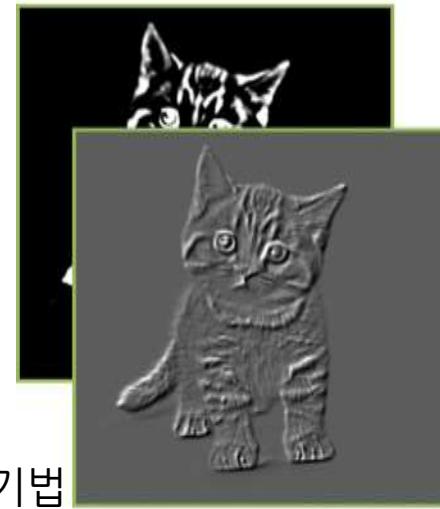
Subsampling 과정

- » Subsampling은 이미지의 크기를 줄이는 것
- » 단순히 Featuremap의 크기만 줄이는것이 아니라 특징은 두드러지게 해주도록 함
- » MAX POOLING기법이 많이 사용됨
- » Average Pooling, Min Pooling등 다양한 Pooling 기법

3	1	0	2
1	2	1	4
1	4	6	2
0	6	8	3



Max Pooling 기법 예시



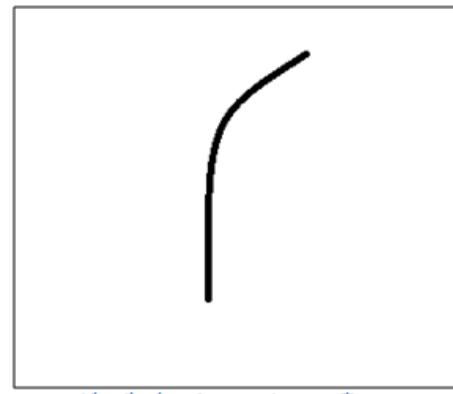
Subsampling

Featuremap의 Subsampling(Pooling)

- » CNN이 잘 되는 이유는 국소적으로 비슷한 이미지를 잘 구분하기 때문
- » 필터(커널)을 조금씩 옆으로 이동하면서 특징을 찾기 때문에 국소이미지 위치 중요치 않음
- » 왼쪽과 같은 필터값을 가진다면 오른쪽 그림과 같이 세로방향에 약간의 커브를 검출가능

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

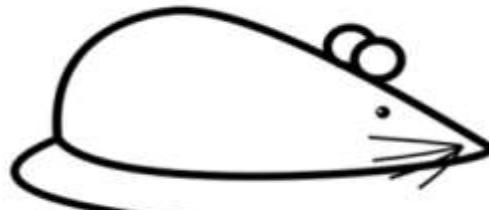


Visualization of a curve detector filter

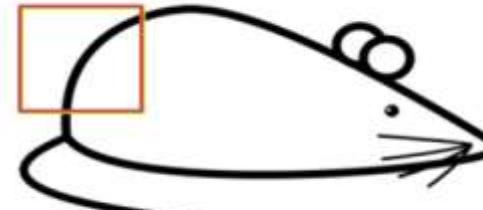
곡선검출기 필터 예

필터 동작 예

» 전체 이미지에 작은 필터를 계속해서 옆으로 옮기다가 등 부분에서 필터와 아주 유사함을 검출 가능



Original image



Visualization of the filter on the image

» 계산적으로도



Visualization of the receptive field

0	0	0	0	0	0	0	30
0	0	0	0	50	50	50	
0	0	0	20	50	0	0	
0	0	0	50	50	0	0	
0	0	0	50	50	0	0	
0	0	0	50	50	0	0	
0	0	0	50	50	0	0	

Pixel representation of the receptive field

0	0	0	0	0	0	30	0
0	0	0	0	0	30	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	0	0	0	0	0

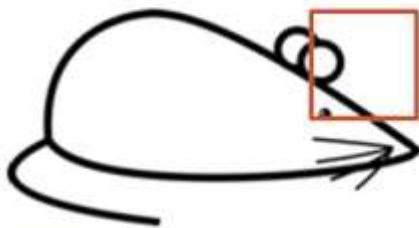
Pixel representation of filter

*

$$\text{곱셈과 덧셈 연산} = (50 \times 30) + (50 \times 30) + (50 \times 30)(20 \times 30) + (50 \times 30) = 6600 \text{ (아주 큰 숫자! 검출!)}$$

필터 동작 예

» 필터로 검출되지 않은 예



Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

*

0	0	0	0	0	30	0	0
0	0	0	0	30	0	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	0	0	0	0	0

Pixel representation of filter

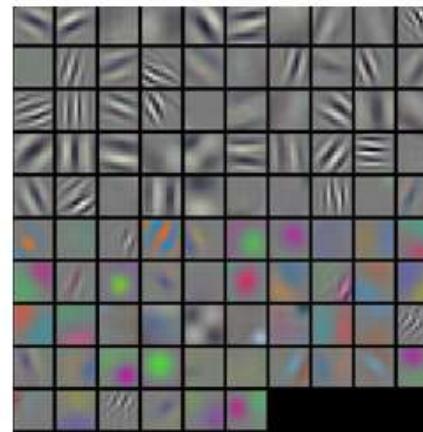
곱셈과 덧셈 연산 = 0

» 주어진 필터로 특징이 검출되지 않는 부분이라면 값이 아주 작은 값이 출력됨

» Convolution의 장점

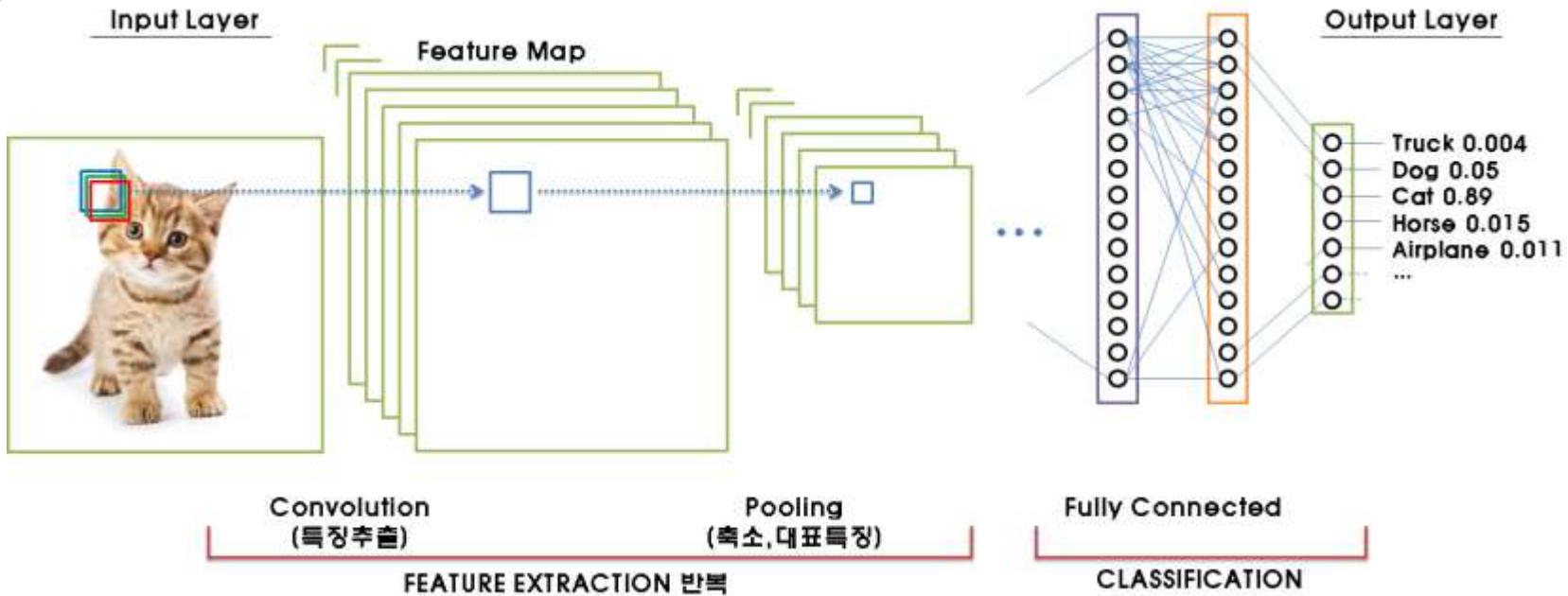
- 1) 필터로 검출하고자 하는 특징이 전체 이미지에서 어떠한 위치에 있더라도 검출할 수 있다
- 2) 완벽히 일치하는 형태가 아니라도 검출할 수 있다

- » 쥐 필터는 간단한 예
- » 실제로는 필터를 학습하면 이러한 형태
 - ◆ 한 개의 작은 조각이 한 개의 필터를 의미



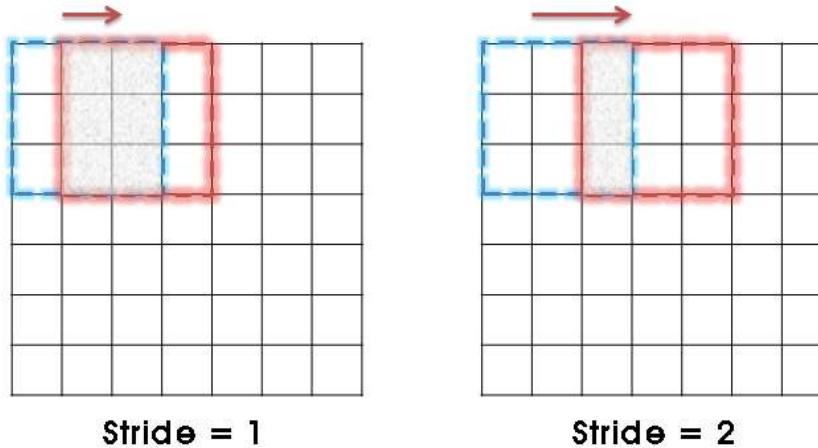
필터들을 시각화 한 예

이미지분류기 전체 알고리즘 도식화



- » 파라미터 수가 적을 수록 overfitting에 빠지지 않고 더욱 정확한 학습이 가능
- » 하지만 Output Layer에 가까운 FC의 파라미터수가 12만개로 아직도 훨씬 많은 수
- » 최근에는 이를 더욱 더 개선하기 위하여 FC를 완전히 없앤 Fully convolutional network도 있음

- » 필터의 이동간격
- » (앞서) 작은 필터를 한 칸씩 움직이면서 Convolution을 계산하는 것
= stride를 1로 계산한 것
- » Stride가 1인 경우 : 필터가 훑고 지나간 부분이 많이 겹침, feature map도 크기 큼
(필터가 겹치는 부분이 많다는 것이 나쁜 것을 의미하는 것은 아님!!)

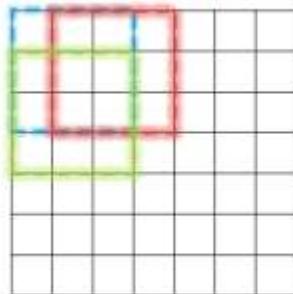


: 이전 필터와 중복되는 영역

Stride=1과 Stride=2의 예

Stride

Stride = 1

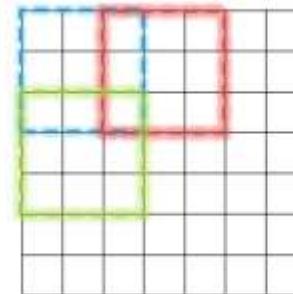


7x7 입력



5x5 출력

Stride = 2



7x7 이미지

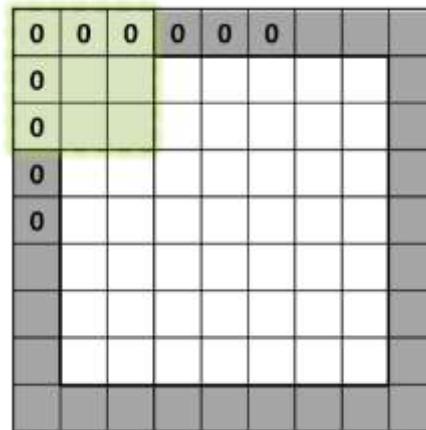


3x3 출력

- » 서로 다른 Stride 값을 사용함에 따라 출력이미지의 크기가 다름

Zero padding과 출력이미지의 크기 계산

- » 주어진 이미지의 가장자리를 0으로 둘러싸는 것
- » Zero padding을 잘 이용하면 출력이미지의 사이즈를 조절가능
- » Stride를 최소값 1로 사용하더라도 출력이미지의 사이즈는 줄어들 수 밖에 없음
- » 이를 보완해 줄 수 있는 것이 Padding
- » convolution 연산 후에도 입력이미지의 사이즈와 출력이미지의 사이즈를 같게 만들 수 있음



7x7 이미지에 Padding 적용 예

» 출력이미지 사이즈, Stride값과 padding값에는 관계

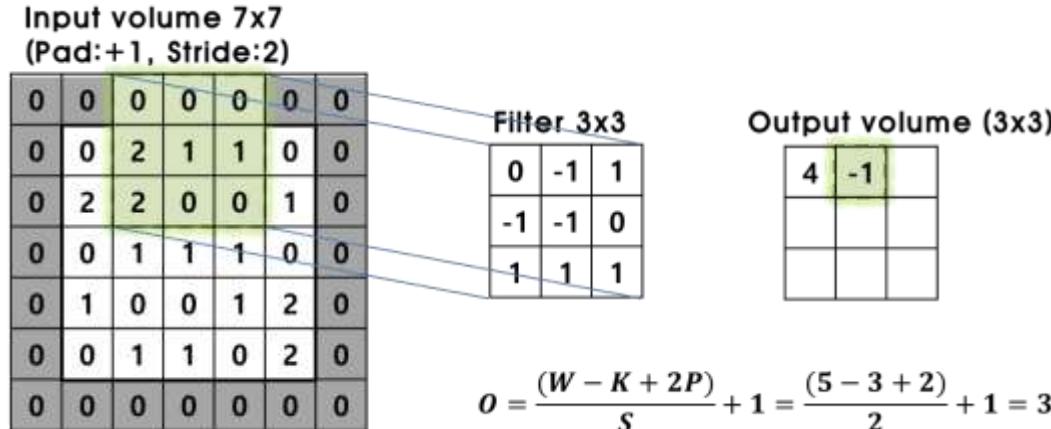
» 가로 또는 세로의 출력사이즈(O)

- ◊ 입력사이즈(W)
- ◊ 필터크기(K)
- ◊ Stride(S)
- ◊ Zero Padding(P)

$$O = \frac{(W - K + 2P)}{S} + 1$$

» 계산결과 값이 정수가 아니라면 일반적으로 Stride가 잘못 정해진 경우

Zero padding과 출력이미지의 크기 계산



Convolution 연산 연습

- » 5x5 이미지에 필터사이즈가 3x3에 Padding= 1, Stride=2
- » 출력이미지 연산공식결과는 3이므로, 3x3이미지가 출력
- » 계산해봅시다!
- » 첫 번째 행 : 픽셀이 모두 0
- 두 번째 행 : $2 * (-1), 1 * (-1)$ 을 계산.
- 세 번째 행 : $2 * 1$ 을 계산. → 모두 더하면 -1

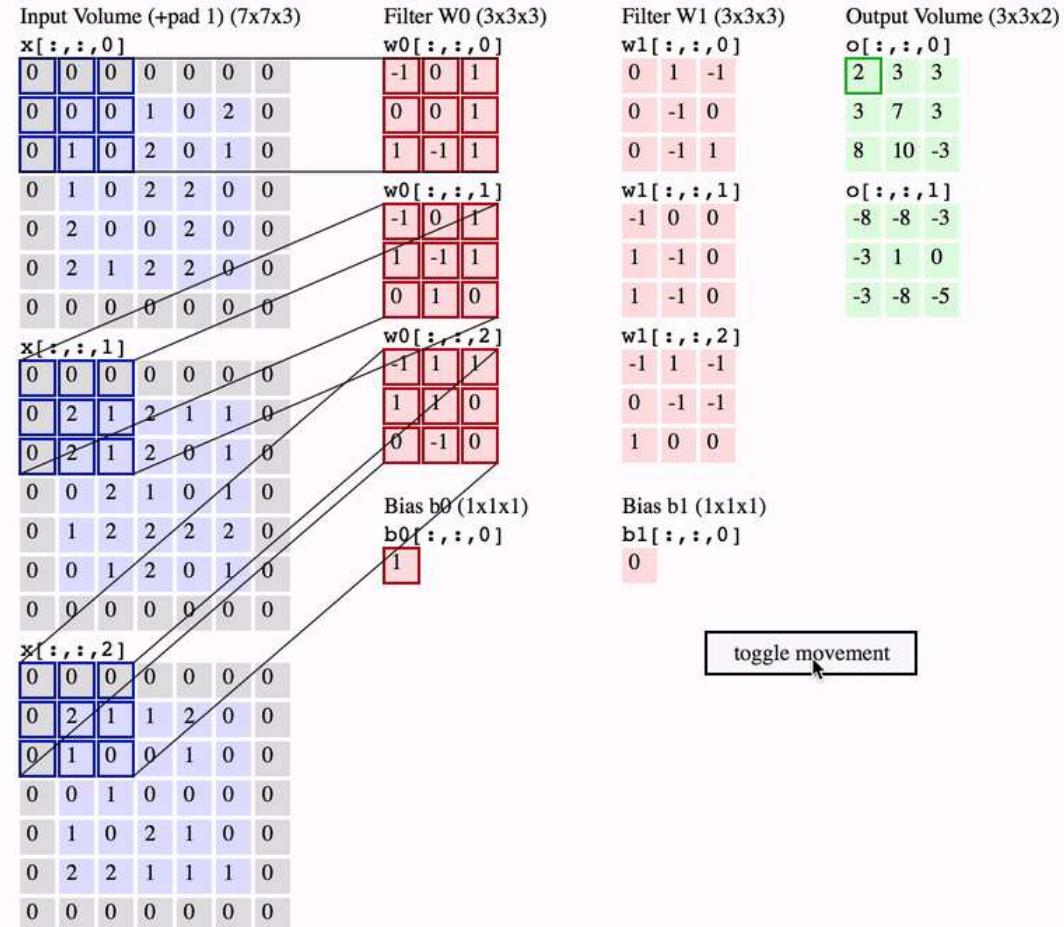
Convolution Demo

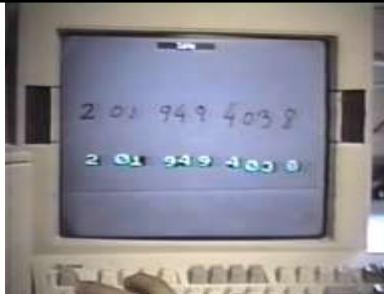
» W=5, F=3, S=2, P=1

$$O = \frac{(W - K + 2P)}{S} + 1$$

$$(5 - 3 + 2)/2 + 1 = 3$$

» (필터2개, RGB 3 Depth 데모)



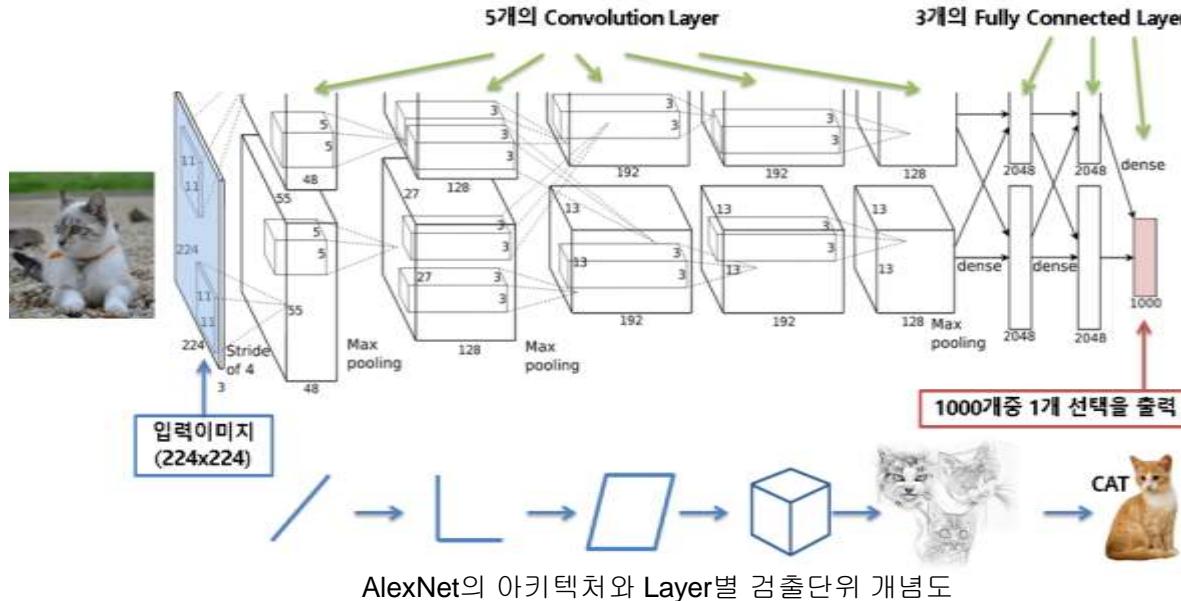


- » CNN을 공부하다 보면 빠질 수 없는 주요 study 리뷰
 - » LeNet은 Convolutional Networks의 첫 번째 성공적인 응용 프로그램

- ◆ 1990년대 Yann LeCun 교수(현재, 뉴욕대학교 교수 겸 facebook 인공지능연구소 소장)가 개발
 - ◆ 우편 번호, 숫자 등을 읽는 데 사용
 - ◆ LeNet이 개발되었을 때에는 컴퓨팅 성능이 연산을 수행하기에는 충분하지 못해 주목 받지는 못함



- » Computer Vision에서 Convolutional Networks를 대중화 한 첫 번째 기록
 - ◊ Alex Krizhevsky와 Ilya Sutskever, Geoff Hinton이 개발
- » ImageNet ILSVRC 에 도전하여 2등과의 큰 격차로 우승
- » 처음으로 CNN을 사용하여 주목할 만한 결과를 낸 아키텍처, drop out 기법은 교과서적 가치



- » ImageNet Large Scale Visual Recognition Challenge ([ILSVRC](#)) 2014 준우승
- » VGG는 이들이 속한 Visual Geometry Group의 첫 글자
- » Convolution Network의 깊이가 높은 정확도를 위한 중요한 요소라는 것을 보여줌
- » 크기가 작은 convolution filter를 사용하여서 학습해야 할 파라미터 개수를 줄임
- » VGG net의 네트워크 구조는 ImageNet 데이터셋 뿐만 아니라 다른 데이터셋에도 잘 동작
- » 미리 학습이 된(pre-trained) VGG net 네트워크를 공개

VGG net의 ConvNet configurations

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
softmax					

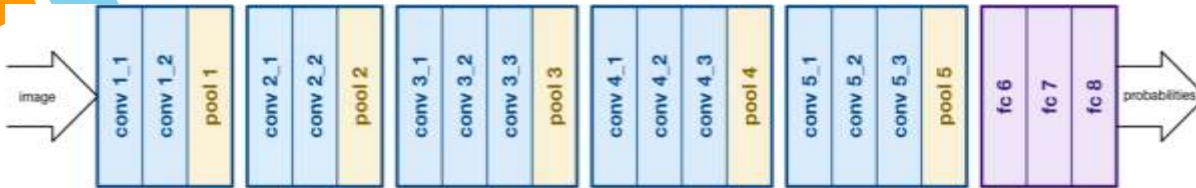
» 레이어를 많이 쌓을수록 좋은 성능을 보이는 것을 비교
A 레이어개수 < E 레이어개수

» VGG Net의 입력은 224×224 의 고정크기 RGB 이미지
» Conv Layer는 stride=1, padding=1, 3x3 Filter 사용
64개의 채널(depth)로 출력

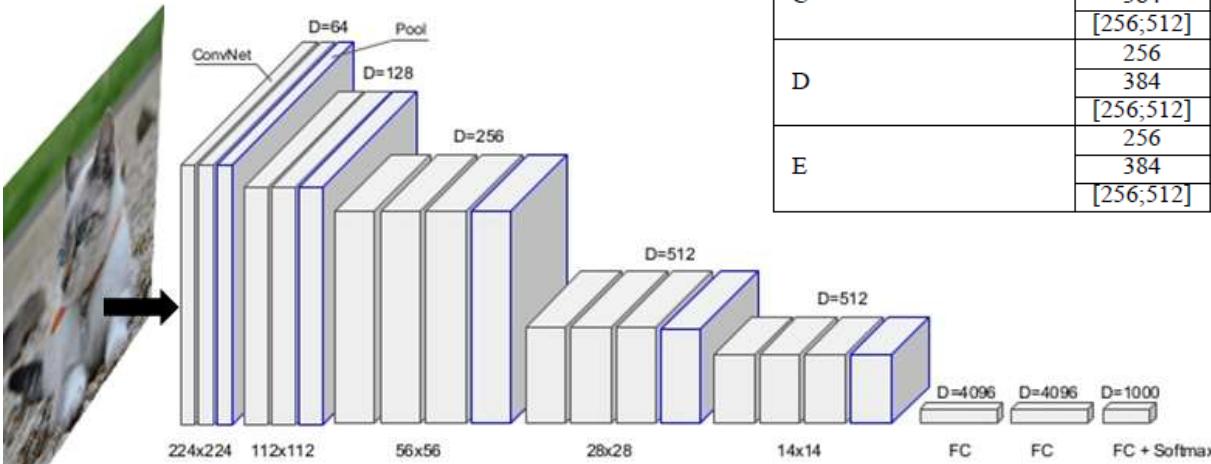
» 3개의 Fully connected(FC) 레이어

» softmax를 통하여 1000개의 채널(Image net 정답범위) 출력

VGG Net



VGG net 16 Layer 의 구성요소



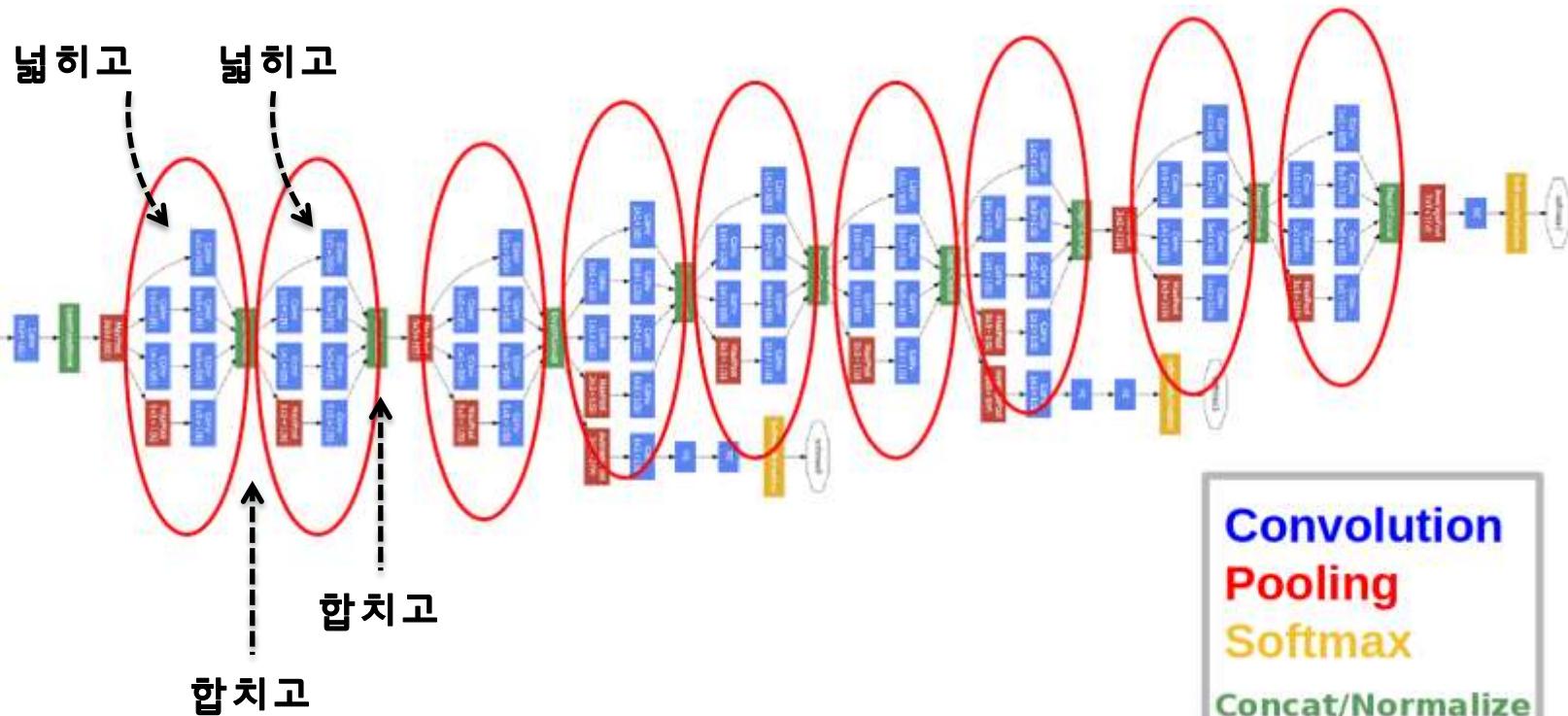
VGG net 16 Layer 아키텍처 입체적 표현

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (<i>S</i>)	test (<i>Q</i>)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
[256,512]	384	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
[256,512]	384	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
[256,512]	384	384	25.5	8.0

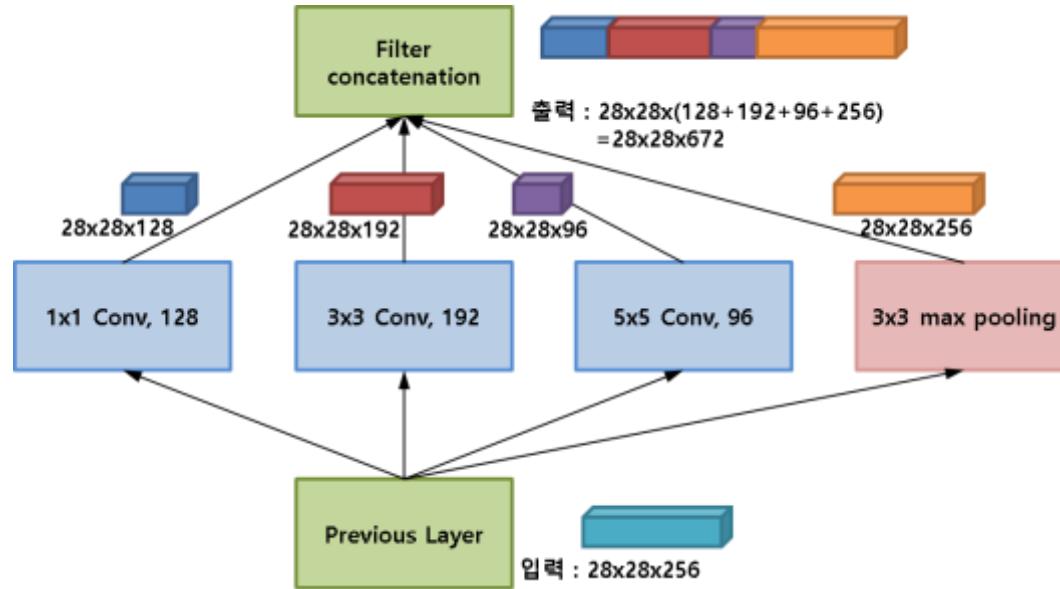
VGG net ConvNet 설계

- » Google이 LeNet에 기반하여서 만든 GoogLeNet
 - » 22개의 레이어로 이루어짐
 - » 파라미터 개수는 최대한 적게 하면서도 레이어는 넓고 깊어지도록 개발됨
 - » Alex net과 비교해 볼 때 파라미터 수는 12분의 1로 줄었지만 더욱 좋은 성능
 - » ILSVRC-2014에서 1등
 - » Inception v1을 사용하여 구현한 네트워크 중 하나 (현재는 Inception v4까지 발표되어 있음)
-
- » 레이어 수를 단순히 늘리는 것이 아니라 몇 개의 레이어와 연산을 묶어 하나의 모듈로 간주
 - » 모듈 안에서는 아주 밀도 높은 연산, 모듈간 연결은 느슨한 형태를 취함
 - » GoogLeNet에는 Inception Module이라 부르는 이런 모듈이 9개 사용

GoogLeNet 아키텍쳐

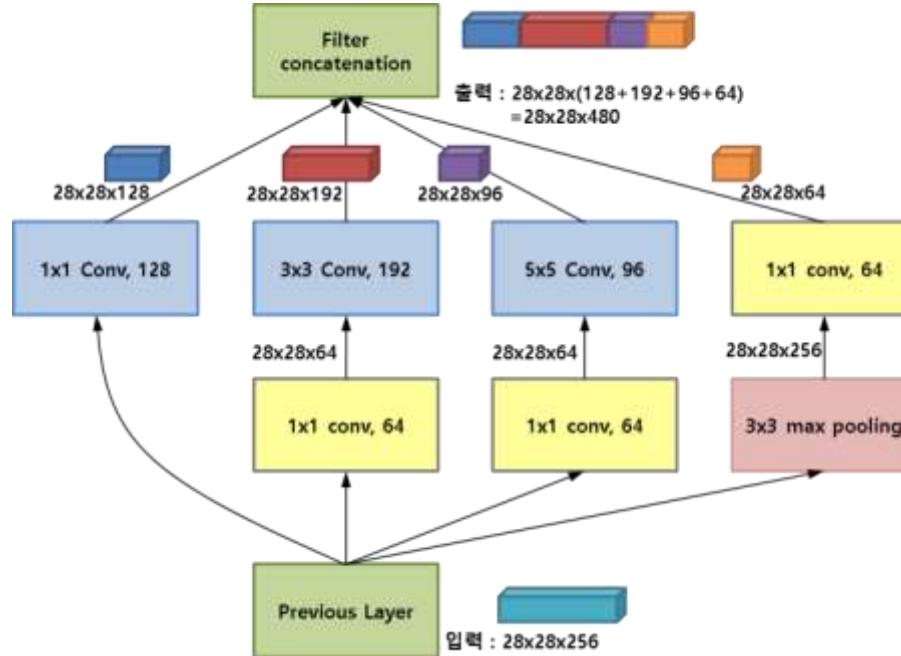


native inception module 연산



- » 이미지의 큰 영역을 커버하면서도 동시에 작은 영역에 대한 정밀함도 유지하는 구조
- » 필터의 크기에 따라 작은 형태에 집중하다 보면 큰 그림에서의 의미를 소홀히 할 수 있고, 반대의 경우도 있을 수 있음
- » 가장 디테일한 1x1 convolution 연산과 큰 5x5 convolution까지 다양한 크기의 convolution 연산을 동시에 수행
- » Max pooling layer를 사용해서 이전 레이어의 정보를 요약한 데이터도 추가
- ➔ 다양한 크기의 개체를 잘 처리

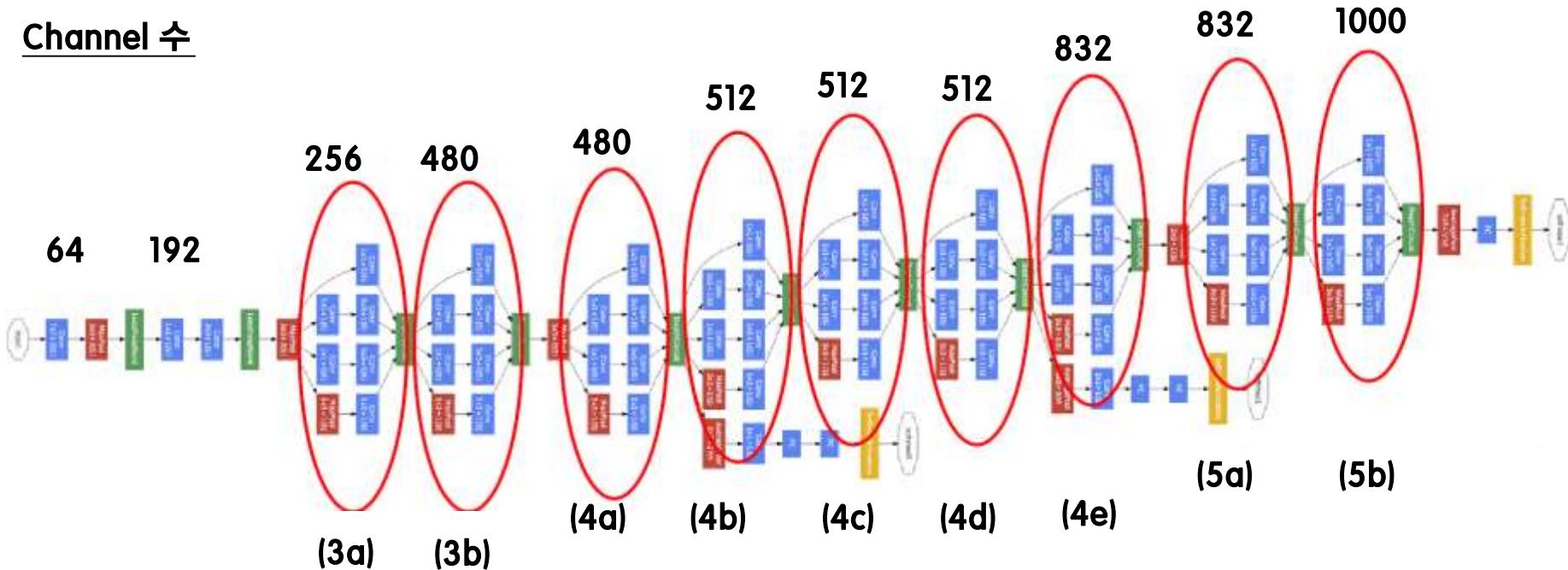
차원감소를 포함한 Inception module



- » 1x1convolution연산을 먼저 수행 : 차원을 늘리는 것이 아니라 줄이는 역할을 하기 때문에 이후 컴퓨터 계산 량이 많이 줄어드는 효과를 기대

레이어별 output되는 채널 수

Channel 수



- » 점점 채널수가 증가해서(이미지의 사이즈는 감소하고)
- » 최종적으로 softmax를 취해서 1000개 중 하나를 선택할 수 있는 $1 \times 1 \times 1000$ 의 형태로 출력

8.

CNN 실습

Inception Module을 활용한 이미지 분류기
이미지 분류기 Retraining - 꽃 분류기

sli.do # Inception Module을 활용한 이미지분류기 데모

- » GoLeNet을 구현할 때 사용한 Inception 모듈을 사용하여 간단한 이미지 분류기
- » 텐서플로우와 Inception모듈은 구글에서 개발하고 공개
- » 텐서플로우 사이트를 통해 누구나 쉽게 접할 수 있음

- » 실습코드는 텐서플로우 사이트에 공개된 내용
 - ◊ Inception-v3를 활용
 - ◊ ImageNet 대회를 위해 이미 학습된 모델을 불러온 후 입력한 이미지가 무엇인지 추론
 - ◊ 정답일 가능성이 높은 5개를 출력
 - ◊ 직접 모델학습 작업에 하려면 너무나 많은 시간과 컴퓨팅자원이 필요
 - ➔ 학습된 파라미터를 다운받아 실습



goo.gl/7JNwGM

giant panda, panda, panda bear, coon bear, Ailuropoda melanoleuca (score = **0.89632**)

indri, indris, Indri indri, Indri brevicaudatus (score = **0.00766**)

lesser panda, red panda, panda, bear cat, cat bear, Ailurus fulgens (score = **0.00266**)

custard apple (score = **0.00138**)

earthstar (score = **0.00104**)

» Top-5 단어와 확률을 출력

» 가용한 옵션들..

옵션	type	기본값	설명
--model_dir	str	/tmp/imagenet	파일다운로드 경로
--image_file	str	없음	입력 이미지 경로
--num_top_predictions	int	5	출력할 정답 개수

sli.do # Inception Module을 활용한 이미지분류기 데모

» 실행명령 예

◇ python classify_image.py --image_file=dog.jpg



Chihuahua (score = 0.96313)
toy terrier (score = 0.00155)
Boston bull, Boston terrier (score = 0.00107)
Brabancon griffon (score = 0.00106)
kelpie (score = 0.00095)



drake (score = 0.53597)
American coot, marsh hen, mud hen, water hen (score = 0.08248)
goose (score = 0.02880)
red-breasted merganser, Mergus serrator (score = 0.01635)
lakeside, lakeshore (score = 0.01044)



goldfish, Carassius auratus (score = 0.92140)
house finch, linnet, Carpodacus mexicanus (score = 0.00169)
macaw (score = 0.00148)
crane (score = 0.00065)
ocarina, sweet potato (score = 0.00061)



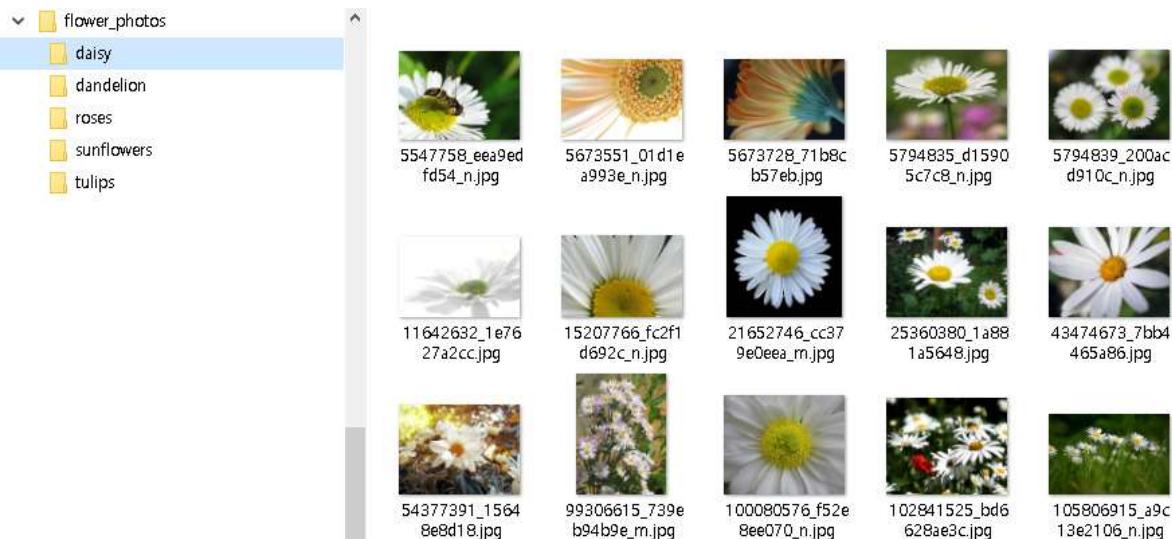
fire engine, fire truck (score = 0.98058)
tow truck, tow car, wrecker (score = 0.00060)
trailertruck, tractor trailer, trucking rig, articulated lorry (score = 0.00047)
garbage truck, dustcart (score = 0.00039)
fireboat (score = 0.00034)

pip install tensorflow-hub

- » 머신러닝 모델을 재사용 할 수 있도록 배포와 검색, 사용을 쉽게 할 수 있도록 도와줌
- » 다른 TASK간에 재사용 가능
- » 모듈은 미리 학습된(pre-trained) 데이터 셋의 정보를 포함
- » 장점
 - ◊ 작은 데이터 셋으로 모델을 훈련시킬 수 있습니다.
 - ◊ 일반화를 향상 시킬 수 있습니다.
 - ◊ 학습 속도가 크게 향상됩니다.

- » 앞에서는 Image Net에서 사용한 1000개의 레이블 중에서 주어진 그림이 무엇인지 맞추는 방법을 알아봄
 - » ‘고양이’ → 페르시안 고양이, 샴고양이, 브리티시 속헤어, 러시안 블루와 같이 고양이 종류를 세분화한 모델이 필요하다면?
 - » 전체 네트워크를 처음부터 학습하는 것은 수백 시간의 GPU 자원이 필요
-
- » 기존에 잘 훈련된 모델을 바탕으로 재학습 하는 방법을 사용
 - ◆ 전체 네트워크를 학습했을 때 보다는 좋은 성능을 내지는 못하겠지만, 상대적으로 짧은 시간(약 30분 정도)만에 제법 훌륭한 성능
 - » 이러한 과정을 “Fine tuning”
 - » 이번 실습에서는 Inception-v3 모듈을 재학습 시켜서 꽃 분류기를 만들어 보기

- » 데이지, 민들레, 장미, 해바라기, 튤립 사진을 여러 개 보여주어서 네트워크를 재학습
- » 학습용 데이터셋 이미지 준비
 - ◊ 분류할 명칭을 폴더이름으로 사용
 - ◊ 즉, daisy라는 단어를 학습하기 위해 필요한 이미지를 daisy 폴더에 모아놓는 식



» retrain.py

- ◊ 재학습을 하기 위한 소스코드
- ◊ 실행방법 : python retrain.py --image_dir flower_photos
- ◊ 4000번 step을 반복 (무작위로 10개의 이미지를 선택)
- ◊ /tmp/retrain_logs에는 로그가 쌓임
- ◊ [실행] python retrain.py --image_dir flower_photos

» label_image.py

- ◊ 직접 학습시킨 결과로 이미지분류 실행
- ◊ [실행] python label_image.py --graph=/tmp/output_graph.pb --labels=/tmp/output_labels.txt --input_layer=Placeholder --output_layer=final_result --image=tulip.jpg

- » 레이블을 이름으로 하는 폴더를 만들고 이미지 파일을 모음
- » 그 레이블 폴더의 루트폴더를 --image_dir의 파라미터로 전달
- » 학습할 이미지는 많으면 많을수록 좋음. >100개
- » 배경에도 신경을 써야 함 (너무 단순하면 분류기준이 배경이 됨)
- » 잘못된 이미지가 섞여 들어가지 않도록 주의 (데이터꽃 vs. 친구이름 데이터)
- » 이미지를 자르거나 회전시키거나 밝기를 변화시키는 것과 같이 이미지를 왜곡 시켜서
한 장의 이미지로 여러 장의 데이터셋을 만들 수 있음
(retrain.py에 준비된 --random_crop, --random_scale, --random_brightnes 옵션을 사용)

- » 실습 예제는 Inception-v3 모델을 사용했지만,
 - ◊ NASNet (notably nasnet_large 과 pnasnet_large)이라는 최신의 더 강력한 모델을 적용하여 더 높은 정확도를 얻을 수도 있음
 - ◊ 모바일과 같은 리소스가 열악한 장치에서 실행시키기 위해서 좀더 가볍고 빠르지만 정확도는 약간 낮은 MobileNet V1 또는 V2 또는 nasnet_mobile 가능

tulips	0.9956002
roses	0.0042540072
sunflowers	7.076714e-05
daisy	6.309221e-05
dandelion	1.19224505e-05



9.

RNN과 LSTM

RNN이란?

RNN의 다양한 입력과 출력 관계

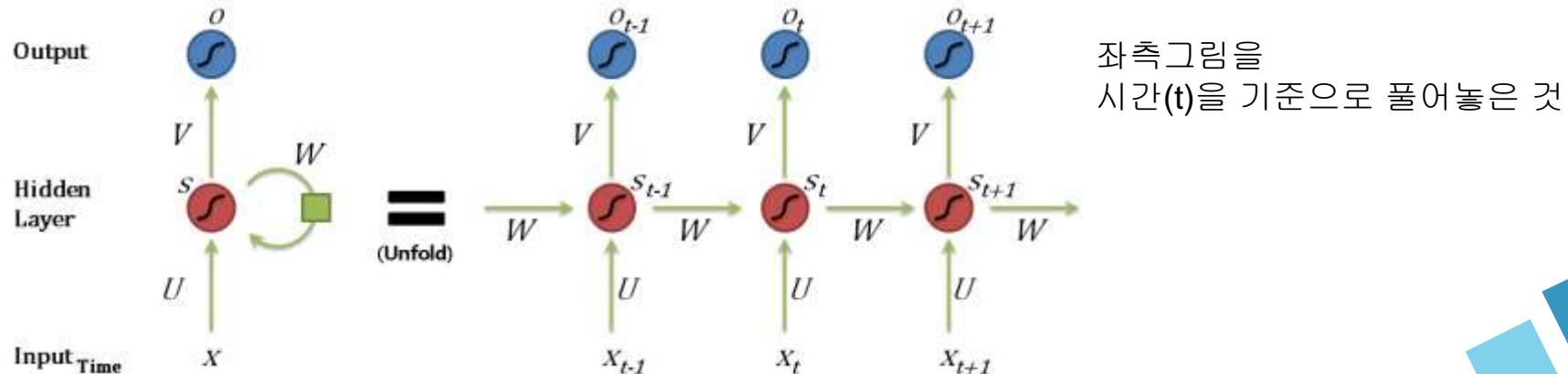
Backpropagation Through Time(BPTT)

RNN 모델에서 Vanishing Gradient Problem

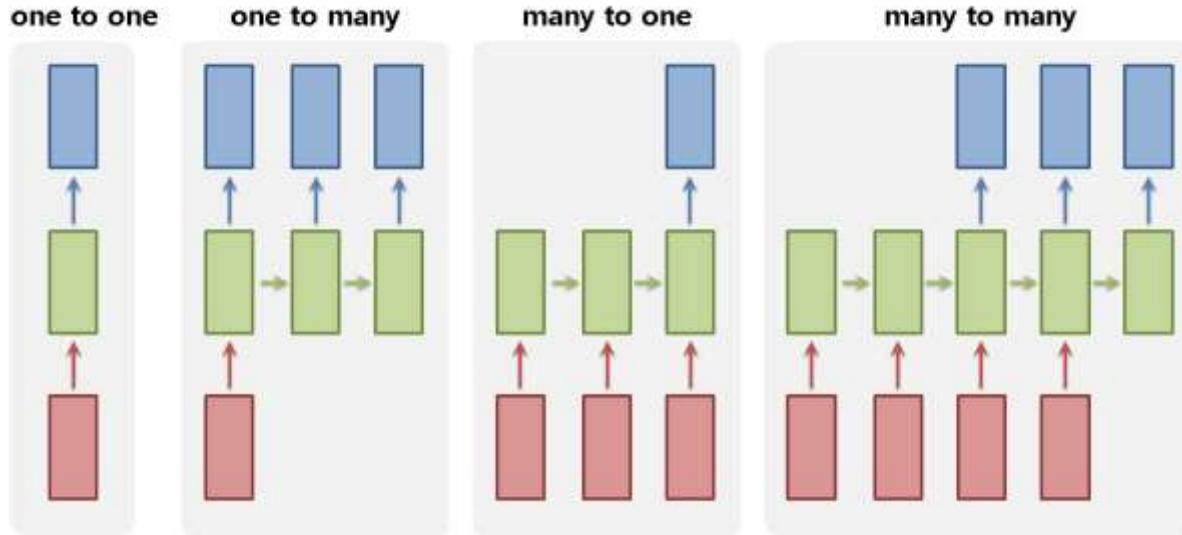
LSTM(Long Short Term Memory) Network에 대한 소개

- » 우리가 일상적으로 접하는 사람의 음성이나 동영상은 연속적인(Sequence) 데이터
 - ◊ 특정 단어나 장면만으로는 그 내용을 파악할 수 없음
 - ◊ 앞뒤로 연결된 데이터 속에 정보가 들어 있음
- » 이전에 입력 받았던 정보를 계속 활용하면서 새로 들어온 데이터를 고려하는 관계
- » RNN은 음성입력이나 여러 글자로 이루어진 문장 또는 동영상과 같이 연속적이며 순서가 의미를 가지는 데이터를 다룰 때 적합한 모델

- » 뉴럴네트워크 내부에 상태가 기억되고 상태정보는 순환하는 연결을 가짐
- » 현재시점(t)의 상태(S_t)는 이전시점(t-1)에서 전달된 가중치(W)의 영향을 받으며, 다음시점(t+1)의 상태(S_{t+1})에도 영향을 주는 구조
- » 잠재적으로 네트워크 내부에 메모리를 가지는 형태



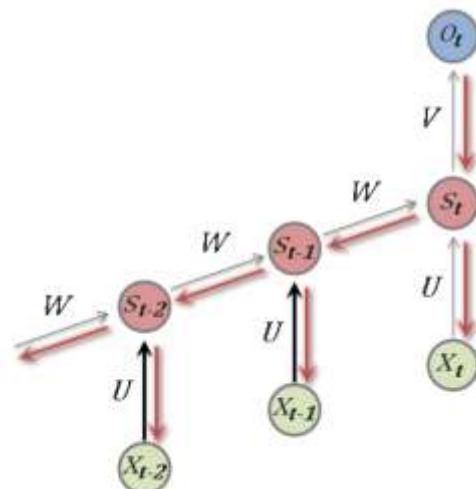
RNN의 다양한 입력과 출력 관계



- » RNN이 다루는 데이터는 연속성 또는 순서가 있는 데이터
 - ◊ (기존의 CNN 뉴럴네트워크는 하나의 입력데이터에 하나의 결과출력)
- » One to Many 예시) 사진을 한 장 입력하고 그것을 설명하는 문장을 출력
- » Many to One 예시) 어떠한 문장을 이야기 했을 때 기쁨, 슬픔과 같이 짧게 감정을 분류
- » RNN의 가장 일반적 사용은 Many to Many 관계

Backpropagation Through Time(BPTT)

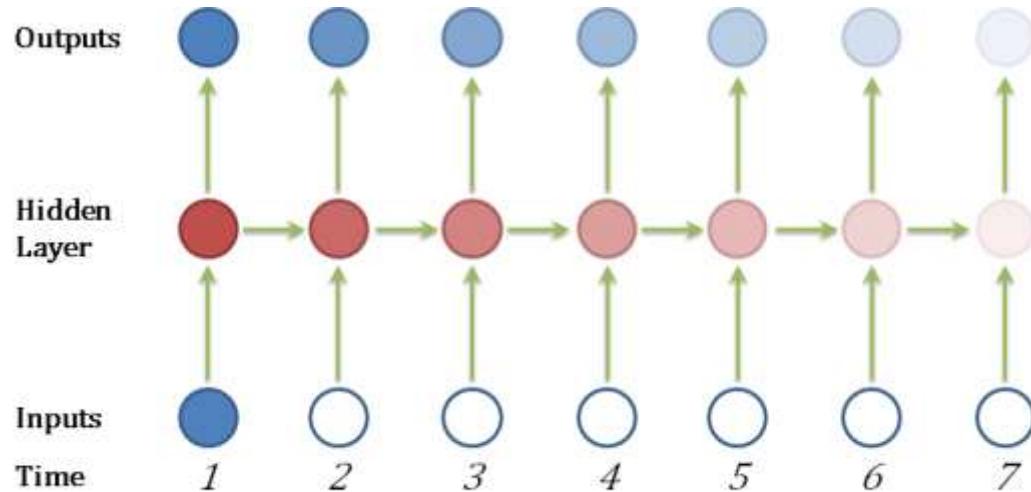
- » RNN은 시계열 데이터를 처리하기 위한 모델.
 - » 그렇다면, 시간상으로 흘러가버린 내용에 대해서 어떻게 학습할 것인가?
- 기존의 뉴럴네트워크와 동일하게 역전파(Back propagation)와 경사로 따라 내려가기(Gradient Descent)를 활용하여 학습



U, V, W에 대해서 오차의 기울기(미분값)를 구하고 기울기가 줄어드는 방향으로 업데이트

sli.do # RNN 모델에서 Vanishing Gradient Problem

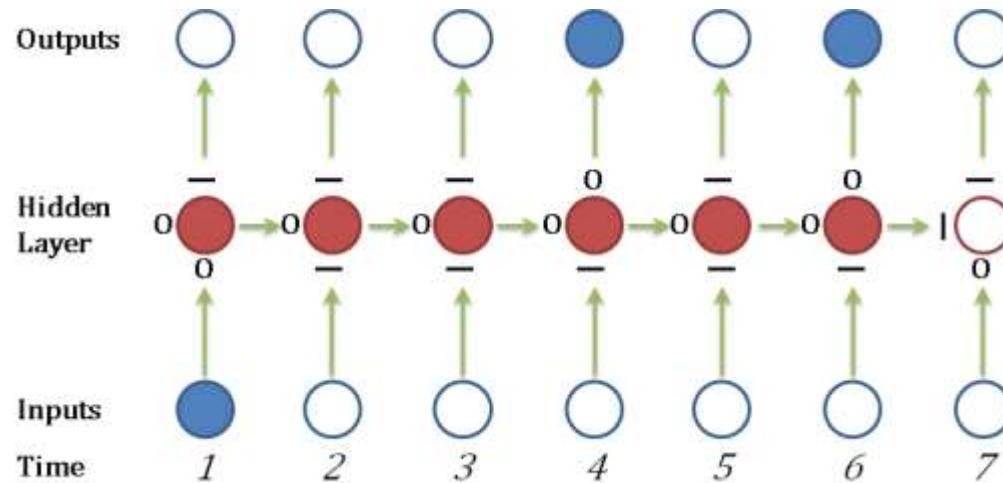
- » RNN은 태생적으로 Vanishing Gradient Problem을 가지고 있음
- » 일찍 입력된 데이터의 영향력은 시간이 지날수록 점점 감소하다가 사라짐 문제 있음



시간 1에서 입력된 데이터가 얼마나 오랫동안
네트워크에 영향을 주는지에 대한 감도

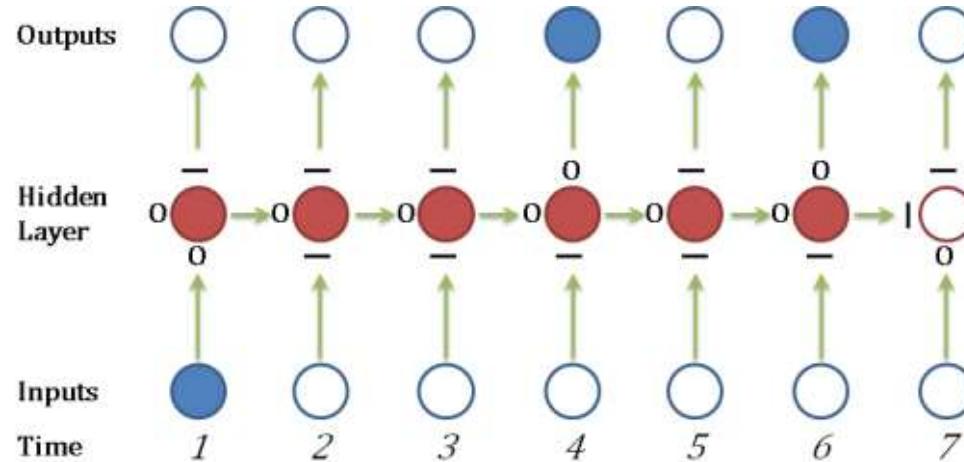
LSTM(Long Short Term Memory) Network

- » Vanishing Gradient Problem 문제를 해결하기 위해서 좀 더 복잡한 형태의 RNN
- » LSTM은 RNN의 아주 대표적인 아키텍처로 RNN을 실제 구현할 때는 LSTM을 사용한다고 이해하셔도 무방함



LSTM에서 Gradient 정보가 저장되는 방식

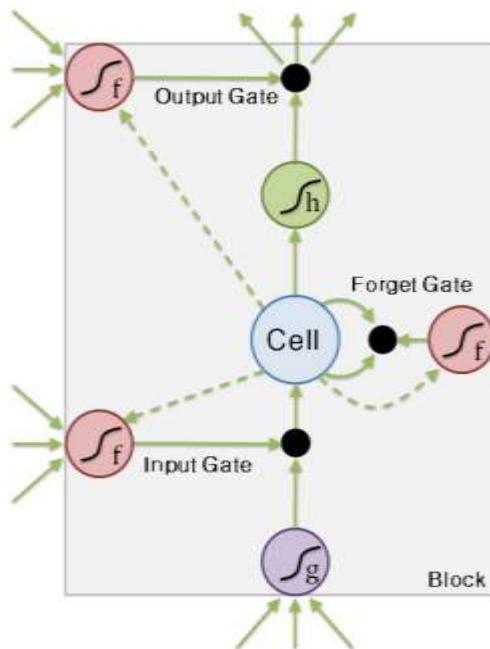
LSTM(Long Short Term Memory) Network



- » 그림에서 'O'와 'I'를 볼 수 있는데, 'O'는 Open을 'I'는 Close
 - ◊ Open은 데이터가 흘러갈 수 있는 것이고 Close는 흘러갈 수 없음
- » 입력된 데이터를 다음 시간에도 계속 전달하며 유지하겠다는 의사를 명확히 함
 - ➔ 입력데이터의 가중치가 희석되지 않고 계속해서 유지됨

LSTM Cell 구조

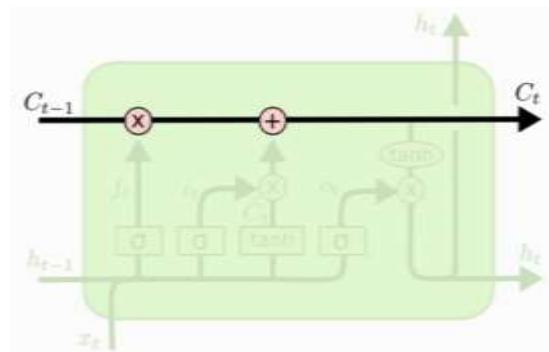
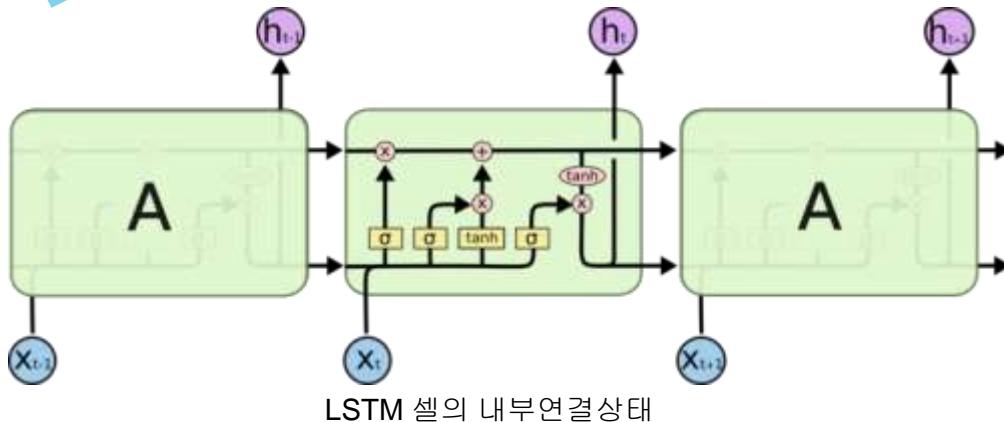
- » Hidden Layer에 Input Gate, Output Gate, Forget Gate 존재
- » 앞에서는 동그라미 하나로 표현했지만 내부적으로는 다음 그림과 같이 약간은 복잡한 연결관계



LSTM Cell 구조

- » Input Gate는 새로운 정보가 cell state에 저장될지 결정
- » Forget gate와 Input gate에서 출력된 값을 Cell state로 업데이트
- » Output gate는 출력 값을 결정하는 게이트
- » 모든 시간에서 같은Weight를 공유하는 전통적인 RNN과 달리
LSTM은 이러한 Gate를 통해서 각 시간대별 Cell이 서로 다른 판단

LSTM 셀의 내부연결상태를 구체화



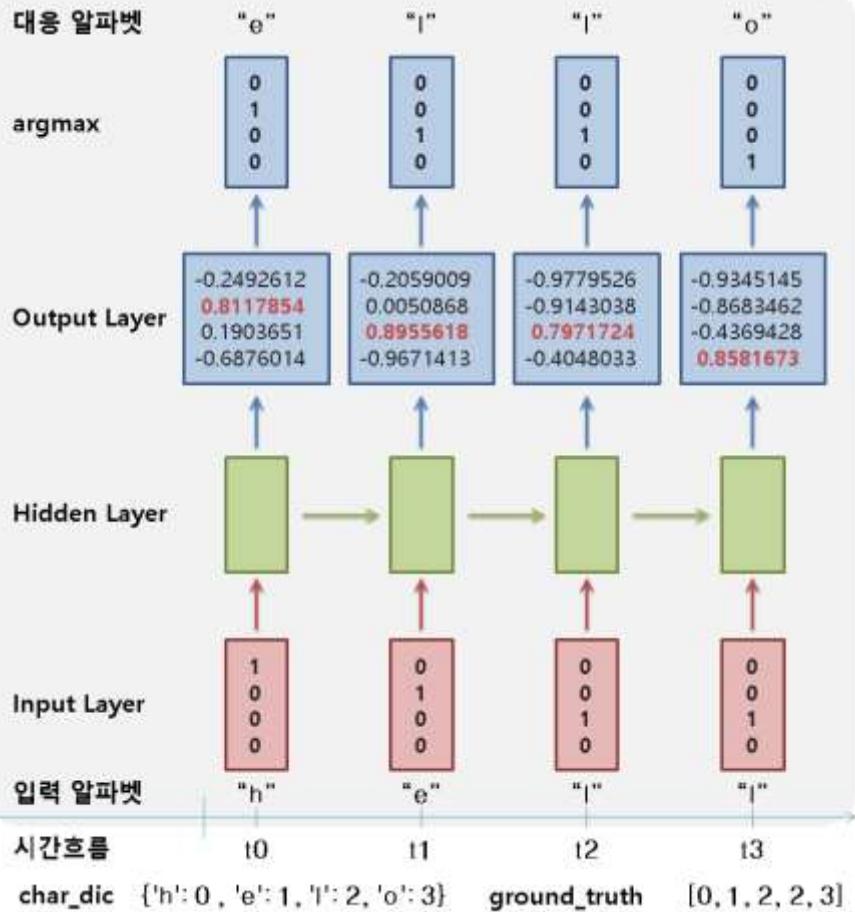
LSTM의 핵심 아이디어

- » 전통적인 RNN은 셀의 내부가 Activation function 하나로 이루어짐
→ 깊은 네트워크의 모두가 동일한 가중치를 공유 = Gradient Vanishing Problem 발생
- » LSTM은 셀 안에 제어네트워크가 구성
→ 각 시간대별로 다음 시간의 셀로 정보를 넘길지 얼마나 넘길지, 이번에 들어온 입력값은 얼마나 반영할지를 결정

10.

RNN 실습

HELLO단어기반 다음알파벳 예측하기



- » hello라는 단어를 알파벳 단위로 학습
- » h를 입력하면 다음단어는 e가 가장 확률이 높다고 출력
- » 데이터셋 : h, e, l, o 4개 알파벳을 one-hot 인코딩으로 변환
- » 출력된 벡터를 다시 알파벳으로 변환하여 잘 학습되었는지 확인

```
12     char_rdic = ['h', 'e', 'l', 'o'] # index를 char로 변환할때 사용  
13     char_dic = {w : i for i, w in enumerate(char_rdic)} # char를 index로 변환할때 사용  
14     ground_truth = [char_dic[c] for c in 'hello']
```

» 데이터셋(dictionary)을 준비

- ◊ ‘char_rdic’ : hello를 이루는 4개 알파벳을 배열로 생성
(RNN의 출력결과가 1이라면 index가 1인 알파벳 ‘e’로 변환하는 데 사용)
- ◊ ‘char_dic’ : h를 입력하면 index인 0을 출력하는 데에 사용
- ◊ ‘ground_truth’ : 학습할 정답으로 ‘hello’의 경우 [0, 1, 2, 2, 3]

```
27  # Configuration  
28  rnn_size = len(char_dic) # 4  
  
34  # RNN Model  
35  rnn_cell = tf.nn.rnn_cell.BasicRNNCell(num_units = rnn_size)
```

- » RNN 모델을 이루는 셀을 정의
- » 파라미터 num_units는 내부적으로는 RNN 셀 내부의 unit의 개수
 - ◊ 아웃풋의 사이즈와 관계가 있음
 - ◊ 실습에 사용하는 데이터셋의 길이에 맞춰서 4로 설정
- » 실습에서는 BasicRNNCell ()을 사용하였지만,
- » BasicLSTMCell()이나 GRUCell()로 클래스 이름만 바꾸어도 해당 RNN 모델로 동작시킬 수 있음

» 결과 확인

[입력]

```
16     x_data = np.array([[1,0,0,0], # h  
17                     [0,1,0,0], # e  
18                     [0,0,1,0], # l  
19                     [0,0,1,0]], # l  
20                     dtype = 'f')
```



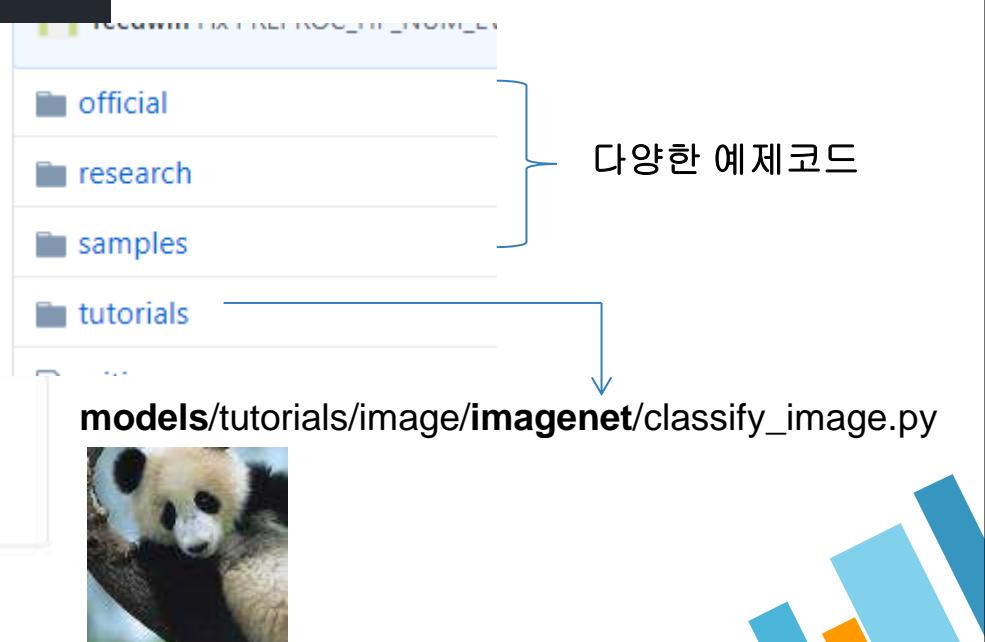
```
[1 2 2 3] ['e', 'l', 'l', 'o']  
[1 2 2 3] ['e', 'l', 'l', 'o']
```

[예측결과]

텐서플로우 GITHUB 둘러보기

» <https://github.com/tensorflow>

The screenshot shows the main page of the TensorFlow GitHub repository. At the top, there's a search bar and navigation links for Pull requests, Issues, Marketplace, and Explore. Below that, the repository name 'tensorflow' is displayed with its logo. It shows 64 repositories, 176 people, and 1 project. A 'Pinned repositories' section features the main TensorFlow repository and a 'models' repository, which is highlighted with a red box. The 'models' repository is described as 'Models and examples built with TensorFlow' and includes Python and C++ versions. At the bottom, there are search and filter options.



TF예제(Neural Image Caption Generator)

» <https://github.com/tensorflow/models/tree/master/research/im2txt>

models /research /im2txt/

A person on a beach
flying a kite.



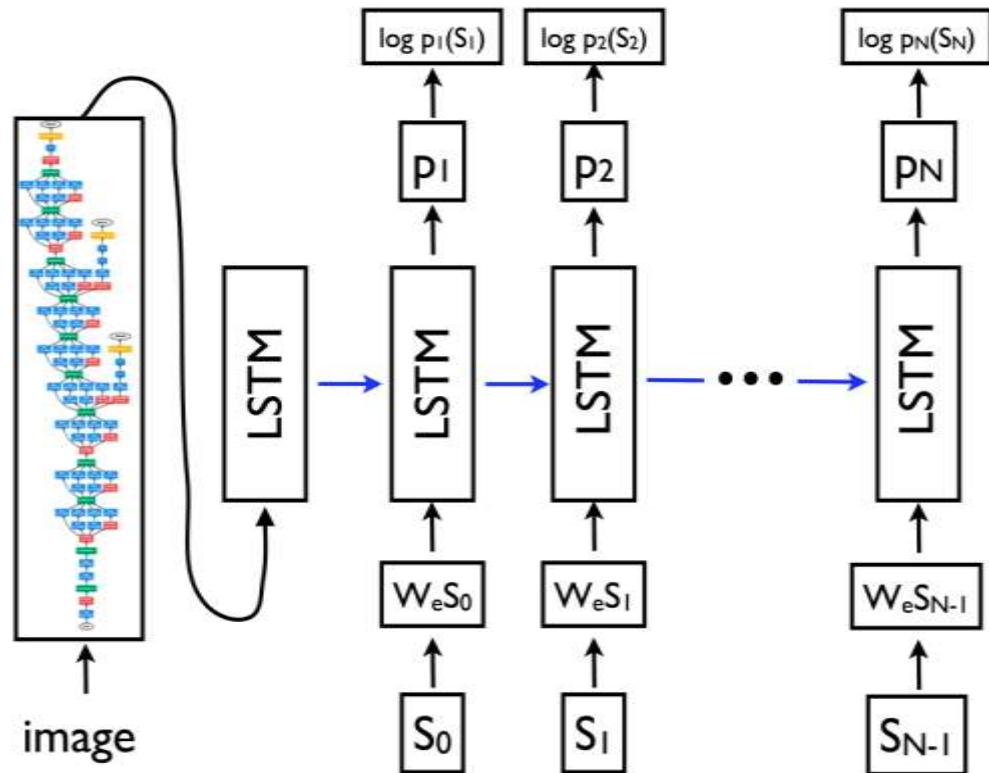
A black and white photo of a train on a train track.



A person skiing down a snow covered slope.



A group of giraffe standing next to each other.



11.

시각화 도구 Tensor Board

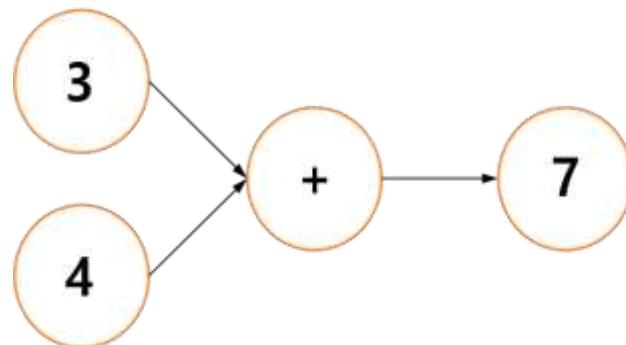
Tensor Board에 대한 소개

Tensor Board 사용을 위한 코드 추가

Tensor Board 실행하기

- » 텐서플로우는 데이터 플로우 그래프(Data flow graph)를 사용하여 연산
- » 그래프 노드 관계를 시각화 해서 볼 수 있다면
전체 아키텍처를 파악하거나 디버깅 할 때 많은 도움

$$3 + 4 = 7$$



- » 텐서플로우가 실행되는 동안 기록된 로그를 사용
- » 스칼라 값, 그래프 연결구조, 히스토그램 등 다양한 형태의 데이터를 표현
- » 모든 정보를 알아서 표시한다? (X)
- » 필요한 로그가 저장될 수 있도록 소스코드에 미리 저장동작 추가

TensorBoard SCALARS IMAGES GRAPHS DISTRIBUTIONS HISTOGRAMS INACTIVE C

Show data download links
 Ignore outliers in chart scaling
Tooltip sorting method: default ▾

Smoothing

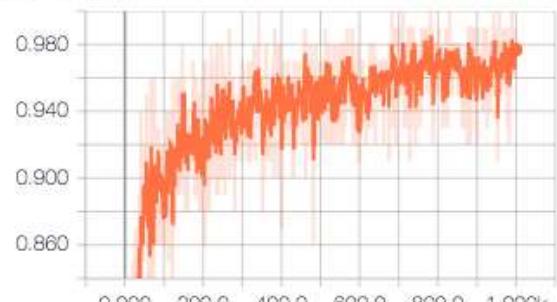
Horizontal Axis STEP RELATIVE WALL

Runs
Write a regex to filter runs

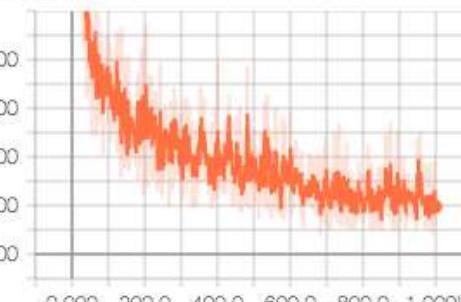
Tags matching ./.* (all tags)

PREVIOUS PAGE NEXT PAGE

accuracy_1



cross_entropy_1



Fit to screen

Run

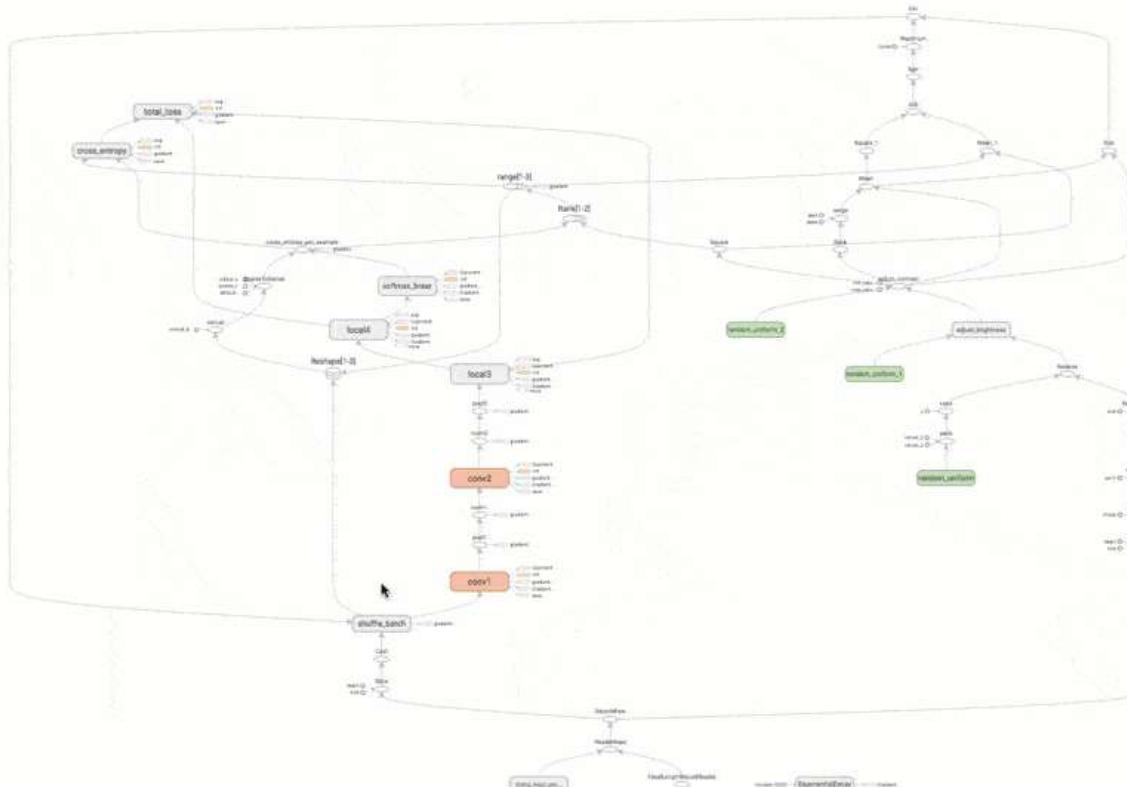
Upload

Color Structure
color: same substructure
gray: unique substructure

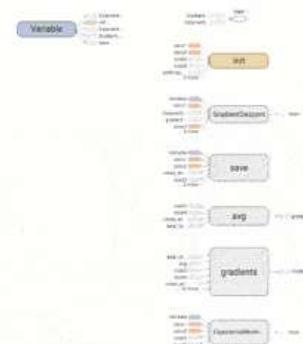
Graph (* = expandable)

- NameSpace*
- OpNode
- Unconnected series*
- Connected series*
- Constant
- Summary
- Dataflow edge
- Control dependency edge
- Reference edge

Main Graph



Auxiliary nodes



- » Step 1. 확인할 변수의 변화를 로그에 남기도록 그래프 구성
 - » Step 2. 로그를 기반으로 텐서보드 실행하여 확인
-
- » Colab 실습시, 마지막에 표시되는 ngrok 링크를 통해 텐서보드 확인



▼ 텐서보드 화면보기

구글 Colab의 6006 포트에 직접 접근할 방법은 없어서, ngrok을 통해 원격자원으로 연결함.

해당주소를 클릭해서 바로 텐서보드에 접근 가능.

```
[4] get_ipython().system_raw('./ngrok http 6006 &')
! curl -s http://localhost:4040/api/tunnels | python3 -c #
  "import sys, json; print(json.load(sys.stdin)['tunnels'][0]['public_url'])"
```

☞ <https://b3cc46da.ngrok.io>

» Step 1. 확인할 변수의 변화를 로그에 남기도록 그래프 구성

```
a = tf.constant(3.0, name="a")
b = tf.constant(4.0, name="b")
c = tf.add(a, b, name="addnode")

# step 1: scalars 기록할 node 선택
tf.summary.scalar('add_result_c', c)
```



» Step 2. 로그를 기반으로 텐서보드 실행하여 확인

- 실행에 필수적인 옵션은 FileWriter의 첫 번째 파라미터로 넣어주었던 파일저장 경로

```
writer = tf.summary.FileWriter('/tmp/tensorboard/addnum')
```

텐서보드 내PC에서 실행하기

- » 커멘드 창에서 텐서보드 실행하면,

```
tensorboard --logdir=/tmp/tensorboard/addnum
```

- » 웹브라우저로 접속할 주소를 표시함 (자신의 IP 주소 + 6006번포트)
- » Ex) <http://0.0.0.0:6006/>

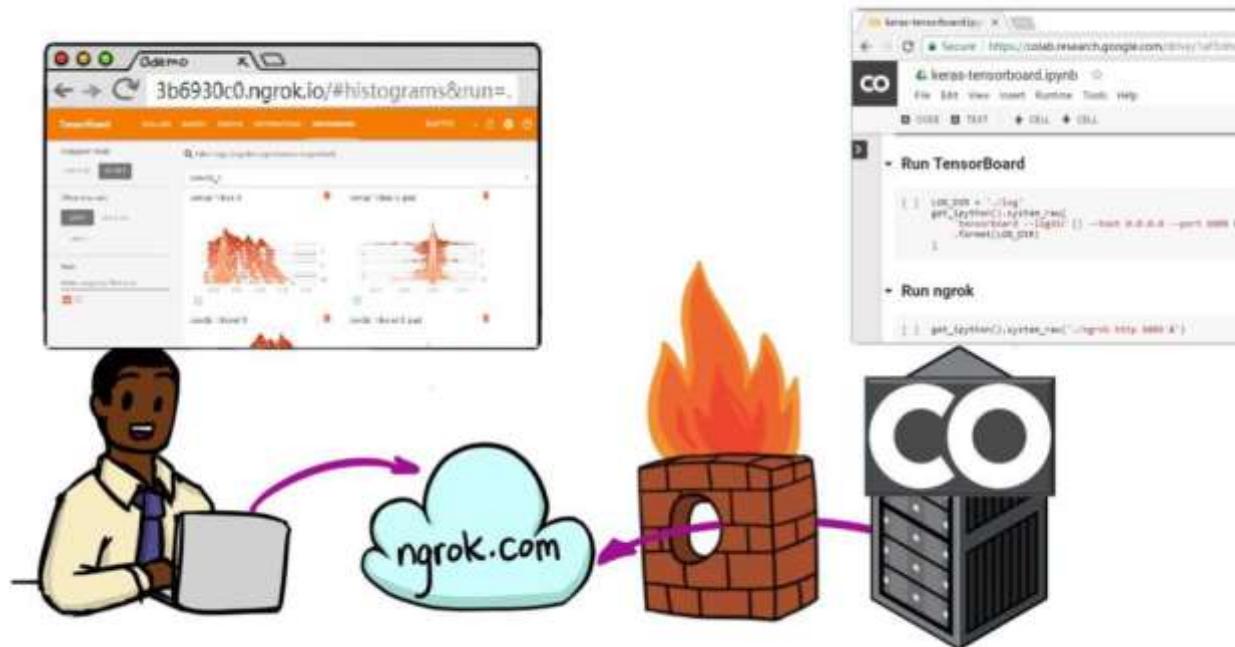
```
(tensorflowtest) K:\#>tensorboard --logdir=/tmp/tensorboard/addnum
TensorBoard 1.8.0 at http://JY-PC:6006 (Press CTRL+C to quit)
```

- » 6006번 포트의 유래(기억하기 쉽도록 구글의 스펠링 GooG 와 비슷한 모양)



sli.do # — 어떻게 Colab실습시 원격서버에 접근했을까?

- » Ngrok을 Colab서버에 설치한 후,
- » Colab서버의 6006번 트래픽을 Ngrok 사이트로 전달



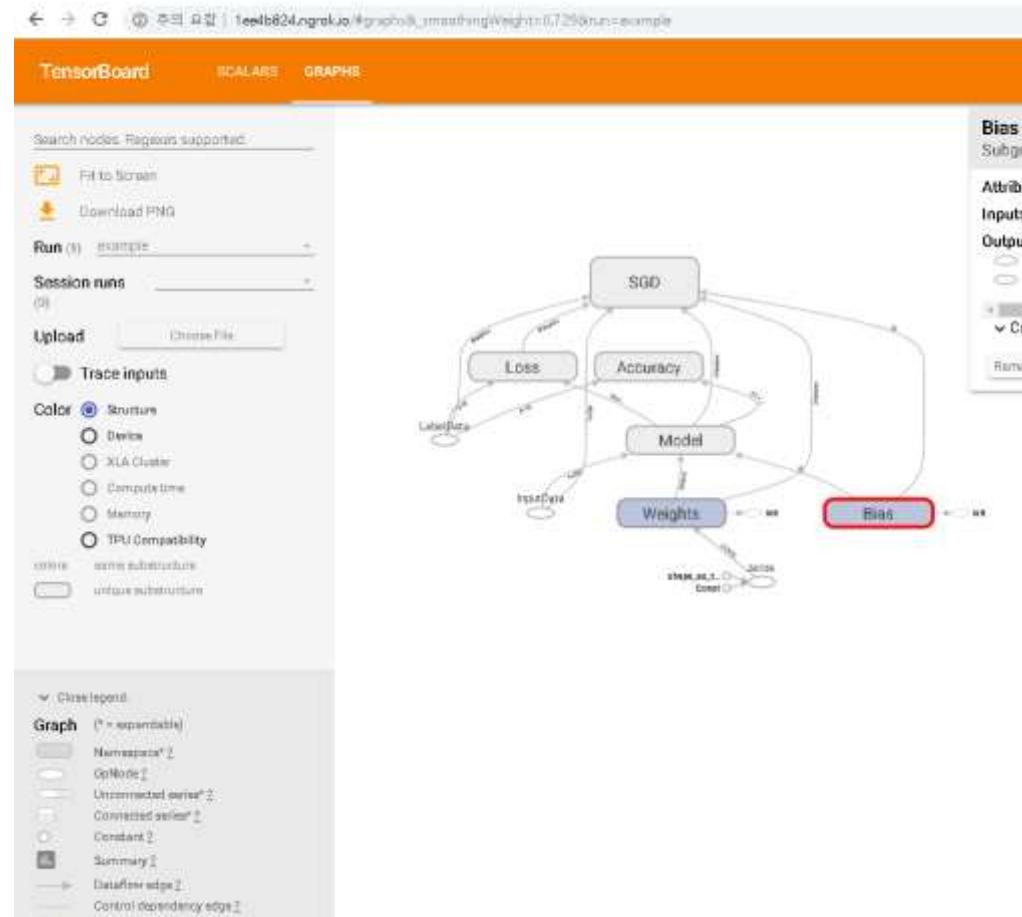
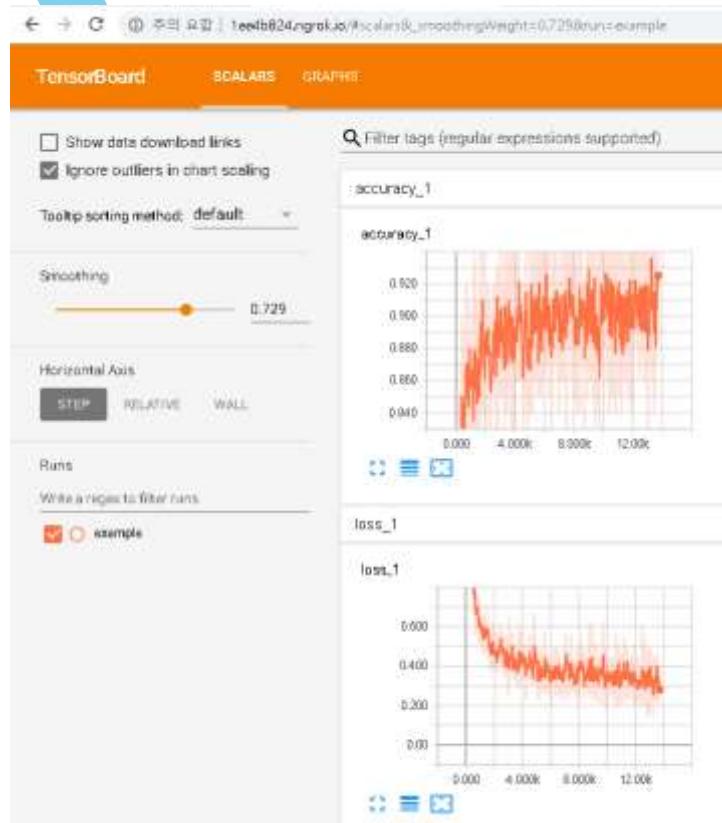
Tensor Board Graph의 범례(Legend)

» 동그라미와 화살표로 이루어 진 것 같아 보이지만 자세히 살펴보면 약간씩 다른 모양

Symbol	의미
	High-Level 노드로 name scope만 표시 중입니다. 더블-클릭하면 내용이 펼쳐집니다.
	독립적인 연산 노드
	번호가 매겨진 노드의 시퀀스. 연결되지 않음.
	번호가 매겨진 노드의 시퀀스. 서로 연결되어 있음.
	상수
	Summary 노드
	Edge. Operation간의 data flow를 보여줌
	Edge. Operation간의 control dependency를 보여줌
	Reference edge. 내보내는 텐서가 변경될 수 있음.

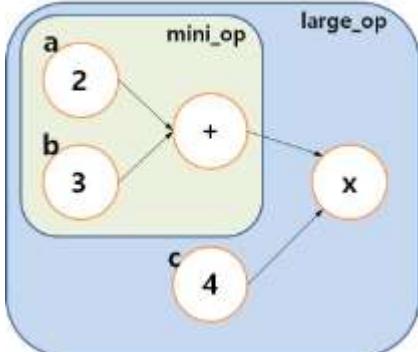
텐서보드실습(2) - MNIST 학습과정 시각화

goo.gl/vX5hdJ



Name Scope로 묶어서 표현하기 (자습)

$$(2 + 3) \times 4$$



1)

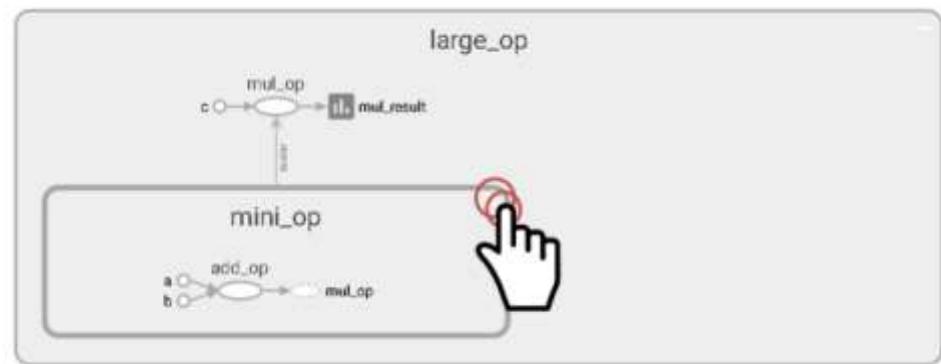
Main Graph



2)



3)



```
a = tf.constant(2.0, name="a")
b = tf.constant(3.0, name="b")
c = tf.constant(4.0, name="c")
```

```
with tf.name_scope('large_op'):
    with tf.name_scope('mini_op'):
        addop = tf.add(a, b, name="add_op")
        mulop = tf.multiply(addop, c, name="mul_op")
```

12.

학습과정을 저장하고 불러오기

- » 학습을 통해 모델을 생성하기 위해서는 많은 시간과 시스템 자원을 필요
- » 모델의 아키텍처가 복잡할수록, 학습에 사용하는 데이터가 크고 방대할수록 학습에 필요한 시간은 계속해서 늘어남
- » 학습에 걸리는 시간은 짧게는 몇 분에서 길게는 며칠 또는 몇 주
- » **[BEST CASE]** 설계한 학습 시퀀스에 따라 처음부터 끝까지 한번에 학습이 실행되고 결과 출력
- » **[현실]** 학습이 진행되는 긴 시간 동안 원하지 않은 일들이 발생
 - ◊ 실수로 프로그램 종료
 - ◊ 시스템 업데이트로 강제 재부팅
 - ◊ 정전으로 인해 전원이 소실
 - ◊ 원격서버의 네트워크가 끊김
- » **공들여 실행했던 모든 내용이 사라져 버림!!**

그래프는 어떻게 저장??

- » ‘그래프’ : 미리 정해진 구조의 변수(variable)로 이루어져있음
- » ‘학습한다’ : 변수들을 초기화한 후에 원하는 방향으로 반복해서 업데이트를 하는 과정
- » 그래프를 이루는 변수는 기본적으로 메모리에서 읽고 쓰기가 반복되지만 파일로 저장할 수 있고 다시 불러올 수 있다면 그래프를 저장하는 것이라 볼 수 있음.
- » 가장 편리한 방법은 tf.train.Saver 클래스를 사용

```
tf.train.Saver.save(...)  
tf.train.Saver.restore(...)
```

변수를 파일로 저장하기

- » Saver는 카운터 값을 사용하여 체크포인트 파일이름에 자동으로 번호를 매겨짐
 - ◊ 디스크가 꽉 차는 것(Disk Full)을 막기 위해서 ..
 - ◊ N개의 최신 파일만을 유지하도록 하거나,
 - ◊ 매 시간마다 체크포인트 파일을 만들 수 있음

예제 코드	체크포인트 파일이름
<code>saver.save(sess, 'my-model', global_step = 0)</code>	my-model-0
<code>saver.save(sess, 'my-model', global_step = 1000)</code>	my-model-1000

변수를 파일로 저장하기

```
# Saver를 생성합니다.  
saver = tf.train.Saver()
```

```
# 그래프를 실행하여 학습을 진행, 매 1000 step마다 모델을 저장
```

```
sess = tf.Session()  
for step in xrange(1000000):  
    sess.run(..training_op..)  
    if step % 1000 == 0:  
        # 체크포인트파일이름에 step번호를 추가  
saver.save(sess, 'my-model', global_step=step)
```

- » Saver는 저장공간상에 최신 체크포인트 리스트와 함께 프로토콜 버퍼를 저장
- » 프로토콜 버퍼 : 체크포인트 개수를 관리하는데 사용, 가장 최신 체크포인트를 찾는 함수인 'latest_checkpoint()'를 위한 것

- » 변수를 파일에서 읽어와서 복구하는 과정은 저장하는 과정보다는 간단
 - ◊ 그래프의 변수를 만들고 값은 0.0으로 설정
 - ◊ 저장할 때와 같은 식으로 Saver()를 생성
 - ◊ Variable(변수) 초기화하는 부분 생략 (파일읽어오기 = 변수를 최신값으로 업데이트이므로)

```
# 파일이 있으면 Variable 복구
ckpt = tf.train.get_checkpoint_state('./saver_bias/')

if tf.train.checkpoint_exists(ckpt.model_checkpoint_path):
    saver.restore(sess, ckpt.model_checkpoint_path)
    print("variable is restored")
```

sli.do #

TF doc 스스로 찾아보기 연습

The screenshot shows a Google search results page for 'tf.train.saver'. Below it is a detailed view of the TensorFlow API documentation for the `tf.train.Saver` class.

Google Search Results:

- Query:** tf.train.saver
- Results:** 약 586,000개 (0.33초)
- Top Result:** [tf.train.Saver | TensorFlow](https://www.tensorflow.org/api_docs/python/tf/train/Saver)

TensorFlow API Documentation (tf.train.Saver):

- API Version:** API v1.12
- Language:** PYTHON
- Code Snippet:**

```
__init__(  
    var_list=None,  
    reshape=False,  
    sharded=False,  
    max_to_keep=5,  
    keep_checkpoint_every_n_hours=10000.0,  
    name=None,  
    restore_sequentially=False,  
    saver_def=None,  
    builder=None,  
    defer_build=False,
```
- Note:** 별다른 설정 없으면 5개만 유지

» 변수하나 Save

```
[ ] import tensorflow as tf

#그래프에서 사용하는 변수
b1= tf.Variable(2.0, name="bias")

#변수사용 준비
init = tf.global_variables_initializer()

# Saver 생성
saver = tf.train.Saver()

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

# 학습과정을 간단히 print로 대체
print("save test bias", sess.run(b1))

# 디스크에 변수를 저장
save_path = saver.save(sess, "./saver_bias/bias.ckpt")
print("Model saved in file: %s" % save_path)
```

↳ save test bias 2.0
Model saved in file: ./saver_bias/bias.ckpt

```
[ ] !ls ./saver_bias/ -al
total 36
drwxr-xr-x 2 root root 4096 Nov 11 02:43 .
drwxr-xr-x 1 root root 4096 Nov 11 02:42 ..
-rw-r--r-- 1 root root 24 Nov 11 02:43 bias.ckpt.data-00000-of-00001
-rw-r--r-- 1 root root 206 Nov 11 02:43 bias.ckpt.index
-rw-r--r-- 1 root root 14607 Nov 11 02:43 bias.ckpt.meta
-rw-r--r-- 1 root root 75 Nov 11 02:43 checkpoint
```

» 변수하나 Restore

Restore 하기

현재는 그래프가 위에 코드에 의해 계속 실행중이므로 그래프를 다시 시작해야 합니다. (실행환경을 깼다켠것처럼)**

Colab 메뉴 -> 런타임 -> 런타임 다시시작



```
import tensorflow as tf  
  
b1= tf.Variable(0.0, name="bias")  
  
# Saver 생성  
saver = tf.train.Saver()  
  
# 세션준비, init을 하지 않습니다.  
sess= tf.Session()  
  
# 파일이 있으면 Variable 복구  
ckpt = tf.train.get_checkpoint_state('./saver_bias/')  
if tf.train.checkpoint_exists(ckpt.model_checkpoint_path):  
    saver.restore(sess, ckpt.model_checkpoint_path)  
    print("variable is restored")  
  
# 복구된 내용 확인  
print("bias:", sess.run(b1))
```

▶ INFO:tensorflow:Restoring parameters from ./saver_bias/bias.ckpt
variable is restored
bias: 2.0

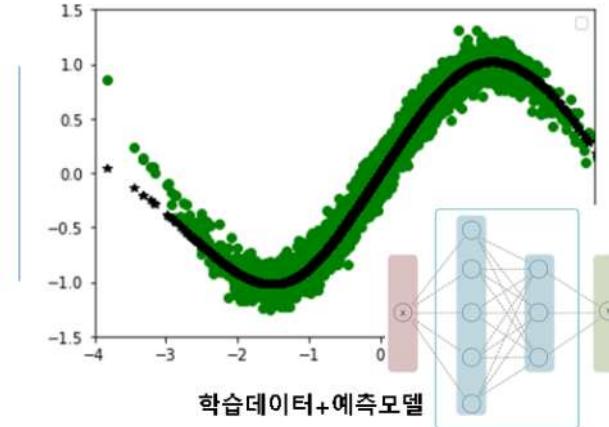
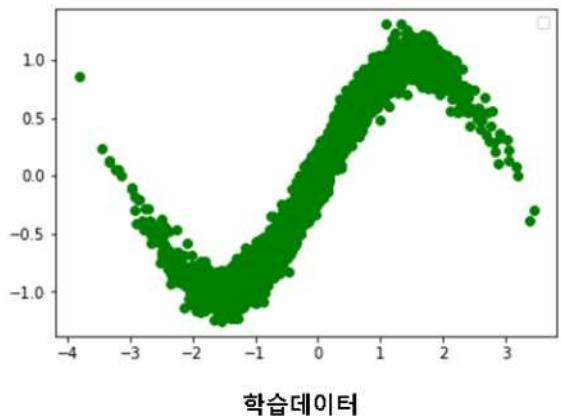
» Sin그래프 Save&Restore

<https://github.com/jaeyong1/deeplearningforus/>

→ Chap 12

→ 전체 다운로드

<https://goo.gl/Hv3YDj>



- » 앞에서 만들었던 SIN 그래프를 학습하는 뉴럴네트워크 소스코드를 다시 한번 사용
- » 예전 실습에는 몇 천 번의 반복을 해야 해서 학습이 오래 걸렸음
- » 이번에는 학습하면서 파일을 만들어 저장
- » 두 번째 실행할 때에는 오랜 기다림 없이 바로 학습된 모델을 파일에서 읽어와서 결과를 출력

LET'S REVIEW

Overview

인공지능과 머신러닝,
딥러닝의 관계.
Linear Regression
Cost function
최적화 함수
데이터셋

CNN

Imagenet 사진분류
Stride, Zero padding
LeNet과 Alex Net
VGG net, GoogLeNet
Retrain을 통한 꽃분류기

TF basic

Python 문법
Sigmoid함수
Placeholder
MNIST 실습

RNN

연속성 데이터
다양한 입력과 출력 관계
Vanishing Gradient Problem
LSTM모델
HELLO 알파벳 예측모델

Neural Network

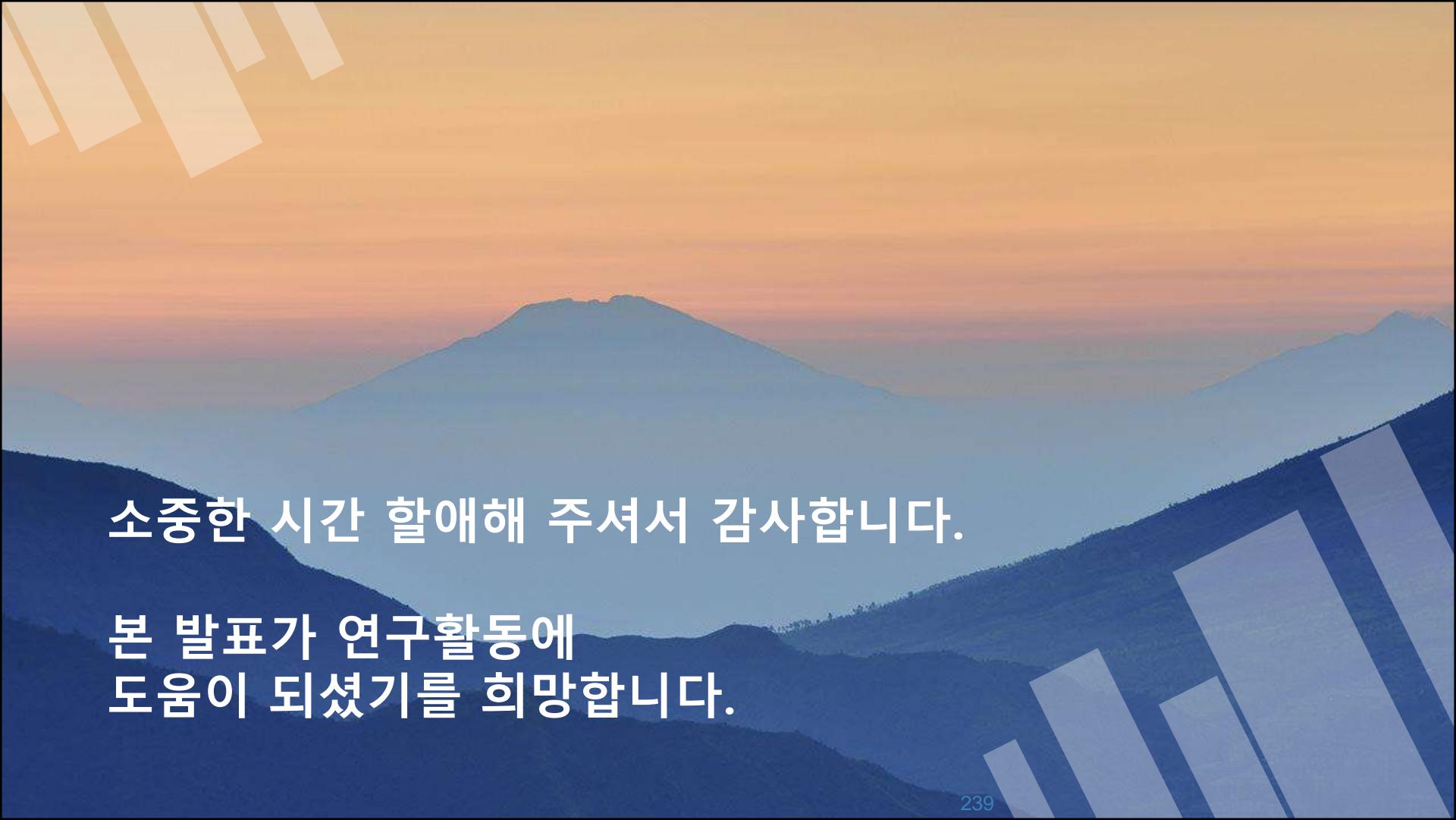
신경망소개
역전파알고리즘
Dropout
Fully connected network
SIN그래프 예측모델
비행기이륙거리 예측모델

Utility

텐서보드
그래프 저장&불러오기

Q&A

>> sli.do



소중한 시간 할애해 주셔서 감사합니다.

본 발표가 연구활동에
도움이 되셨기를 희망합니다.



THANKS!

You can find me at

» jaeyong1@naver.com [박재용]

