Let's Learn Deep Learning with TensorFlow

Wanho Choi
(wanochoi.com)

Two Artificial Heroes

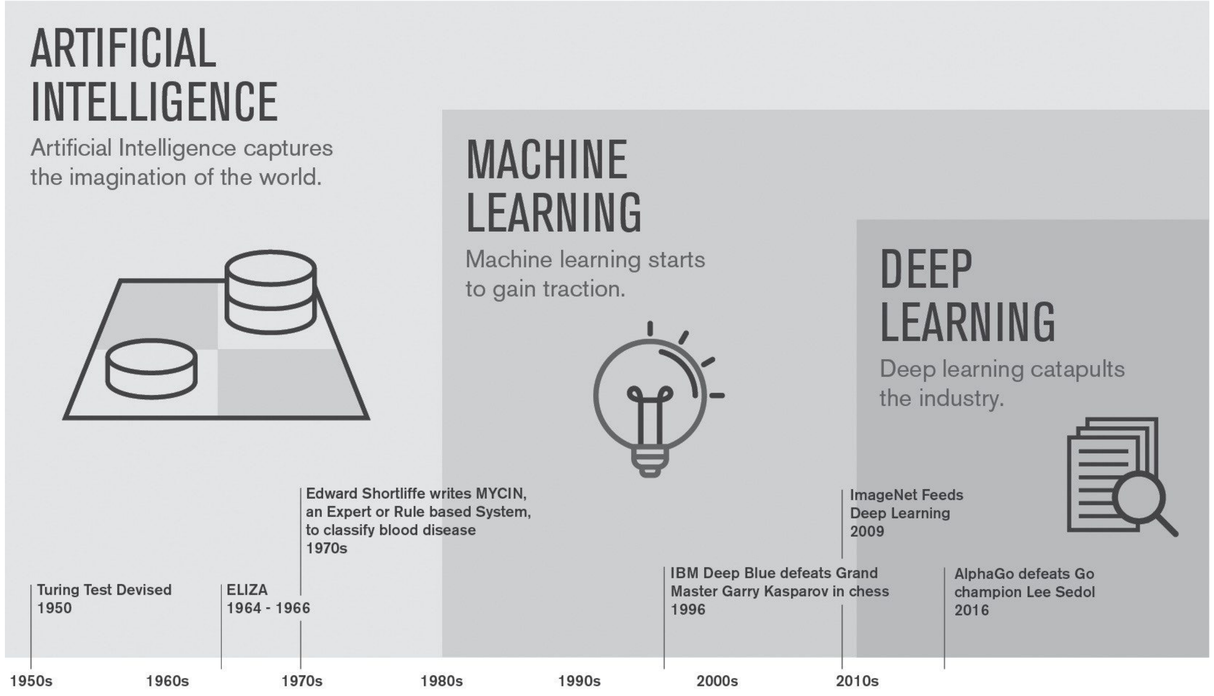
PIENSE
THINK
ΣΚΕΨΟΥ
DENKE
PENSER
Watson
$300,000
$1,000,000
$200,000
KEN
WATSON
BRAD

KBA
Korea Baduk Association
Google DeepMind
Challenge Match
8 - 15 March 2016
AlphaGo
ALPHAGO
01:58:22
LEE SEDOL
01:59:27
AlphaGo
Lee Sedol
Google DeepMind
Challenge Match
AlphaGo
Sedol
Lee

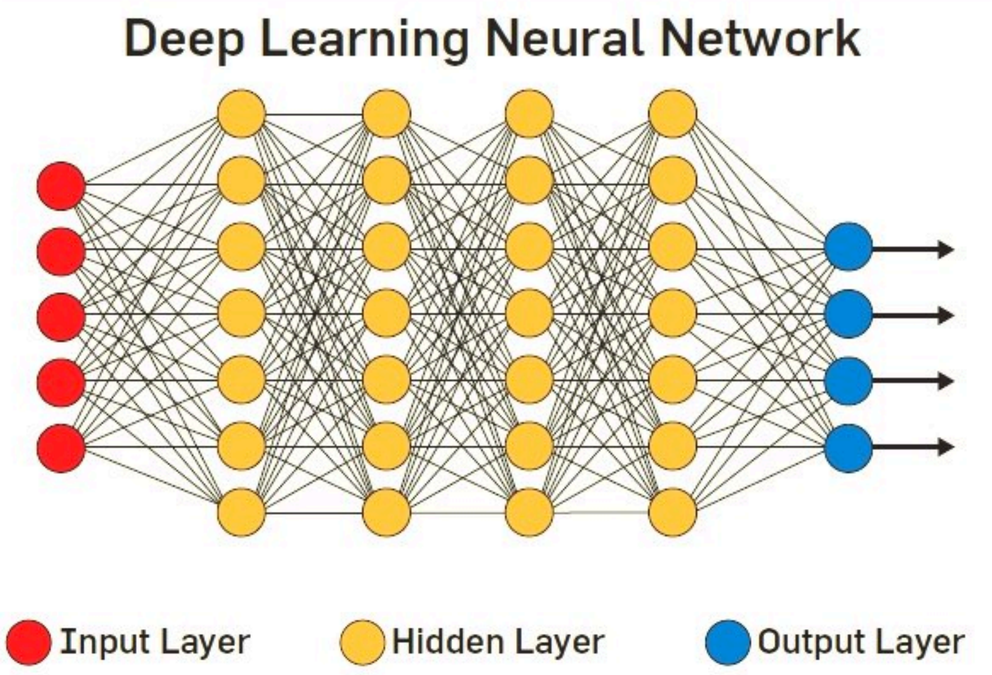
Introduction

딥러닝 (DL: Deep Learning)

- It is a class of **machine learning algorithm** that uses **multiple stacked layers** of processing units to **learn high-level representations** from structured/unstructured data.



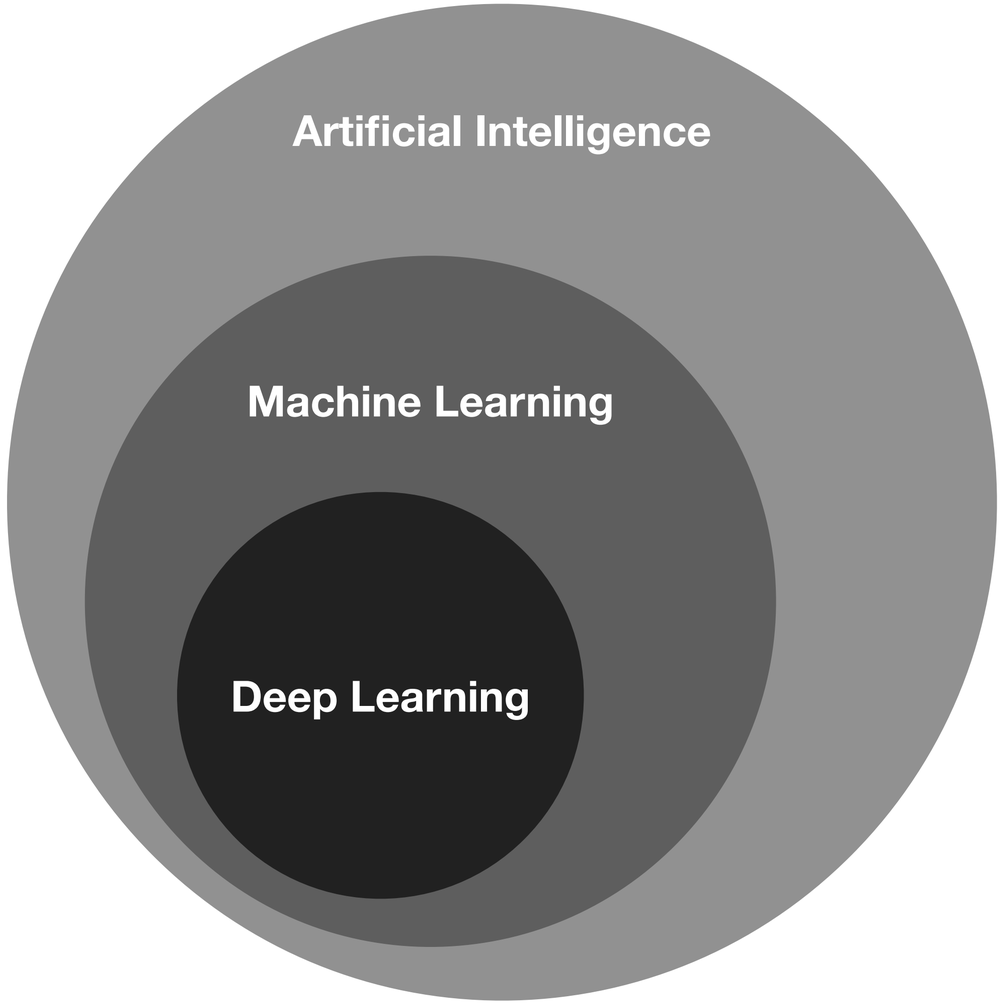
https://apttus.com/blog/artificial-intelligence-works-intelligent-enterprise/



https://cacm.acm.org/magazines/2018/6/228030-deep-learning-hunts-for-signals-among-the-noise/fulltext

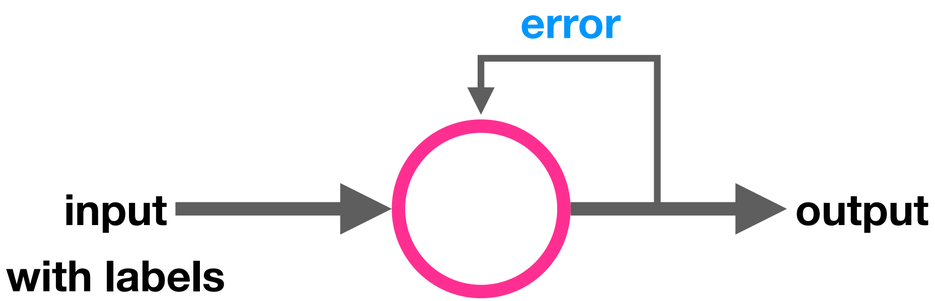
인공지능? 기계학습?

- **인공지능 (AI: Artificial Intelligence)**
  - 기계나 컴퓨터가 인간을 흉내낼 수 있도록 하는 모든 기술
- **기계학습 (ML: Machine Learning)**
  - 기계나 컴퓨터가 학습과정을 거쳐 목적하는 기능을 향해 점차 개선되도록 하는 인공지능의 한 가지 기법
- **딥러닝 (DL: Deep Learning)**
  - 다계층 인공 신경망(multi-layered neural network)을 이용하여 기계학습을 하는 기법

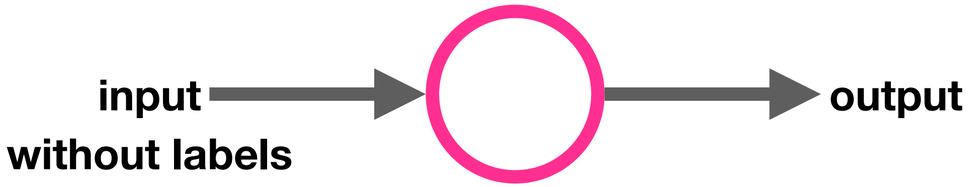


기계학습(ML)의 종류

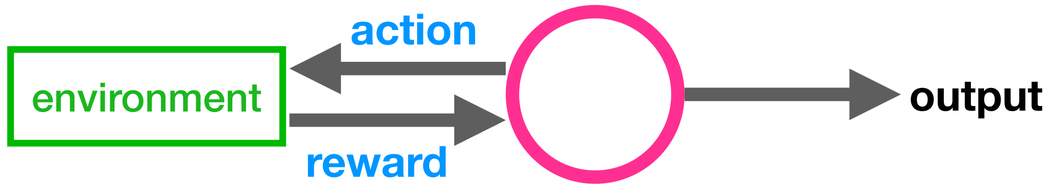
- 지도 학습 (supervised learning)



- 비지도 학습 (unsupervised learning)



- 강화 학습 (reinforcement learning)



기계학습(ML)의 종류

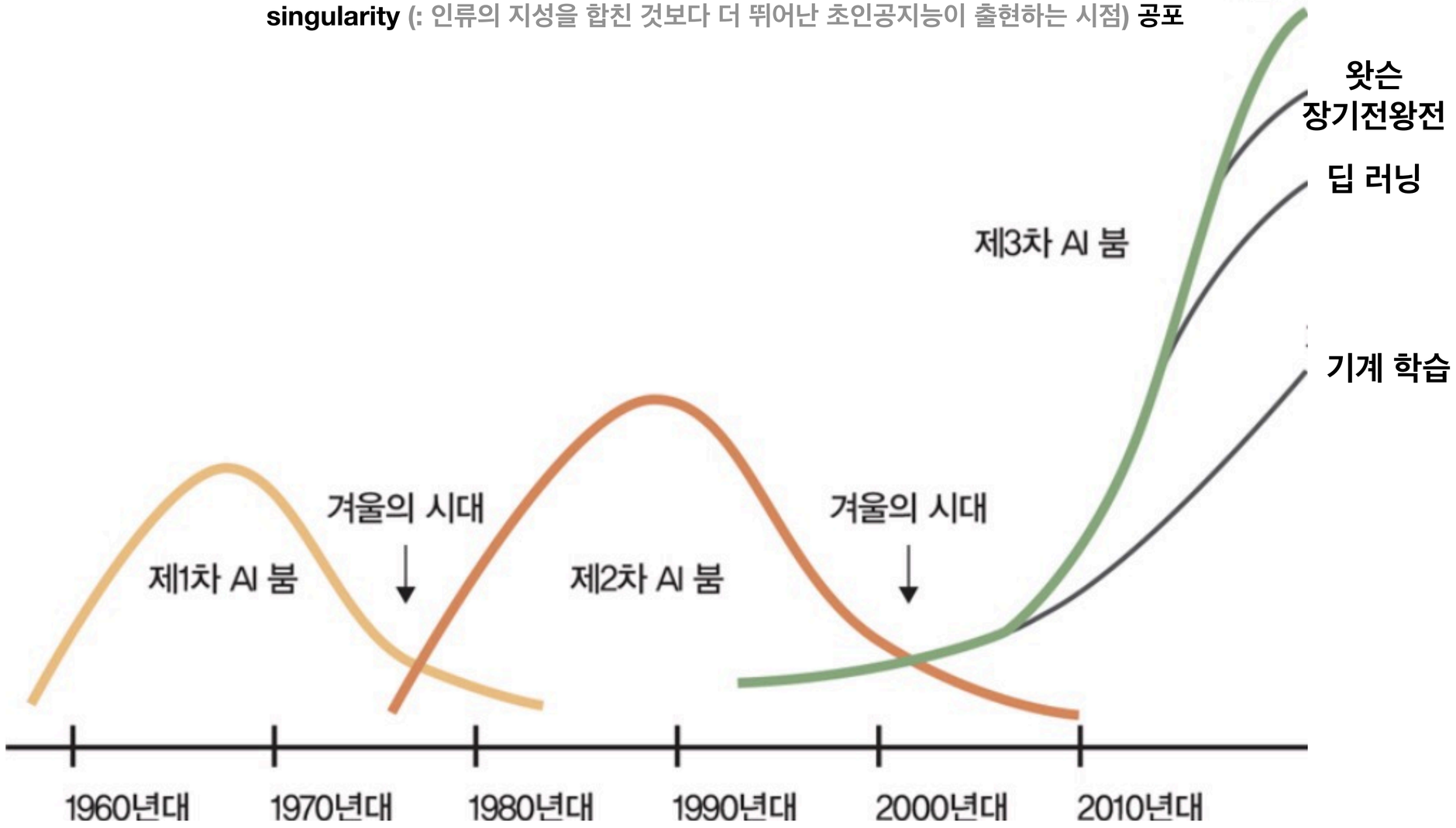
- **지도 학습 (supervised learning)**
  - 입력값(input)이 주어졌을 때 출력값(output)에 대한 정답을 알려주고 이를 하나씩 대조하며 학습함
  - 예) 회귀(regression), 분류(classification), …

- **비지도 학습 (unsupervised learning)**
  - 입력값(input)이 주어졌을 때 출력값(output)에 대한 정답을 알려주지 않아도 스스로 학습함
  - 예) 군집(clustering), 비슷한 주제의 뉴스끼리 분류하기, ...

- **강화 학습 (reinforcement learning)**
  - 데이터가 먼저 주어지는 지도 학습이나 비지도 학습과 달리 주어진 환경 속에서 스스로 데이터를 수집함
  - 이 환경 속에서 어떠한 동작(action)을 취하고 그에 따른 보상(reward)를 얻으면서 학습이 진행됨
  - 예) DeepMind의 DQN(Deep Q-network), 자율 주행, ...

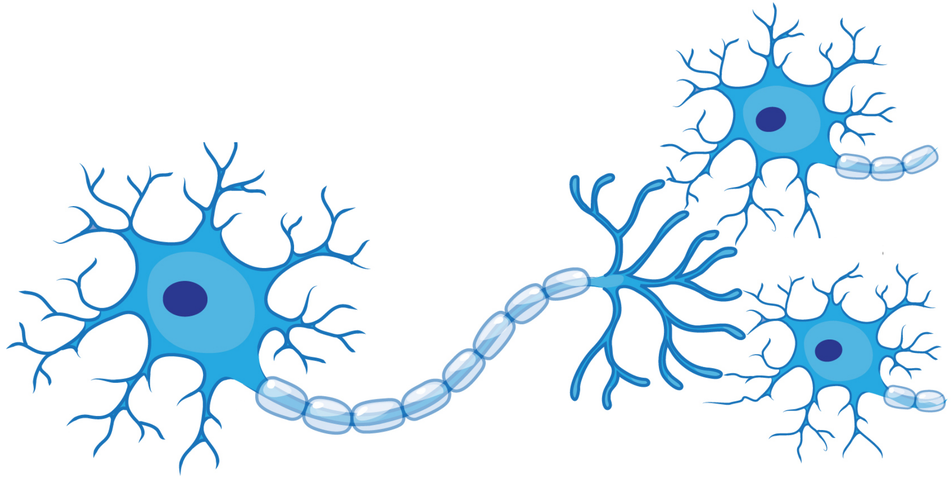
인공지능 암흑기 (AI Winter)

singularity (: 인류의 지성을 합친 것보다 더 뛰어난 초인공지능이 출현하는 시점) **공포**

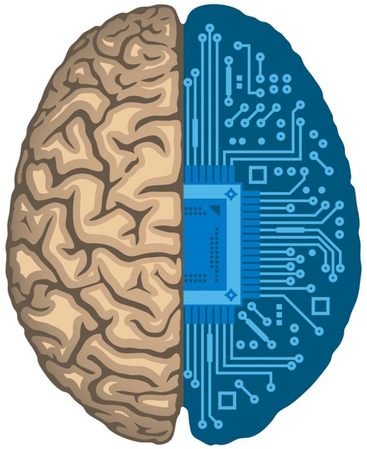


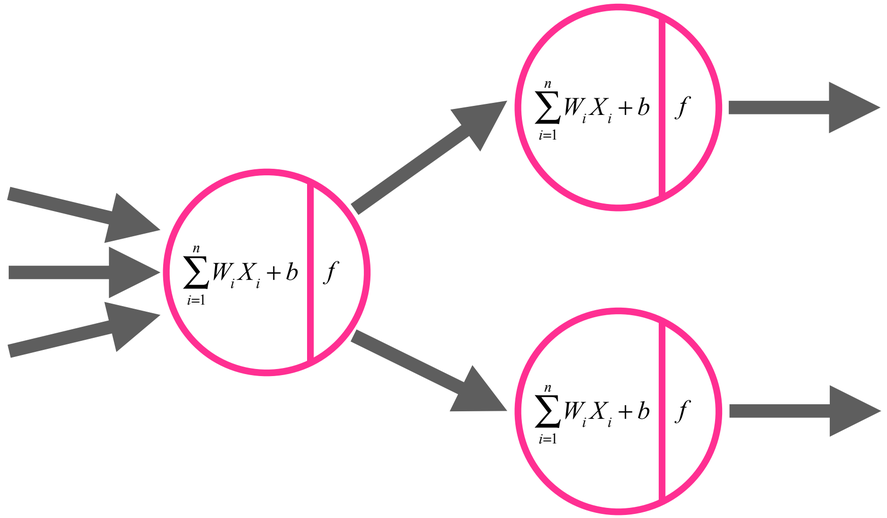
"인공지능과 딥러닝" by 마쓰오 유타카

신경망 (NN: Neural Network)



생물학적 신경망
(Biological NN)

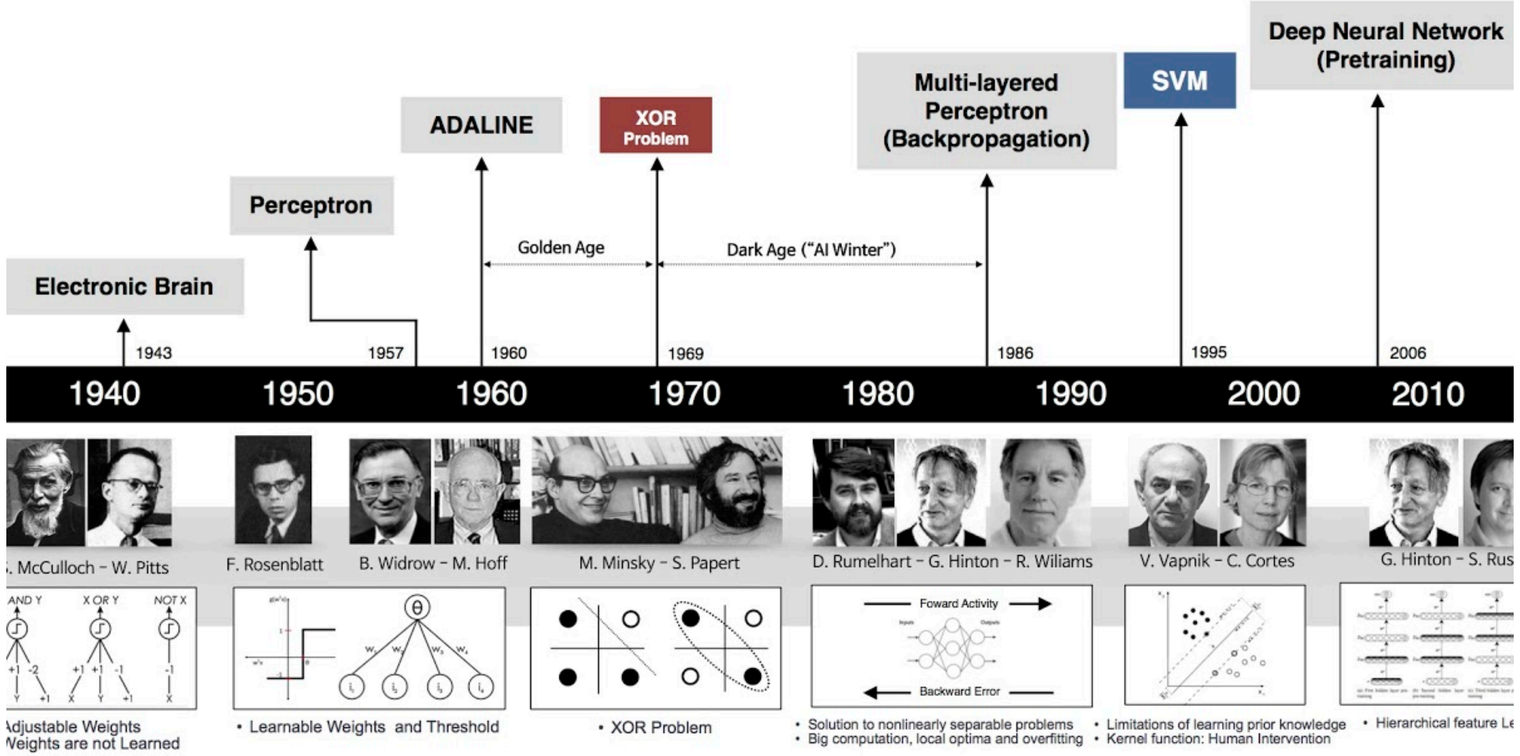




인공 신경망
(Artificial NN)

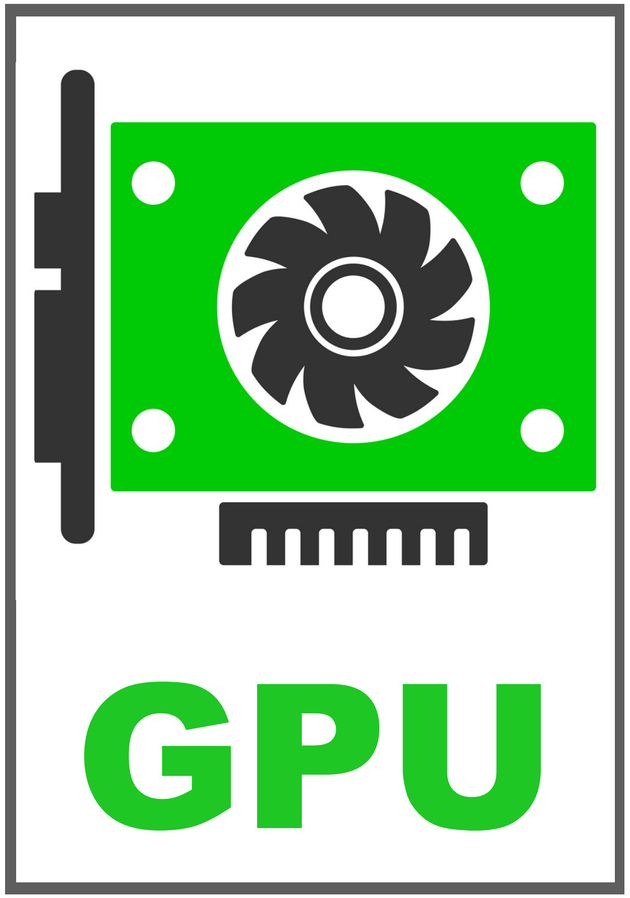
인공 신경망 (ANN)

- 인간의 **뇌(brain)** 구조를 **모방**하여 **기계 학습**을 시키는 기법
  - **인공 뉴런(artificial neurons)**을 **네트워크(network)**의 **노드(node)**로 구성
- 이미 **1940년대** 부터 연구하던 기술

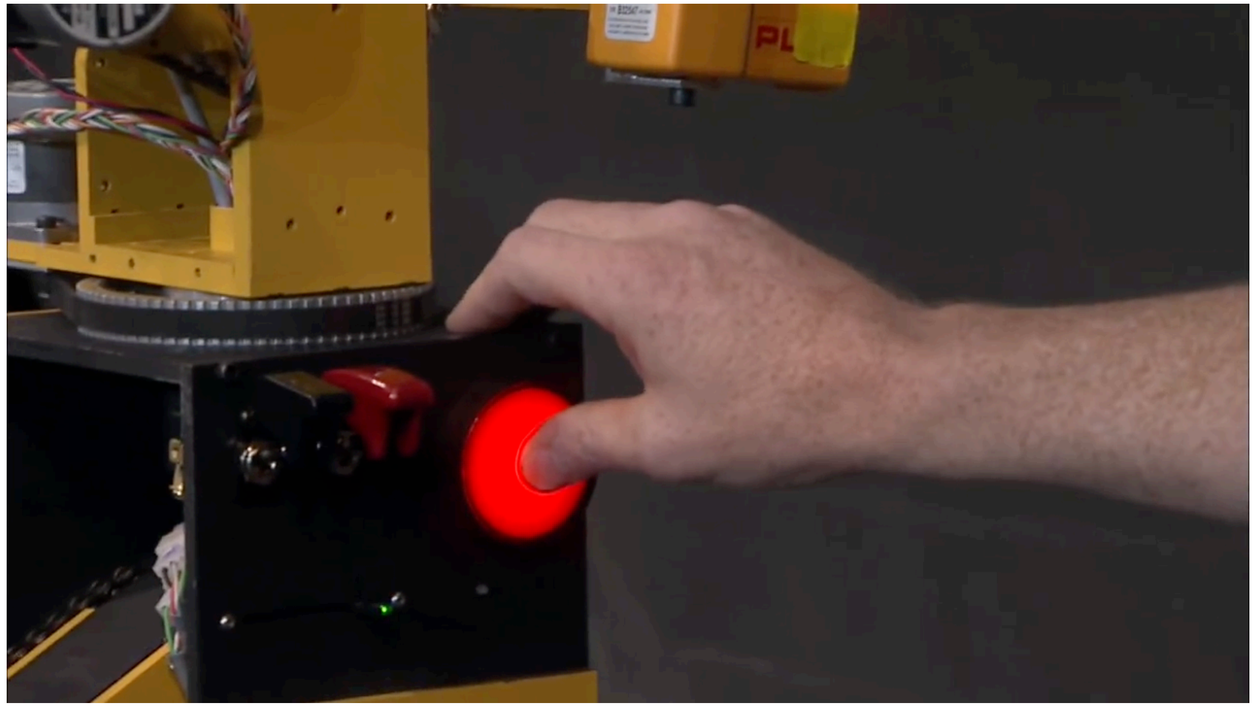


http://grjenkin.com/articles/category/machine-learning/974537/machine-learning-3-artificial-neural-networks-part-1-basics

2010년 이후에야 부상한 이유



CPU vs GPU



CPU

GPU

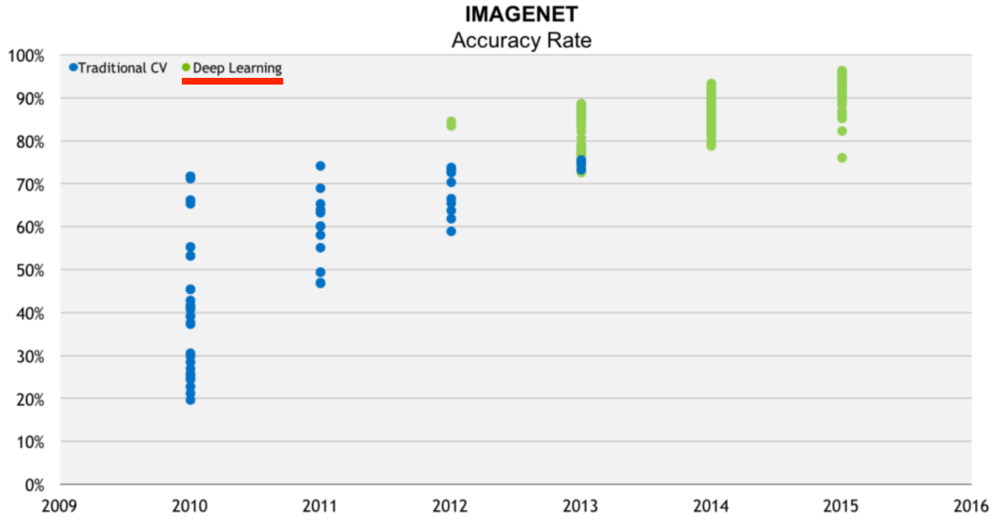
https://www.youtube.com/watch?v=-P28LKWTzrI

Image Classification Challenge

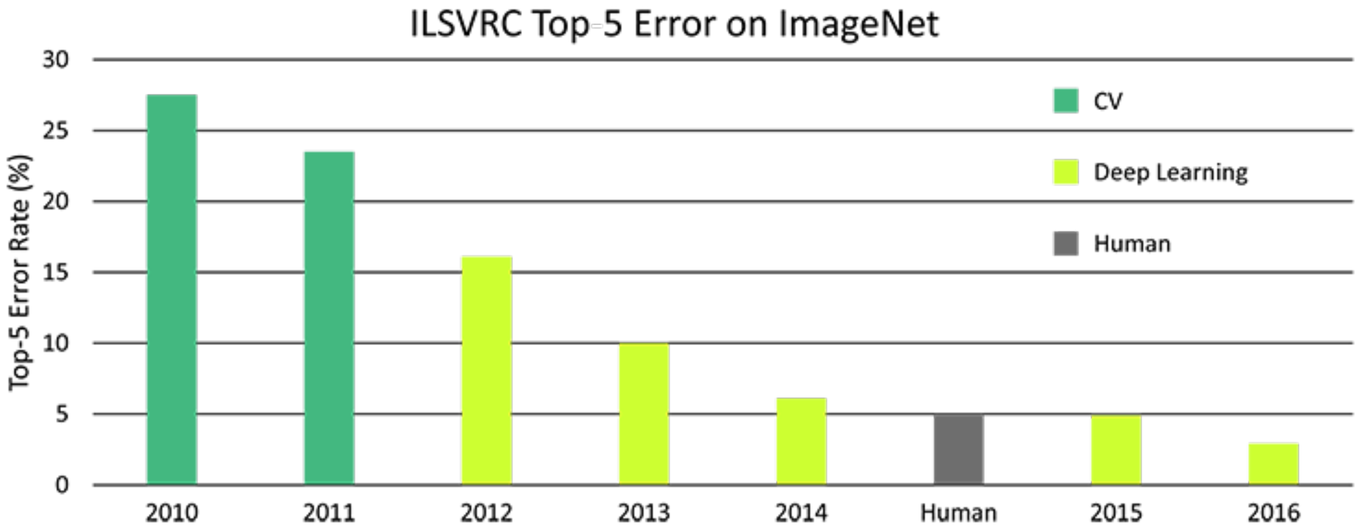
- **ImangeNet**
  - 연구 목적의 이미지 데이터베이스를 제공하는 것을 목표로하는 프로젝트
  - 1,400만개 이상의 이미지, 2만개 이상의 클래스를 제공

- **ILSVRC** (**I**mageNet **L**arge **S**cale **V**isual **R**ecognition **C**ompetition)
  - ImageNet에서 주최하는 연례 경쟁 대회 (2010년 부터 개최되고 있음)
  - 이미지 인식률로 컴퓨터 비전 알고리즘을 평가
  - 학습 데이터: 1,000 개의 클래스에 속하는 120 만 개의 이미지가있는 ImageNet의 하위 집합

Image Classification Challenge



https://blogs.nvidia.com/blog/2016/06/29/deep-learning-6/

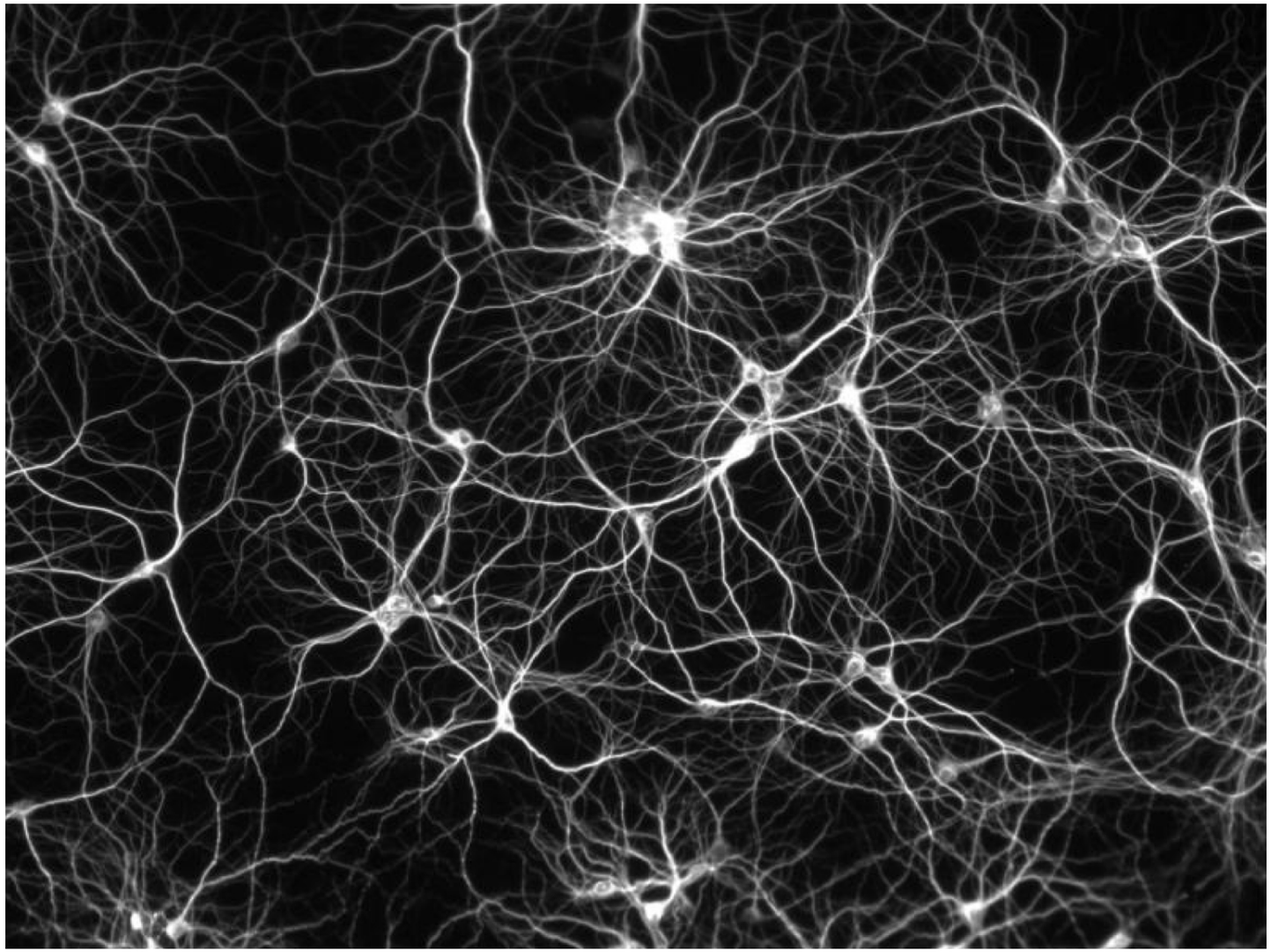


https://medium.com/nanonets/how-to-automate-surveillance-easily-with-deep-learning-4eb4fa0cd68d

인간의 뇌 (Human Brain)

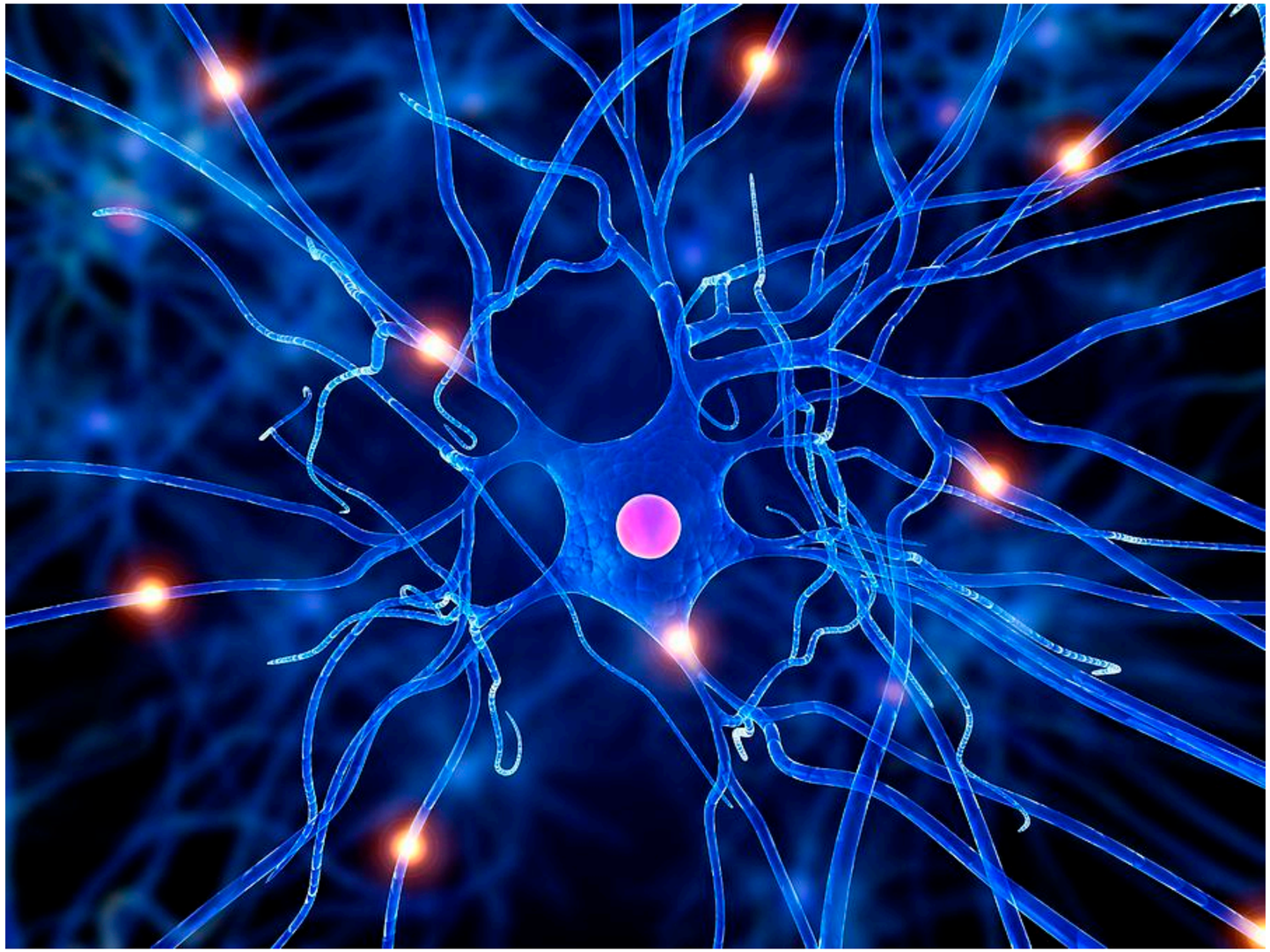
https://www.popsci.com/most-sophisticated-brain-ever-grown-lab

인간의 뇌 (Human Brain)



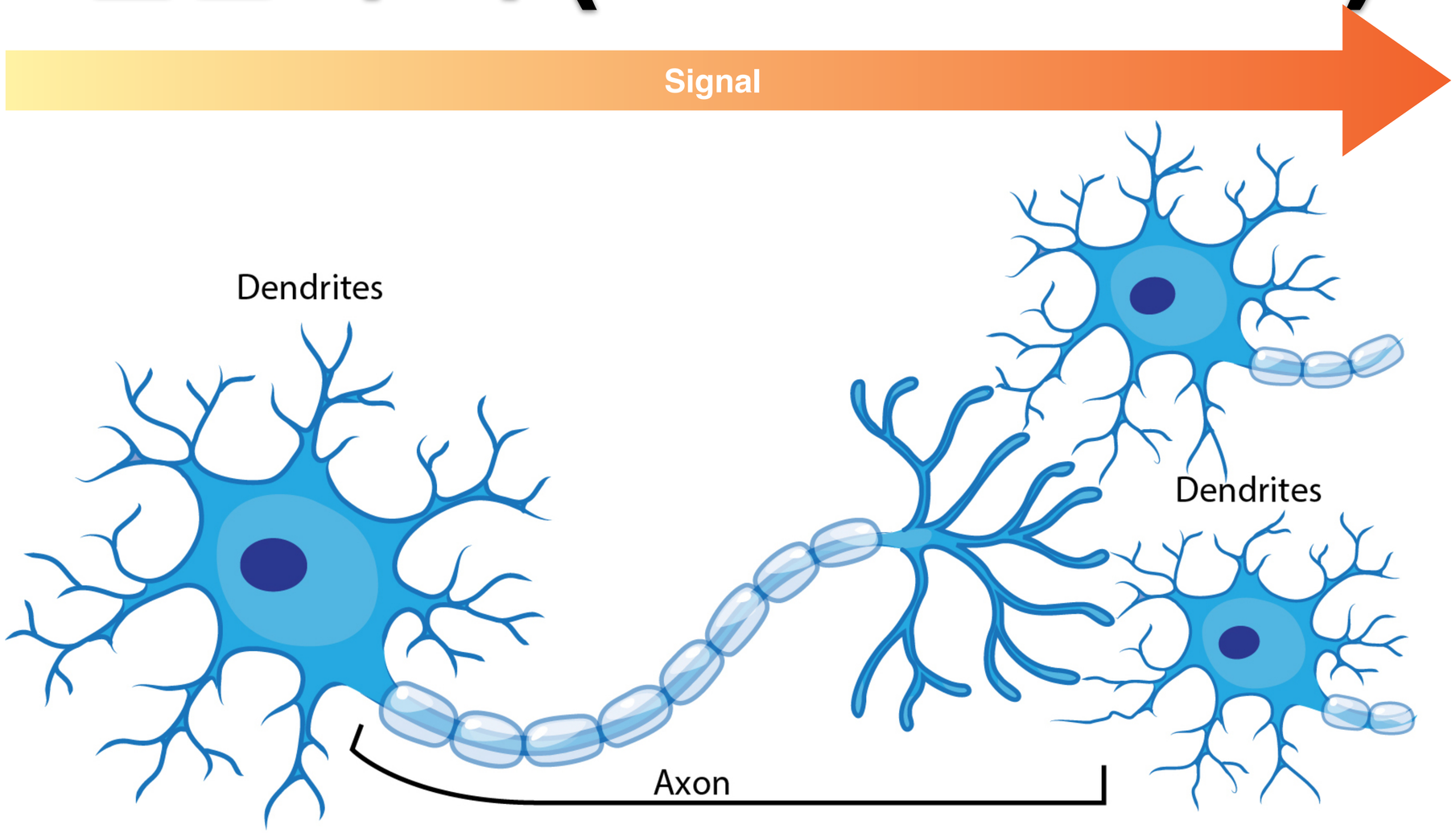
https://wccftech.com/scientists-artificial-neurons-mimics-human-brain-cells/

인간의 뇌 (Human Brain)



https://madwayoga.wordpress.com/2015/05/26/balancing-the-branches-of-your-nervous-system/

인간의 뇌 (Human Brain)

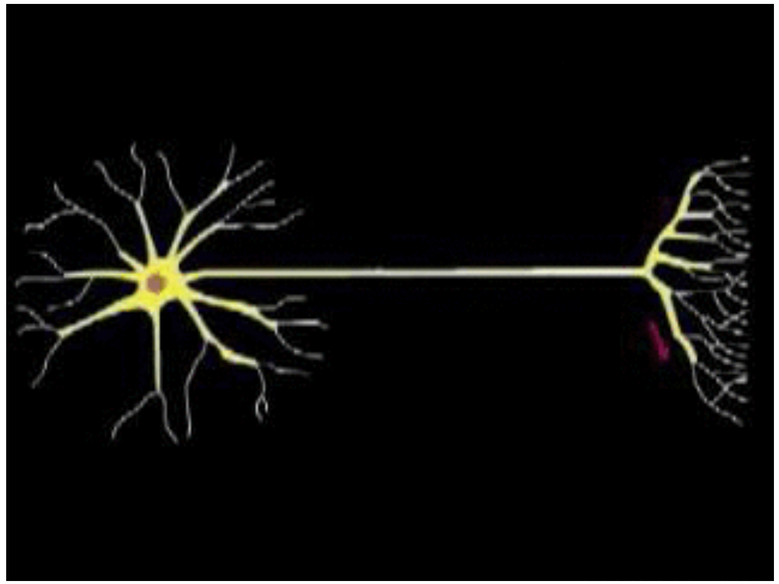


https://askabiologist.asu.edu/plosable/speed-human-brain

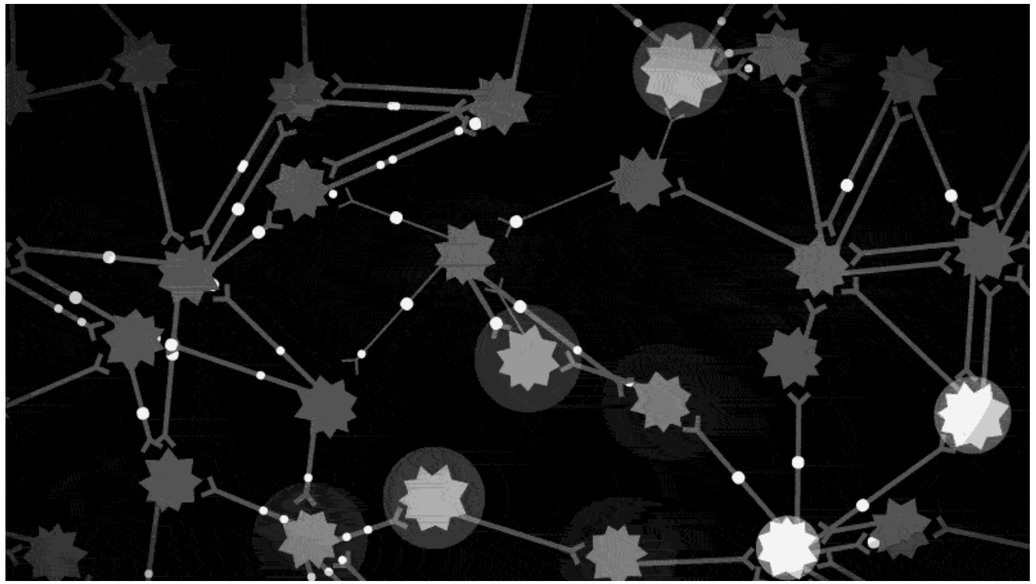
인간의 뇌는 복잡해 보이지만 단순한 정보를 처리하는 많은 신경세포들의 집합일 뿐이다.

뉴런 (Nueron)

- **전기적 신호**에 의해 반응하는 **신경 세포**
- 다른 뉴런들로 부터 **받은** 전기적 신호를 **변형**하여 또 다른 뉴런들에게로 **전달**
- 수 많은 뉴런들이 서로 연결되어 **네트워크**를 구성
- 일련의 과정을 통해 전달된 전기적 신호는 최종적으로 **중추신경계**로 도달
- 중추신경계에서 처리한 정보를 다시 우리 몸의 각 부분에 전달하여 **명령**을 **수행**



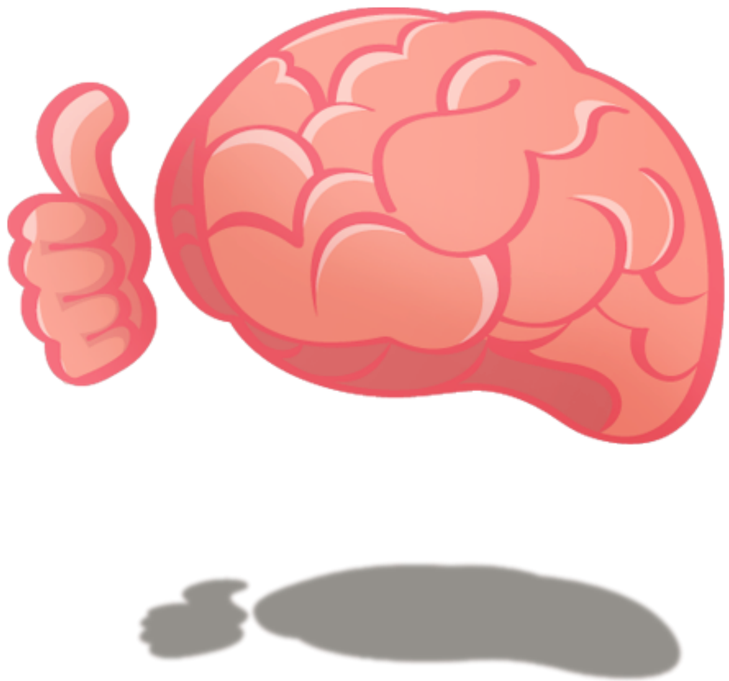
https://gfycat.com/ko/gifs/detail/AffectionateImmaterialAfricanjacana



https://gifer.com/en/7Sar

머리가 좋다? or 운동 신경이 좋다?

- **많은** 수의 **뉴런(neuron)**을 가지고 있다. **[선천적]**
- 뉴런(neuron)들이 네트워크(network)로 **잘 연결**되어 있다. **[후천적]**



머리가 좋다? or 운동 신경이 좋다?

“우리 애는 머리는 좋은데 공부를 못해.”

||

“우리 애는 뉴런은 많은데 제대로 연결이 안되어 있어.”

머리가 좋다? or 운동 신경이 좋다?

“우리 애는 머리는 좋은데 공부를 못해.”

||

“우리 애는 뉴런은 많은데 제대로 연결이 안되어 있어.”

Training!!

학습 (Learning) or 훈련 (Training)

- 가지고 있는 **뉴런**들을 잘 **조직화**하여 뇌가 **목적**하는 대로 **반응**할 수 있도록 하는 **과정**

http://www.iconsumer.or.kr/news/articleView.html?idxno=5182

인공신경망(ANN)을 이용한 학습

- 앞에서 설명했듯이..
  인공신경망은 인간 뇌 속에 있는 **뉴런**의 행동 방식에 **영감**을 받아서 고안된 **알고리즘**이다.

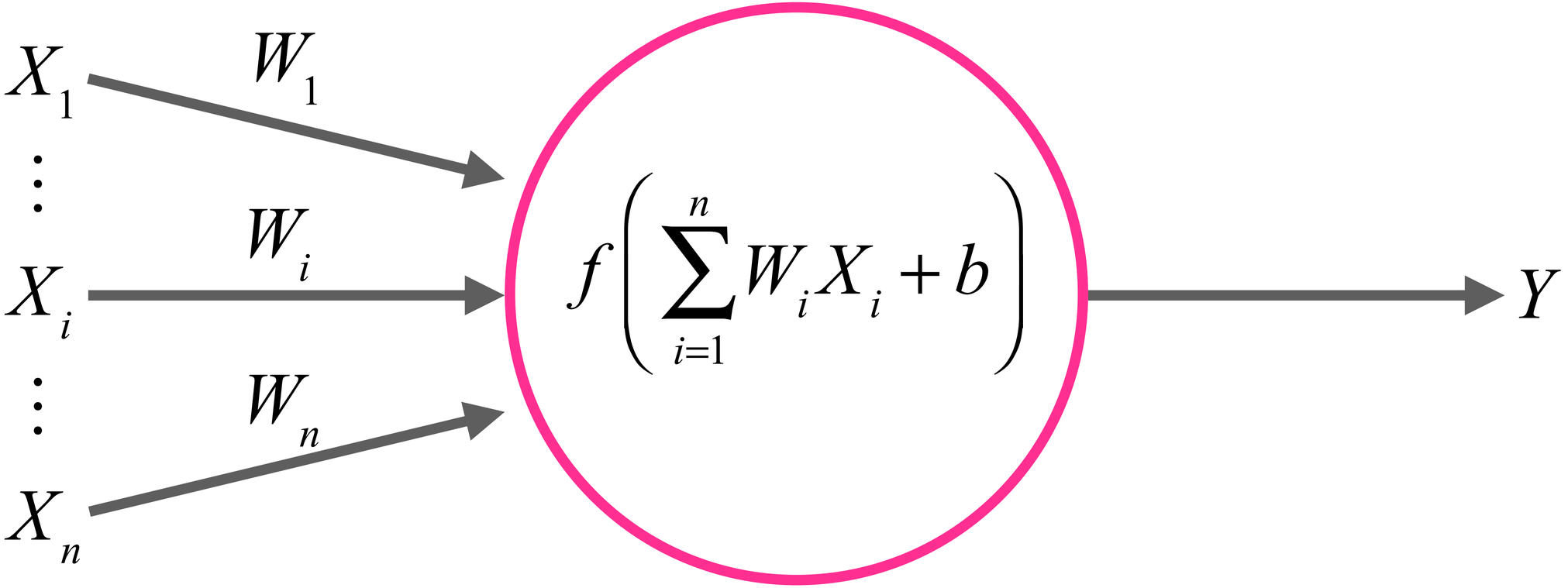
- **이전 층(layer)**에 있는 뉴런들의 출력값들을 **입력** 받아서,
  가중치(weights)를 곱하고 **편향(bias)**를 더해서 나온 결과를
  다시 **다음 층(layer)**에 있는 뉴런의 입력값으로 **전달**한다.

- 때로는 **비선형 함수(non-linear function)**를 근사(approximation)하기 위해,
  출력값(output)에 **활성화 함수(activation function)**를 적용하기도 한다.

- **딥러닝(deep learning)**은 이러한 **다수의 층들(layers)**을 사용하여 **복잡한 함수**를 **표현**할 수 있다.

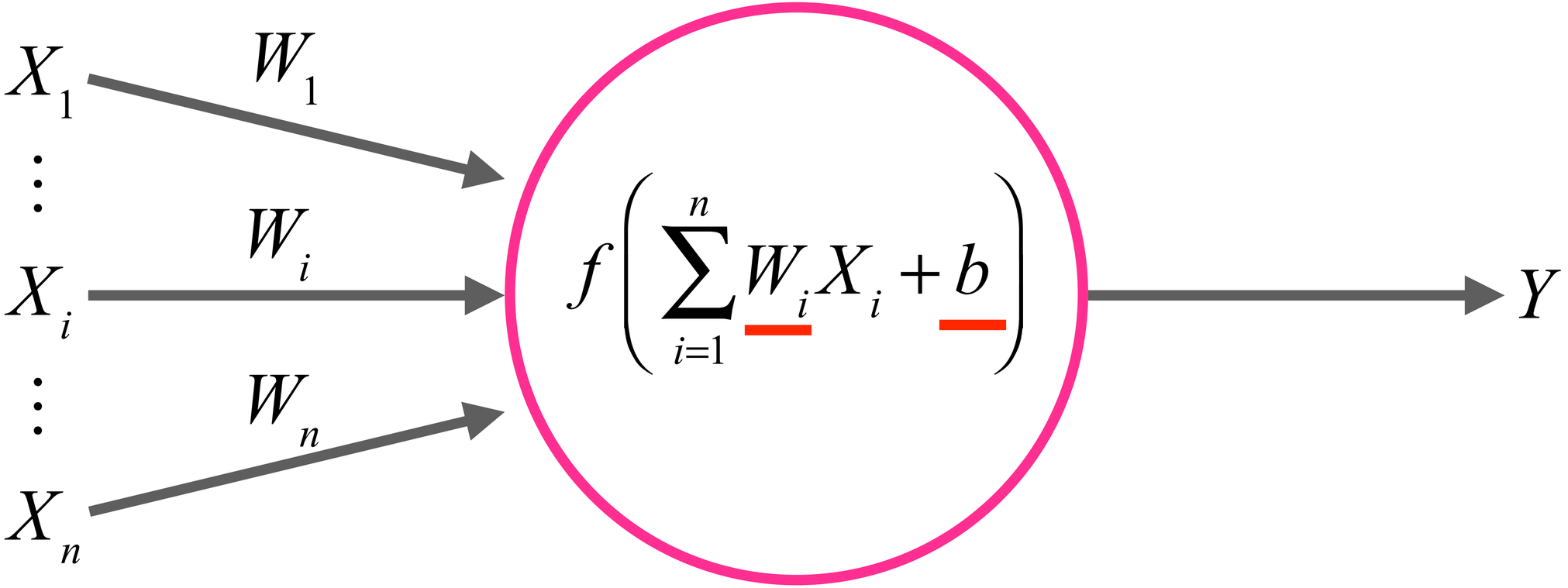
- DNN에서 **학습(training)**을 시킨다는 것은 모델의 **손실 함수(loss function)**가 **최소화(minimization)**되는 방향으로 조금씩 움직여서 **최적**의 **가중치(weights)**와 **편향(bias)**을 찾는 과정을 의미한다.

인공 뉴런 (Artificial Neuron)



학습 (Learning) or 훈련 (Training)

- W_i와 b를 정하는 과정

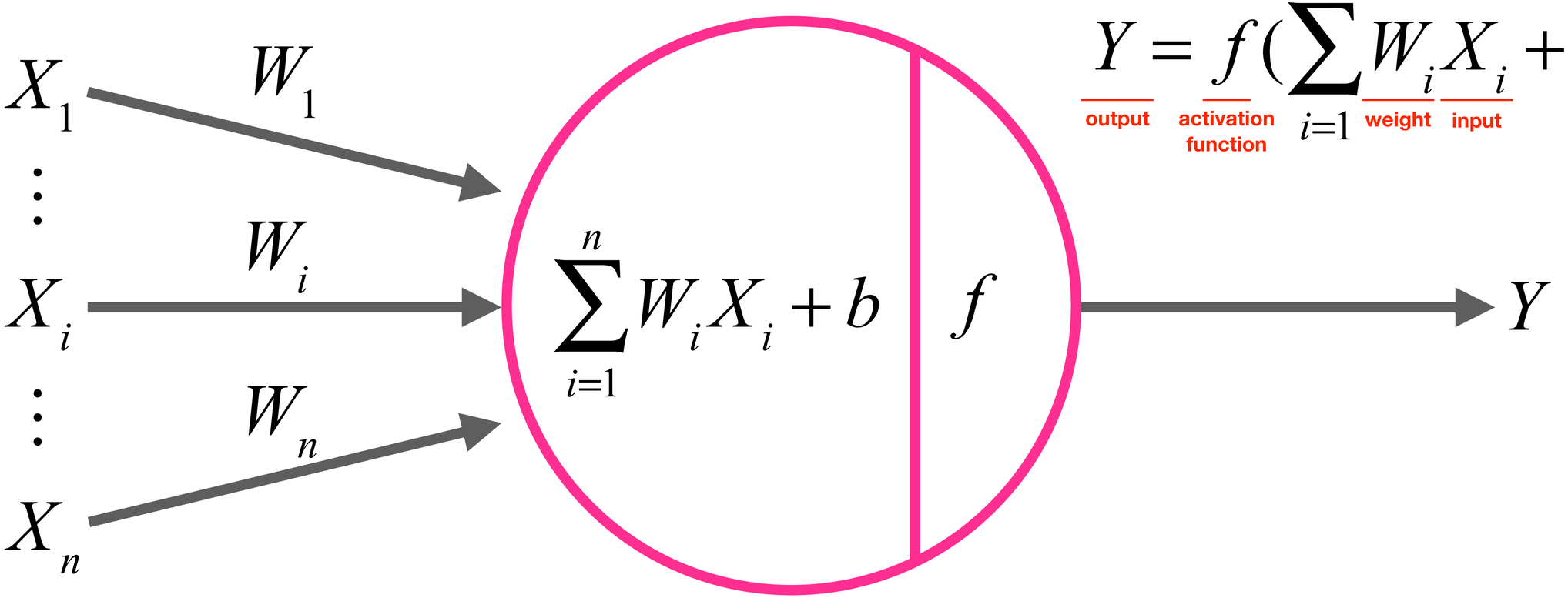


활성화 함수 (Activation Function)

- 입력신호(input)를 조작한 결과를 출력신호(output)로 **변환**하는 **함수**
- 특정 **임계치(threshold)** 이상의 값일 때에만 다음 뉴런(neuron)으로 **신호**를 **전달**하도록 하는 역할

$$\underbrace{Y}_{\text{output}} = \underbrace{f}_{\text{activation function}}\left(\sum_{i=1}^{n} \underbrace{W_i}_{\text{weight}} \underbrace{X_i}_{\text{input}} + \underbrace{b}_{\text{bias}}\right)$$

n : # of inputs

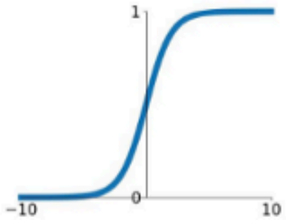


활성화 함수 (Activation Function)

- To produce **non-linearity**
- **Non-linear activation functions** are preferred
  as they allow the nodes to learn more complex structures in the data.

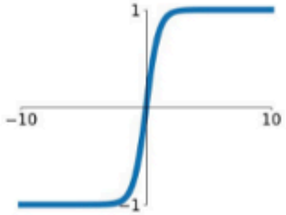
Sigmoid

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



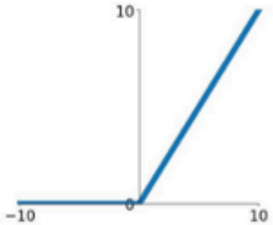
tanh

$$\tanh(x)$$



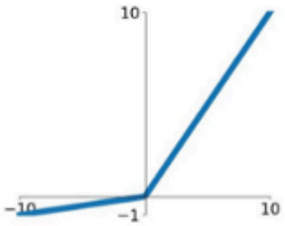
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

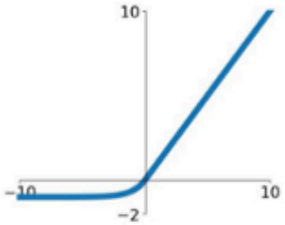


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



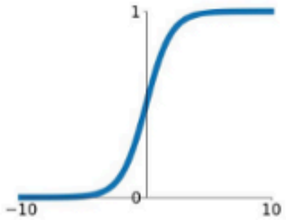
https://mc.ai/complete-guide-of-activation-functions/

활성화 함수 (Activation Function)

- To produce **non-linearity**
- **Non-linear activation functions** are preferred
  as they allow the nodes to learn more complex structures in the data.

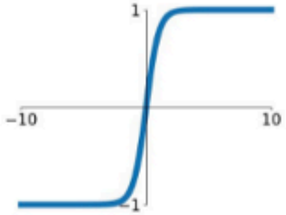
Sigmoid

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



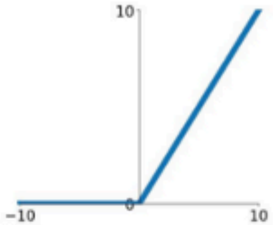
tanh

$$\tanh(x)$$



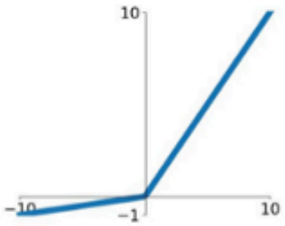
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

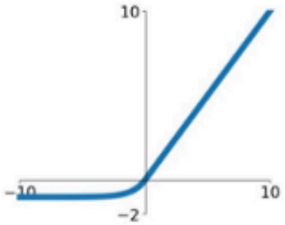


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



https://mc.ai/complete-guide-of-activation-functions/

Rectified Linear Unit (ReLU)

- It is a **piecewise linear** function.
- It will **output** the input **directly** if it is **positive**, otherwise, it will output **zero**.
- It has become the **default activation function** for many types of neural networks because it is easier to train and often achieve **better performance**.
- The **sigmoid** and **hyperbolic tangent** activation functions cannot be used in networks with many layers due to the **vanishing gradient problem**.
- ReLU **overcomes** the vanishing gradient problem.
- ReLU helps to train the network **much faster** (due to its **non-saturating** nature) and made it possible to train a **much deeper** network in an end-to-end fashion.

왜 비선형 활성화 함수를 사용해야 하는가?

- A **combination** of **linear** functions in a linear manner is still another **linear** function.

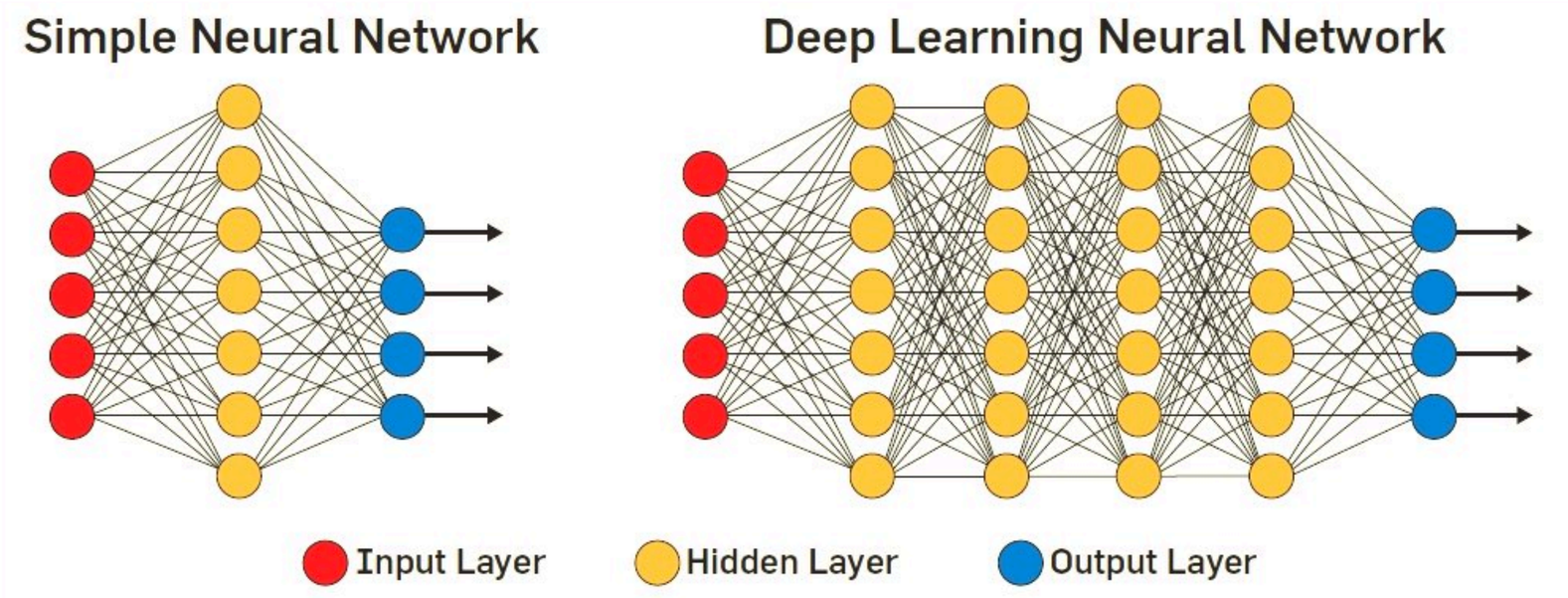
$$f(x) = \alpha x$$
$$g(x) = \beta x$$

$$\Rightarrow \quad g(f(x)) = \beta(\alpha x) = (\alpha\beta)x = \gamma x$$

- **Non-linearity** is the powerhouse of neural networks.

- Without it, neural networks lack **expressiveness**
  and simply boil down to a **linear transformation** of input data.

심층 신경망 (Deep Neural Network)



https://cacm.acm.org/magazines/2018/6/228030-deep-learning-hunts-for-signals-among-the-noise/fulltext

Machine Learning Process

- **Data munging** (or **wrangling**)

  input data: $X = \{x_1, x_2, \ldots, x_n\}$

  output data: $Y = \{y_1, y_2, \ldots, y_n\}$

  $\Rightarrow D = \{(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)\}$

- **Model definition**

  output: $Y_\theta = f_\theta(X)$

  parameter set: $\theta = \{\theta_1, \theta_2, \ldots, \theta_m\}$

  loss: $L(Y, Y_\theta)$

- **Learning** (or **training**)

  $$\theta^* = \underset{\theta}{argmin}\, L(Y, Y_\theta)$$

- **Testing** (or **prediction**)

  $$Y_{new} = f_{\theta^*}(X_{new})$$

https://www.slideshare.net/NaverEngineering/ss-96581209

Deep Learning Process

- **Data munging** (or **wrangling**)

input data: $X = \{x_1, x_2, \ldots, x_n\}$

output data: $Y = \{y_1, y_2, \ldots, y_n\}$

$$D = \{(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)\}$$

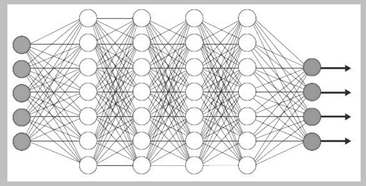
- **Model definition**

output: $Y_\theta = f_\theta(X)$

parameter set: $\theta = \{\theta_1, \theta_2, \ldots, \theta_m\}$

loss: $L(Y, Y_\theta)$

Deep Neural Network



$$Y_\theta = f_\theta(X)$$

$$\theta = \{(W_1, b_1), (W_2, b_2), \ldots, (W_l, b_l)\}$$

- **Learning** (or **training**)

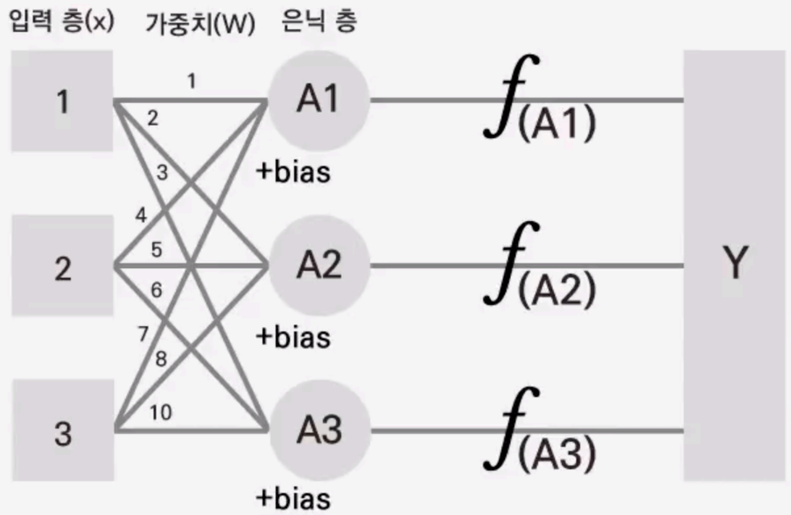
$$\theta^* = \underset{\theta}{argmin}\, L(Y, Y_\theta)$$

- **Testing** (or **prediction**)

$$Y_{new} = f_{\theta^*}(X_{new})$$

https://www.slideshare.net/NaverEngineering/ss-96581209

ANN = Matrix Multiplication



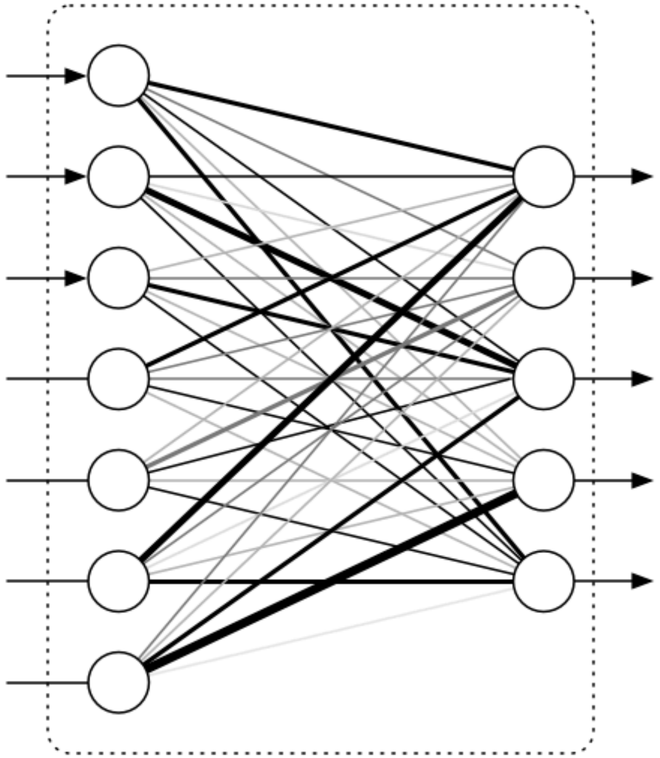
1. 입력층의 값을 가져온다. x
2. 입력층과 가중치를 곱한다. Wx
3. 편향치(Bias)를 더한다. +bias
4. 활성화 함수를 적용한다. $f(x) = x$

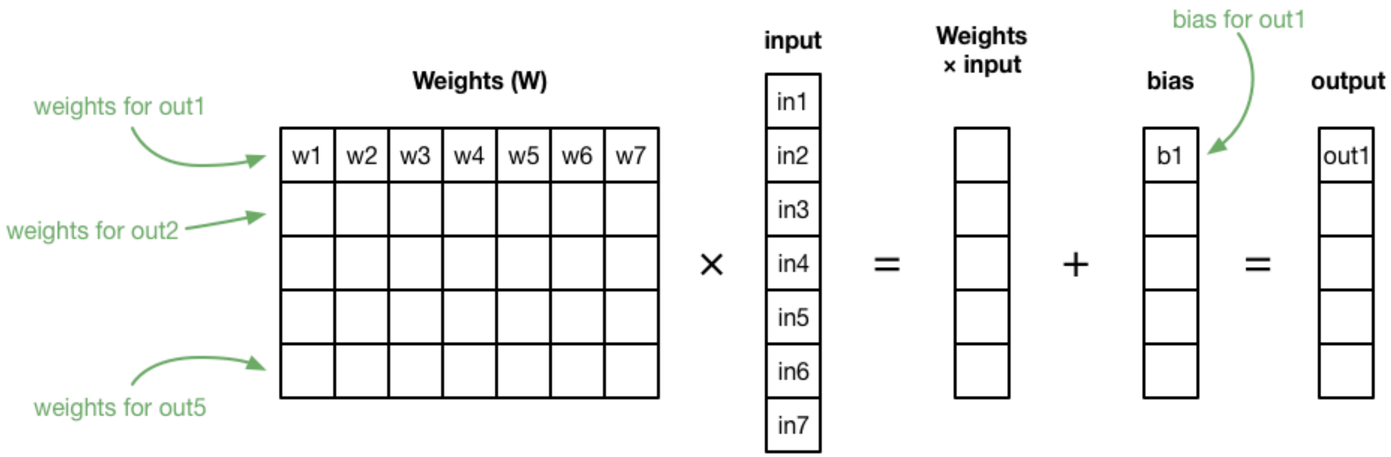
$$\overset{W}{\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 10 \end{bmatrix}} \times \overset{x}{\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}} + \overset{bias}{\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}} = \begin{bmatrix} 31 \\ 37 \\ 46 \end{bmatrix} \Rightarrow f\left(\begin{bmatrix} 31 \\ 37 \\ 46 \end{bmatrix}\right) = \overset{Y}{\begin{bmatrix} 31 \\ 37 \\ 46 \end{bmatrix}}$$

$$Y=f(Wx+b)$$

https://www.youtube.com/watch?v=SY5ZN7xZxlg

ANN = Matrix Multiplication

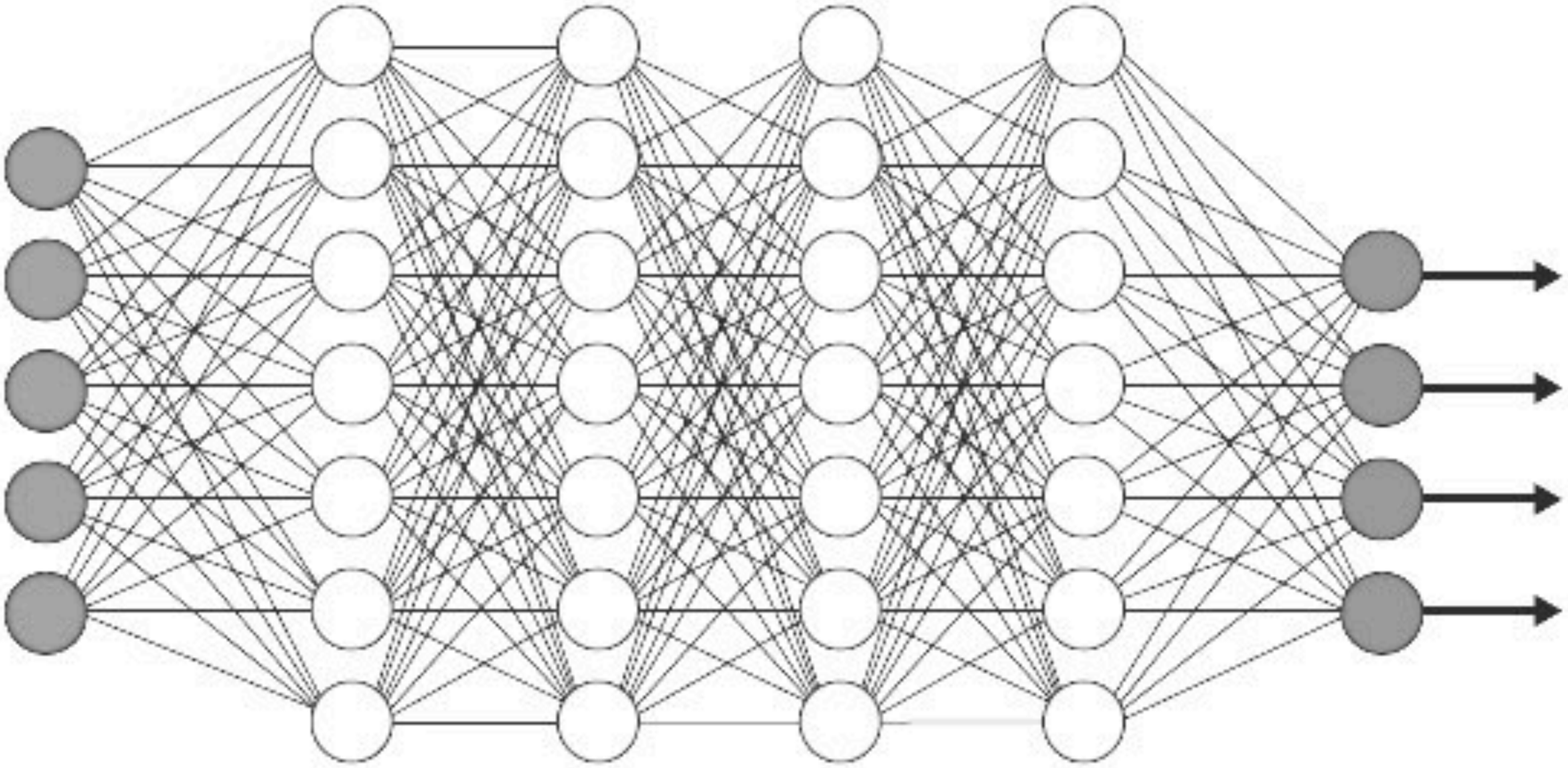




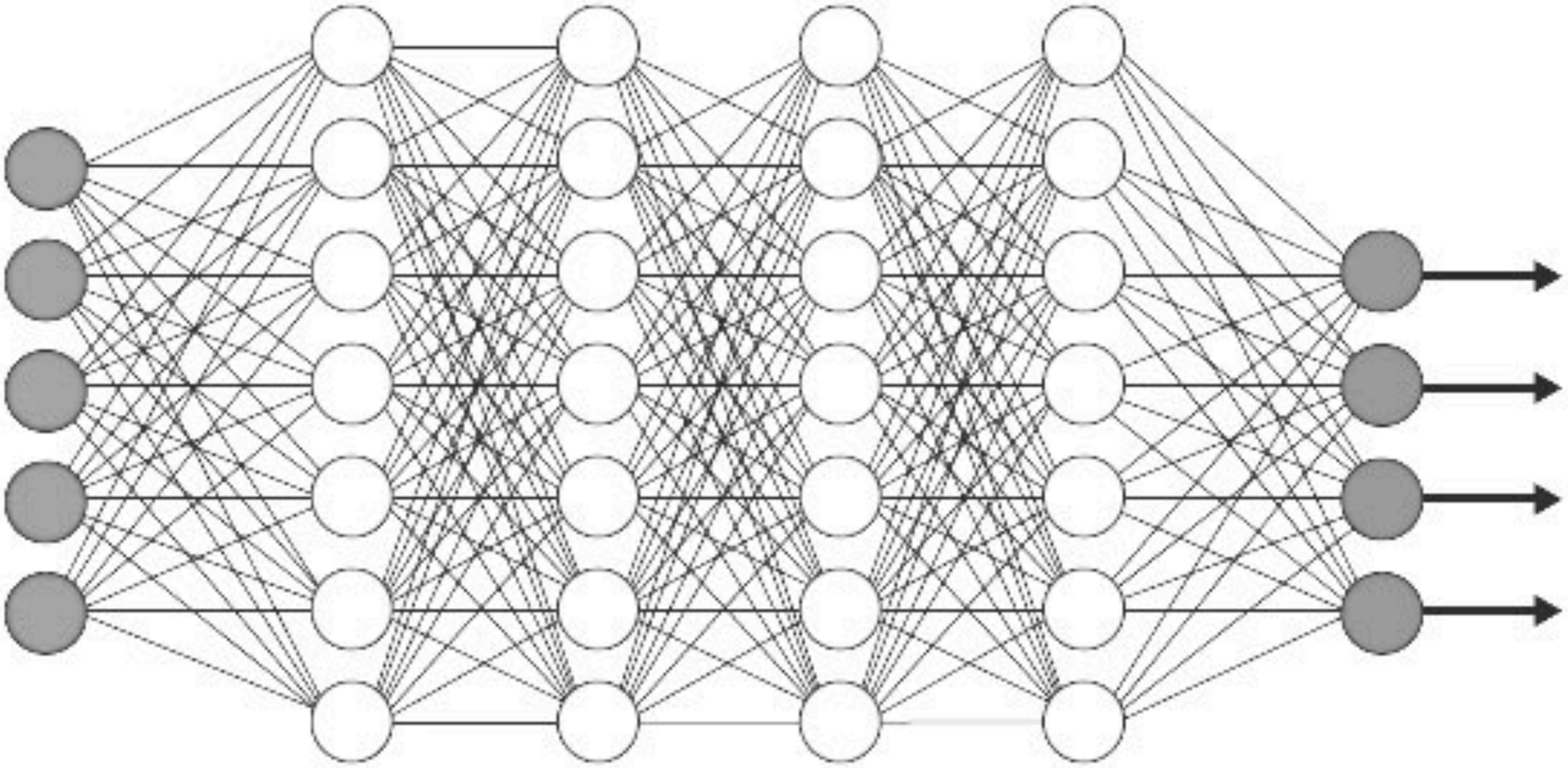
output = f(Weights × input + bias)

https://machinethink.net/blog/convolutional-neural-networks-on-the-iphone-with-vggnet/

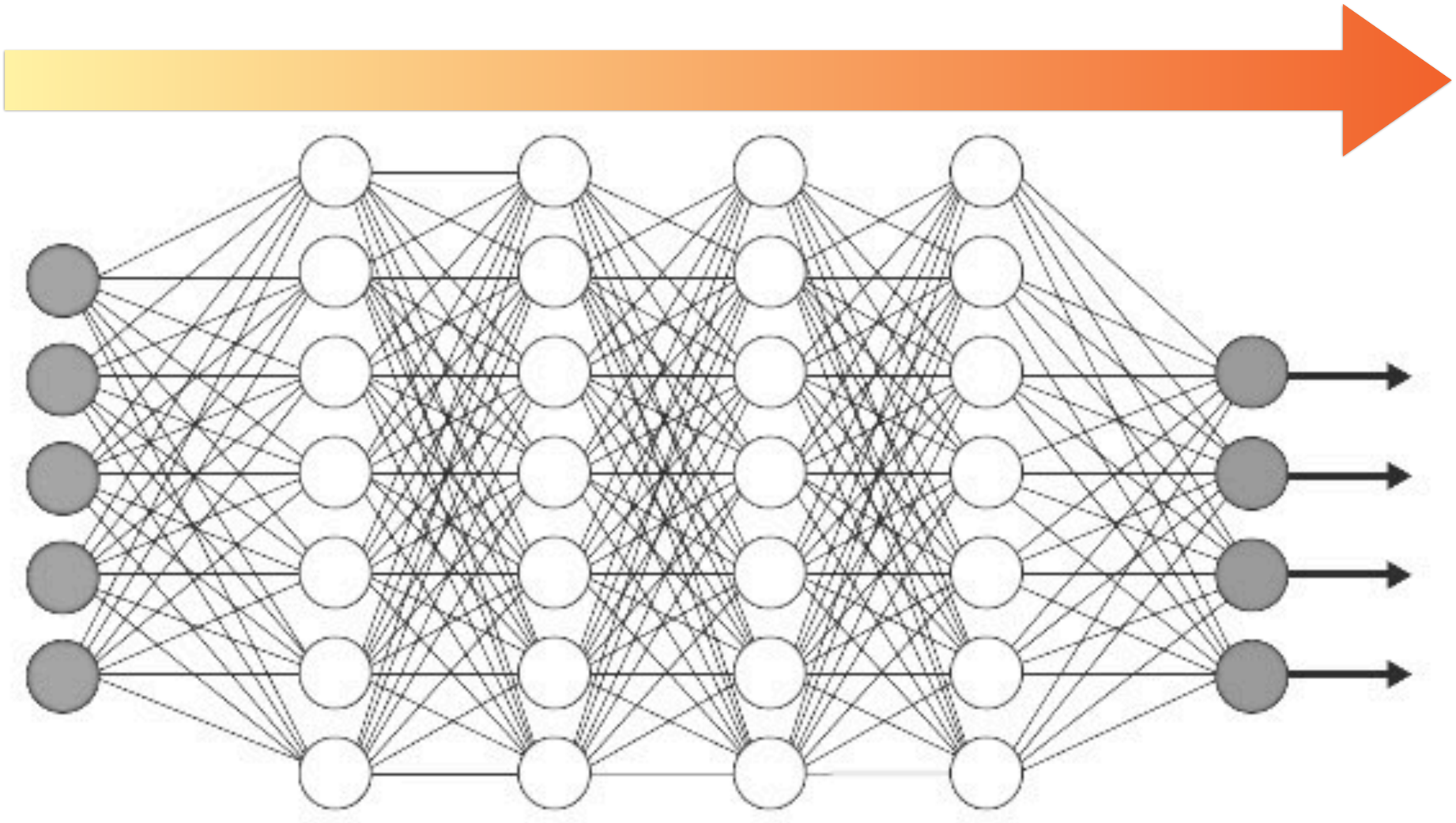
Step 1) Build: Neural Network



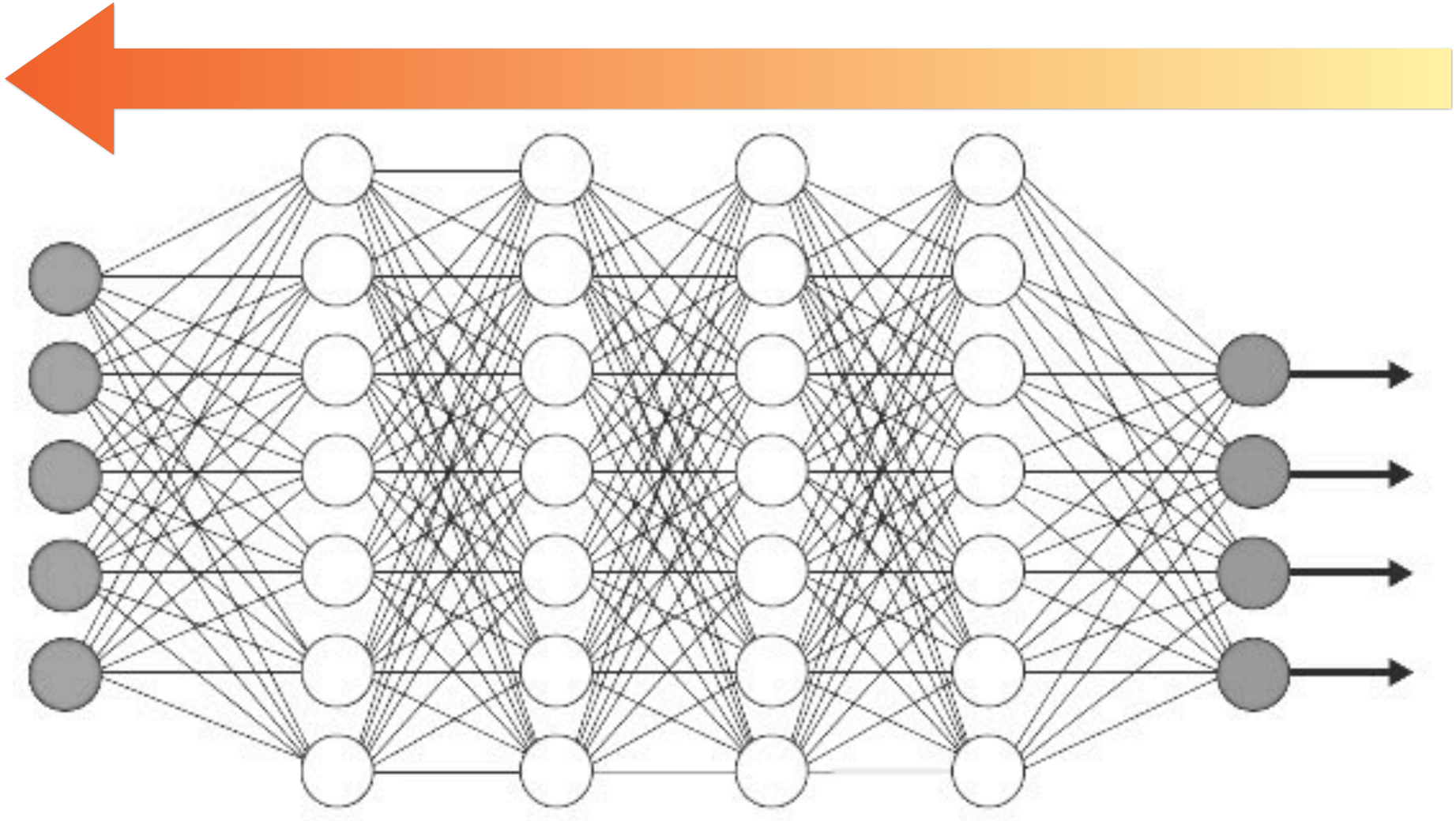
Step 2) Initialize: Weights & Biases



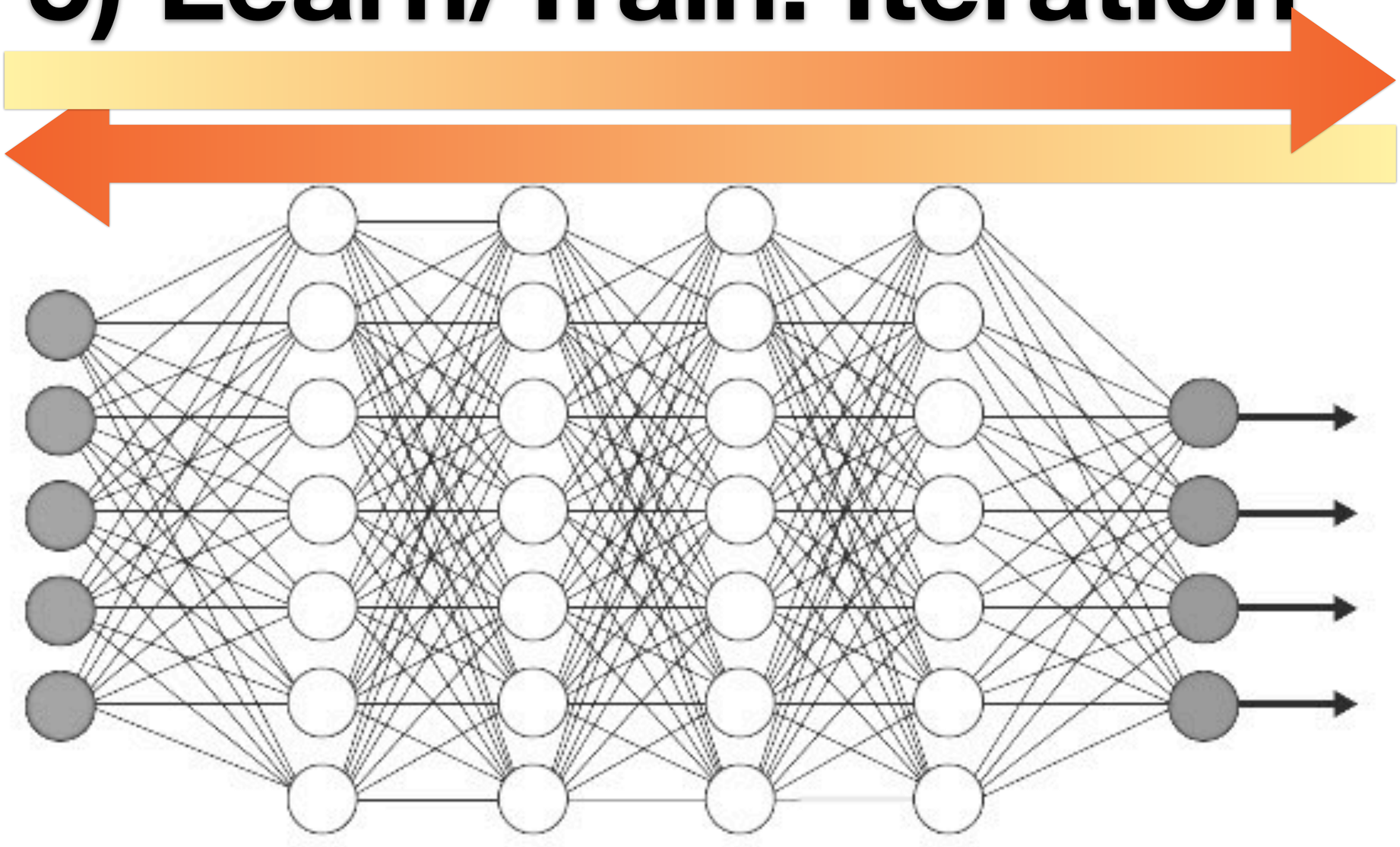
Step 3) Estimate: Feed Forward



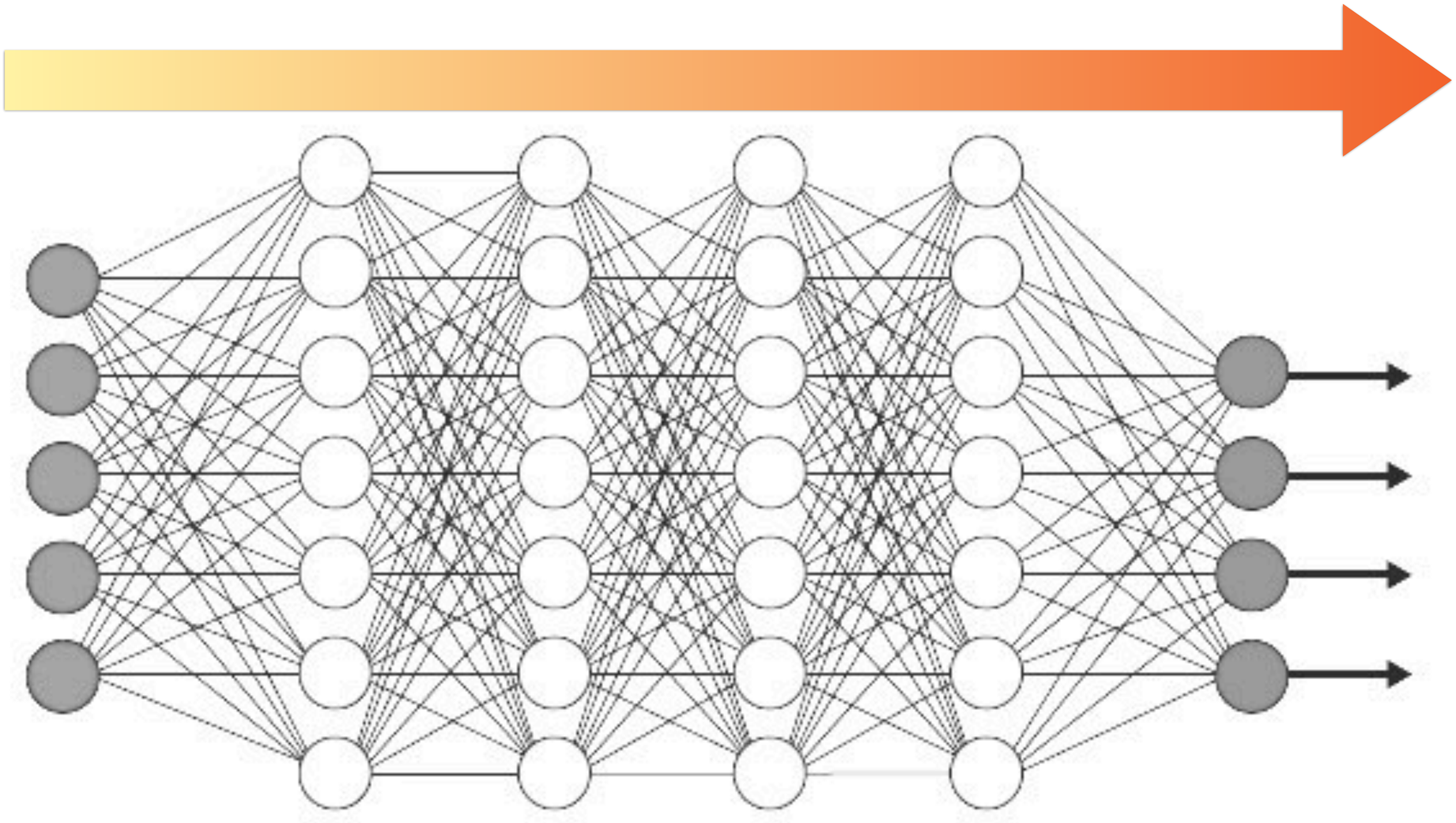
Step 4) Update: Back Propagation



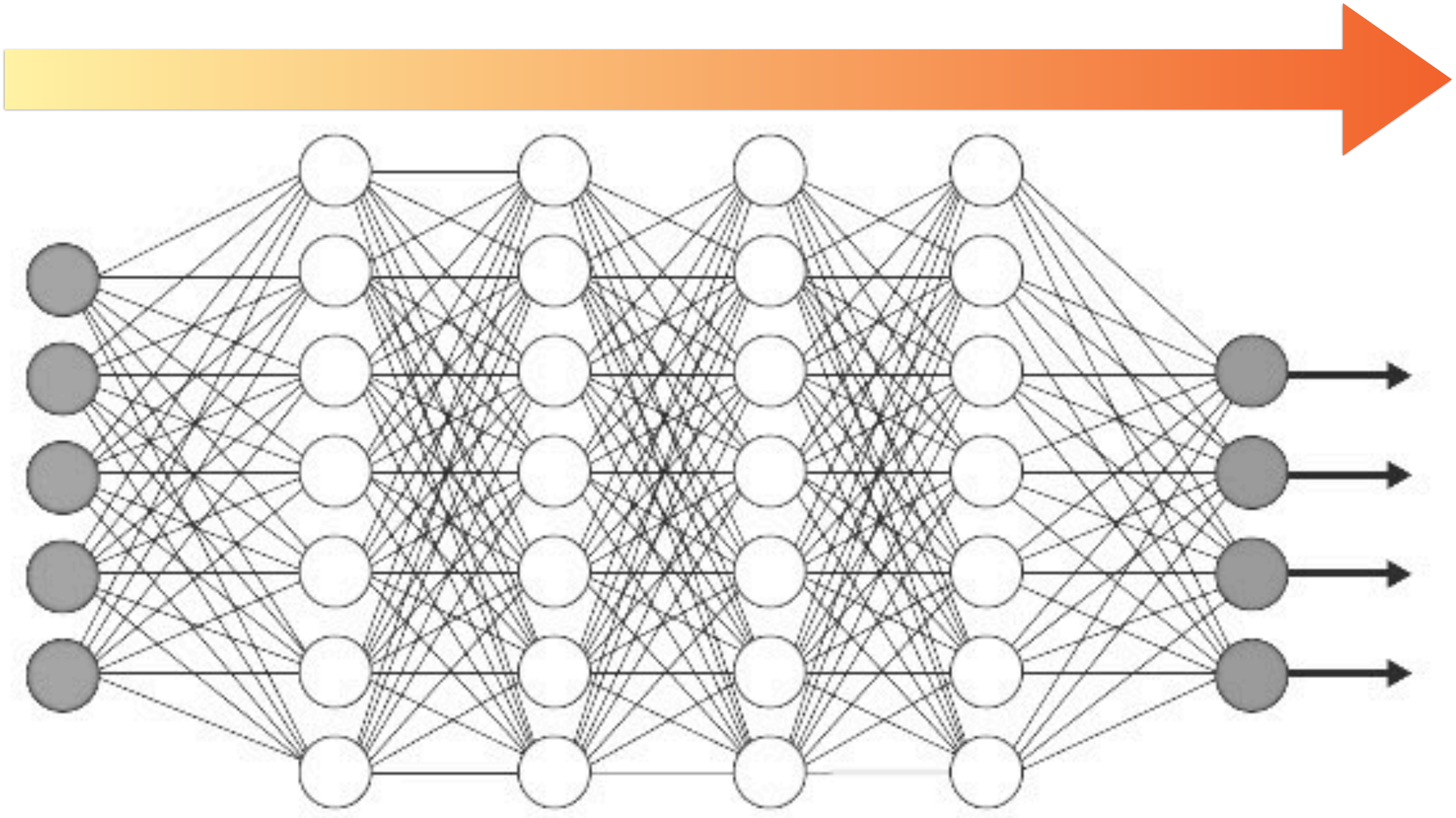
Step 5) Learn/Train: Iteration



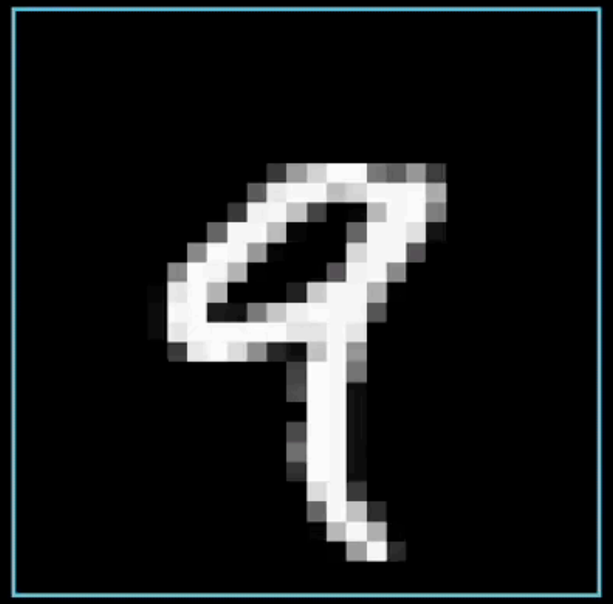
Step 6) Test



Step 7) Infer/Predict: w/ New Data



Classification Example



https://www.youtube.com/watch?v=aircAruvnKk

어떻게 작동하는가?

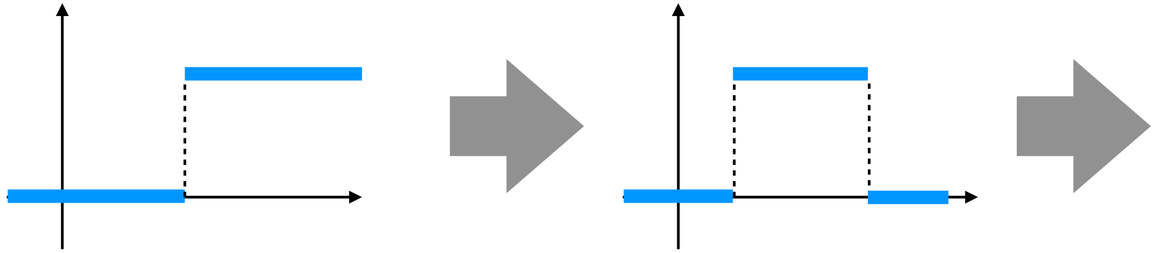
- 만능 근사 정리 (**Universal Approximation Theorem**)
  - **1개**의 **hidden layer**를 가진 **neural network**를 이용해 **어떠한 함수든지 근사시킬 수 있다**는 이론 (단, 비선형의 활성화 함수를 사용해야 함)
  - 하지만, 이 때 원하는 정확도를 얻기 위해서는 노드의 개수가 감당하지 못할 정도로 많아질 수도 있다.
  - 따라서, 더 많은 수의 hidden layer를 사용하는 것이 좋다.

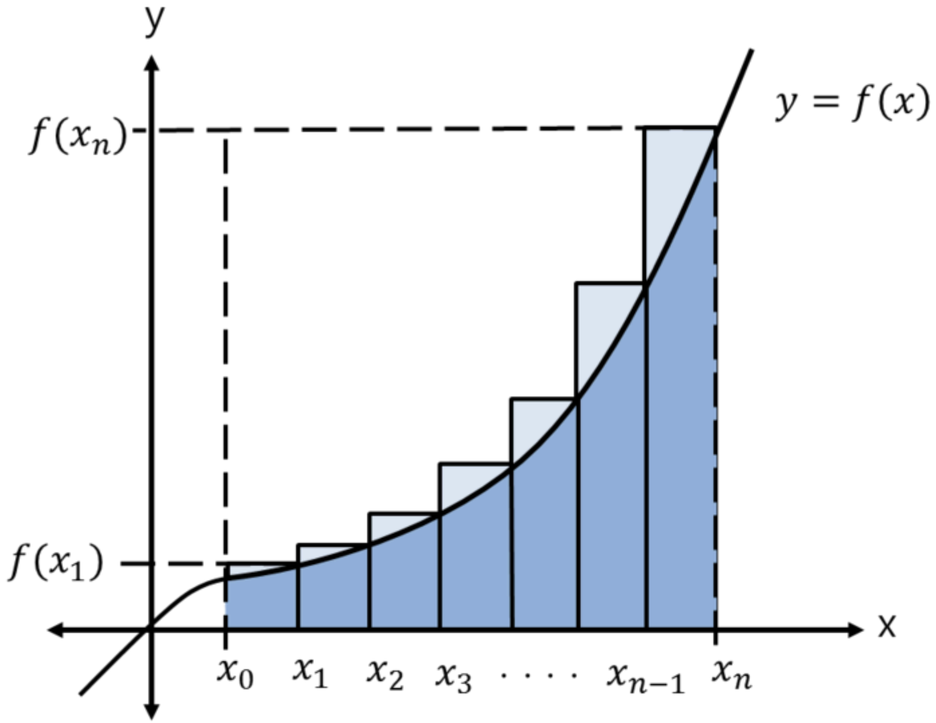
> A feedforward network with a single layer is sufficient to represent any function, but the layer may be infeasibly large and may fail to learn and generalize correctly.
>
> *Ian Goodfellow*

https://www.youtube.com/watch?v=7AloljyhrBc

Universal Approximation Theorem

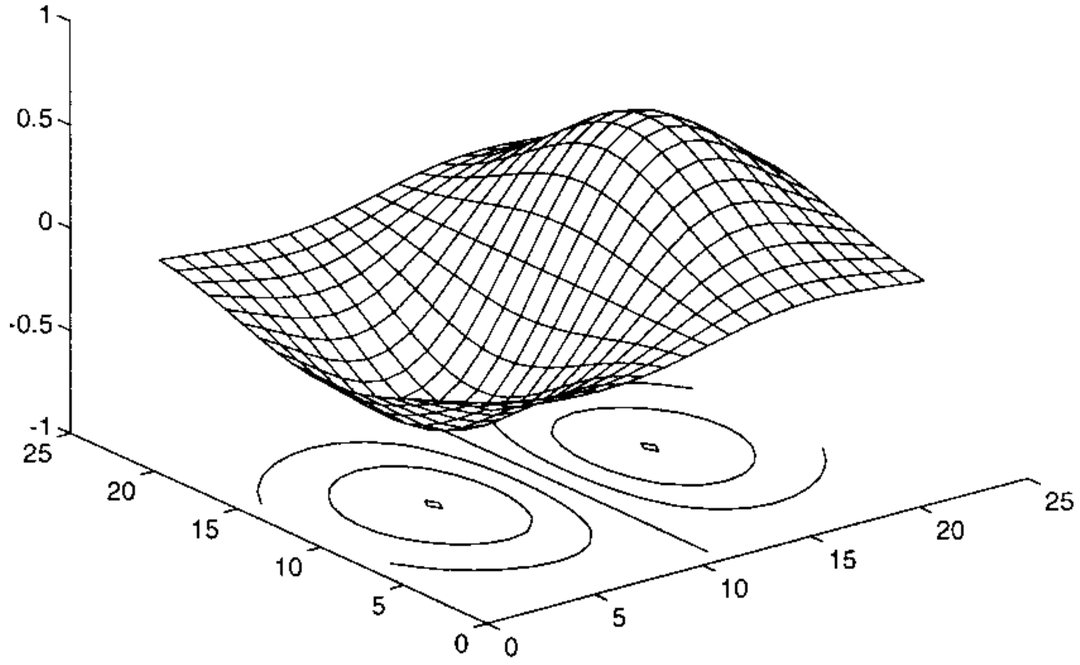
- Therefore, a NN can be an **approximation of some function** we wish to model.
- A neural network is an approximation of some function we wish to model.

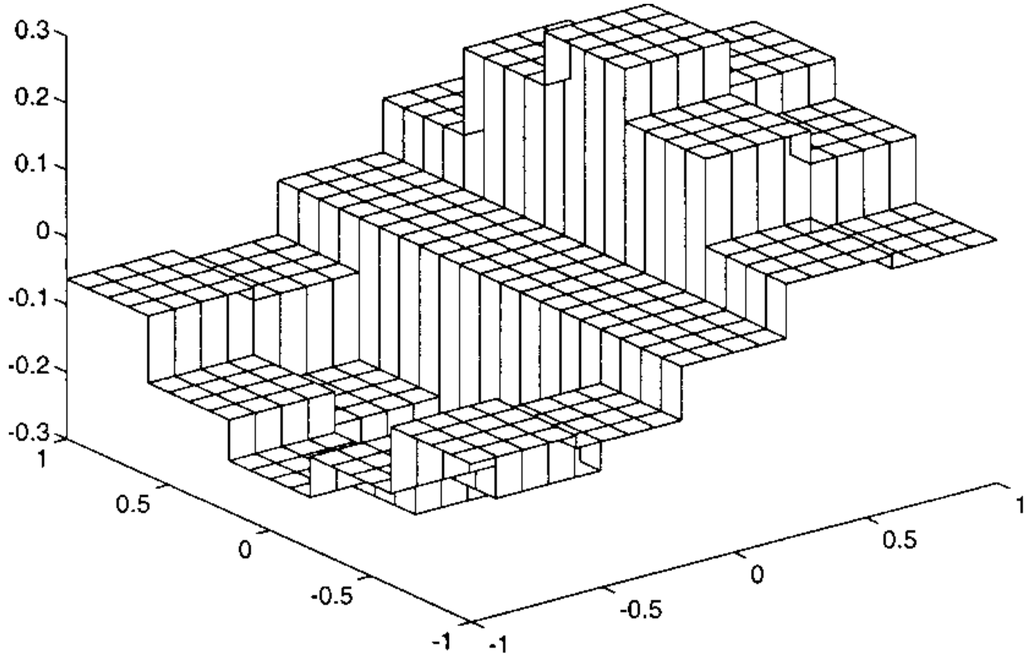




Universal Approximation Theorem

- Therefore, a NN can be an **approximation of some function** we wish to model.
- A neural network is an approximation of some function we wish to model.





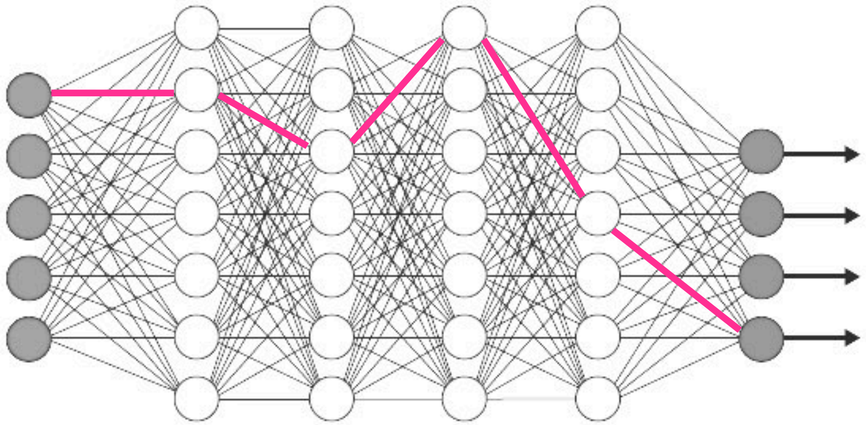
https://www.semanticscholar.org/paper/Universal-Approximation-Using-Feedforward-Neural-A-Scarselli-Tsoi/5126188f4b8180a648b7fda958e354d923afda35

Universal Approximation Theorem

- A feed-forward NN with **a single hidden** layer and continuous **non-linear activation function** can approximate **any continuous function** with arbitrary precision.

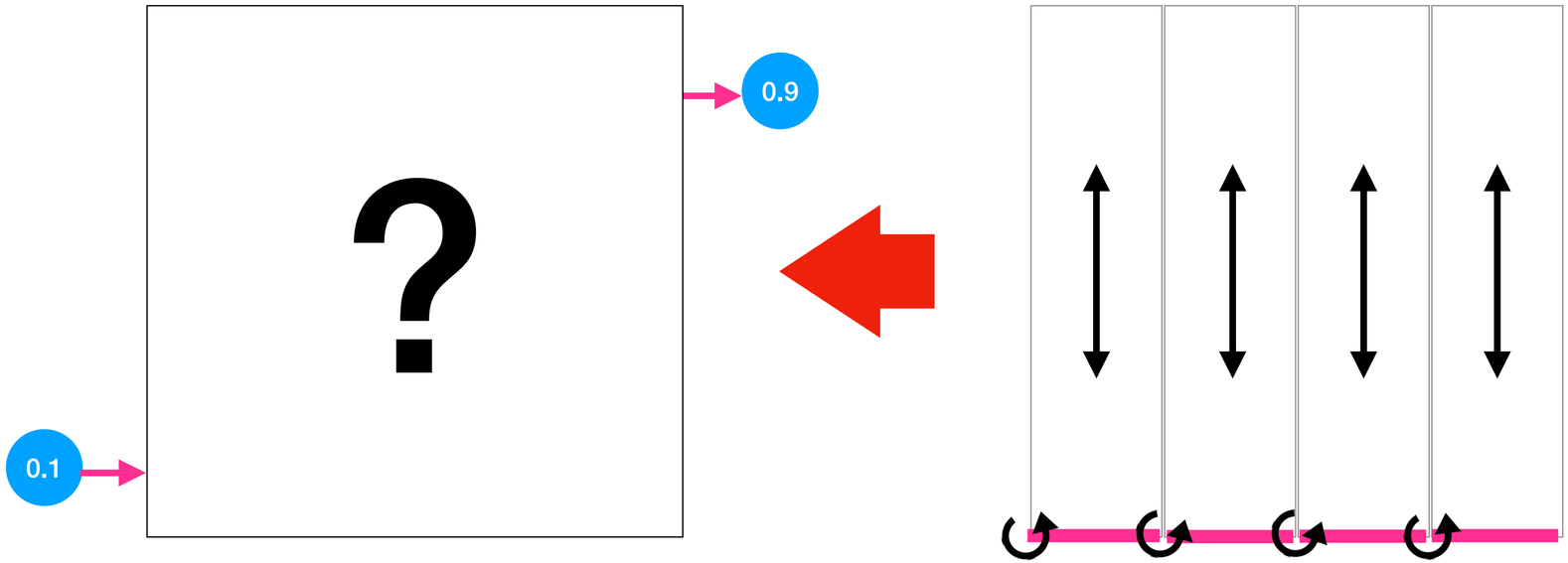
https://www.youtube.com/watch?time_continue=296&v=ljqkc7OLenI

Slope Puzzle

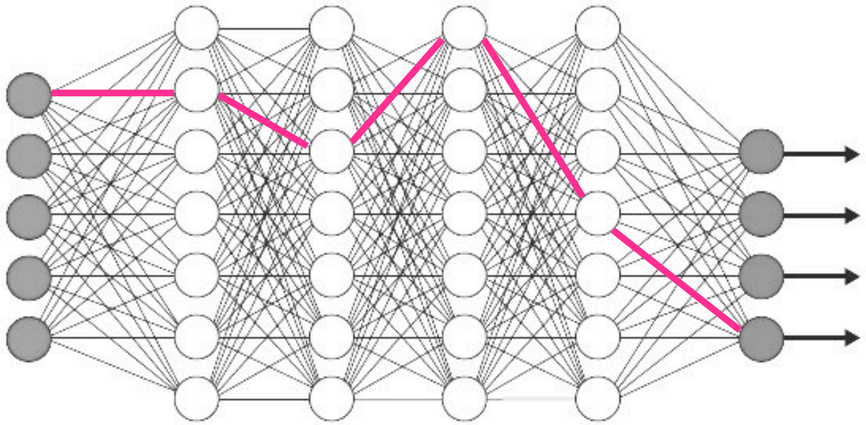


w : **slope**

b : **lift**

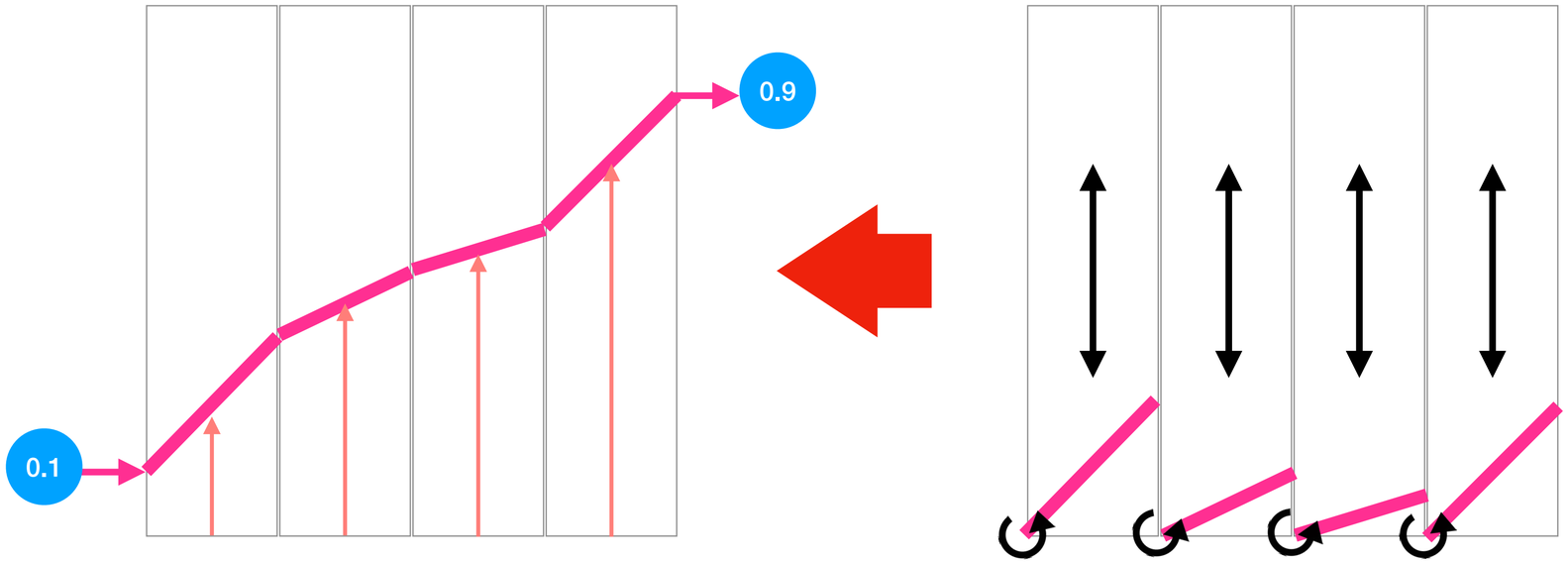


Slope Puzzle

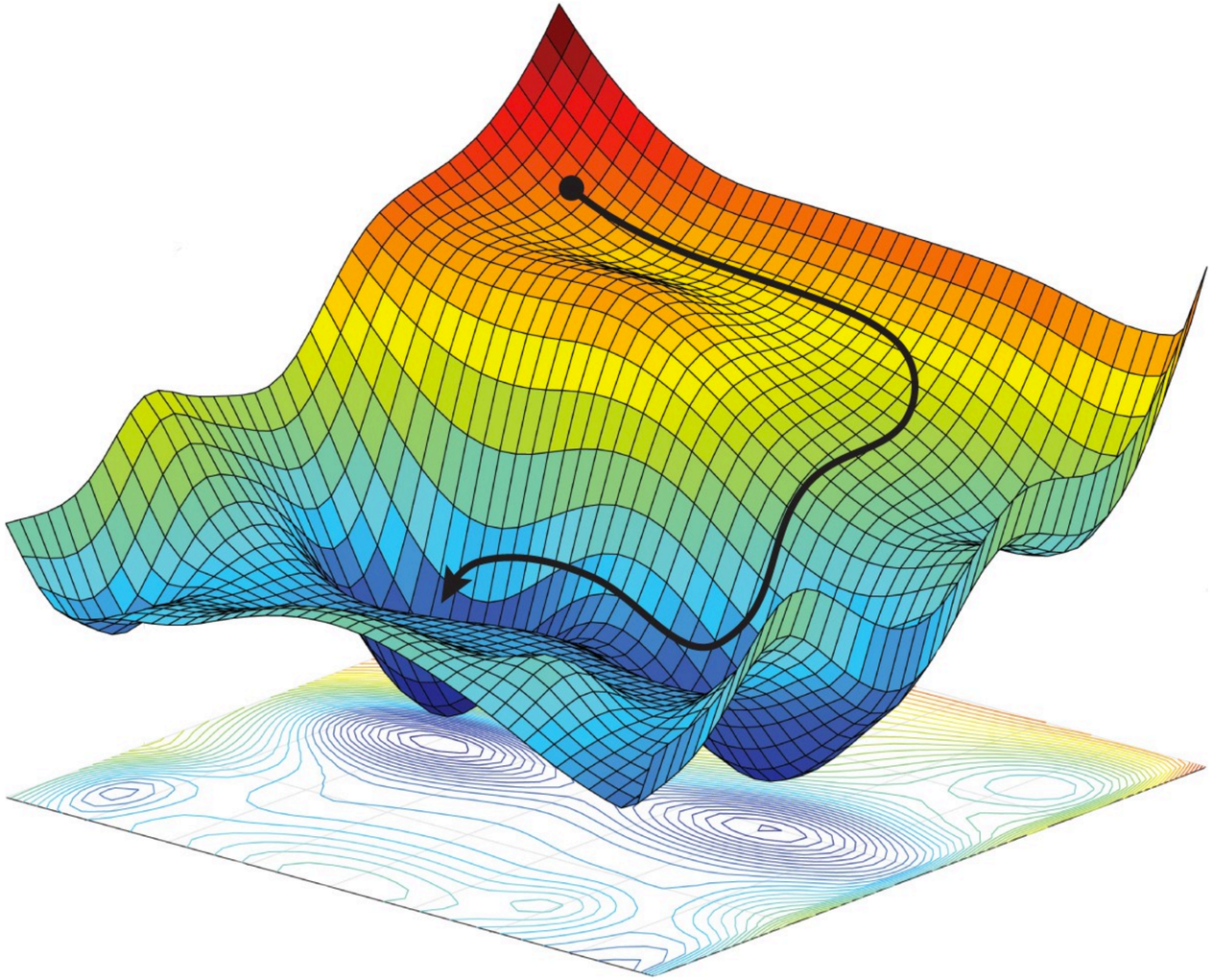


w : **slope**

b : **lift**



최적화 문제 (Optimization Problem)



https://towardsdatascience.com/on-optimization-of-deep-neural-networks-21de9e83e1

최적화 문제 (Optimization Problem)

- **Data munging** (or **wrangling**)

input data: $X = \{x_1, x_2, \ldots, x_n\}$

output data: $Y = \{y_1, y_2, \ldots, y_n\}$

$$D = \{(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)\}$$

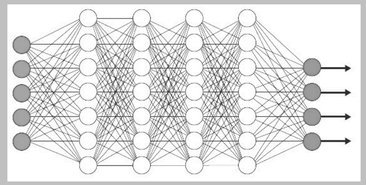
- **Model definition**

output: $Y_\theta = f_\theta(X)$

parameter set: $\theta = \{\theta_1, \theta_2, \ldots, \theta_m\}$

loss: $L(Y, Y_\theta)$

Deep Neural Network



$$Y_\theta = f_\theta(X)$$

$$\theta = \{(W_1, b_1), (W_2, b_2), \ldots, (W_l, b_l)\}$$

- **Learning** (or **training**)

$$\boxed{\theta^* = \underset{\theta}{argmin}\, L(Y, Y_\theta)}$$

- **Testing** (or **prediction**)

$$Y_{new} = f_{\theta^*}(X_{new})$$

https://www.slideshare.net/NaverEngineering/ss-96581209

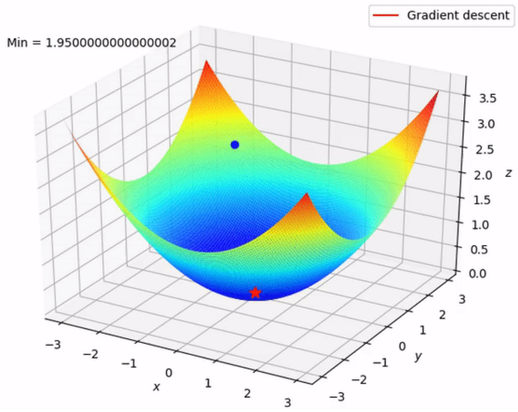
최적화 문제 (Optimization Problem)

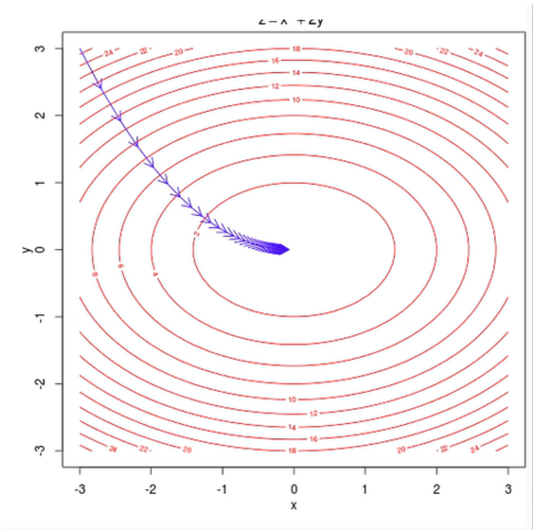
- **출력값**과 주어진 **정답값**과의 **차이**가 **줄어들도록** 네트워크의 **가중치**와 **편향값**을 **재조정**하는 **과정**

- **경사(gradient) 하강(descent) 법(method)**: 최적화(optimization) 문제를 푸는 기법 중 하나

- **비용(cost) 함수**는 출력값과 정답값과의 거리 차이의 제곱을 한 형태이기 때문에
  3차원 그래프로 그려보면 단면이 포물선(parabola), 즉 아래로 오목한 그릇(bowl) 형태가 된다.
  quadratic form

- 이 그릇(bowl) 상의 임의의 지점에서 시작하여
  cost function **initial guess**
  최대 경사 방향으로 조금씩 내려갈 때
  -gradient **learning rate** **iteration**
  더 이상 내려갈 수 없는 지점이 바로
  optimal solution
  우리가 찾는 비용(cost) 함수를 최소화 하는 지점이 된다.





http://www.xpertup.com/2018/05/11/loss-functions-and-optimization-algorithms/

https://giphy.com/gifs/gradient-6QlTwkiggq4yk

Iterative Approach

$$\textit{Repeat until} \quad \| L(\theta + \Delta\theta) - L(\theta) \| < \epsilon$$

$$\{$$

$$\theta \rightarrow \theta + \Delta\theta \qquad s.t. \quad L(\theta + \Delta\theta) < L(\theta)$$

$$\}$$

direction? ∇L : *to the steepest descent*

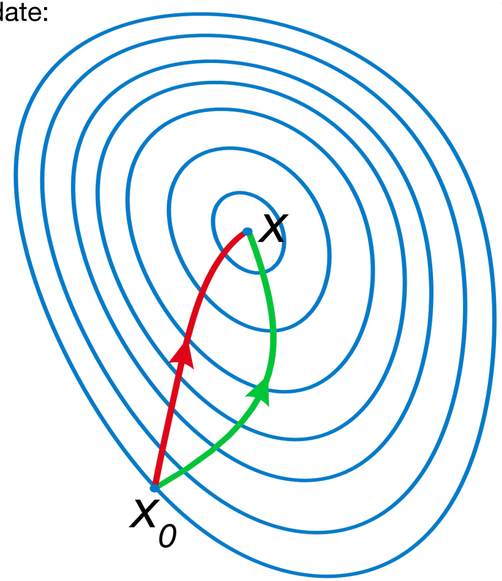
how much? α : *user defined learning rate*

Linear Search Algorithm

a given cost function: $y = f(\boldsymbol{x})$

goal: $\underset{x}{argmin} f(x)$

update:



https://en.wikipedia.org/wiki/Newton%27s_method_in_optimization

The current position: $\boldsymbol{x}^n$

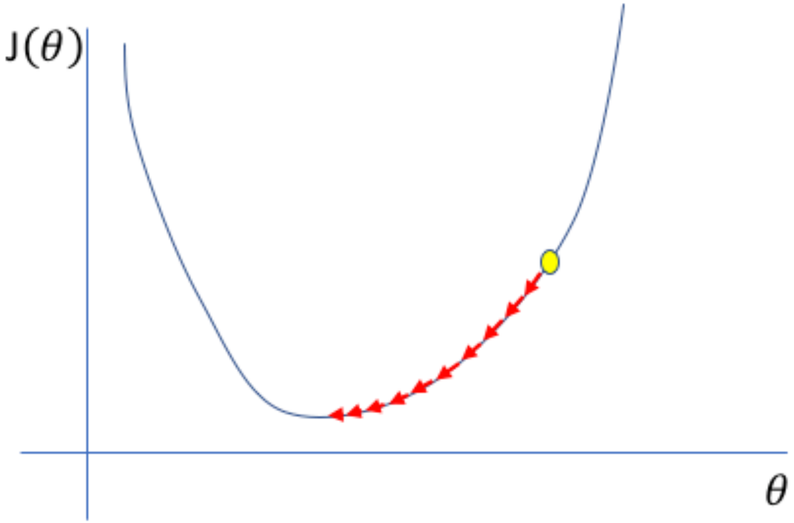
$$y^{n+1} = f(\boldsymbol{x}^n + \underbrace{\alpha^n}_{\text{moving distance}} \cdot \underbrace{\boldsymbol{d}^n}_{\text{moving direction}})$$

$$\boldsymbol{d}^n = -\underbrace{\nabla f(x^n)}_{\text{gradient}}$$

gradient
: the steepest direction

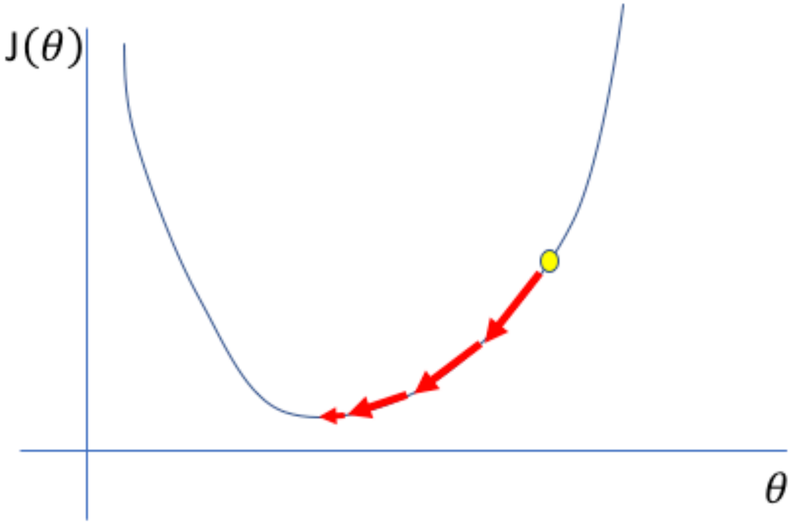
Learning Rate

Too low



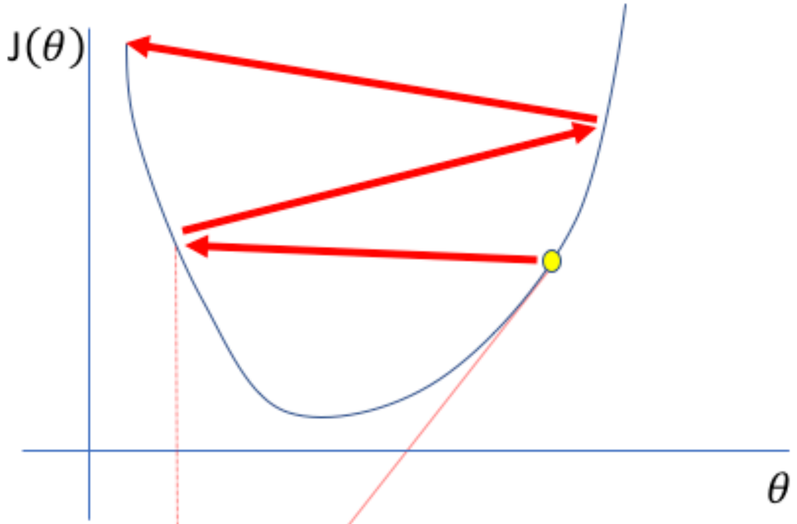
A small learning rate requires many updates before reaching the minimum point

Just right



The optimal learning rate swiftly reaches the minimum point

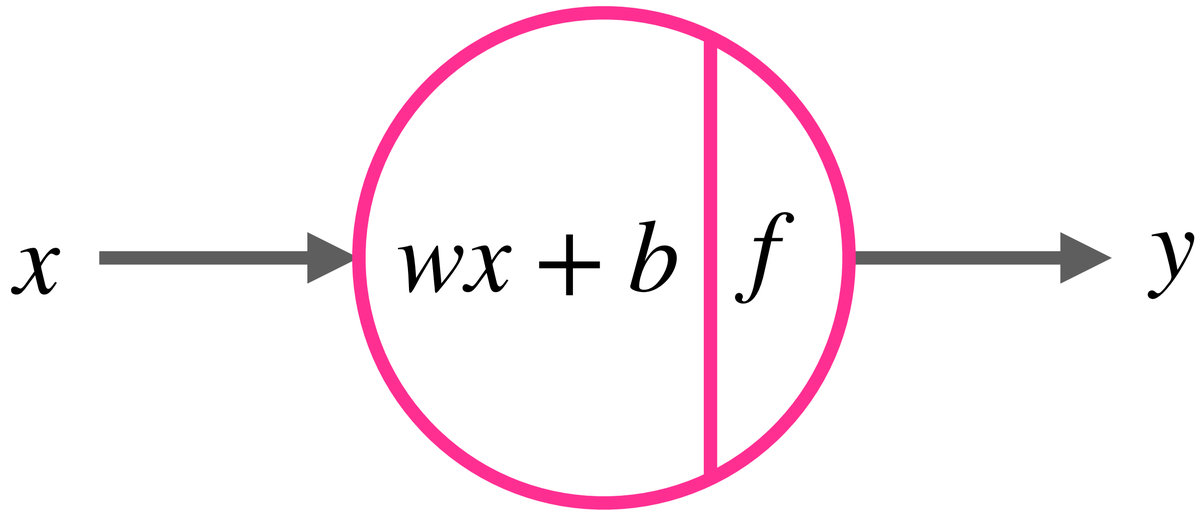
Too high



Too large of a learning rate causes drastic updates which lead to divergent behaviors

https://www.jeremyjordan.me/nn-learning-rate/

How to Update Weight & Bias



$$E = \frac{1}{2}\left(y - y_{target}\right)^2$$

assumption)

$f(\sigma) = \sigma$: simple linear function

$$w^{n+1} = w^n - \alpha\frac{\partial E}{\partial w} \quad : \quad \frac{\partial E}{\partial w} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial f}\frac{\partial f}{\partial \sigma}\frac{\partial \sigma}{\partial w} = (y - y_{target}) \cdot 1 \cdot 1 \cdot x$$

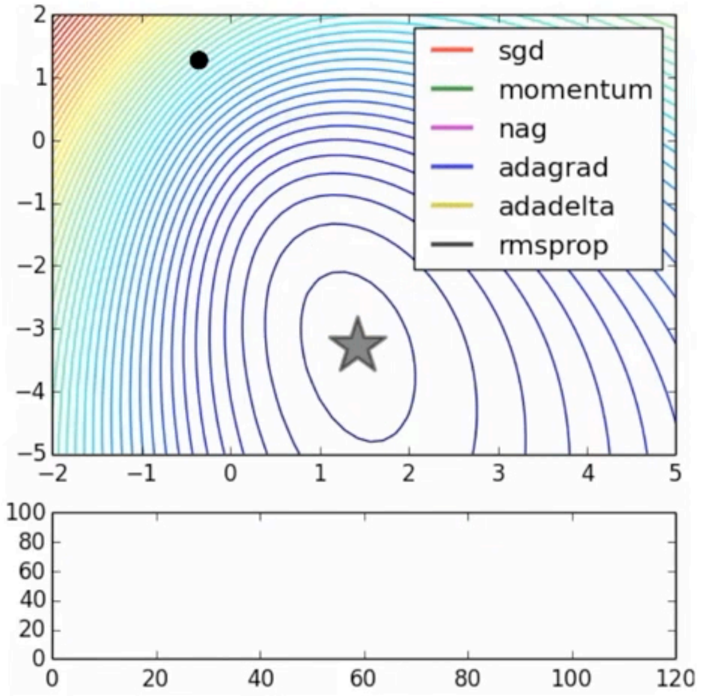
$$b^{n+1} = b^n - \alpha\frac{\partial E}{\partial b} \quad : \quad \frac{\partial E}{\partial b} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial f}\frac{\partial f}{\partial \sigma}\frac{\partial \sigma}{\partial b} = (y - y_{target}) \cdot 1 \cdot 1 \cdot 1$$

Optimizers

Weights update process

$$W_{new} = W - \eta \cdot \nabla_W L$$

```
weights_grad = evaluate_gradient(
    loss_fun, data, weights)
weights += -step_size * weights_grad
```



https://www.youtube.com/watch?v=kGAo32JgY48&t=42s

Optimizers & Momentum

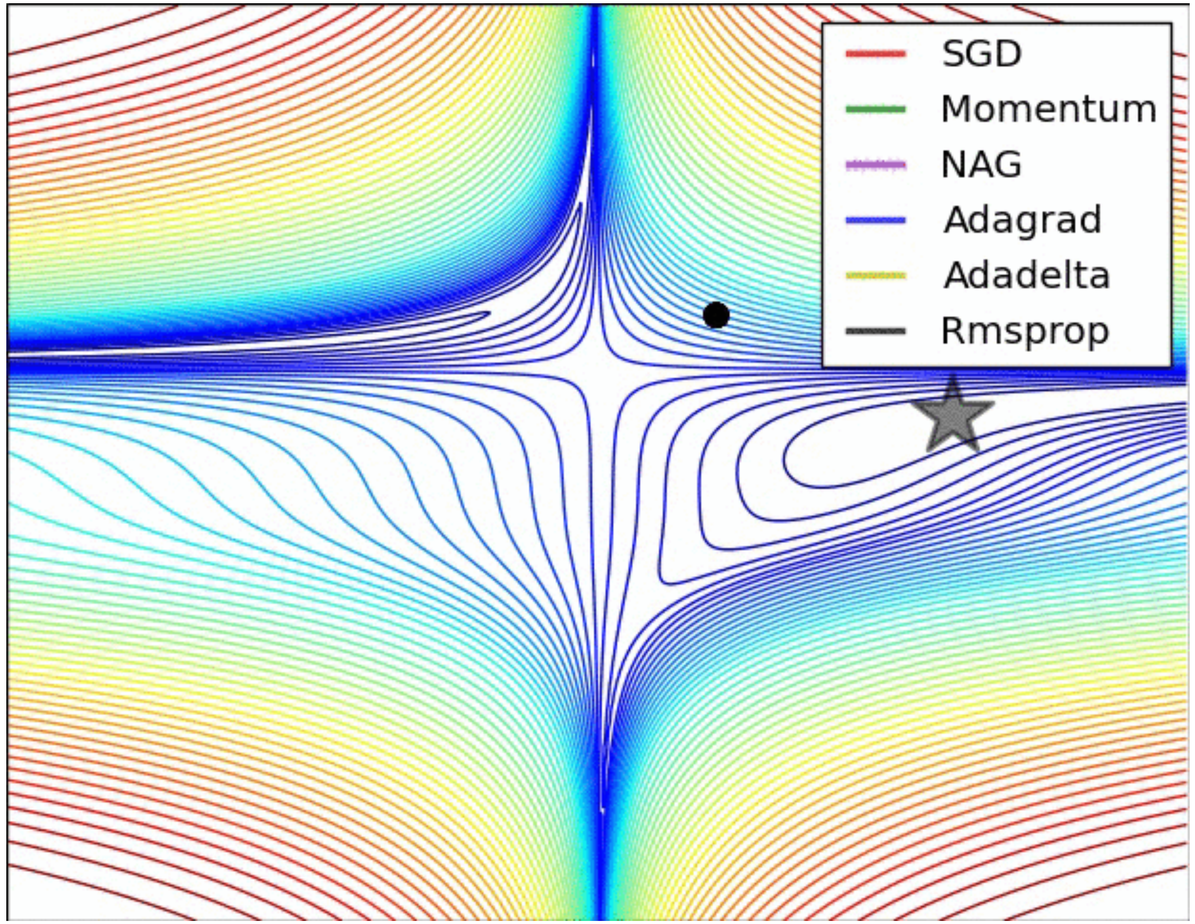
- The inspiration came from physics in the form of **momentum**.

- One of the most commonly used optimizers is Stochastic gradient descent (SGD). Unfortunately, **SGD** is inherently limiting as it employs **first-order information only**.

- It cannot compete with optimizers that use **second-order information**, the so-called **momentum**. By utilizing second-order dynamics, optimizers can effectively get an **adaptive learning rate** for each parameter. This, in turn, lends itself to extremely fast convergence.

- Imagine two balls, one rolling on a slope and another rolling inside a bowl. What will happen if both the balls are left to their destiny? For the first bowl, it keeps increasing its velocity (in other words momentum), and for the second it will simply come to stand-still at the bottom of the bowl after a few swings.

Optimizers: Comparison



https://towardsdatascience.com/on-optimization-of-deep-neural-networks-21de9e83e1

Software 1.0 vs Software 2.0

- **Andrej Karpathy**
- Director of AI at **Tesla**.
- Previously Research Scientist at OpenAI and PhD student at Stanford.

https://medium.com/@karpathy/software-2-0-a64152b37c35

https://gist.github.com/haje01/d2518ea998ab2de102b072fed600c0a4

Software 1.0

$$\mathbf{f} = m\mathbf{a}$$

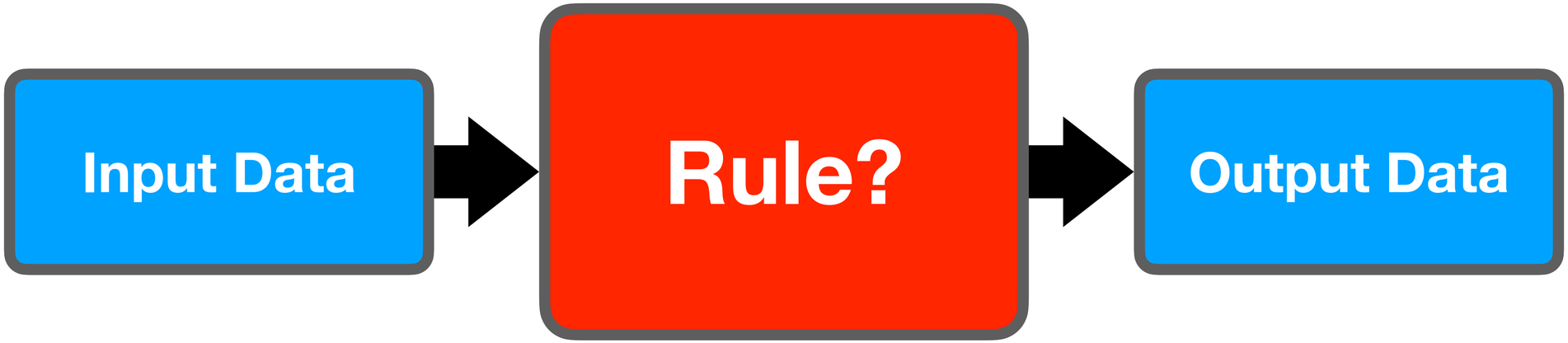
$$\frac{\partial \mathbf{u}}{\partial t} = -\left(\mathbf{u}\cdot\nabla\right)\mathbf{u} + \nabla\cdot(\nu\nabla\mathbf{u}) - \frac{1}{\rho}\nabla p + \mathbf{f}$$

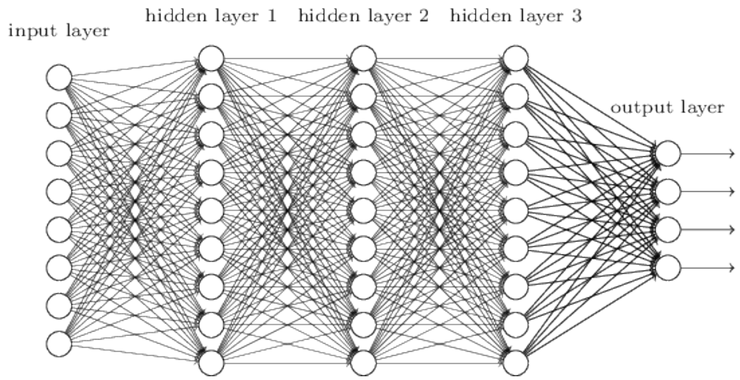
$$m\ddot{\mathbf{x}}(t) + c\dot{\mathbf{x}}(t) + k\mathbf{x}(t) = \mathbf{f}(t)$$

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0$$

$$\vdots$$

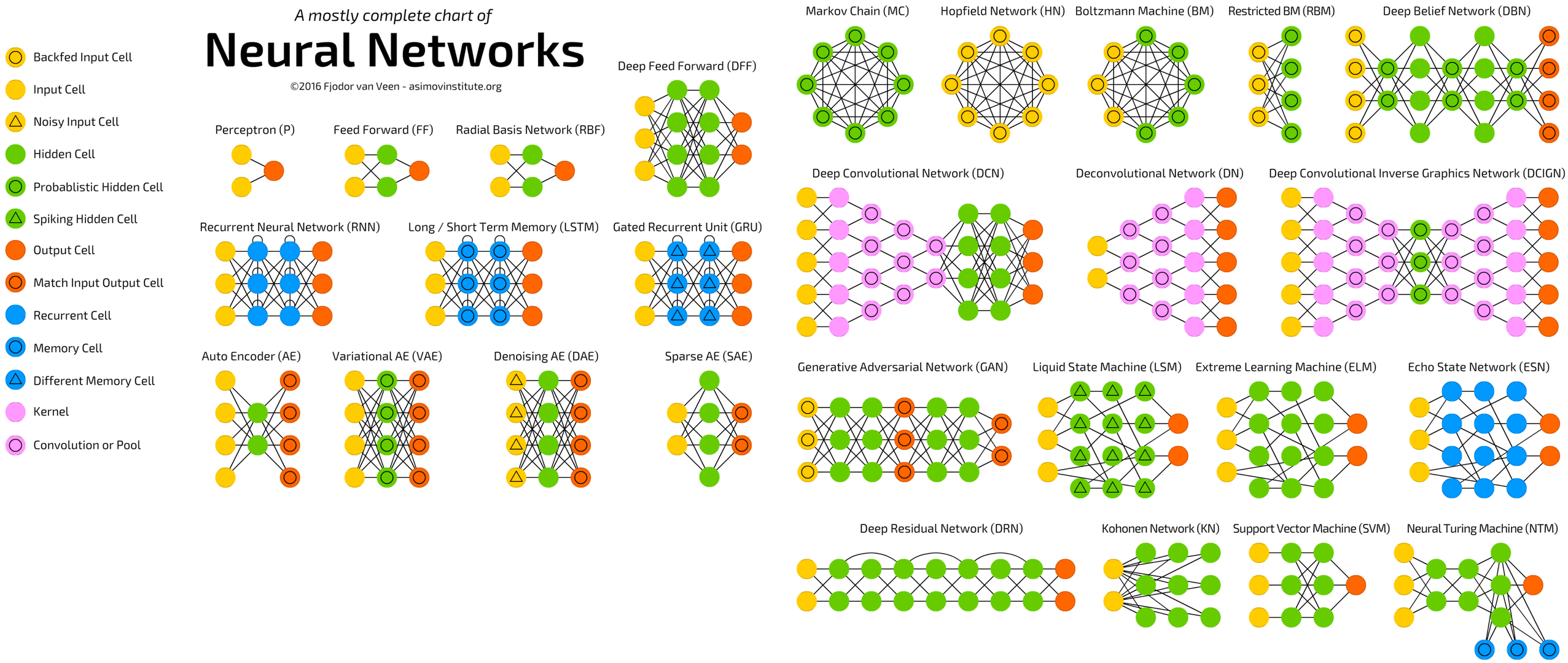
Software 2.0





Hey computer! I'll give you data. You find the rules yourself!

Variations



https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464

Flexible Architecture

- Deep neural networks are completely **flexible by design**.
- There really are **no fixed rules** when it comes to model architecture.
- A point to remember is that it is the **layers** rather than the network itself.
- Like a child with a set of **building blocks**, the design of your neural network is only limited by your own **imagination**—and, crucially, your understanding of how the various layers fit together.

Deep Learning Problems

어떠한 **네트워크 구조**를 사용할 것인가

미분 가능한 **손실 함수**를 어떻게 설계할 것인가

필요한 **데이터**를 어떻게 확보할 것인가

어떠한 **파라미터**들로 학습을 시킬 것인가

It is a Black Box!

#56 수랏간 식당(이하 식선각)

정상궁 (앞에 앉아있는 아이들을 보며)모두 그리 생각하느냐?
모두 맞는다는 듯 아무소리 못 하고 있다.

정상궁 그 래? (하며 한 번 먹어보고 음미하고는)내 생각엔..

장금 홍시입니다.

정상궁 어찌 홍시라 생각하느냐?

장금 예?
저는.. (기 죽어가며)제 입에서는.. 고기를 씹을 때..
홍시 맛이 났는데.. (중얼중얼)
어찌 홍시라 생각했느냐 하시면
그냥.. 홍시맛이 나서 홍시라 생각한 것이온데.

정상궁 (크게 웃으며)호오! 타고난 미각은 따로 있었구나!
그렇지! 홍시가 들어있어 홍시 맛이 난걸
생각으로 알아내라 한 ,내가 어리석었다!

Always Keep in Your Mind

GARBAGE
IN

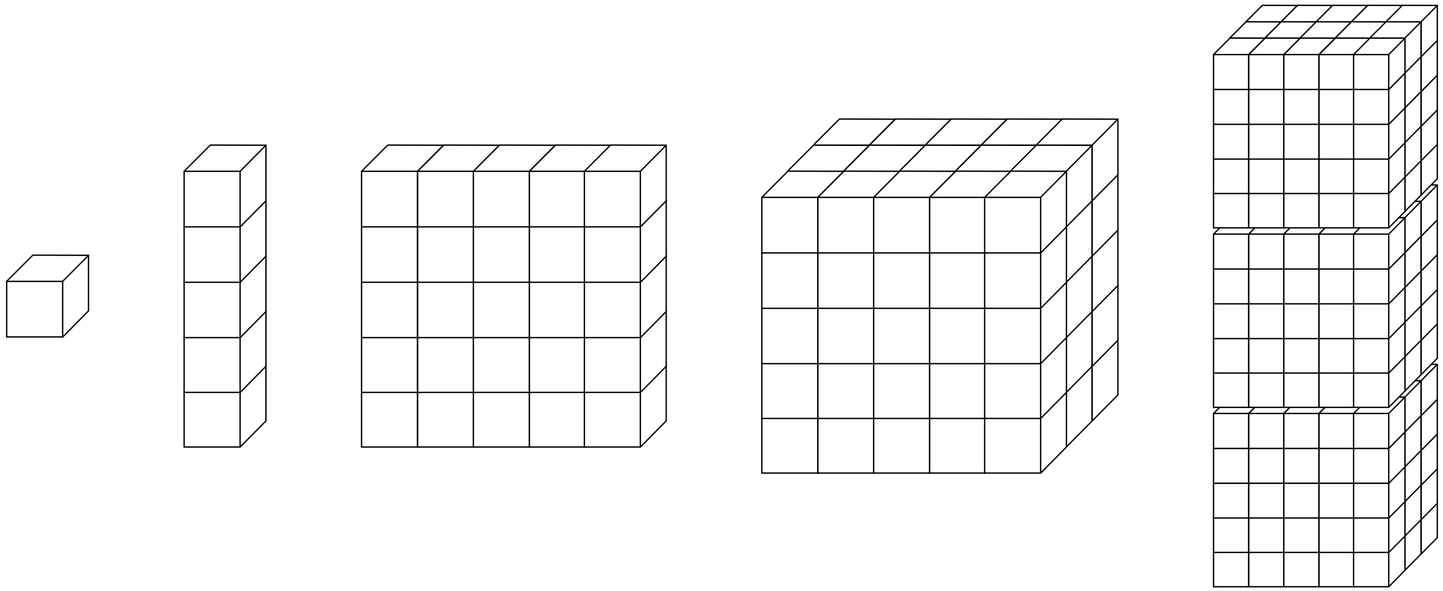
GARBAGE
OUT

NumPy Basics

Tensor

- Simply
  - Multi-dimensional array
  - Generalized matrix
- Strictly
  - Matrix: just a collection of numbers inside brackets
  - Tensors have some transformation properties when changing coordinate system.
- In TensorFlow
  - Unit of data

Tensor



0D tensor

scalar

1D tensor

vector

2D tensor

matrix

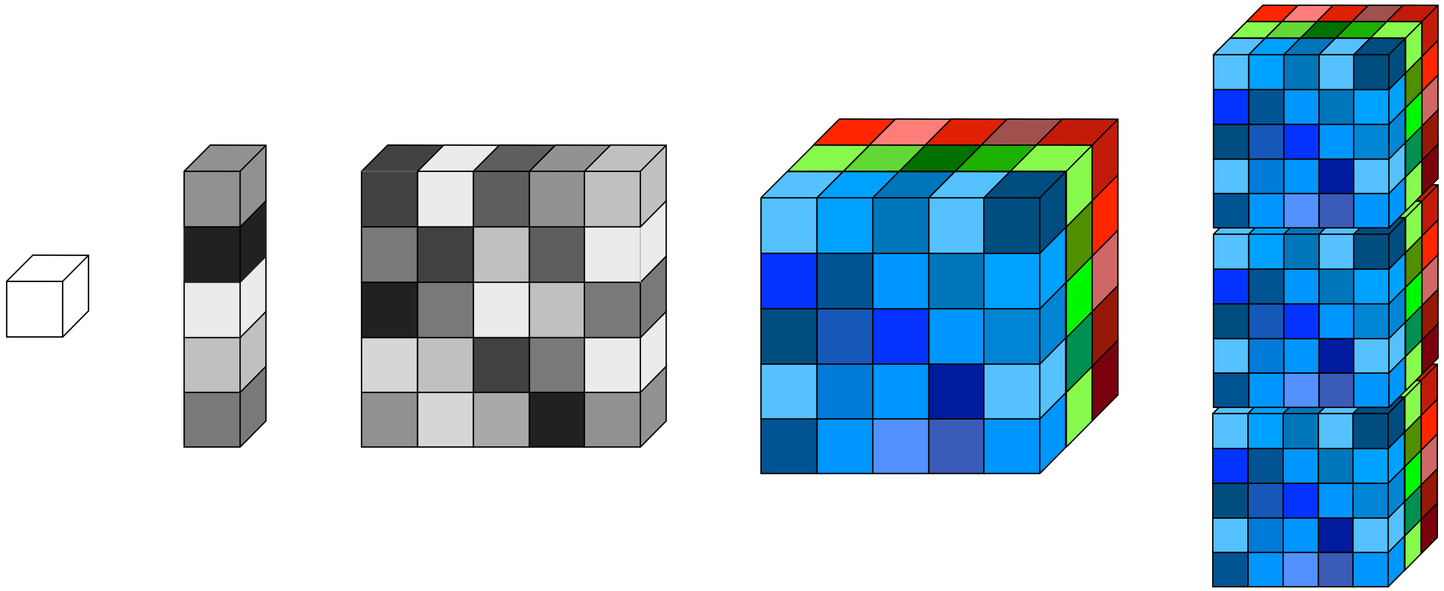
3D tensor

cube

4D tensor

a vector of cube

Tensor



0D tensor

variable

1D tensor

array

2D tensor

gray scale image

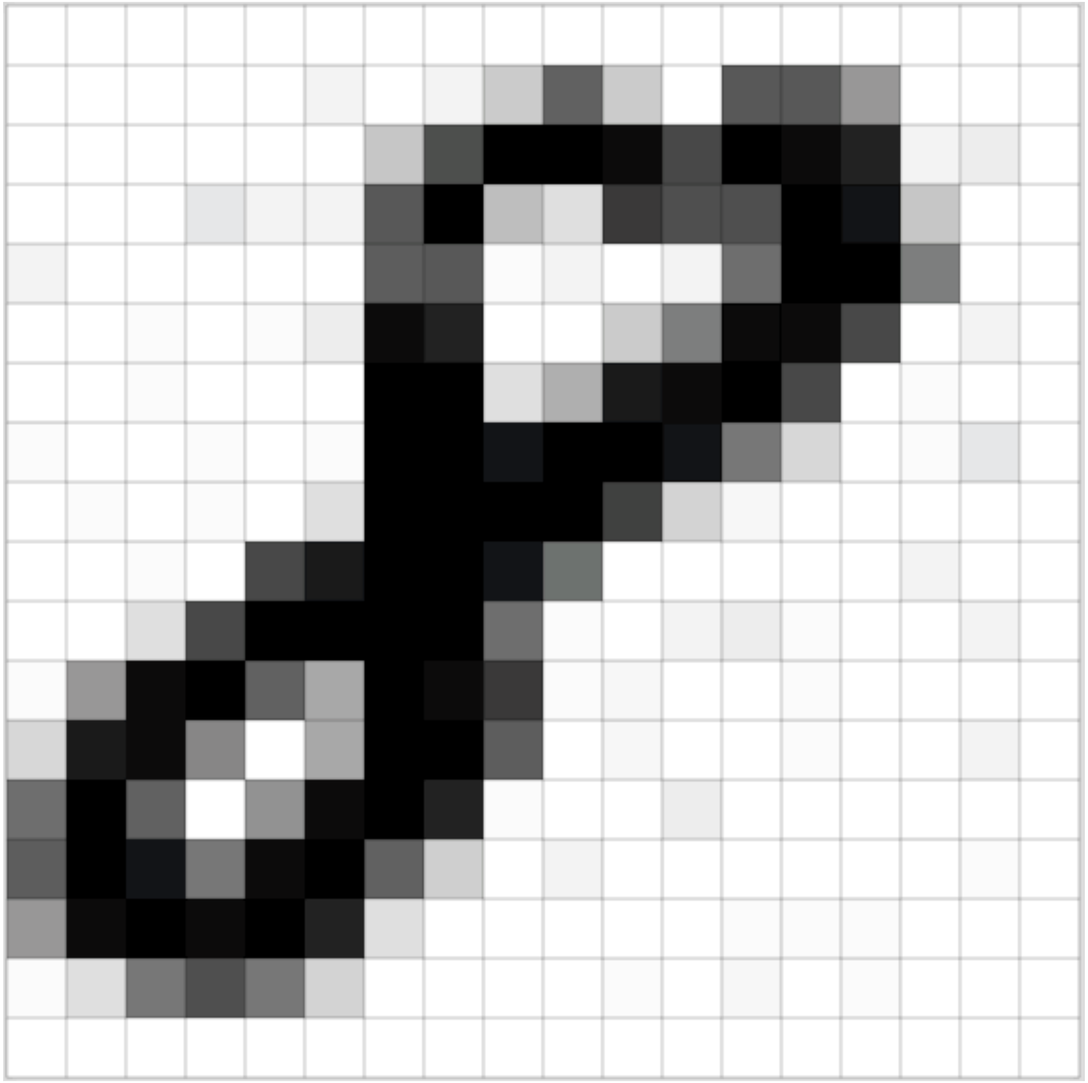
3D tensor

RGB image

4D tensor

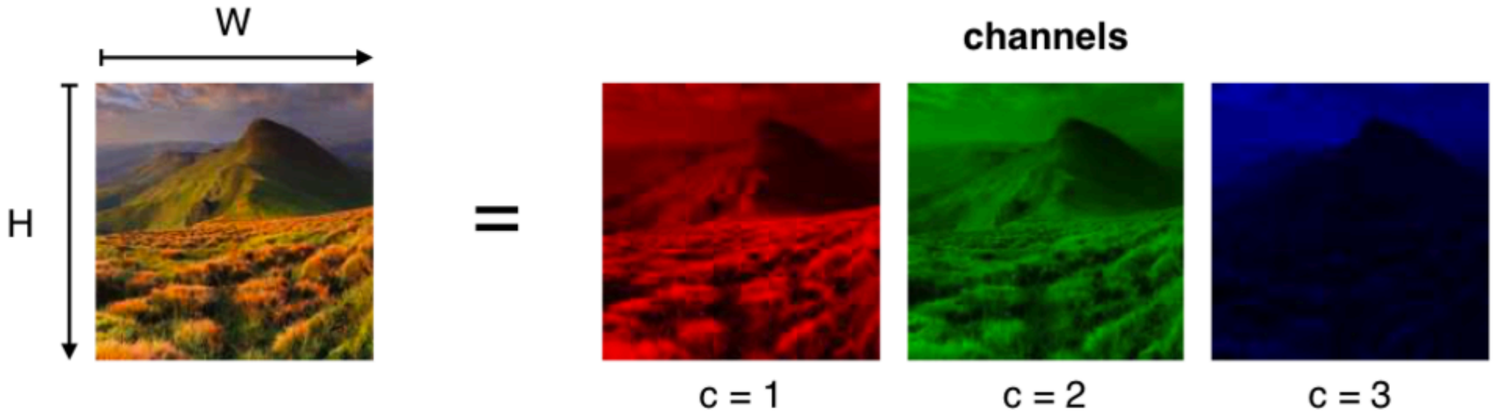
video

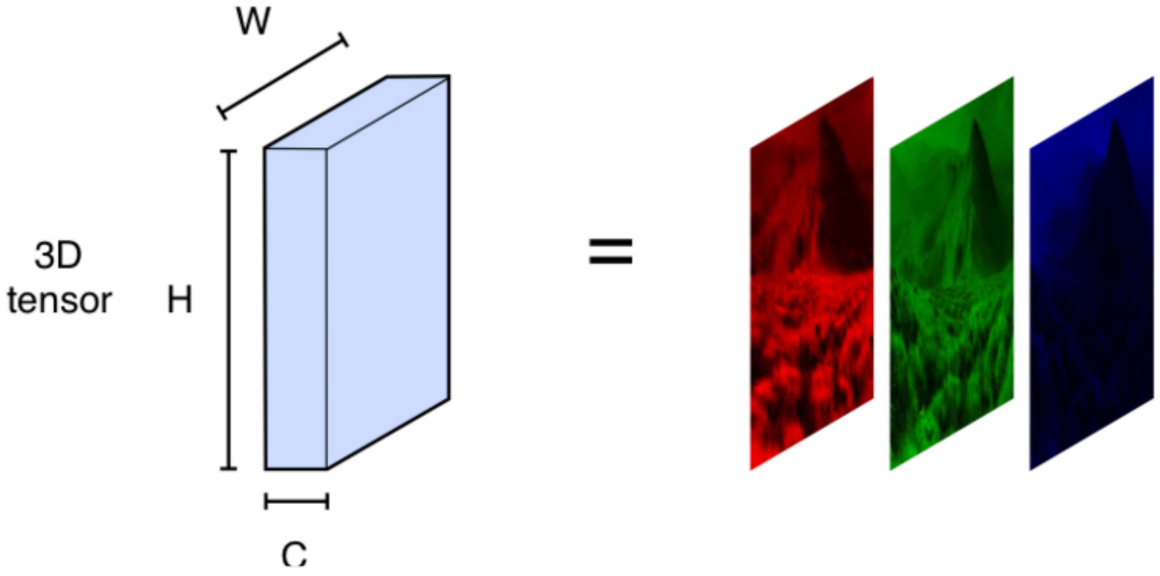
Gray Scale Image as Matrix



https://www.esantus.com/blog/2019/1/31/convolutional-neural-networks-a-quick-guide-for-newbies

RGB Image as Tensor





https://www.esantus.com/blog/2019/1/31/convolutional-neural-networks-a-quick-guide-for-newbies

NumPy

- **행렬(matrix)**이나 대규모 다차원 **배열(array)**을 쉽게 처리 할 수 있도록 지원하는 **Python library**
- NumPy는 **데이터 구조** 외에도 **수치 계산**을 위해 효율적으로 구현된 기능을 제공한다.
- TensorFlow의 텐서(tensor)는 **사용법**에 있어 NumPy n차원 배열(array)과 매우 **유사**하다.
- TensorFlow의 텐서(tensor)와 NumPy의 배열(array)는 서로 **변환**(conversion)이 **가능**하다.

NumPy

NumPy Array Creation

- Conversion from other Python structures (e.g., lists, tuples)
- Intrinsic numpy array creation objects (e.g., arange, ones, zeros, etc.)
- Reading arrays from disk, either from standard or custom formats
- Creating arrays from raw bytes through the use of strings or buffers
- Use of special library functions (e.g., random)

https://docs.scipy.org/doc/numpy-1.15.0/user/basics.creation.html

NumPy Array Basics


```
import numpy as np

print(np.__version__)                                   # 1.15.4

a = np.zeros(5)
print(a)                                                # [0. 0. 0. 0. 0.]
print(type(a))                                          # <class ‘numpy.ndarray'>
print(type(a[0]))                                       # <class ‘numpy.float64’>
print(a.dtype)                                          # float64
print(a.size)                                           # 5
print(a.ndim)                                           # 1
print(a.shape)                                          # (5,)

a = np.array([1, 2, 3], dtype=‘int')
print(a.dtype)                                          # int64
a = a.astype(‘float’)                                   # int64 -> float64
print(a.dtype)                                          # float64

l = [1, 2, 3, 4, 5]                                     # Python’s list
a = np.array(l)
b = np.array([l])
c = np.array([[l]])
print(a, a.size, a.ndim, a.shape)                       # [1 2 3 4 5] 5, 1 (5,)
print(b, b.size, b.ndim, b.shape)                       # [[1 2 3 4 5]] 5 2 (1, 5)
print(c, c.size, c.ndim, c.shape)                       # [[[1 2 3 4 5]]] 5 3 (1, 1, 5)
```

NumPy Array Basics


```
print(np.sum(a))                    # 15 = 1+2+3+4+5
print(np.prod(a))                   # 120 = 1*2*3*4*5
print(np.mean(a))                   # 3.0 = (1+2+3+4+5)/5
print(np.var(a))                    # 2.0: variance
print(np.min(a))                    # 1: minimum
print(np.max(a))                    # 5: maximum
print(np.argmin(a))                 # 0: the index of the min. value
print(np.argmax(a))                 # 4: the index of the max. value

b = np.linspace(2, 10, 5)           # from 2 to 10, with 5 elements
print(b)                            # [2. 4. 6. 8. 10.]
print(a+b)                          # [3. 6. 9. 12. 15.]: element-wise addition
print(a*b)                          # [2. 8. 18. 32. 50.]: element-wise multiplication
print(a+1)                          # [2 3 4 5 6]: lift
print(a*2)                          # [2 4 6 8 10]: scale
print(a@b)                          # 110: dot product

print(a[1:3])                       # [2 3]
print(a[1:])                        # [2 3 4 5]
print(a[:3])                        # [1 2 3]
print(a[-1])                        # 5: the last element
print(a[:-1])                       # [1 2 3 4]
```

NumPy Array Basics


```
print(a[[0, 4])                                    # [1 5]
print(a<3)                                         # [True True False False False]
print(a[a<3])                                      # [1 2]

a = np.linspace(1, 10, 10, dtype='i')
print(a)                                           # [1 2 3 4 5 6 7 8 9 10]
print(a[::2])                                      # [1 3 5 7 9]
print(a[::3])                                      # [1 4 7 10]
print(a[::-1])                                     # [10 9 8 7 6 5 4 3 2 1]
print(a[::-2])                                     # [10 8 6 4 2]
print(a[::-3])                                     # [10 7 4 1]
print(a[2:10:2])                                   # [3 5 7 9]

a = np.array([[1, 2, 3], [4, 5, 6]])
print(a.ndim, a.shape)                             # 2 (2, 3)
print(a)                                           # [[1 2 3] [4 5 6]]
print(a[::-1, :])                                  # [[4 5 6] [1 2 3]]
print(a[:, ::-1])                                  # [[3 2 1] [6 5 4]]

print(np.arange(5))                                # [1 2 3 4 5]
print(np.arange(3, 10))                            # [3 4 5 6 7 8 9]
print(np.arange(3, 10 step=2))                     # [3 5 7 9]
print(np.arange(3, 10 step=2, dtype='float32'))    # [3. 5. 7. 9.]
```

NumPy Array Basics


```
a = np.arange(3, 10)
print(a)                                  # [3 4 5 6 7 8 9]
print(np.random.choice(a, 3))             # [6 3 5]: different for each run

np.random.shuffle(a)
print(a)                                  # [5 3 8 4 9 6 7]: different for each run

print(np.random.randint(7))               # 3 (0~7 by uniform distrib.): different for each run
print(np.random.randint(0, 3, size=(2,2)) # [[0 1] [2 2]]: different for each run

# uniform distribution
print(np.random.rand(2))                  # [0.72916817 0.68116289]: different for each run

# Gaussian distribution
print(np.random.randn(2))                 # [0.64210077 -0.56222559]: different for each run

np.random.seed(123)
a = np.random.randint(10, size=5)
print(a)                                  # [2 2 6 1 3]: always same
print(np.sort(a))                         # [0 3 3 5 7]
```

Python List vs NumPy Array

- **Python List**
  - 다양한 데이터형을 혼합하여 각 성분으로 가질 수 있음: integer, string, float, list, etc.
  - 생성한 후 크기(size)를 조절할 수 있음
- **NumPy Array**
  - 여러 가지 데이터형을 혼합하여 가질 수 없음
  - 생성 후 크기(size)를 조절할 수 없음

Python List vs NumPy Array

```
import numpy as np

a = [1, 2]
b = [3, 4]

print(len(a), a)          # 2 [1, 2]
print(len(b), b)          # 2 [3, 4]
print(a + b)              # [1, 2, 3, 4]: appending

aa = np.array(a)
bb = np.array(b)

print(aa.ndim, aa)        # 1 [1,2]
print(bb.ndim, bb)        # 1 [3,4]
print(aa + bb)            # [4, 6]: element-wise addition

#######################################################

a = [123, ‘abc’]
print(a)                  # [123, ‘abc’]: integer & string
aa = np.array(a)
print(aa)                 # [‘123’, ‘abc’]: string & string
print(aa.dtype)           # <U21: not int64
```

from Python List to NumPy Array

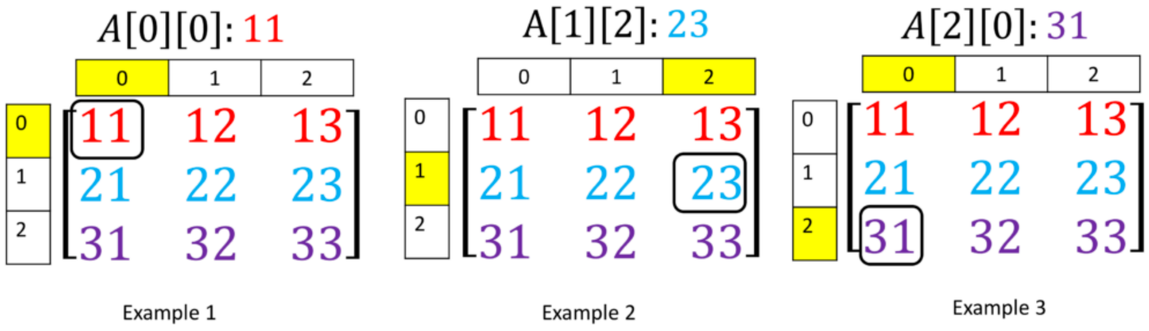
$$A = [[11, 12, 13], [21, 22, 23], [31, 32, 33]]$$

$$\begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{bmatrix}$$

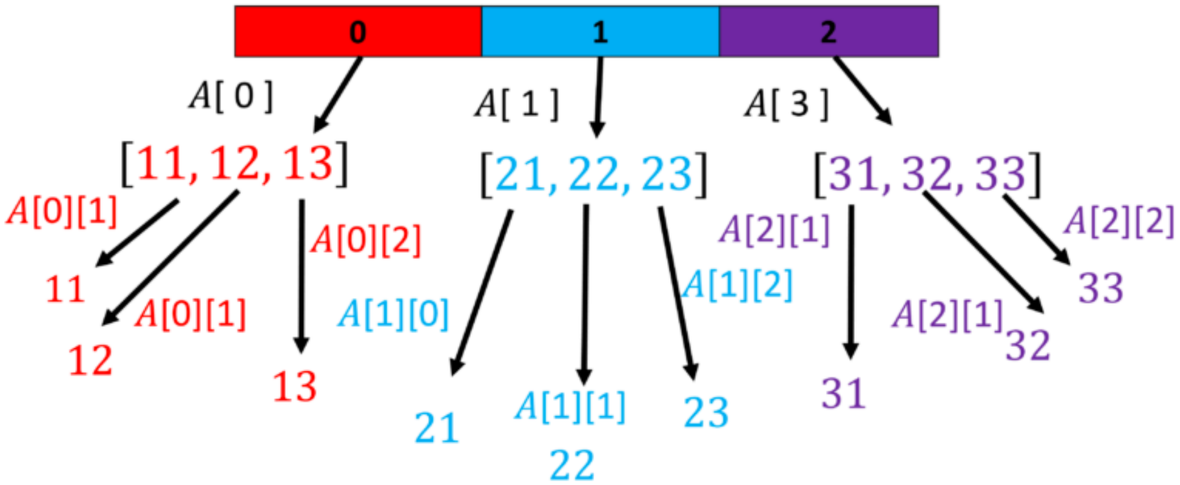
$$A = [[A[0][0], A[0][1], A[0][2]], [A[1][0], A[1][1], A[1][2]][A[2][0], A[2][1], A[2][2]]]$$

$$\begin{bmatrix} A[0][0] & A[0][1] & A[0][2] \\ A[1][0] & A[1][1] & A[1][2] \\ A[2][0] & A[2][1] & A[2][2] \end{bmatrix}$$

$$A = [[11, 12, 13], [21, 22, 23], [31, 32, 33]]$$



$$A = [[11, 12, 13], [21, 22, 23], [31, 32, 33]]$$



https://cognitiveclass.ai/blog/nested-lists-multidimensional-numpy-arrays/

3D NumPy Array

A=[[[111,112,113],[121,122,123],[131,132,133]],[[211,212,213],[221,222,123],[231,232,233]],[[311,212,213],[321,222,123],[331,332,333]]]

A[0]

Contents in list A[0]

[[111,112,113],[121,122,123],[131,132,133]]

rectangular table representing the values

111	112	113
121	122	123
131	132	133

rectangular table representing the index

A[0][0][0]	A[0][0][1]	A[0][0][2]
A[0][1][0]	A[0][1][1]	A[0][1][2]
A[0][2][0]	A[0][2][1]	A[0][2][2]

A[1]

Contents in list A[1]

[[211,212,213],[221,222,123],[231,232,233]]

rectangular table representing the values

211	212	213
221	222	223
231	232	233

rectangular table representing the index

A[1][0][0]	A[1][0][1]	A[1][0][2]
A[1][1][0]	A[1][1][1]	A[1][1][2]
A[1][2][0]	A[1][2][1]	A[1][2][2]

A[2]

Contents in list A[2]

[[311,212,213],[321,222,123],[331,332,333]]]

rectangular table representing the values

311	312	313
321	322	323
331	332	333

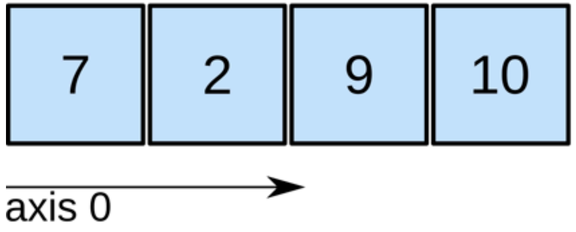
rectangular table representing the index

A[2][0][0]	A[2][0][1]	A[1][0][2]
A[2][1][0]	A[2][1][1]	A[1][1][2]
A[2][2][0]	A[2][2][1]	A[2][2][2]

https://cognitiveclass.ai/blog/nested-lists-multidimensional-numpy-arrays/

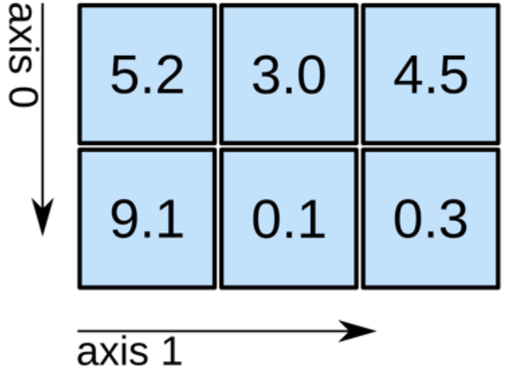
Shape & Axis

1D array



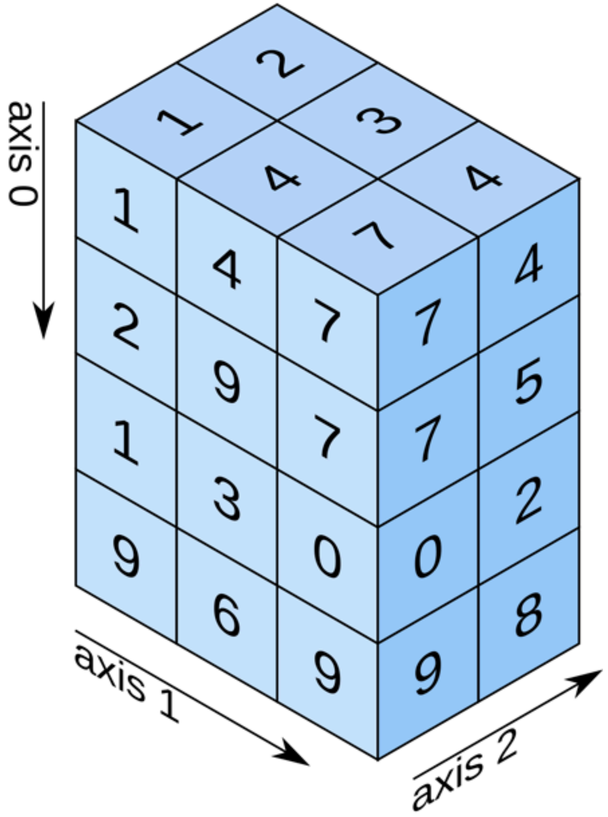
shape: (4,)

2D array



shape: (2, 3)

3D array



shape: (4, 3, 2)

https://medium.com/datadriveninvestor/artificial-intelligence-series-part-2-numpy-walkthrough-64461f26af4f

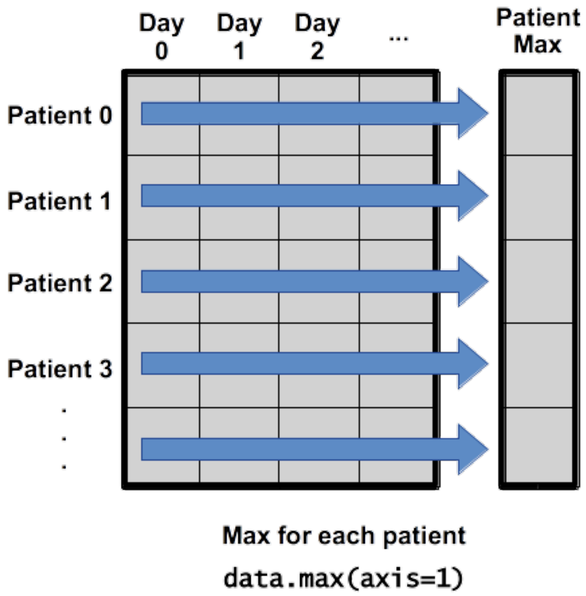
squeeze()

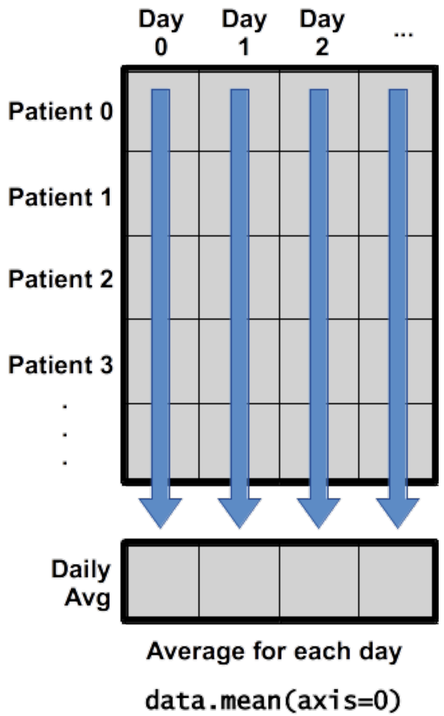
- 차원(dimension)이 1인 축(axis)을 찾아서 없애준다.

```
import numpy as np

a = np.random.randn(1, 3)
print(a.shape)        # (1, 3)
print(a)              # [[-0.26091455 -1.13381306  0.77830865]]

b = np.squeeze(a)
print(b)              # [-0.26091455 -1.13381306  0.77830865]
```

NumPy Array Axis

```
import numpy as np

a = np.array([[1,2,3], [4,5,6]])

print(a)              # [[1 2 3] [4 5 6]]
print(a[::-1,])       # [[4 5 6] [1 2 3]]
print(a[:,::-1)       # [[3 2 1] [6 5 4]]
print(a[::-1,::-1)    # [[6 5 4] [3 2 1]]

print(a.flatten())    # [1 2 3 4 5 6]

print(a.min(axis=0)) # [1 2 3]
print(a.min(axis=1)) # [1 4]

print(a.max(axis=0)) # [4 5 6]
print(a.max(axis=1)) # [3 6]

print(a.sum(axis=0)) # [5 7 9]
print(a.sum(axis=1)) # [6 15]
```





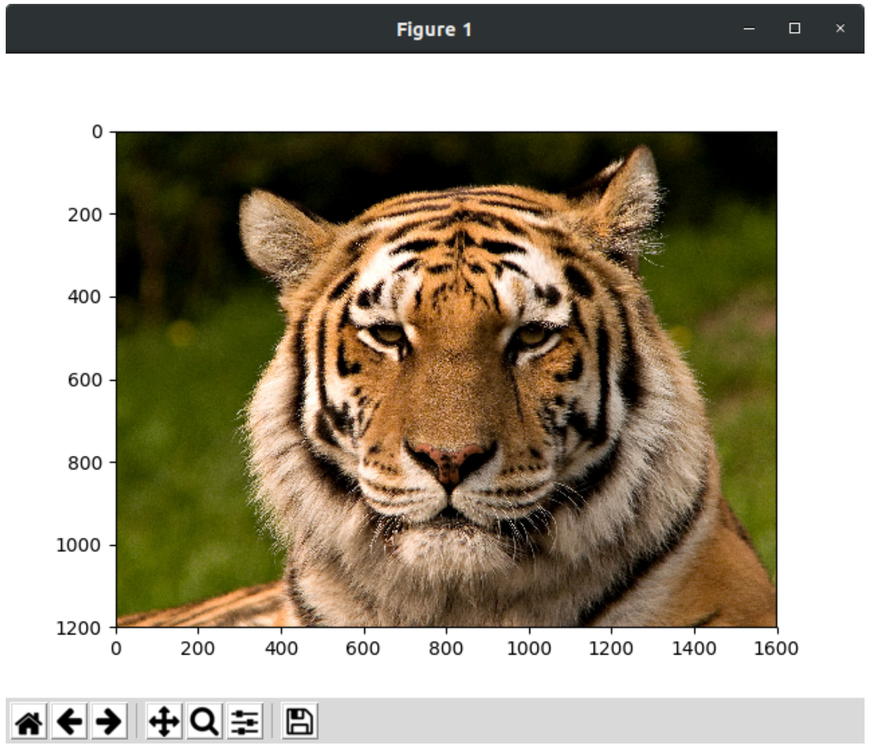
https://swcarpentry.github.io/python-novice-inflammation/01-numpy/

from NumPy Array to Python List

```
import numpy as np
l = [[1,2,3], [4,5,6], [7,8,9]]  # Python List
a = np.array(l)                  # NumPy Array

aa = np.split(a, 3, axis=1)

x=aa[0]; y=aa[1]; z=aa[2]
print(type(x), type(y), type(z)) # <class 'numpy.ndarray'> <class ‘numpy.ndarray'> <class ‘numpy.ndarray'>
print(x, y, z)                   # [[1] [4] [7]]            [[2] [5] [8]]           [[3] [6] [9]]

x=aa[0].tolist(); y=aa[1].tolist(); z=aa[2].tolist()
print(type(x), type(y), type(z)) # <class 'list'>           <class 'list'>        <class ‘list'>
print(x, y, z)                   # [[1], [4], [7]]          [[2], [5], [8]]       [[3], [6], [9]]

x=aa[0].flatten().tolist(); y=aa[1].flatten().tolist(); z=aa[2].flatten().tolist()
print(type(x), type(y), type(z)) # <class 'list'>           <class 'list'>        <class ‘list'>
print(x, y, z)                   # [1, 4, 7]                [2, 5, 8]             [3, 6, 9]

x=a[:,0]; y=a[:,1]; z=a[:,2];
print(type(x), type(y), type(z)) # <class 'numpy.ndarray'> <class 'numpy.ndarray'> <class ‘numpy.ndarray'>
print(x, y, z)                   # [1 4 7]                  [2 5 8]               [3 6 9]
```

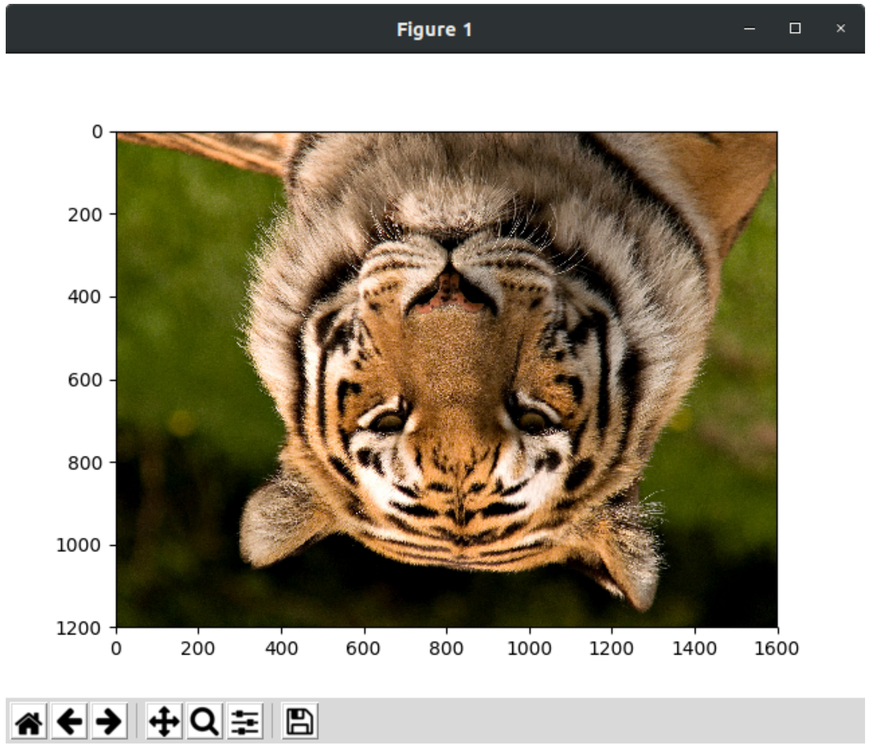
NumPy Image Basics

```
import numpy as np
from skimage import io
import matplotlib.pyplot as plt

image = io.imread('../data/image/tiger.jpg')
print(type(image)) # <class 'numpy.ndarray'>
print(image.shape) # (1200, 1600, 3)
print(image.dtype) # uint8

plt.imshow(image)
plt.show()
```



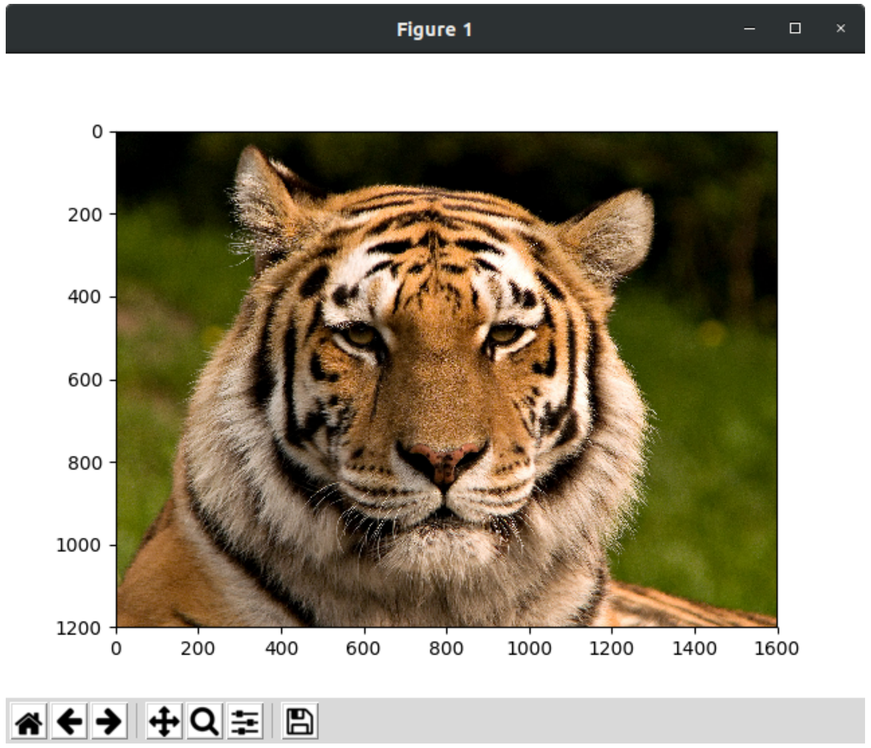
NumPy Image Basics

```
# flipping
plt.imshow(image[::-1, :, :])
plt.show()

plt.imshow(image[:, ::-1, :])
plt.show()

plt.imshow(image[:, :, ::-1])
plt.show()

# cropping
plt.imshow(image[400:800, 600:1100])
plt.show()

# resizing
plt.imshow(image[::10, ::10])
plt.show()

# thresholding
plt.imshow(image[:,:,0] > 200, cmap=plt.cm.gray)
plt.show()
```



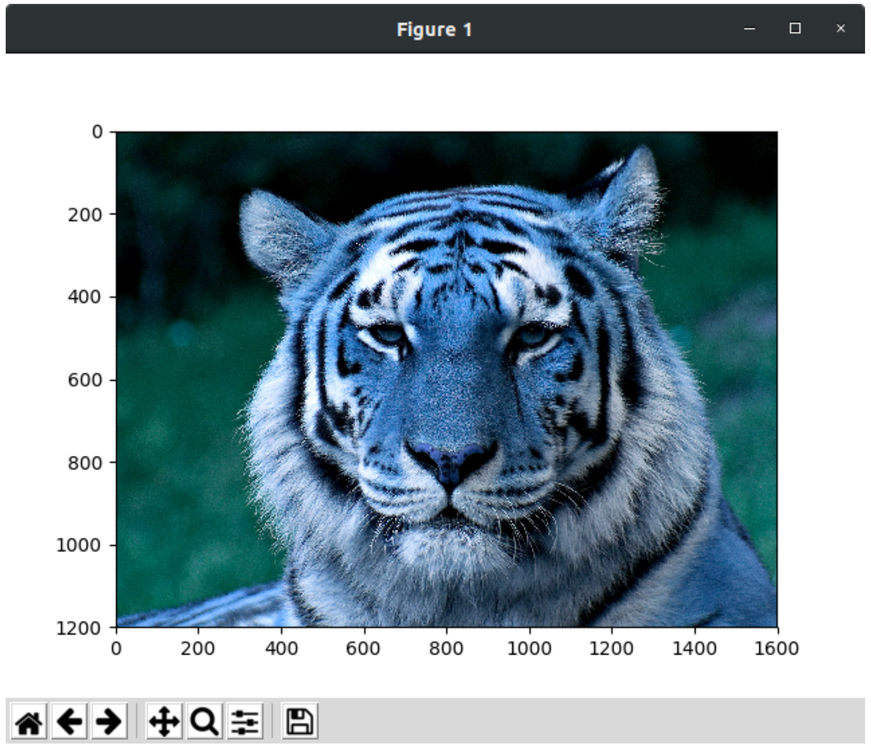
NumPy Image Basics

```
# flipping
plt.imshow(image[::-1, :, :])
plt.show()

plt.imshow(image[:, ::-1, :])
plt.show()

plt.imshow(image[:, :, ::-1])
plt.show()

# cropping
plt.imshow(image[400:800, 600:1100])
plt.show()

# resizing
plt.imshow(image[::10, ::10])
plt.show()

# thresholding
plt.imshow(image[:,:,0] > 200, cmap=plt.cm.gray)
plt.show()
```



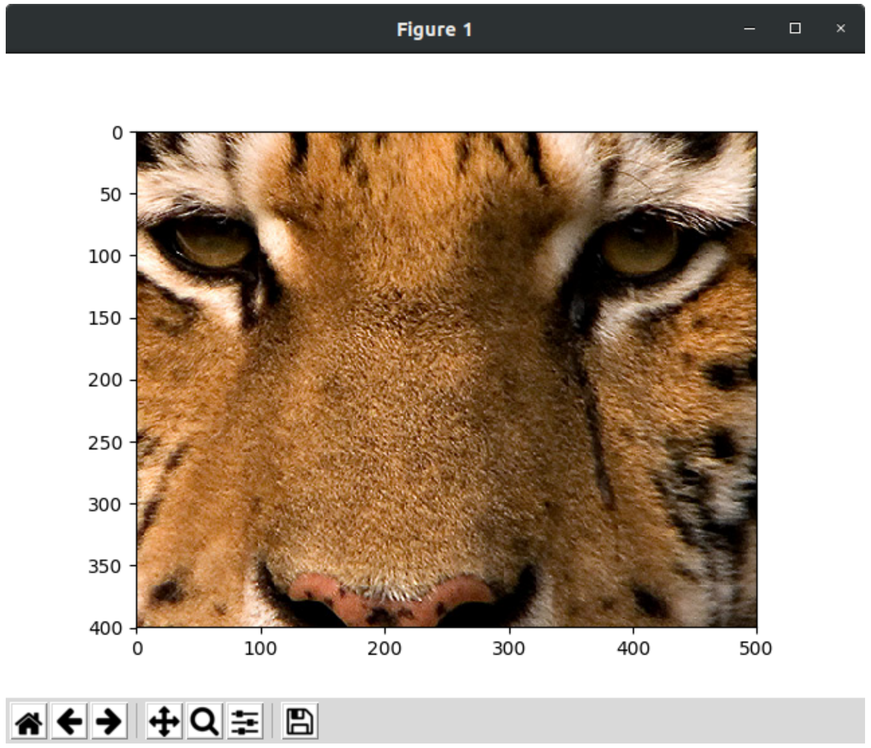
NumPy Image Basics

```
# flipping
plt.imshow(image[::-1, :, :])
plt.show()

plt.imshow(image[:, ::-1, :])
plt.show()

plt.imshow(image[:, :, ::-1])
plt.show()

# cropping
plt.imshow(image[400:800, 600:1100])
plt.show()

# resizing
plt.imshow(image[::10, ::10])
plt.show()

# thresholding
plt.imshow(image[:,:,0] > 200, cmap=plt.cm.gray)
plt.show()
```



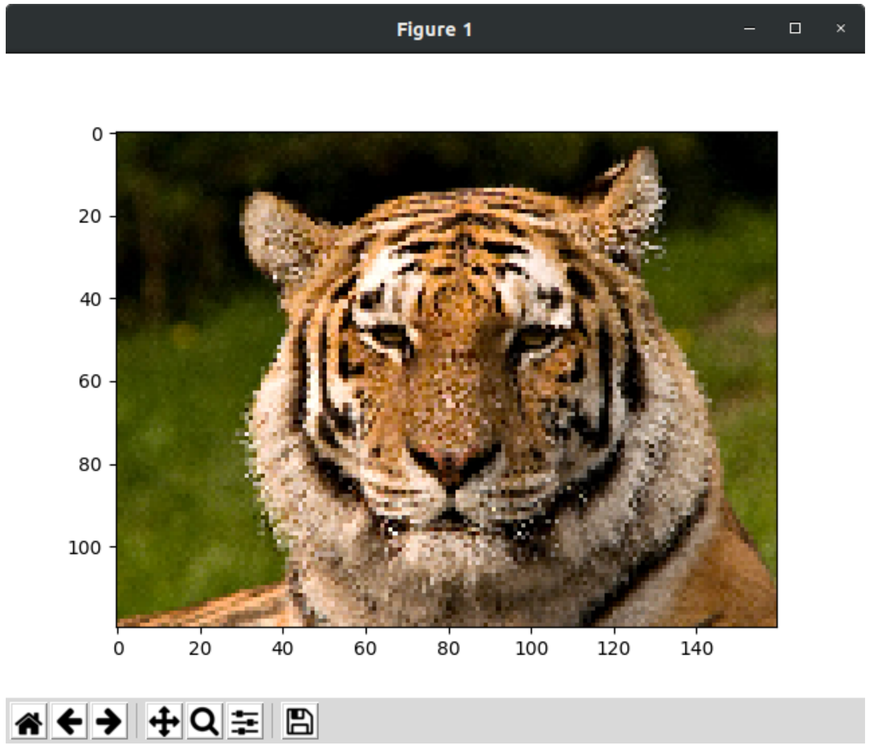
NumPy Image Basics

```
# flipping
plt.imshow(image[::-1, :, :])
plt.show()

plt.imshow(image[:, ::-1, :])
plt.show()

plt.imshow(image[:, :, ::-1])
plt.show()

# cropping
plt.imshow(image[400:800, 600:1100])
plt.show()

# resizing
plt.imshow(image[::10, ::10])
plt.show()

# thresholding
plt.imshow(image[:,:,0] > 200, cmap=plt.cm.gray)
plt.show()
```



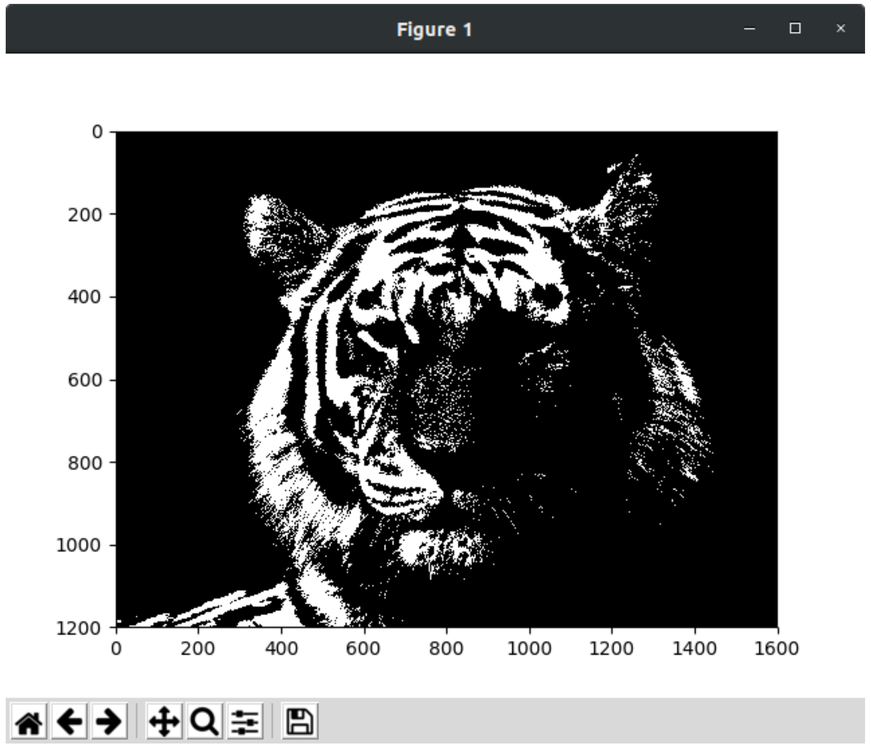
NumPy Image Basics

```
# flipping
plt.imshow(image[::-1, :, :])
plt.show()

plt.imshow(image[:, ::-1, :])
plt.show()

plt.imshow(image[:, :, ::-1])
plt.show()

# cropping
plt.imshow(image[400:800, 600:1100])
plt.show()

# resizing
plt.imshow(image[::10, ::10])
plt.show()

# thresholding
plt.imshow(image[:,:,0] > 200, cmap=plt.cm.gray)
plt.show()
```



NumPy Image Basics

```
# flipping
plt.imshow(image[::-1, :, :])
plt.show()

plt.imshow(image[:, ::-1, :])
plt.show()

plt.imshow(image[:, :, ::-1])
plt.show()

# cropping
plt.imshow(image[400:800, 600:1100])
plt.show()

# resizing
plt.imshow(image[::10, ::10])
plt.show()

# thresholding
plt.imshow(image[:,:,0] > 200, cmap=plt.cm.gray)
plt.show()
```



TensorFlow Basics

Hello TensorFlow!

```
# the canonical import statement
import tensorflow as tf

print(tf.__version__)
print(tf.VERSION)

print(tf.keras.__version__)
```

```
import tensorflow as tf

a = tf.constant("Hello TensorFlow!")

sess = tf.Session()

print(sess.run(a)) # b'Hello TensorFlow!

sess.close()
```

```
import tensorflow as tf

a = tf.constant(3)
b = tf.constant(4)
c = tf.add(a, b) # same as c = a + b

d = tf.constant("Hello ")
e = tf.constant("TensorFlow!")
f = tf.add(d, e) # same as f = d + e

sess = tf.Session()

print(sess.run(c)) # 7
print(sess.run(f)) # b'Hello TensorFlow!

sess.close()
```

Log Level Control

- 안보이게 하려면 다음과 같이 환경변수 설정: **export TF_CPP_MIN_LOG_LEVEL=2**

```
>>> import tensorflow as tf
>>> sess = tf.Session()

2018-07-06 10:00:14.552604: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions
that this TensorFlow binary was not compiled to use: AVX2 FMA
2018-07-06 10:00:15.231119: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:898] successful NUMA node read
from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2018-07-06 10:00:15.232472: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1356] Found device 0 with
properties:
name: GeForce GTX 1070 major: 6 minor: 1 memoryClockRate(GHz): 1.797
pciBusID: 0000:01:00.0
totalMemory: 7.93GiB freeMemory: 7.40GiB
2018-07-06 10:00:15.232548: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1435] Adding visible gpu devices: 0
2018-07-06 10:00:27.698002: I tensorflow/core/common_runtime/gpu/gpu_device.cc:923] Device interconnect
StreamExecutor with strength 1 edge matrix:
2018-07-06 10:00:27.698029: I tensorflow/core/common_runtime/gpu/gpu_device.cc:929] 0
2018-07-06 10:00:27.698035: I tensorflow/core/common_runtime/gpu/gpu_device.cc:942] 0:N
2018-07-06 10:00:27.710571: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1053] Created TensorFlow device (/
job:localhost/replica:0/task:0/device:GPU:0 with 7150 MB memory) -> physical GPU (device: 0, name: GeForce GTX
1070, pci bus id: 0000:01:00.0, compute capability: 6.1)
```

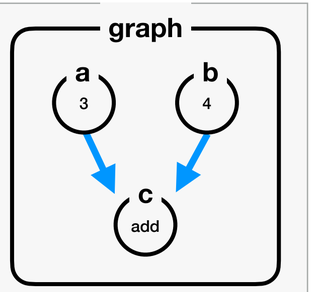
Session

- A **class** for **run**ning TensorFlow **operations**.
- **지연 실행** (**lazy evaluation** or **lazy computation**): graph의 생성과 실행 과정 분리

```
import tensorflow as tf

# Create a graph.
a = tf.constant(3)
b = tf.constant(4)
c = a + b # = tf.add(a, b)
```

name과 data type이 자동으로 지정됨

```
print(a) # Tensor("Const:0", shape=(), dtype=int32)
print(b) # Tensor("Const_1:0", shape(), dtype=int32)
print(c) # Tensor("Const_2:0", shape=(), dtype=int32)

# Create a session, and launch the graph in the session.
sess = tf.Session()

# Evaluate the graph.
print(sess.run(a),"+",sess.run(b),"=",sess.run(c)) # 3 + 4 = 7

# Release the resource
sess.close()
```




```
import tensorflow as tf

a = tf.constant(3)
b = tf.constant(4)

print(a, b)

# Resource Release Method 1)
# Using the close() method
sess = tf.Session()
sess.run(a + b)
sess.close()

# Resource Release Method 2)
# Using the context manager
with tf.Session() as sess:
  sess.run(a + b)
```

Session

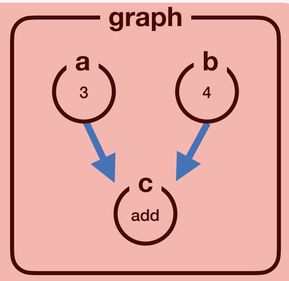
- A **class** for **run**ning TensorFlow **operations**.
- **지연 실행** (**lazy evaluation** or **lazy computation**): graph의 생성과 실행 과정 분리

```
import tensorflow as tf

# Create a graph.
a = tf.constant(3)
b = tf.constant(4)
c = a + b # = tf.add(a, b)
```

Graph Creation

name과 data type이 자동으로 지정됨

```
print(a) # Tensor("Const:0", sh    =(), dtype=int32)
print(b) # Tensor("Const_1:0",    pe(), dtype=int32)
print(c) # Tensor("Const_2:0",    pe=(), dtype=int32)
```




```
# Create a session, and launch    graph in the session.
sess = tf.Session()

# Evaluate the graph.
print(sess.run(a),"+",sess.run(b),"=",sess.run(c)) # 3 + 4 = 7

# Release the resource
sess.close()
```

Graph Evaluation

```
import tensorflow as tf

a = tf.constant(3)
b = tf.constant(4)

print(a, b)

# Resource Release Method 1)
# Using the close() method
sess = tf.Session()
sess.run(a + b)
sess.close()

# Resource Release Method 2)
# Using the context manager
with tf.Session() as sess:
  sess.run(a + b)
```

TF가 GPU 가속 확인 방법

```
import tensorflow as tf

# 7
with tf.device('/gpu:0'):
    a = tf.constant(3)
    b = tf.constant(4)
    c = tf.add(a, b)

with tf.Session() as sess:
    print(sess.run(c))

# [_DeviceAttributes(/job:localhost/replica:0/task:0/device:CPU:0, CPU, 268435456,
11611345940698844734), _DeviceAttributes(/job:localhost/replica:0/task:0/device:XLA_CPU:0,
XLA_CPU, 17179869184, 4807632777874203135), _DeviceAttributes(/job:localhost/replica:0/task:0/
device:XLA_GPU:0, XLA_GPU, 17179869184, 16295880932032667 39), _DeviceAttributes(/job:localhost/
replica:0/task:0/device:GPU:0, GPU, 10123594957, 3978246611461485481)
with tf.Session() as sess:
    devices = sess.list_devices()
    print(devices)

# Device mapping:
# /job:localhost/replica:0/task:0/device:XLA_CPU:0 -> device: XLA_CPU device
# /job:localhost/replica:0/task:0/device:XLA_GPU:0 -> device: XLA_GPU device
# /job:localhost/replica:0/task:0/device:GPU:0 -> device: 0, name: GeForce RTX 2080 Ti, pci bus
id: 0000:01:00.0, compute capability: 7.5
sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))
```

Session::run() and Its Return

```
import tensorflow as tf

a = tf.constant(1.0)
b = tf.constant(2.0)

sess = tf.Session()

print(sess.run(a + b)) # 3.0
print(sess.run(a - b)) # -1.0
print(sess.run(a * b)) # 2.0
print(sess.run(a / b)) # 0.5

sess.close()
```

```
import tensorflow as tf

a = tf.constant(1.0)
b = tf.constant(2.0)

c = a + b
d = a - b
e = a * b
f = a / b

sess = tf.Session()

# C = sess.run(c)
# E = sess.run(e)
C, _, E, _ = sess.run([c,d,e,f])

print(C) # 3
print(E) # 2.0

sess.close()
```

Constant / Variable / Placeholder

- **Constant**
  - 한 번 값을 설정하면 변경될 수 없는 상수
  - Global constant variable
  - C언어에서의 const variable과 비슷한 개념
- **Variable**
  - 실행중에 값이 변경될 수 있는 변수
  - Weight, bias 등
  - C언어에서의 variable과 비슷한 개념
- **Placeholder**
  - 일단 초기값이 없이 선언한 후에 후에 실행시 필요한 값을 받아서 채워지는 변수
  - 딥러닝에 사용되는 학습 데이터 (training data) 등
  - C언어에서의 pointer와 비슷한 개념

Constant / Variable / Placeholder

```
import tensorflow as tf
tf.logging.set_verbosity(tf.logging.ERROR)

# constant
a = tf.constant(1)
b = tf.constant(2)

# variable
c = tf.Variable(1)
d = tf.Variable(b)

# placeholder
e = tf.placeholder(tf.int32)
f = tf.placeholder(tf.int32)

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

print(sess.run(a+b))                          # 3
print(sess.run(c+d))                          # 3
print(sess.run(e+f, feed_dict={e:1, f:2}))    # 3
print(sess.run(a+b+c+d))                      # 6
print(sess.run(a+e, feed_dict={e:3}))         # 4

sess.close()
```

만약 colocate_with is deprecated.. 경고가 발생한다면 이 구문을 사용하여 경구 문구가 안나오도록 해준다.

variable은 constant, placeholder와는 달리 대문자 V로 시작한다.

variable은 constant로 부터 초기화하는 것이 가능하다.

placeholder는 처음 생성시 데이터 타입을 지정해주어야 한다.

TensorFlow에서 variable은 그래프를 실행하기 전에 초기화를 해주어야 설정한 값이 변수에 지정된다.

placeholder는 feed_dict를 사용하여 그래프 실행시에 값을 지정해 준다.

constant, variable, placeholder 끼리의 연산이 가능하다.

Variable Initialization #1

```
import tensorflow as tf
tf.logging.set_verbosity(tf.logging.ERROR)

a = tf.Variable(1)
b = tf.Variable(2)

sess = tf.Session()

# Initialization Method 1)
sess.run(tf.variables_initializer([a,b]))

# Initialization Method 2)
sess.run(tf.variables_initializer(tf.global_variables()))

# Initialization Method 3)
sess.run(tf.global_variables_initializer())

print(sess.run(a+b)) # 3

sess.close()
```

Variable Initialization #2

```
import tensorflow as tf
tf.logging.set_verbosity(tf.logging.ERROR)

a = tf.Variable(1)
b = tf.Variable(2)

# Initialization Method 1)
init = tf.variables_initializer([a,b])

# Initialization Method 2)
init = tf.variables_initializer(tf.global_variables())

# Initialization Method 3)
init = tf.global_variables_initializer()

sess = tf.Session()

sess.run(init)

print(sess.run(a+b)) # 3

sess.close()
```

Data Types

Data Type	Python Type	Description
DT_INT8	tf.int8	8-bit signed integer
DT_INT16	tf.int16	16-bit signed integer
DT_INT32	tf.int32	32-bit signed integer
DT_INT64	tr.int64	64-bit signed integer
DT_FLOAT	tf.float32	32-bit single-precision floating-point
DT_DOUBLE	tf.float64	64-bit double-precision floating-point
DT_BOOL	tf.bool	boolean
DT_STRING	tf.string	string

https://www.tensorflow.org/api_docs/python/tf/dtypes/DType

Constant: Name & Type

```
import tensorflow as tf

a = tf.constant(1)
print(a.name, a.dtype) # Const:0 <dtype: 'int32'>

a = tf.constant(1, tf.float32)
print(a.name, a.dtype) # Const_1:0 <dtype: 'float32'>

a = tf.constant(1, name="aaa")
print(a.name, a.dtype) # aaa:0 <dtype: 'int32'>

a = tf.constant(1, tf.float32, name="aaa")
print(a.name, a.dtype) # aaa_1:0 <dtype: 'float32'>

a = tf.constant(1, tf.float64, name="aaa")
print(a.name, a.dtype) # aaa_2:0 <dtype: 'float64'>

a = tf.constant(1.0)
print(a.name, a.dtype) # Const_2:0 <dtype: 'float32'>

a = tf.constant(True, tf.bool)
print(a.name, a.dtype) # Const_3:0 <dtype: 'bool'>

a = tf.constant("abc", tf.string)
print(a.name, a.dtype) # Const_4:0 <dtype: 'string'>
```

소수점을 사용하지 않으면 자동으로 int_32 정수로 인식된다.

name을 명시해주지 않으면 이름이 자동으로 할당된다.

소수점을 사용하지 않더라도 dtype을 명시해주면 실수 형태로 사용할 수 있다.

name을 명시해줄 수 있다.

dtype과 name을 동시에 명시해줄 수 있다.

1 대신 1.0을 써주면 자동으로 float_32 형태로 인식된다.

Variable: Name & Type

```
import tensorflow as tf

a = tf.Variable(1)
print(a.name, a.dtype) # Variable:0 <dtype: 'int32_ref'>

a = tf.Variable(1, tf.float32)
print(a.name, a.dtype) # Variable_1:0 <dtype: 'int32_ref'>

a = tf.Variable(1, name="aaa")
print(a.name, a.dtype) # aaa:0 <dtype: 'int32_ref'>

a = tf.Variable(1, tf.float32, name="aaa")
print(a.name, a.dtype) # aaa_1:0 <dtype: 'int32_ref'>

a = tf.Variable(1, tf.float64, name="aaa")
print(a.name, a.dtype) # aaa_2:0 <dtype: 'int32_ref'> ➜ bug??

a = tf.Variable(1.0)
print(a.name, a.dtype) # Variable_2:0 <dtype: 'float32_ref'>

a = tf.Variable(True, tf.bool)
print(a.name, a.dtype) # Variable_3:0 <dtype: 'bool_ref'>

a = tf.Variable("abc", tf.string)
print(a.name, a.dtype) # Variable_4:0 <dtype: 'string_ref'>
```

소수점을 사용하지 않으면 자동으로 int_32 정수로 인식된다.
constant와는 달리 variable은 reference 형태를 가진다.

dtype을 명시해주더라도 소수점을 사용하지 않으면 int_32로 인식된다.
따라서, dtype을 사용할 필요가 없다.

constant와 동일한 방식으로 name을 명시해줄 수 있다.

Data Shape

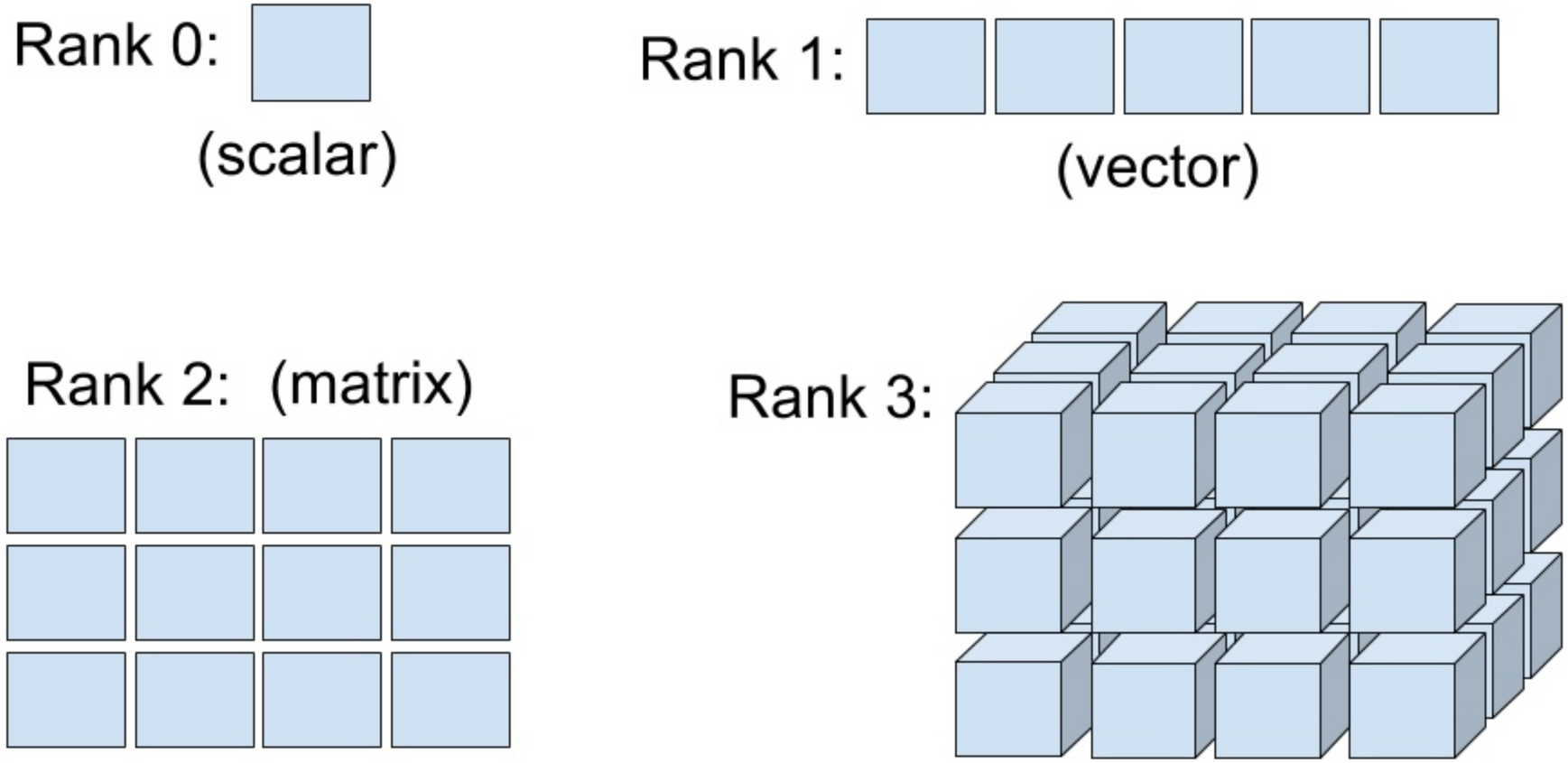
Rank	Math Entity	Example
0	scalar	a=123
1	vector	a=[1, 2, 3]
2	matrix	a=[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
3	3-tensor	a=[[[1], [2], [3]], [[4], [5], [6]], [[7], [8], [9]]]
n	n-tensor	…

a=[[[1], [2], [3]], [[4], [5], [6]], [[7], [8], [9]]]

[[[의 개수=rank

]]]의 개수=rank

Rank



https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781787125933/14/ch14lvl1sec85/tensorflow-ranks-and-tensors

Constant: Data Shape

```
import tensorflow as tf

a = tf.constant(1)
print(a.shape) # ()

a = tf.constant([1, 2, 3])
print(a.shape) # (3,)

a = tf.constant([[1, 2, 3]])
print(a.shape) # (1,3)

a = tf.constant([1, 2, 3, 4, 5, 6])
print(a.shape) # (6,)

a = tf.constant([[1, 2, 3], [4, 5, 6]])
print(a.shape) # (2, 3)

a = tf.constant([[1, 2], [3, 4], [5, 6]])
print(a.shape) # (3, 2)

a = tf.constant([[[1],[2],[3]], [[4],[5],[6]]])
print(a.shape) # (2, 3, 1)

a = tf.constant([[[1], [2]], [[3], [4]], [[5], [6]]])
print(a.shape) # (3, 2, 1)

a = tf.constant([[[1,1,1],[2,2,2],[3,3,3]], [[4,4,4],[5,5,5],[6,6,6]]])
print(a.shape) # (2, 3, 3)
```

Variable: Data Shape

```
import tensorflow as tf

a = tf.Variable(1)
print(a.shape) # ()

a = tf.Variable([1, 2, 3])
print(a.shape) # (3,)

a = tf.constant([[1, 2, 3]])
print(a.shape) # (1,3)

a = tf.Variable([1, 2, 3, 4, 5, 6])
print(a.shape) # (6,)

a = tf.Variable([[1, 2, 3], [4, 5, 6]])
print(a.shape) # (2, 3)

a = tf.Variable([[1, 2], [3, 4], [5, 6]])
print(a.shape) # (3, 2)

a = tf.Variable([[[1],[2],[3]], [[4],[5],[6]]])
print(a.shape) # (2, 3, 1)

a = tf.Variable([[[1], [2]], [[3], [4]], [[5], [6]]])
print(a.shape) # (3, 2, 1)

a = tf.Variable([[[1,1,1],[2,2,2],[3,3,3]], [[4,4,4],[5,5,5],[6,6,6]]])
print(a.shape) # (2, 3, 3)
```

Reshape

- tf.reshape(tensor, shape, name=None): 텐서의 구조를 변경할때 사용함
- tensor의 원소로 shape의 구조가 채워진 텐서를 반환함
- shape에 의해 지정된 전체 원소의 개수는 원래 tensor의 원소의 개수와 동일해야 함
- -1을 사용하면 전체 크기가 일정하게 유지되도록 해당 차원의 길이가 계산됨
  (-1은 한 개 원소만 사용 가능)

Constant: Reshape

```
import tensorflow as tf
tf.logging.set_verbosity(tf.logging.ERROR)

a = tf.constant([1, 2, 3, 4])
print(a.shape) # (4,)

a = tf.reshape(a, [2,2])
print(a.shape) # (2,2)

a = tf.reshape(a, [1,4])
print(a.shape) # (1,4)

a = tf.reshape(a, [-1,2])
print(a.shape) # (2,2)

a = tf.reshape(a, [2,-1])
print(a.shape) # (2,2)

a = tf.reshape(a, [-1])
print(a.shape) # (4,)
```

Variable: Reshape

```
import tensorflow as tf
tf.logging.set_verbosity(tf.logging.ERROR)

a = tf.Variable([1, 2, 3, 4])
print(a.shape) # (4,)

a = tf.reshape(a, [2,2])
print(a.shape) # (2,2)

a = tf.reshape(a, [1,4])
print(a.shape) # (1,4)

a = tf.reshape(a, [-1,2])
print(a.shape) # (2,2)

a = tf.reshape(a, [2,-1])
print(a.shape) # (2,2)

a = tf.reshape(a, [-1])
print(a.shape) # (4,)
```

Constant: Initializer

```
import tensorflow as tf

x = [11, 22]
a = tf.constant(x)                array x로 부터 초기화를 하였음
b = tf.constant([11, 22])
c = tf.zeros([1, 2], tf.int32)    모든 원소를 0으로 채움
d = tf.zeros_like(a)
e = tf.ones([1, 2], tf.int32)     모든 원소를 1로 채움
f = tf.ones_like(a)
g = tf.fill([1, 2], 3)            모든 원소를 지정한 숫자로 채움
h = tf.linspace(1.0, 5.0, 3) # start, stop, num
i = tf.range(1.0, 3.0, 0.5)  # start, limit, delta

sess = tf.Session()

print(a.shape, sess.run(a)) # (2,) [11 22]
print(b.shape, sess.run(b)) # (2,) [11 22]
print(c.shape, sess.run(c)) # (1, 2) [[0 0]]
print(d.shape, sess.run(d)) # (2,) [0 0]
print(e.shape, sess.run(e)) # (1, 2) [[1 1]]
print(f.shape, sess.run(f)) # (2,) [1 1]
print(g.shape, sess.run(g)) # (1, 2) [[3 3]]
print(h.shape, sess.run(h)) # (3,) [1. 3. 5.]
print(i.shape, sess.run(i)) # (4,) [1.  1.5 2.  2.5]

sess.close()
```

Variable: Initializer

- Variable은 constant로 부터 초기화 설정이 가능하다.

```
import tensorflow as tf
tf.logging.set_verbosity(tf.logging.ERROR)

x = [11, 22]
a = tf.Variable(x)
b = tf.Variable(tf.constant([11, 22]))
c = tf.Variable(tf.zeros([1, 2], tf.int32))
d = tf.Variable(tf.zeros_like(a))
e = tf.Variable(tf.ones([1, 2], tf.int32))
f = tf.Variable(tf.ones_like(a))
g = tf.Variable(tf.fill([1, 2], 3))
h = tf.Variable(tf.linspace(1.0, 5.0, 3)) # start, stop, num
i = tf.Variable(tf.range(1.0, 3.0, 0.5))  # start, limit, delta

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

print(a.shape, sess.run(a)) # (2,) [11 22]
print(b.shape, sess.run(b)) # (2,) [11 22]
print(c.shape, sess.run(c)) # (1, 2) [[0 0]]
…

sess.close()
```

Constant: Random Tensor

```
a = tf.constant([[1,2], [3,4], [5,6]])
b = tf.random_shuffle(a)
```

a가 constant가 아닌 Variable일 때에도 동일하게 random_shuffle 함수를 사용할 수 있다.

```
tf.set_random_seed(1)

# random_uniform() outputs random values from a uniform distribution. [minVal, maxVal)
# random_normal() outputs random values from a normal distribution.
# [] determines the shape of the output tensor.

c = tf.random_uniform([])
d = tf.random_normal([])

e = tf.random_uniform([1])
f = tf.random_normal([1])

x = tf.random_uniform([2,3], minval=-1, maxval=1, seed=0)
y = tf.random_normal([2,3], mean=0, stddev=1, seed=0)
z = tf.truncated_normal([2,3], mean=0, stddev=1, seed=0)
```

https://tensorflowkorea.gitbooks.io/tensorflow-kr/content/g3doc/api_docs/python/constant_op.html

Math Operations (1/3)

```
import tensorflow as tf

a = tf.constant(3.0)
b = tf.constant(2.0)

sess = tf.Session()

print(sess.run(tf.abs(-a)))   # 3.0
print(sess.run(tf.sign(a)))   # 1.0
print(sess.run(tf.square(a))) # 9.0
print(sess.run(tf.round(a)))  # 3.0
print(sess.run(tf.sqrt(a)))   # 1.7320508
print(sess.run(tf.exp(a)))    # 20.085537
print(sess.run(tf.log(a)))    # 1.0986123
print(sess.run(tf.sin(a)))    # 0.14112
print(sess.run(tf.cos(a)))    # -0.9899925

sess.close()
```

```
import tensorflow as tf

a = tf.constant(3.0)
b = tf.constant(2.0)

sess = tf.Session()

print(sess.run(a + b))            # 5.0
print(sess.run(a - b))            # 1.0
print(sess.run(a * b))            # 6.0
print(sess.run(a / b))            # 1.5

print(sess.run(tf.mod(a, b)))     # 1.0
print(sess.run(tf.pow(a, b)))     # 9.0

print(sess.run(tf.minimum(a, b))) # 2.0
print(sess.run(tf.maximum(a, b))) # 3.0

sess.close()
```

https://www.tensorflow.org/api_docs/cc/group/math-ops

Math Operations (2/3)

```
import tensorflow as tf

a = tf.constant([-1.0, -2.0])
b = tf.Variable([+3.0, +4.0])

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

print(sess.run(tf.abs(a)))        # [1. 2.]
print(sess.run(tf.abs(b)))        # [3. 4.]
print(sess.run(tf.sign(a)))       # [-1. -1.]
print(sess.run(tf.sign(b)))       # [1. 1.]
print(sess.run(tf.square(a)))     # [1. 4.]
print(sess.run(tf.square(b)))     # [ 9. 16.]
print(sess.run(tf.maximum(a,b)))  # [3. 4.]
print(sess.run(tf.minimum(a,b)))  # [-1. -2.]
print(sess.run(tf.add(a,b)))      # [2. 2.]
print(sess.run(tf.subtract(a,b))) # [-4. -6.]
print(sess.run(tf.multiply(a,b))) # [-3. -8.]
print(sess.run(tf.divide(a,b)))   # [-0.33333334 -0.5]

sess.close()
```

Math Operations (3/3)

```
import tensorflow as tf
tf.logging.set_verbosity(tf.logging.ERROR)

a = tf.constant([-1.0, -2.0])
b = tf.Variable([+3.0, +4.0])

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

print(sess.run(tf.equal(a,b)))        # [False False]

print(sess.run(tf.reduce_mean(a)))    # -1.5
print(sess.run(tf.reduce_mean(b)))    # 3.5

print(sess.run(tf.cast(a, tf.int32))) # [-1 -2]
print(sess.run(tf.cast(b, tf.int32))) # [3 4]

sess.close()
```

Variable: Assign

```
import tensorflow as tf
tf.logging.set_verbosity(tf.logging.ERROR)

a = tf.constant(1)
b = tf.Variable(0)

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

print(sess.run(b.assign(b+a)))   # 1
print(sess.run(b.assign_add(a))) # 2
print(sess.run(b.assign_sub(a))) # 1

sess.close()
```

Variable: Accumulation

```
import tensorflow as tf

state = tf.Variable(0, name="state")

increase = tf.constant(1)
new_value = tf.add(state, increase)
update = tf.assign(state, new_value)

sess = tf.Session()

sess.run(tf.global_variables_initializer())

print(sess.run(state))

for _ in range(10):
    sess.run(update)
    print(sess.run(state))

sess.close()
```

result:

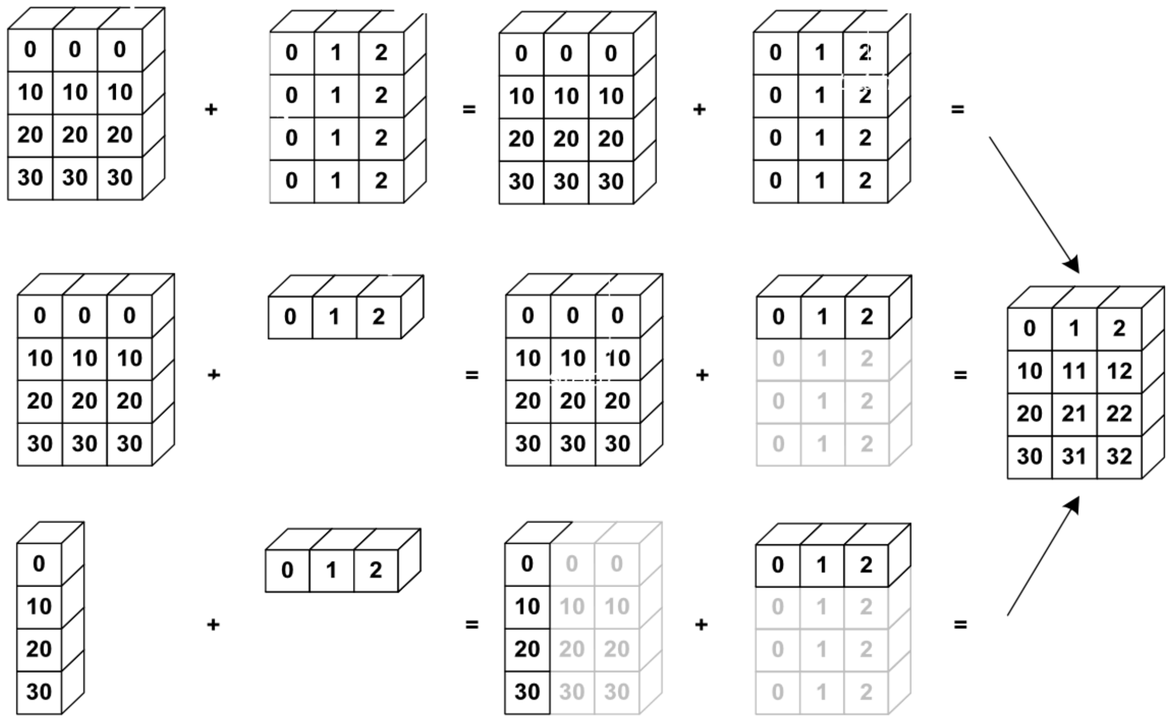
```
0
1
2
3
4
5
6
7
8
9
10
```

Matrix Operations

```
A = tf.constant([[1,2,3], [4,5,6]])       # shape=(2,3)
B = tf.constant([[1,1], [0,1], [-1,-1]]) # shape=(3,2)
AB = tf.matmul(A, B)                      # shape=(2,2)
c = tf.constant([10,20])                  # shape=(2,)
d = tf.constant([[10], [20]])             # shape=(2,1)
e = tf.constant(1)                        # shape=()

f = tf.constant([[1,2,3]])                # shape=(1,3)
g = tf.constant([[1], [2], [3]])          # shape=(3,1)

sess = tf.Session()

print(sess.run(A))                        # [[1 2 3] [4 5 6]]
print(sess.run(B))                        # [[1 1] [0 1] [-1 -1]]
print(sess.run(c))                        # [10 20]
print(sess.run(d))                        # [[10] [20]]
print(sess.run(e))                        # 1
print(sess.run(AB))                       # [[-2 0] [-2 3]]
print(sess.run(AB+c))                     # [[8 20] [8 23]]
print(sess.run(AB+d))                     # [[8 10] [18 23]]
print(sess.run(AB+e))                     # [[-1 1] [-1 4]]
print(sess.run(tf.matmul(f,g)))           # [[14]]

sess.close()
```

Broadcasting

- Shape이 다르더라도 연산이 가능하도록 해준다.
- 새로 추가되는 곳의 값을 기존값들을 이용해서 채워준다.
- 잘못 사용하면 예상하지 못한 결과를 얻을 수 있기 때문에 사용에 유의해야 한다.



https://jungwoon.github.io/tensorflow/toe/2017/12/19/MT-04-Tensor-Manipulation/

Broadcasting

```
import tensorflow as tf

a = tf.constant([[0,0,0], [10,10,10], [20,20,20], [30,30,30]])
b = tf.constant([0,1,2], [0,1,2], [0,1,2], [0,1,2]])

c = tf.constant([[0,0,0], [10,10,10], [20,20,20], [30,30,30]])
d = tf.constant([[0,1,2]])

e = tf.constant([[0], [10], [20], [30]])
f = tf.constant([[0,1,2]])

sess = tf.Session()

print(sess.run(a + b))
print(sess.run(c + d))
print(sess.run(e + f))

sess.close()
```

ArgMax / ArgMin

```
import tensorflow as tf

sess = tf.Session()

a = [0, 1, 0, 0]
print(sess.run(tf.argmax(a))) # 1
print(sess.run(tf.argmin(a))) # 0 (the first one)

b = [0.1, 0.3, 0.2, 0.4]
print(sess.run(tf.argmax(b))) # 3
print(sess.run(tf.argmin(b))) # 0

# | 4 1 6 |
# | 2 5 7 |
# | 3 9 3 |
c = [[4,1,6], [2,5,7], [3,9,3]]
print(sess.run(tf.constant(c))
print(sess.run(tf.argmax(c)))   # [0 2 1]: max. index in each column
print(sess.run(tf.argmax(c,0))) # [0 2 1]: max. index in each column
print(sess.run(tf.argmax(c,1))) # [2 2 1]: max. index in each row

sess.close()
```

Softmax

- 모든 요소들(elements)의 합(sum)이 1이 되도록 **정규화(normalization)** 해준다.
- 최종 **출력값**을 **확률**로서 사용하기 위해 많이 사용한다.

```
import tensorflow as tf

sess = tf.Session()

x = [1., 2., 3., 4., 5.]
y = tf.nn.softmax(x)

print(x)                    # [1.0, 2.0, 3.0, 4.0, 5.0]
print(sess.run(y))          # [0.01165623 0.03168492 0.08612854 0.23412165 0.6364086]

sum = 0.0

for i in range(len(x)):
    sum += sess.run(y[i])
    print(sum)              # 0.999999969266355

sess.close()
```

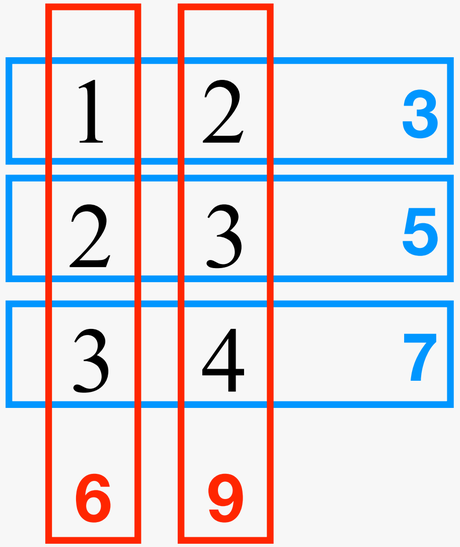
Placeholder

```
import tensorflow as tf

a = tf.placeholder(tf.float32)
b = tf.placeholder(tf.float32)

op1 = a + b
op2 = a - b
op3 = a * b
op4 = a / b

sess = tf.Session()

print(sess.run(op1, feed_dict={a:3, b:4})) # 7.0
print(sess.run(op2, feed_dict={a:3, b:4})) # -1.0
print(sess.run(op3, feed_dict={a:3, b:4})) # 12.0
print(sess.run(op4, feed_dict={a:3, b:4})) # 0.75

sess.close()
```

```
import tensorflow as tf
tf.logging.set_verbosity(tf.logging.ERROR)

X = tf.placeholder(tf.float32, [2,3])
x_data = [[1,2,3], [4,5,6]]

W = tf.Variable(tf.random_normal([3,2]))
b = tf.Variable(tf.random_normal([2,1]))

Y = tf.matmul(X,W) + b

sess = tf.Session()

sess.run(tf.global_variables_initializer())

print(sess.run(Y, feed_dict={X:x_data}))

sess.close()
```

ReduceSum & ReduceMean

```
import tensorflow as tf

x = tf.constant([1.,2.], [2.,3.], [3.,4.]])

a = tf.reduce_sum(x)
b = tf.reduce_sum(x, axis=0)
c = tf.reduce_sum(x, axis=1)

d = tf.reduce_mean(x)
e = tf.reduce_mean(x, axis=0)
f = tf.reduce_mean(x, axis=1)

sess = tf.Session()

print(a.eval(session=sess)) # 15.0 = (1+2)+(2+3)+(3+6)
print(b.eval(session=sess)) # [6. 9.] = [(1+2+3) (2+3+4)]
print(c.eval(session=sess)) # [3. 5. 7.] = [(1+2) (2+3) (3+4)]

print(d.eval(session=sess)) # 2.5 = 15/6
print(e.eval(session=sess)) # [2. 3.] = [(6/3) (9/3)]
print(f.eval(session=sess)) # [1.5 2.5 3.5] = [(3/2) (5/2) (7/2)]

sess.close()
```



Saving / Restoring Variables

```
import tensorflow as tf
tf.logging.set_verbosity(tf.loggin.ERROR)

v1 = tf.Variable(1, name="v1")
v2 = tf.Variable(2, name="v2")

saver = tf.train.Saver()

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

print(sess.run(v1)) # 1
print(sess.run(v2)) # 2

path = saver.save(sess, "/tmp/aaa")

sess.close()
```

```
import tensorflow as tf
tf.logging.set_verbosity(tf.loggin.ERROR)

v1 = tf.Variable(3, name="v1")
v2 = tf.Variable(4, name="v2")

saver = tf.train.Saver()

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

path = saver.restore(sess, "/tmp/aaa")

print(sess.run(v1)) # 1
print(sess.run(v2)) # 2

sess.close()
```

Exception Handling

```
import tensorflow as tf
tf.logging.set_verbosity(tf.loggin.ERROR)

v1 = tf.Variable(3, name="v1")
v2 = tf.Variable(4, name="v2")

saver = tf.train.Saver()

sess = tf.Session()

try:
    path = saver.restore(sess, "/tmp/bbb")
except ValueError:
    sess.run(tf.global_variables_initializer())

print(sess.run(v1)) # 3
print(sess.run(v2)) # 4

sess.close()
```

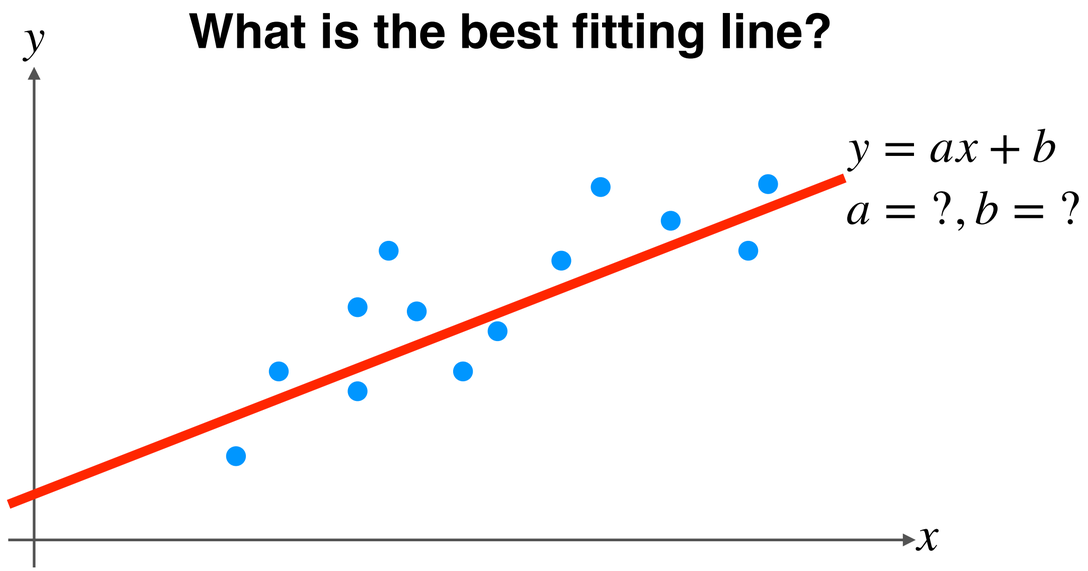
Linear Regression

Linear Regression

- **선형 회귀**(線型回歸, linear regression)는

  종속 변수 Y와 한 개 이상의 독립 변수 (또는 설명 변수) X 사이의

  선형 상관 관계를 모델링하는 수치기법



주어진 데이터의 추이를 가장 잘 설명하는 직선은 무엇인가?

Loss Function / Cost

- **손실 함수 (loss function)**
  - 하나의 단위 데이터에 대한 손실값을 계산하는 함수
  - 손실값: 실제값과 예측한 값의 차이
  - 손실값이 작을수록 정확하게 예측하고 있다는 의미
    (손실값이 커질수록 예측에서 벗어나고 있다는 의미)
  - 보통의 경우 **| 실제값 - 예측값 |**^{2}의 형태를 사용
    (음수 처리 회피 및 예측에서 멀어질수록 손실값이 커지도록 하기 위함)

- **비용 함수 (cost function)**
  - 모든 데이터에 대해서 손실값을 구한 결과
  - 보통의 경우 $\underbrace{\sum_{i=1}^{n} | \text{실제값}_i - \text{예측값}_i |^2}_{\text{MSE (Mean Square Error)}}$의 형태를 사용한다.

Formulation

Model: $y_i = wx_i + b \quad (i = 1,2,3,\cdots,N)$

Error: $E_i = \underbrace{\hat{y}_i}_{\text{known}} - \underbrace{y_i}_{\text{predicted}} = \hat{y}_i - (wx_i + b) = \hat{y}_i - wx_i - b$

$$\frac{\partial E_i}{\partial w} = -x_i$$

$$\frac{\partial E_i}{\partial b} = -1$$

Loss: $L_i = \frac{1}{2}(\hat{y}_i - y_i)^2 = \frac{1}{2}E_i^2$

$$\frac{\partial L_i}{\partial w} = \frac{\partial L_i}{\partial E_i}\frac{\partial E_i}{\partial w} = E_i \cdot (-x_i) = -x_i E_i$$

$$\frac{\partial L_i}{\partial b} = \frac{\partial L_i}{\partial E_i}\frac{\partial E_i}{\partial b} = E_i \cdot (-1) = -E_i$$

Cost: $C = \frac{1}{N}\sum_{i=1}^{N} L_i$

$$\boxed{\frac{\partial C}{\partial w}} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L_i}{\partial w} = \frac{1}{N}\boxed{\sum_{i=1}^{N}(-x_i E_i)}$$

$$\boxed{\frac{\partial C}{\partial b}} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L_i}{\partial b} = \frac{1}{N}\boxed{\sum_{i=1}^{N}(-E_i)}$$

NumPy Implementation

```
import numpy as np

# y = w*x+b: w=2, b=1
x = np.array([1, 2, 3, 4, 5], dtype='float32')
y = np.array([3, 5, 7, 9, 11], dtype='float32')

w = np.random.randn(1) # weight
b = np.random.randn(1) # bias

# iteration
for epoch in range(10000):
    # prediction
    y_predicted = w * x + b
    error = y - y_predicted

    # gradient
    w_grad = (-x * error).mean()
    b_grad = (-error).mean()

    # update
    w -= 0.01 * w_grad
    b -= 0.01 * b_grad

print(w, b) # [2.00000001] [0.99999998]
```

$$E_i = \underbrace{\hat{y}_i}_{\text{known}} - \underbrace{y_i}_{\text{predicted}}$$

$$\frac{\partial C}{\partial w} = \sum_{i=1}^{N} (-x_i E_i)$$

$$\frac{\partial C}{\partial b} = \sum_{i=1}^{N} (-E_i)$$

NumPy Implementation

```
import numpy as np

# y = w*x+b: w=2, b=1
x = np.array([1, 2, 3, 4, 5], dtype='float32')
y = np.array([3, 5, 7, 9, 11], dtype='float32')

w = np.random.randn(1) # weight
b = np.random.randn(1) # bias

# iteration
for epoch in range(10000):
    # prediction
    y_predicted = w * x + b
    error = y - y_predicted

    # gradient
    w_grad = (-x * error).mean()
    b_grad = (-error).mean()

    # update
    w -= 0.01 * w_grad
    b -= 0.01 * b_grad

print(w, b) # [2.00000001] [0.99999998]
```

$$w^{n+1} = w^n - \alpha \frac{\partial E}{\partial w}$$

$$b^{n+1} = b^n - \alpha \frac{\partial E}{\partial b}$$

How does it work?

- **Exactly the same process as solving Ax = b with the steepest descent method**

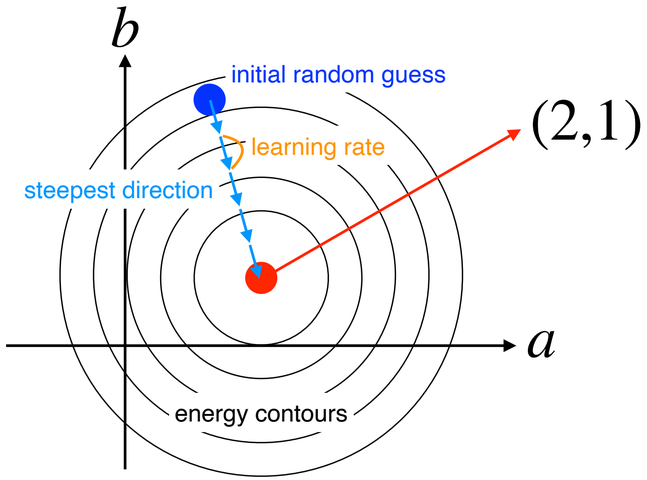
Model: $y_i = ax_i + b \quad (i = 1,2,3,\cdots,N)$ $\longrightarrow$ $E = \sum_{i=1}^{N} (y_i - ax_i - b)^2$ $\xrightarrow{\underset{a,b}{\mathrm{argmin}} E}$

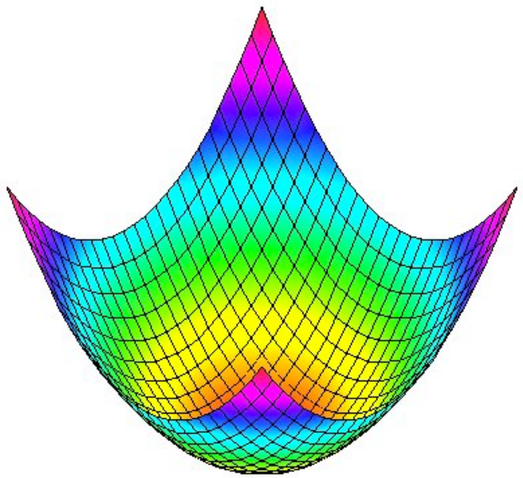
$$\begin{cases} \dfrac{\partial E}{\partial a} = 2\sum_{i=1}^{N} (y_i - ax_i - b)\cdot(-x_i) = 0 \\ \dfrac{\partial E}{\partial b} = 2\sum_{i=1}^{N} (y_i - ax_i - b)\cdot(-1) = 0 \end{cases}$$

$$\longrightarrow \begin{cases} a\left(\sum_{i=1}^{N} x_i^2\right) + b\left(\sum_{i=1}^{N} x_i\right) = \left(\sum_{i=1}^{N} x_i y_i\right) \\ a\left(\sum_{i=1}^{N} x_i\right) + bN = \left(\sum_{i=1}^{N} y_i\right) \end{cases} \longrightarrow \begin{bmatrix} \sum_{i=1}^{N} x_i^2 & \sum_{i=1}^{N} x_i \\ \sum_{i=1}^{N} x_i & N \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} x_i y_i \\ \sum_{i=1}^{N} y_i \end{bmatrix} \quad \mathbf{Ax = b}$$

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = 0$$

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} + \mathbf{c} \longrightarrow \mathbf{Ax} = \mathbf{b}$$





1D Example #1

```
import tensorflow as tf

# y = w*x+b: w=2, b=1
x_data = [1, 2, 3, 4, 5]
y_data = [3, 5, 7, 9, 11]

x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)

w = tf.Variable(tf.random_uniform([1], -1.0, 1.0), name='weight')
b = tf.Variable(tf.random_uniform([1], -1.0, 1.0), name='bias')

y_predicted = (x * w) + b
cost = tf.reduce_mean(tf.squared_difference(y, y_predicted))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(cost)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for epoch in range(10000):
        cost_, _ = sess.run([cost, train], feed_dict={x:x_data, y:y_data})
        w_, b_ = sess.run([w, b])

print(w_, b_) # [2.000005] [0.9999834]
```

1D Example #2

```
import tensorflow as tf

# y = w*x+b: w=2, b=1
x_data = [[1], [2], [3], [4], [5]]  # 5 by 1
y_data = [[3], [5], [7], [9], [11]] # 5 by 1

x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)

w = tf.Variable(tf.random_uniform([1, 1], -1.0, 1.0), name='weight')
b = tf.Variable(tf.random_uniform([1], -1.0, 1.0), name='bias')

y_predicted = (x * w) + b
cost = tf.reduce_mean(tf.squared_difference(y, y_predicted))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(cost)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for epoch in range(10000):
        cost_, _ = sess.run([cost, train], feed_dict={x:x_data, y:y_data})
        w_, b_ = sess.run([w, b])

print(w_, b_) # [[2.000005]] [0.9999834]
```

2D Example #1

```
import tensorflow as tf

# y = (w1*x1)+(w2*x2)+b: w1=1, w2=-1, b=0
x1_data = [1, 2, 3, 4, 5]
x2_data = [0, 1, 0, 2, 0]
y_data = [1, 1, 3, 2, 5]

x1 = tf.placeholder(tf.float32)
x2 = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)

w1 = tf.Variable(tf.random_uniform([1], -1.0, 1.0), name='weight1')
w2 = tf.Variable(tf.random_uniform([1], -1.0, 1.0), name='weight2')
b = tf.Variable(tf.random_uniform([1], -1.0, 1.0), name='bias')

y_predicted = (x1 * w1) + (x2 * w2) + b
cost = tf.reduce_mean(tf.squared_difference(y, y_predicted))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(cost)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for epoch in range(10000):
        cost_, _ = sess.run([cost, train], feed_dict={x1:x1_data, x2:x2_data, y:y_data})
        w1_, w2_, b_ = sess.run([w1, w2, b])

print(w1_, w2_, b_) # [1.0000012] [-0.9999982] [-4.6192463e-06]
```

2D Example #2

```
import tensorflow as tf

x_data = [[1, 2, 3, 4, 5],  [0, 1, 0, 2, 0]] # 2 by 5
y_data = [1, 1, 3, 2, 5]

x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)

w = tf.Variable(tf.random_uniform([1, 2], -1.0, 1.0), name='weight')
b = tf.Variable(tf.random_uniform([1], -1.0, 1.0), name='bias')

y_predicted = tf.matmul(w, x) + b
cost = tf.reduce_mean(tf.square(y - y_predicted))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(cost)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for epoch in range(10000):
        cost_, _ = sess.run([cost, train], feed_dict={x:x_data, y:y_data})
        w_, b_ = sess.run([w, b])

print(w_, b_) # [[ 0.99999976 -1.0000044 ]] [3.2783096e-06]
```

2D Example #3

```
import tensorflow as tf

x_data = [[1, 0], [2, 1], [3, 0], [4, 2], [5, 0]] # 5 by 2
y_data = [[1], [1], [3], [2], [5]]

x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)

w = tf.Variable(tf.random_uniform([2, 1], -1.0, 1.0), name='weight')
b = tf.Variable(tf.random_uniform([1], -1.0, 1.0), name='bias')

y_predicted = tf.matmul(x, w) + b
cost = tf.reduce_mean(tf.square(y - y_predicted))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(cost)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for epoch in range(10000):
        cost_, _ = sess.run([cost, train], feed_dict={x:x_data, y:y_data})
        w_, b_ = sess.run([w, b])

print(w_, b_) # [[ 1.0000012] [-0.9999982]] [-4.619085e-06]
```

2D Example #4


```
import tensorflow as tf

w = tf.Variable(tf.random_uniform([2, 1], -1.0, 1.0), name='weight')
b = tf.Variable(tf.random_uniform([1], -1.0, 1.0), name='bias')

def Data():
    x_data = [[1, 0], [2, 1], [3, 0], [4, 2], [5, 0]] # 5 by 2
    y_data = [[1], [1], [3], [2], [5]]
    return tf.to_float(x_data), tf.to_float(y_data)

def FeedForward(x):
    return tf.matmul(x, w) + b

def Cost(x, y):
    y_predicted = FeedForward(x)
    return tf.reduce_mean(tf.square(y - y_predicted))

def Train(cost, learning_rate):
    optimizer = tf.train.GradientDescentOptimizer(learning_rate)
    return optimizer.minimize(cost)
```

2D Example #4


```
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())

    x, y = Data()
    cost = Cost(x, y)
    train = Train(cost, 0.01)

    for epoch in range(10000):
        cost_, _ = sess.run([cost, train])
        w_, b_ = sess.run([w, b])

print(w_, b_) # [[ 0.99999976] [-1.0000044]] [3.2786731e-06]
```

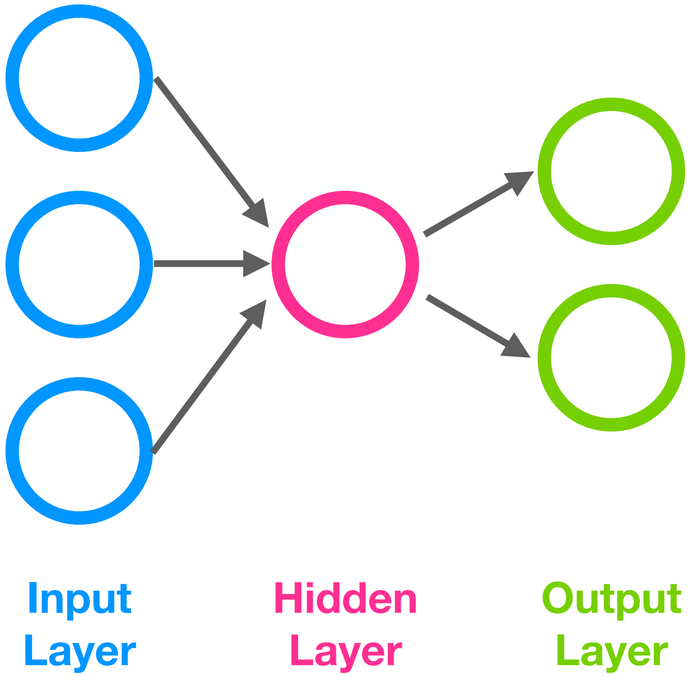
Practical Example

Region	Temp. (F)	Rainfall (mm)	Humidity (%)	Apples (ton)	Oranges (ton)
Kanto	73	67	43	56	70
Johto	91	88	64	81	101
Hoenn	87	134	58	119	133
Sinnoh	102	43	37	22	37
Unova	69	96	70	103	119

```
yield_apple  = w11 * temp + w12 * rainfall + w13 * humidity + b1
yield_orange = w21 * temp + w22 * rainfall + w23 * humidity + b2
```

$$X \quad \times \quad W^T \quad + \quad b$$

$$\begin{bmatrix} 73 & 67 & 43 \\ 91 & 88 & 64 \\ \vdots & \vdots & \vdots \\ 69 & 96 & 70 \end{bmatrix} \times \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix} + \begin{bmatrix} b_1 & b_2 \\ b_1 & b_2 \\ \vdots & \vdots \\ b_1 & b_2 \end{bmatrix}$$



https://medium.com/dsnet/linear-regression-with-pytorch-3dde91d60b50

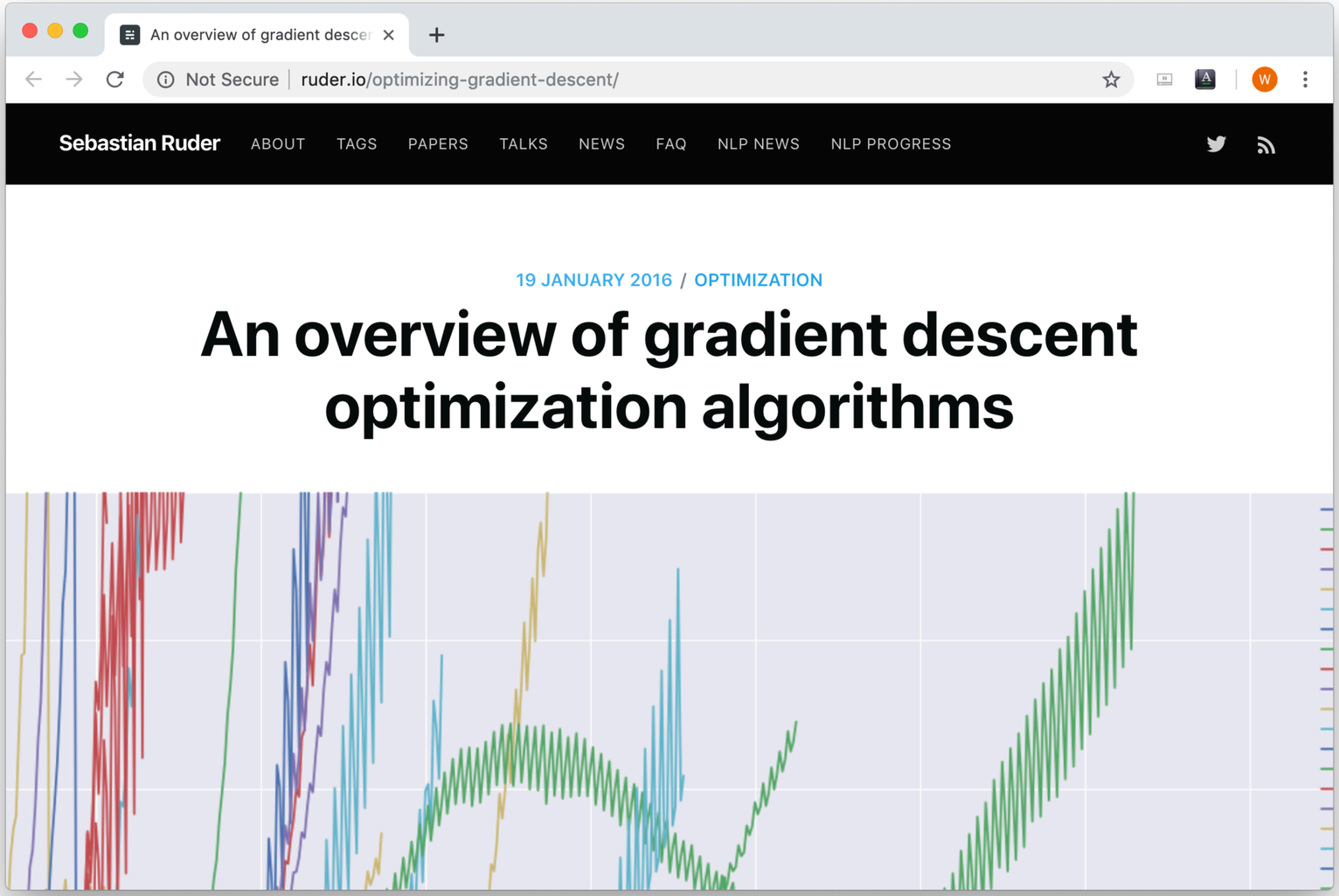
Practical Example


```
import tensorflow as tf

# input data: (temperature, rainfall, humidity)
x_train = [[73, 67, 43], [91, 88, 64], [87, 134, 58], [102, 43, 37], [69, 96, 70]]

# output data: (apples, oranges)
y_train = [[56, 70], [81, 101], [119, 133], [22, 37], [103, 119]]

x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)

w = tf.Variable(tf.random_normal([3, 2], stddev=0.01), name='weight')
b = tf.Variable(tf.random_normal([2], stddev=0.01), name='bias')

y_predicted = tf.matmul(x, w) + b
cost = tf.reduce_mean(tf.square(y - y_predicted))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.0001)
train = optimizer.minimize(cost)
```

Practical Example


```
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for epoch in range(10000):
        cost_, _ = sess.run([cost, train], feed_dict={x:x_train, y:y_train})
        w_, b_ = sess.run([w, b])

# test
y = tf.matmul(x, w_) + b_
with tf.Session() as sess:
    print(sess.run(y, feed_dict={x: x_train}))

# prediction
x_test = [[80, 70, 50]]
with tf.Session() as sess:
    print(sess.run(y, feed_dict={x: x_test}))
```

Several Optimizers

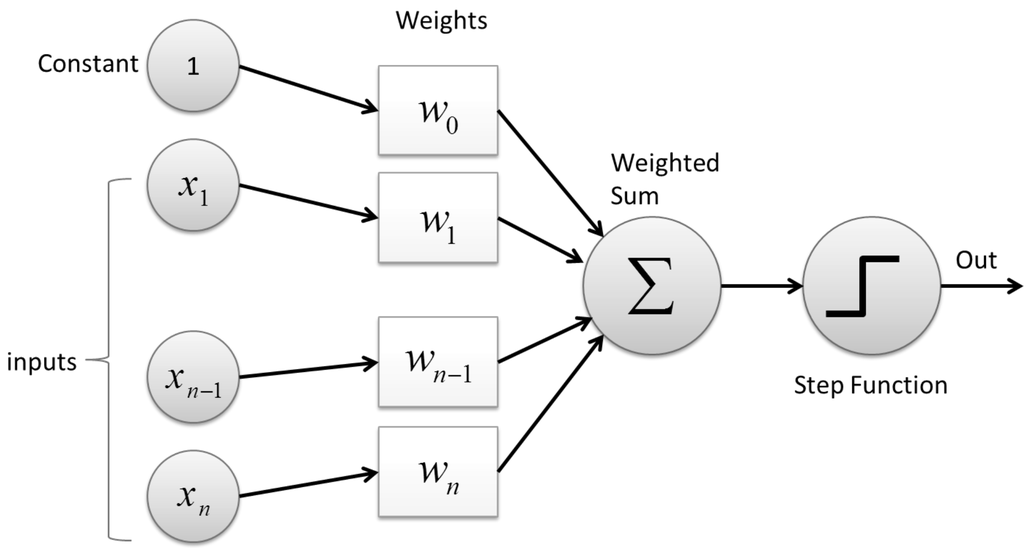


http://ruder.io/optimizing-gradient-descent/

Logistic Regression

퍼셉트론 (Perceptron)

- 단순한 **step function**을 activation function으로 사용하는 neural network
  (즉, 신호가 흐른다(1), 안흐른다(0) 2가지 상태만 표현 가능)



$$Y = \begin{cases} 0 & w_0 + w_1 X_1 + w_2 X_2 + \cdots + w_n X_n \le \alpha \\ 1 & w_0 + w_1 X_1 + w_2 X_2 + \cdots + w_n X_n > \alpha \end{cases}$$

http://abhay.harpale.net/blog/machine-learning/a-hands-on-tutorial-on-the-perceptron-learning-algorithm/

퍼셉트론 (Perceptron)

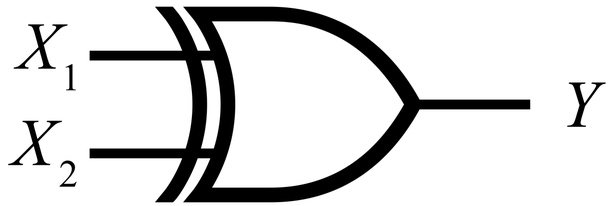
- 1세대 인공 신경망 (1957년 **Frank Rosenblatt**에 의해 제안됨)
- **인공신경망**의 **시초**가 되었지만 명백한 **한계**로 인해 오랫동안 인공신경망 **연구**를 **정체**시킨 요인이기도 함
- 1969년 Marvin **Minsky** (MIT AI Lab. 설립자) 교수가 **"Perceptrons"**라는 그의 저서에서 **[XOR 문제]**와 같은 간단한 문제의 학습 조차도 **불가능**하다는 것을 수학적으로 증명함
- **MLP(multi-layer perception)**를 사용하면 [XOR 문제]를 푸는 것이 **가능**하지만 MLP를 사용할 경우 각 perceptron의 weight와 bias를 **학습시킬 수 없다**고 하여 많은 사람들의 기대에 **찬물**을 확 끼얹음

"No one on earth had found a viable way to train MLPs good enough to learn such simple functions."

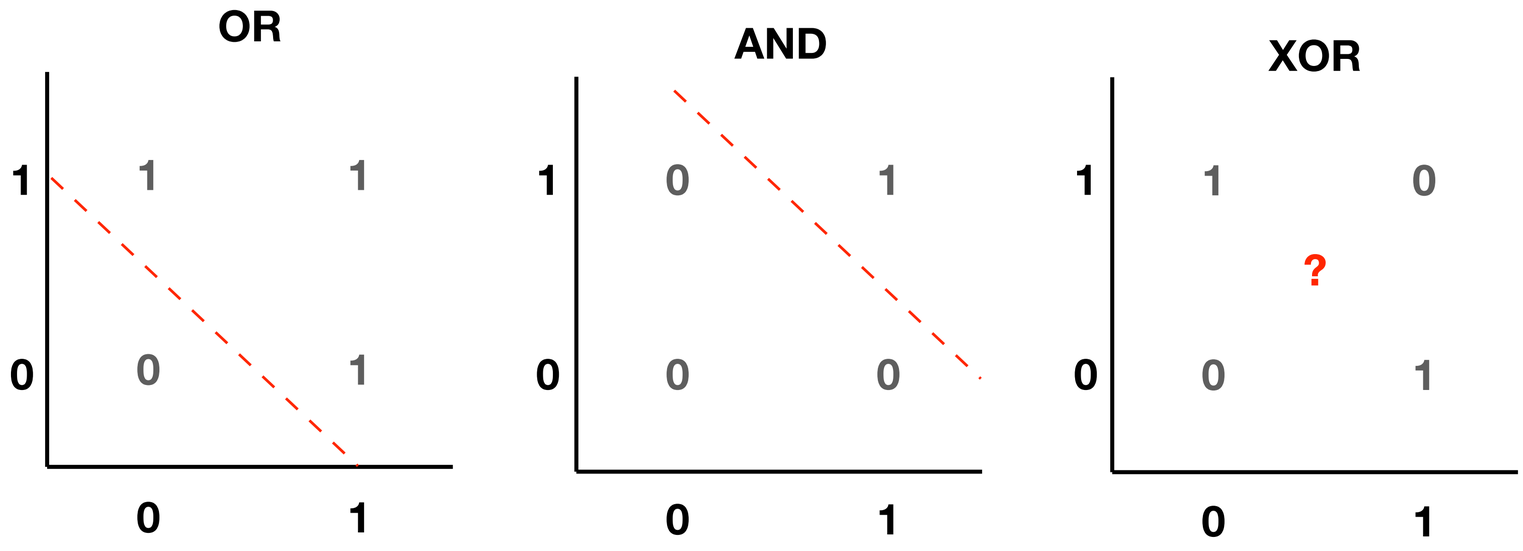
- 하지만, 20~30년 후에 **back-propagation**(1974, 1982 by Paul Werebos, 1986 by Hinton) 알고리즘이 등장하면서 MLP를 학습시킬 수 있게 되었음
(앞에서 설명한 3가지 요인 중 하나: 새로운 알고리즘의 등장)

XOR Problem

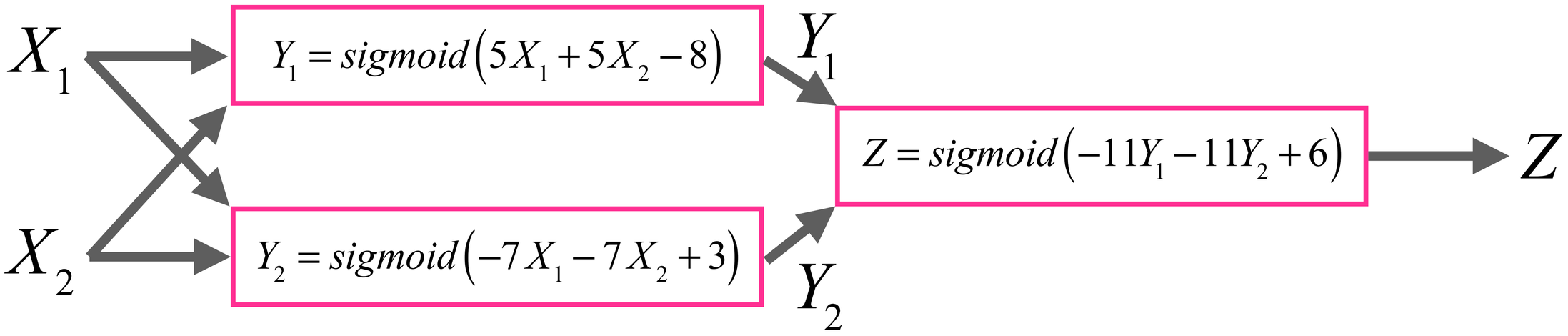
- 선형(linear) 방식으로 데이터를 구분할 수 없는 대표적인 문제
  - OR 문제나 AND 문제는 직선을 사용하여 명확하게 결과를 만들어낼 수 있음
  - XOR 문제는 직선만으로는 맞는 결과를 만들어낼 수 없음



X_1	X_2	Y
0	0	0
1	0	1
0	1	1
1	1	0



XOR 문제를 푸는 인공 신경망의 예



$(X_1, X_2) = (0,0)$

$$Y_1 = sigmoid(5\times0+5\times0-8) = sigmoid(-8) \approx 0$$
$$Y_2 = sigmoid(-7\times0-7\times0+3) = sigmoid(3) \approx 1$$
$$Z = sigmoid(-11\times0-11\times1+6) = sigmoid(-5) \approx 0$$

$(X_1, X_2) = (1,0)$

$$Y_1 = sigmoid(5\times1+5\times0-8) = sigmoid(-3) \approx 0$$
$$Y_2 = sigmoid(-7\times1-7\times0+3) = sigmoid(-4) \approx 0$$
$$Z = sigmoid(-11\times0-11\times0+6) = sigmoid(6) \approx 1$$

$(X_1, X_2) = (0,1)$

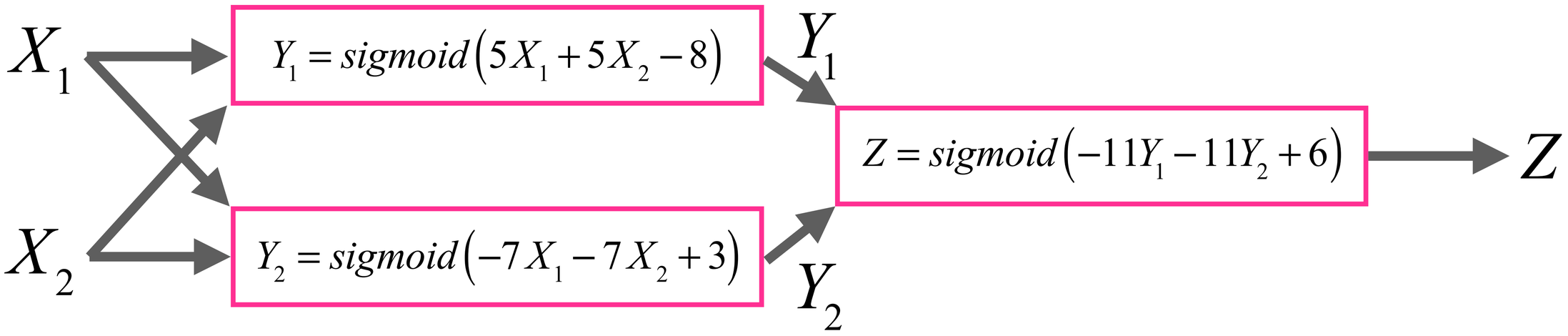
$$Y_1 = sigmoid(5\times0+5\times1-8) = sigmoid(-3) \approx 0$$
$$Y_2 = sigmoid(-7\times0-7\times1+3) = sigmoid(-4) \approx 0$$
$$Z = sigmoid(-11\times0-11\times0+6) = sigmoid(6) \approx 1$$

$(X_1, X_2) = (1,1)$

$$Y_1 = sigmoid(5\times1+5\times1-8) = sigmoid(2) \approx 1$$
$$Y_2 = sigmoid(-7\times1-7\times1+3) = sigmoid(-11) \approx 0$$
$$Z = sigmoid(-11\times1-11\times0+6) = sigmoid(-5) \approx 0$$

https://www.youtube.com/watch?v=GYecDQQwTdI&index=14&list=PLPIA-cliHVSuaBofb24zMwfHsrFy603Qy

XOR 문제를 푸는 인공 신경망의 예



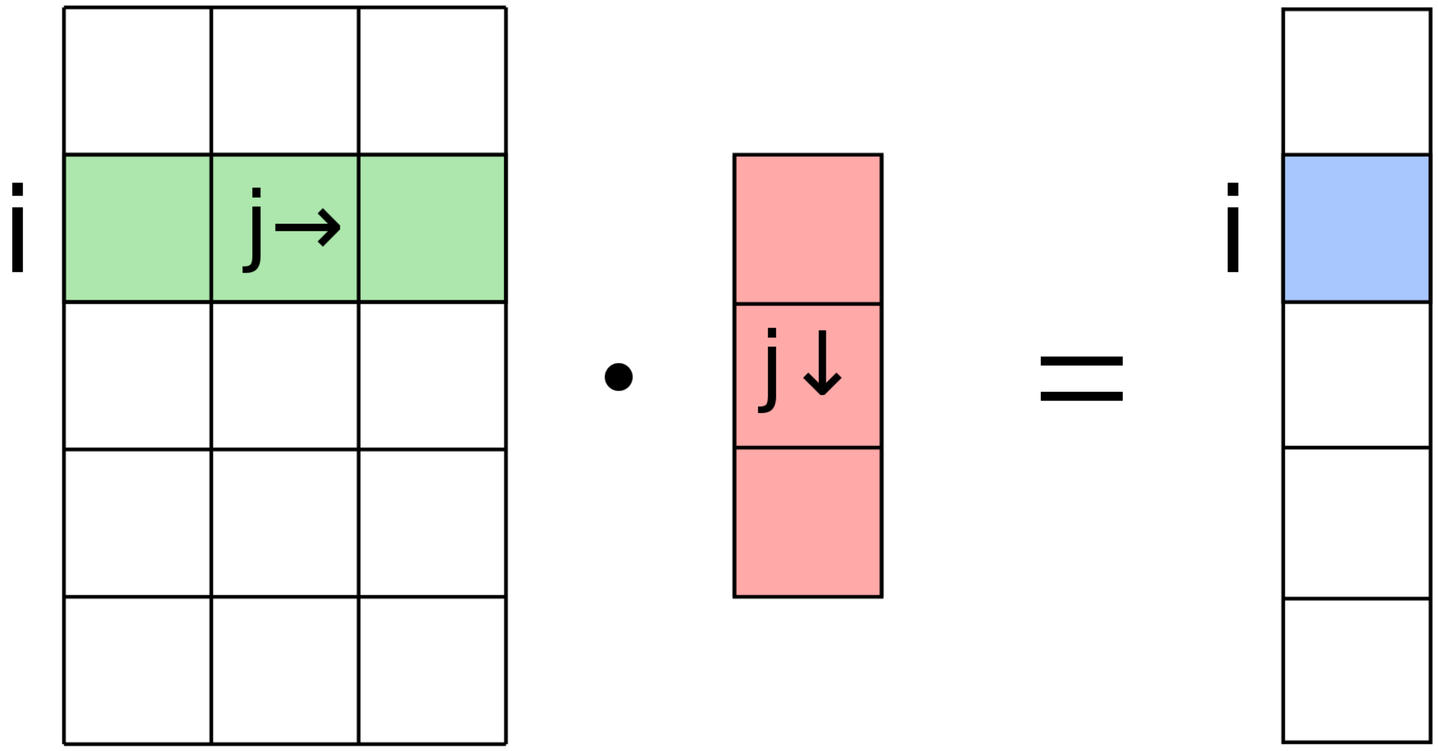
각 계층(layer)을 행렬(matrix)을 이용하여 간단하게 나타낼 수 있음

$$\begin{bmatrix} Y_1 & Y_2 \end{bmatrix} = sigmoid\left(\begin{bmatrix} X_1 & X_2 \end{bmatrix} \begin{bmatrix} 5 & -7 \\ 5 & -7 \end{bmatrix} + \begin{bmatrix} -8 & 3 \end{bmatrix} \right) \quad \Leftrightarrow \quad \mathbf{Y} = sigmoid(\mathbf{X}\mathbf{W}_1 + \mathbf{b}_1)$$

$$[Z] = sigmoid\left(\begin{bmatrix} Y_1 & Y_2 \end{bmatrix} \begin{bmatrix} -11 \\ -11 \end{bmatrix} + [6] \right) \quad \Leftrightarrow \quad \mathbf{Z} = sigmoid(\mathbf{Y}\mathbf{W}_2 + \mathbf{b}_2)$$

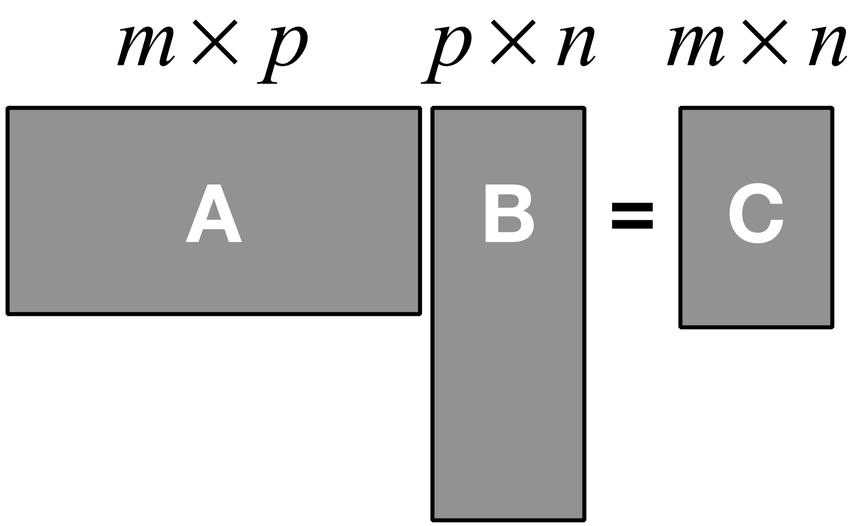
https://www.youtube.com/watch?v=GYecDQQwTdI&index=14&list=PLPIA-cliHVSuaBofb24zMwfHsrFy603Qy

Matrix / Vector Multiplication



https://commons.wikimedia.org/wiki/File:Matrix_vector_multiplication.svg

Matrix / Matrix Multiplication



$$C_{ij} = \sum_{k=1}^{p} A_{ik} \times B_{kj}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} a \times 1 + b \times 3 & a \times 2 + b \times 4 \\ c \times 1 + d \times 3 & c \times 2 + d \times 4 \end{bmatrix}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} a \times 1 + b \times 3 + c \times 5 & a \times 2 + b \times 4 + c \times 6 \\ d \times 1 + e \times 3 + f \times 5 & d \times 2 + e \times 4 + f \times 6 \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} a \times 1 + b \times 3 & a \times 2 + b \times 4 \\ c \times 1 + d \times 3 & c \times 2 + d \times 4 \\ e \times 1 + f \times 3 & e \times 2 + f \times 4 \end{bmatrix}$$

One-Hot Encoding

- 데이터의 종류
  - **수치형** 데이터 (**numerical** data): **숫자**로 나타내는 데이터. 컴퓨터 연산 가능.
  - **범주형** 데이터 (**categorical** data): **단어**로 나타내는 데이터. 컴퓨터 연산 불가능.
- **One-Hot Encoding**: 범주형 데이터를 수치형 데이터인 **One-Hot Vector로 변환**하는 과정.
- **One-Hot Vector**: 하나의 요소(element)만 **1**이고 나머지 요소(element)는 **0**인 벡터(vector)

  : 제일 뜨거운 하나만 1로 남겨놓고, 나머지들은 다 0을로 만들겠다!
- One-hot encoding 과정에서 각각의 one-hot vector가 n 차원의 **기저(basis)**가 된다.
- 예) DOG=[1,0,0], CAT=[0,1,0], TIGER=[0,0,1]

  Which Animal? = (0.6 × [1,0,0]) + (0.1 × [0,1,0]) + (0.3 × [0,0,1])

  = [**0.6** 0.1 0.3]: Maybe DOG!

로지스틱 회귀 (Logistic Regression)

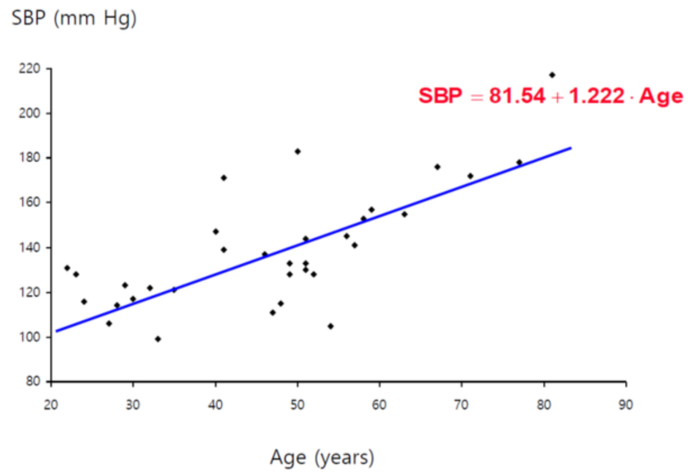
- **범주형(categorical) 변수**를 **예측**하는 모델

- 33명의 성인 여성에 대한 나이와 혈압 데이터의 선형 회귀 분석 (Linear Regression):

Age	SBP
22	131
23	128
24	116
27	106
28	114
29	123
30	117
32	122
33	99
35	121
40	147

Age	SBP
41	139
41	171
46	137
47	111
48	115
49	133
49	128
50	183
51	130
51	133
51	144

Age	SBP
52	128
54	105
56	145
57	141
58	153
59	157
63	155
67	176
71	172
77	178
81	217



- 유의미한 회귀 직선이 얻어진다.

https://ratsgo.github.io/machine%20learning/2017/04/02/logistic/

로지스틱 회귀 (Logistic Regression)

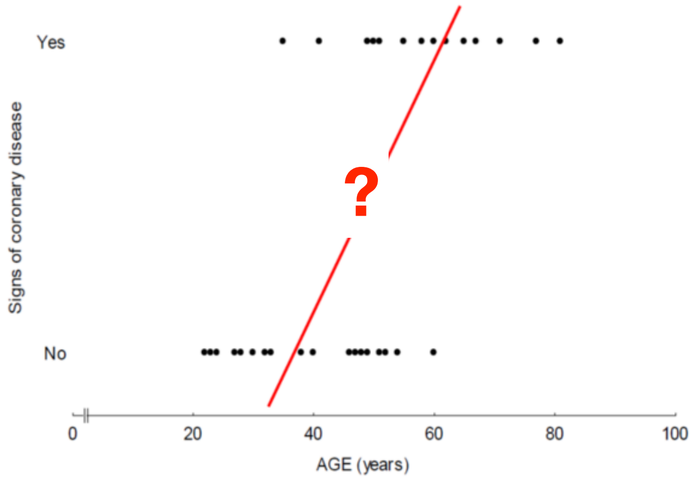
- **범주형(categorical) 변수**를 **예측**하는 모델

- 나이와 암 발생여부(1이면 발병, 0이면 정상) 데이터의 선형 회귀 분석 (Linear Regression):

Age	CD
22	0
23	0
24	0
27	0
28	0
30	0
30	0
32	0
33	0
35	1
38	0

Age	CD
40	0
41	1
46	0
47	0
48	0
49	1
49	0
50	1
51	0
51	1
52	0

Age	CD
54	0
55	1
58	1
60	1
60	0
62	1
65	1
67	1
71	1
77	1
81	1



- 유의미한 회귀직선이 아님

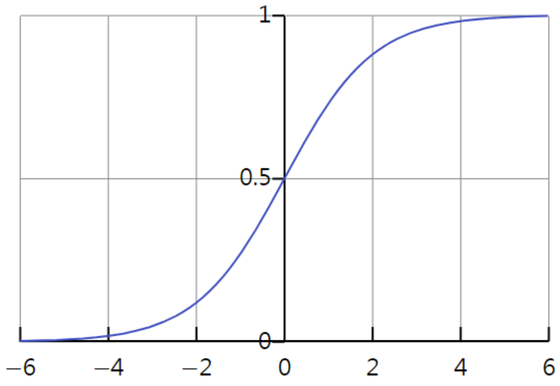
- 0과 1이 수치가 아닌 범주형 데이터이기 때문에 나타나는 현상

https://ratsgo.github.io/machine%20learning/2017/04/02/logistic/

로지스틱 함수 (Logistic Function)

- 많은 자연 및 사회 현상들을 확률변수로 나타내었을 때 확률값이 선형이 아닌 **S-Curve**을 따른다.
  : smooth transition from one to another

- **로지스틱 함수**: 이러한 S-Curve를 함수로 표현한 것 = **Sigmoid**



$$Y = \frac{1}{1 + e^{-X}}$$

- 따라서, **가설(hypothesis): Y = sigmoid(Wx + b)**

- 이 때, **sigmoid** 함수가 **활성화 함수(activation function)**가 된다.

Sigmoid as Activation Function

- **활성화 함수(activation function)가 없는 경우**에는 **"평균 제곱 오차 (Mean Square Error)"**를 사용해도 역전파(back-propagation) 과정에서 경사하강법(gradient descent method)을 사용하면 **global minimum solution**을 찾는 것이 **보장** 된다.

평균 제곱 오차 (Mean Square Error)

$$E = \frac{1}{N}\sum_{i=1}^{N}\left(Y_i - H(X_i)\right)^2$$

H: hypothesis (예측값)

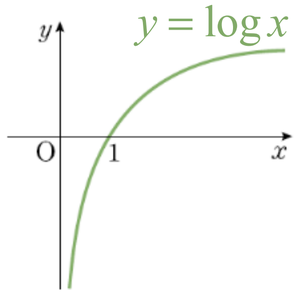
- 하지만, **sigmoid 함수**를 활성화 함수(activation function)로 사용하게 되면 비용 함수(cost function)가 매끈한 그릇(bowl) 형상이 아닌 **평평하거나 울퉁불퉁한 모양**이 되기 때문에 역전파(back-propagation) 과정에서 경사하강법(gradient descent method)을 사용할 때 항상 global minimum solution을 찾는다는 보장이 없다. (**local minimum**에 빠질 가능성이 높다.)

- 따라서, one-hot encoding과 logistic regression에 맞는 **새로운 비용 함수**가 필요하다!

https://www.youtube.com/watch?v=6vzchGYEJBc&index=12&list=PLlMkM4tgfjnLSOjrEJN31gZATbcj_MpUm

New Loss Function for Sigmoid

$$\text{loss} = -y\log(H(x)) - (1-y)\log(1-H(x))$$



$\log\varepsilon = -10000$

$\log(1-\varepsilon) = 0.0001$

	$H(x) = \varepsilon$ **예측값(가설)이 거의 0일 때**	$H(x) = 1-\varepsilon$ **예측값(가설)이 거의 1일 때**
$y = 0$	$-0\cdot\log\varepsilon - 1\cdot\log(1-\varepsilon)$ $= -0\times(-10000) - 1\times(-0.0001) = 0.0001$ 매우 작은 값 (거의 0)	$-0\cdot\log(1-\varepsilon) - 1\cdot\log\varepsilon$ $= -0\times(-0.0001) - 1\times(-10000) = 10000$ 매우 큰 값
$y = 1$	$-1\cdot\log\varepsilon - 0\cdot\log(1-\varepsilon)$ $= -1\times(-10000) - 0\times(-0.0001) = 10000$ 매우 큰 값	$-1\cdot\log(1-\varepsilon) - 0\cdot\log\varepsilon$ $= -1\times(-0.0001) - 0\times(-10000) = 0.0001$ 매우 작은 값 (거의 0)

New Cost Function for Sigmoid

$$\text{loss} = -y\log(H(x)) - (1-y)\log(1-H(x))$$

$$\text{cost} = \frac{1}{N}\sum_{i=1}^{N}\left[-Y_i\log(H(X_i)) - (1-Y_i)\log(1-H(X_i))\right]$$

Cross Entropy between two Bernoulli distribution

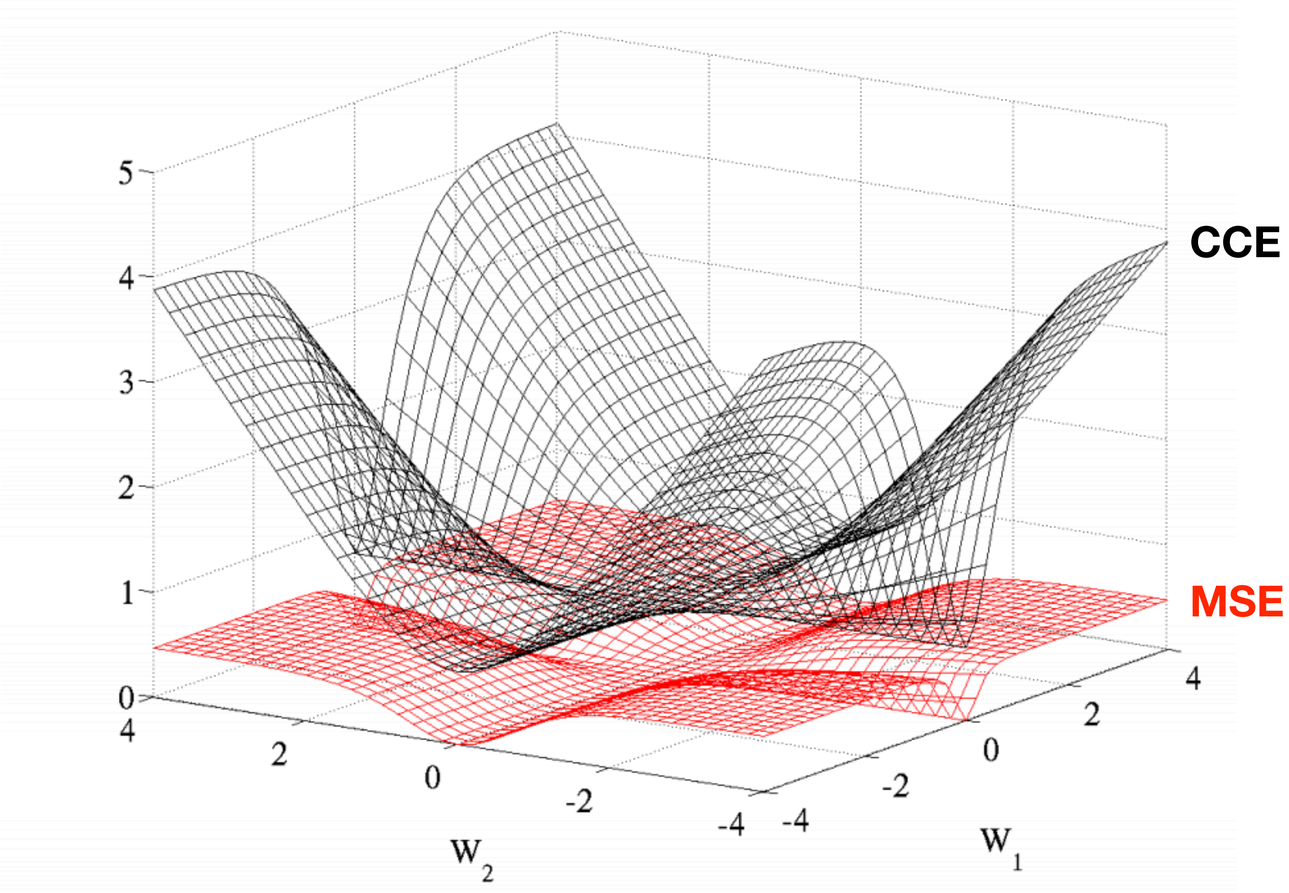
Average Cross Entropy Error

Comparison

	Linear Regression	Logistic Regression
Training Data	$\left(X_i, Y_i\right)\ (i = 1, \cdots, N)$ Y_i: real number	$\left(X_i, Y_i\right)\ (i = 1, \cdots, N)$ Y_i: 0 or 1 (on-hot encoding)
Hypothesis	$H = \sum_{j=1}^{M} W_j X_j + b$ **output: any value**	$H = \sigma\left(\sum_{j=1}^{M} W_j X_j + b\right)$ **output: between 0 and 1**
Cost Function	$\frac{1}{N}\sum_{i=1}^{N}\left(H(X_i) - Y_i\right)^2$ **평균 제곱 오차 (Mean Square Error)**	$\frac{1}{N}\sum_{i=1}^{N}\left[-Y_i \log(H(X_i)) - (1 - Y_i)\log(1 - H(X_i))\right]$ **평균 교차 엔트로피 오차 (Average Cross Entropy Error)**

https://www.youtube.com/watch?v=hSXFuypLukA

Why CCE rather than MSE?



http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf

MSE vs CCE

$$C = \frac{1}{N}\sum \frac{1}{2}\left(y_i - h_i\right)^2 \Rightarrow C = \frac{1}{2}\left(y - h\right)^2 \text{ 꼴}$$

$$h = \sigma\left(wx + b\right) = \sigma\left(z\right)$$

$$\frac{\partial C}{\partial w} = \left(y - h\right)\frac{\partial h}{\partial w}$$

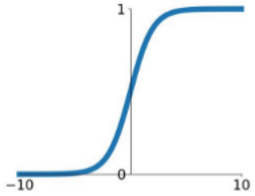
$$= \left(y - h\right)\frac{\partial \sigma(z)}{\partial z}\frac{\partial z}{\partial w}$$

$$= \left(h - y\right)\frac{\partial \sigma(z)}{\partial w}x$$

vanishing

Sigmoid

$$\sigma(w) = \frac{1}{1+e^{-w}}$$



$$C = -\frac{1}{N}\sum\left(y_i \ln h_i + (1 - y_i)\ln(1 - h_i)\right) \Rightarrow C = y\ln h + (1 - y)\ln(1 - h) \text{ 꼴}$$

$$h = \sigma\left(wx + b\right) = \sigma\left(z\right)$$

$$\frac{\partial C}{\partial w} = \frac{y}{h}\frac{\partial h}{\partial w} - \frac{1 - y}{1 - h}\frac{\partial h}{\partial w}$$

$$= \left(\frac{y}{h} - \frac{1 - y}{1 - h}\right)\frac{\partial h}{\partial w} = \frac{(1 - h)y - h(1 - y)}{h(1 - h)}\frac{\partial h}{\partial w} = \frac{y - \cancel{hy} - h + \cancel{hy}}{h(1 - h)}\frac{\partial h}{\partial w}$$

$$= \frac{y - h}{h(1 - h)}\frac{\partial h}{\partial w} = \frac{y - h}{h(1 - h)}\frac{\partial \sigma(z)}{\partial z}\frac{\partial z}{\partial w}$$

$$\frac{1}{\sigma(z)\left(1 - \sigma(z)\right)}\frac{\partial \sigma(z)}{\partial z} = \frac{1}{\frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right)}\frac{e^{-z}}{(1+e^{-z})^2} = \frac{1}{\frac{1}{1+e^{-z}}\frac{e^{-z}}{1+e^{-z}}}\frac{e^{-z}}{(1+e^{-z})^2} = 1$$

: no-zero gradient

https://www.youtube.com/watch?v=uMYhthKw1PU

What are good initial weights?

- The **zero initialization** will most probably not learn anything as the **derivatives** will also be **zero**. Thus, **no weight update**.

- If they start from zero, or even with **same values** (not random), all neurons might learn the **same features**, making them **redundant**.

https://courses.learnopencv.com/p/computer-vision-for-faces

```
import tensorflow as tf

x_data = [[0,0], [1,0], [0,1], [1,1]]
y_data = [[0], [1], [1], [0]]
                                [# of samples, data size]
X = tf.placeholder(tf.float32, [4, 2])
Y = tf.placeholder(tf.float32, [4, 1])
                                [# of inputs, # of outputs]
W1 = tf.Variable(tf.random_normal([2, 2]), name="weight1")
b1 = tf.Variable(tf.random_normal([2]), name="bias1")
L1 = tf.sigmoid(tf.matmul(X, W1) + b1) # layer1
                                [# of inputs, # of outputs]
W2 = tf.Variable(tf.random_normal([2, 1]), name="weight2")
b2 = tf.Variable(tf.random_normal([1]), name="bias2")
L2 = tf.sigmoid(tf.matmul(L1, W2) + b2) # layer2

model = L2 # = y_predicted (hypothesis)
cost = -tf.reduce_mean(Y * tf.log(model) + (1 - Y) * tf.log(1 - model))
optimizer = tf.train.GradientDescentOptimizer(0.1).minimize(cost)
predicted = tf.cast(model > 0.5, dtype=tf.int32)

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

for step in range(10000):
    sess.run(optimizer, feed_dict={X: x_data, Y: y_data})
hypothesis_ret, predicted_ret = sess.run([model, predicted], feed_dict={X: x_data})

print(predicted_ret)
sess.close()
```

Wider Neural Network

```
import tensorflow as tf

x_data = [[0,0], [1,0], [0,1], [1,1]]
y_data = [[0], [1], [1], [0]]
                                [# of samples, data size]
X = tf.placeholder(tf.float32, [4, 2])
Y = tf.placeholder(tf.float32, [4, 1])
                                [# of inputs, # of outputs]
W1 = tf.Variable(tf.random_normal([2, 10]), name="weight1")
b1 = tf.Variable(tf.random_normal([10]), name="bias1")
L1 = tf.sigmoid(tf.matmul(X, W1) + b1) # layer1
                                [# of inputs, # of outputs]
W2 = tf.Variable(tf.random_normal([10, 1]), name="weight2")
b2 = tf.Variable(tf.random_normal([1]), name="bias2")
L2 = tf.sigmoid(tf.matmul(L1, W2) + b2) # layer2

model = L2 # = y_predicted (hypothesis)
cost = -tf.reduce_mean(Y * tf.log(model) + (1 - Y) * tf.log(1 - model))
optimizer = tf.train.GradientDescentOptimizer(0.1).minimize(cost)
predicted = tf.cast(model > 0.5, dtype=tf.int32)

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

for step in range(10000):
    sess.run(optimizer, feed_dict={X: x_data, Y: y_data})
hypothesis_ret, predicted_ret = sess.run([model, predicted], feed_dict={X: x_data})

print(predicted_ret)
sess.close()
```

Deeper Neural Network

```
import tensorflow as tf

x_data = [[0,0], [1,0], [0,1], [1,1]]
y_data = [[0], [1], [1], [0]]
                                [# of samples, data size]
X = tf.placeholder(tf.float32, [4, 2])
Y = tf.placeholder(tf.float32, [4, 1])
                                [# of inputs, # of outputs]
W1 = tf.Variable(tf.random_normal([2, 10]), name="weight1")
b1 = tf.Variable(tf.random_normal([10]), name="bias1")
L1 = tf.sigmoid(tf.matmul(X, W1) + b1) # layer1
                                [# of inputs, # of outputs]
W2 = tf.Variable(tf.random_normal([10, 10]), name="weight2")
b2 = tf.Variable(tf.random_normal([10]), name="bias2")
L2 = tf.sigmoid(tf.matmul(L1, W2) + b2) # layer2

W3 = tf.Variable(tf.random_normal([10, 10]), name="weight3")
b3 = tf.Variable(tf.random_normal([10]), name="bias3")
L3 = tf.sigmoid(tf.matmul(L2, W3) + b3) # layer3

W4 = tf.Variable(tf.random_normal([10, 1]), name="weight4")
b4 = tf.Variable(tf.random_normal([1]), name="bias4")
layer4 = tf.sigmoid(tf.matmul(L3, W4) + b4) # layer4

model = L4 # = y_predicted (hypothesis)

.
.
.
```

Another Classification

```
import tensorflow as tf

x_data = [[0,0], [1,0], [0,1], [1,1], [0,0], [0,1]] # [feather, wing]
y_data = [[1,0,0], [0,1,0], [0,0,1], [0,0,1], [1,0,0], [0,0,1]] # none, mammal, bird, bird, none, bird

X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

W1 = tf.Variable(tf.random_normal([2, 5]), name="weight1") # [특징 수, 은닉층 노드 수]
b1 = tf.Variable(tf.random_normal([5]), name="bias1")
L1 = tf.sigmoid(tf.matmul(X, W1) + b1) # layer1

W2 = tf.Variable(tf.random_normal([5, 3]), name="weight2") # [은닉층 노드 수, 분류 수]
b2 = tf.Variable(tf.random_normal([3]), name="bias2")
L2 = tf.sigmoid(tf.matmul(L1, W2) + b2) # layer2

model = L2

# cost = -tf.reduce_mean(Y * tf.log(model) + (1 - Y) * tf.log(1 - model))
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits=model, labels=Y))

.
.
.
```

MNIST

Don’t Forget.

Garbage in, garbage out!

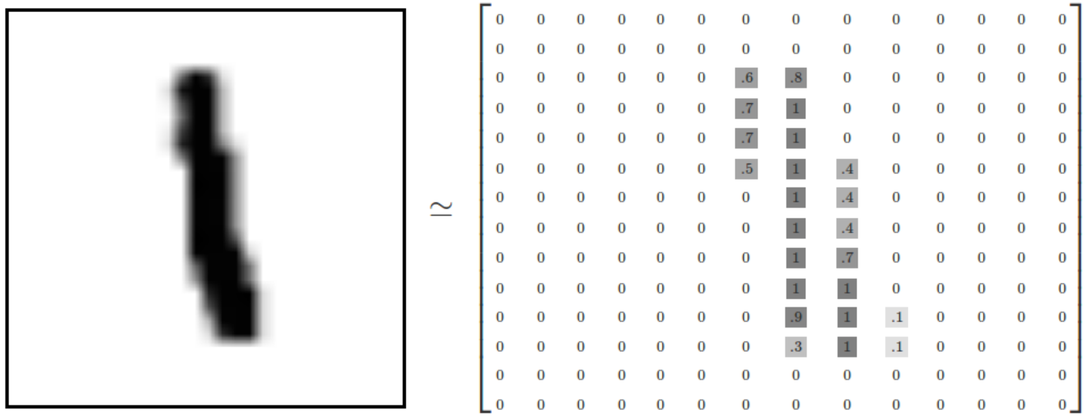
데이터 전처리 (Data Pre-processing)

- Anyone who have worked with raw data knows how ugly they get.
- 또한, 머신 러닝에 사용되는 실제 데이터에는 잘못된 데이터가 있을 가능성이 높다. (누락, 오기 등)
- 이러한 잘못된 데이터를 입력으로 사용하면 결과적으로 머신 러닝의 성능을 저하될 수 밖에 없다.
- 따라서, 오류가 없는 깨끗한 실제 데이터가 입력되도록 사용하기 전에 데이터를 정리하는 과정이 필요하다.
- 이러한 과정을 **데이터 먼징(data munging)**, 또는 **데이터 랭글링(data wrangling)**이라고 한다.
- 쉽게 사용할 수 있도록 전처리 과정을 거친 공개 데이터: **MNIST**, **CIFAR-10**, **CIFAR-100** 등

MNIST

- **MNIST**: Modified National Institute of Standards and Technology (NIST: 미국 국립표준기술연구소)
- 손으로 쓴 숫자(**0~9**) 이미지들로 구성된 데이터셋
- 각 이미지 해상도: **28×28** (**784**개 픽셀)
- **60,000**개의 트레이닝 이미지, **10,000**개의 테스트 이미지
- 각 이미지 마다 어떤 숫자인지에 대한 **라벨(label)**이 붙어있어서 테스트용 데이터로 많이 사용됨

https://tensorflowkorea.gitbooks.io/tensorflow-kr/content/g3doc/tutorials/mnist/beginners/



https://tensorflowkorea.gitbooks.io/tensorflow-kr/content/g3doc/tutorials/mnist/beginners/

MNIST 수동 다운로드 방법

MNIST handwritten digit datab

yann.lecun.com/exdb/mnist/

THE MNIST DATABASE

of handwritten digits

Yann LeCun, Courant Institute, NYU
Corinna Cortes, Google Labs, New York
Christopher J.C. Burges, Microsoft Research, Redmond

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

Four files are available on this site:

train-images-idx3-ubyte.gz:	training set images (9912422 bytes)
train-labels-idx1-ubyte.gz:	training set labels (28881 bytes)
t10k-images-idx3-ubyte.gz:	test set images (1648877 bytes)
t10k-labels-idx1-ubyte.gz:	test set labels (4542 bytes)

please note that your browser may uncompress these files without telling you. If the files you downloaded have a larger size than the above, they have been uncompressed by your browser. Simply rename them to remove the .gz extension. Some people have asked me "my application can't open your image files". These files are not in any standard image format. You have to write your own (very simple) program to read them. The file format is described at the bottom of this page.

The original black and white (bilevel) images from NIST were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. the images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field.

http://yann.lecun.com/exdb/mnist/

MNIST 자동 다운로드 방법

- 다음과 같은 TensorFlow 코드를 이용하여 MNIST 데이터를 자동으로 다운로드 받을 수 있다.

```
from tensor flow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets(“MNIST_data”, one_hot=True)
```

- 현재 위치에 MNIST_data라는 이름의 디렉토리를 만들고 이곳에 파일들을 다운로드 받는다.

- 하지만, 위의 코드를 실행하면 아래와 같은 경고가 나타난다. (사용하는 데에는 큰 문제 없음)

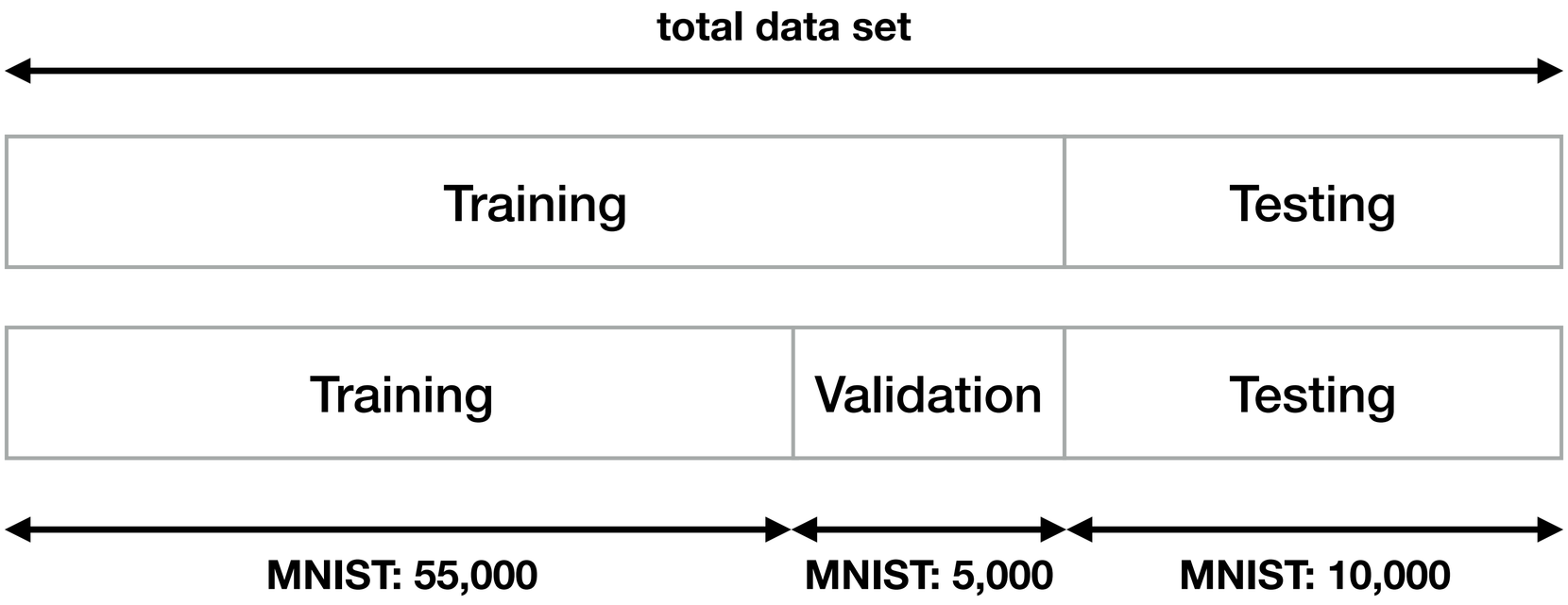
```
WARNING:tensorflow:From 23_mnist_matplot.py:5: read_data_sets (from
tensorflow.contrib.learn.python.learn.datasets.mnist) is deprecated and will be removed in a future
version.
```

- 만약 이 경고가 안나오도록 하고 싶다면 자동 다운로드 코드 앞에 다음 코드를 추가해주면 된다.

```
tf.logging.set_verbosity(tf.loggin.ERROR)

from tensor flow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets(“MNIST_data”, one_hot=True)
```

Three Types of Data Set

- **mnist.train** : training data
- **mnist.validation** : validation data
- **mnist.test** : test data



Holdout Validation

- 주어진 데이터를 모두 학습 과정에 사용한다면 이 데이터에 대해서는 100%에 가까운 정확도를 가지겠지만 최종적인 목적은 **새로운 데이터(unseen data)**에 대해 **결과**를 **예측**하는 것이기 때문에 큰 의미가 없다.

- 이처럼 학습 데이터에 대해서는 정확도가 높게 나오지만 새로운 데이터를 예측할 때 정확도가 떨어지는 현상을 **과적합(overfitting)**이라고 한다.

- 따라서, 주어진 데이터를 **training data**와 **test data**로 나누고, training data를 이용한 학습이 끝나면 test data를 이용해서 사용 가능한 수준이 되었는지 학습된 모델의 성능을 검증한다.

- 만약 학습 데이터가 충분하다면 training data 중에서 **validation data**를 구성하여 사용하기도 한다. 테스트 단계 이전에 소수의 데이터를 이용해서 미리 한 번 확인을 해보는 개념으로 생각하면 된다.

Holdout Validation

- **Training data**
  - 분석 모델을 만들기 위한 학습용 데이터
- **Validation data**
  - 여러 분석 모델 중 어떤 모델이 적합한지 선택하기 위한 검증용 데이터
- **Test data**
  - 최종적으로 선택된 분석 모델이 얼마나 잘 작동하는지 확인하기 위한 결과용 데이터

Validation Data를 사용하는 이유

- **모델**의 **성능**을 **평가**하기 위해서 사용 (이를 위해서 training set의 일부를 희생)
  - 학습을 통해 만들어진 모델이 학습에 사용되지 않은 데이터에 대해서 얼마나 예측을 잘 하는가
  - 만약 정확도가 training 과정보다 현저하게 낮아졌다면 과적합이 발생한 것으로 판단할 수 있다.

- 각 data set의 역할
  - **Validation data**: 학습 과정에 관여. 여러 모델 중에서 최종 모델을 선정하기 위한 성능 평가에 관여.
    하지만 cost function을 최소화하는데 사용하지는 않는다.
  - **Test data**: 학습 과정에 관여하지 않음. 모델의 최종 성능을 평가하기 위해 사용됨.

- 만약 여러 후보 모델들 중에서 성능 평가를 통해 가장 좋은 모델을 선택하고 싶지 않은 경우에는 굳이 validation set을 만들지 않아도 된다. (하지만 test accuracy 예측 불가능 및 overfitting 등의 문제가 발생할 가능성이 높아진다.)

Cross-Validation (교차검증)

- Training data 중 일부를 validation data로 사용하지 못할 정도로
  training data를 충분히 확보하지 못했을 때 사용하는 방법

- When we split our training dataset to get a validation set,
  there's always **a risk of losing some crucial data** from the training set, or **of losing patterns** which might go unnoticed by the model.
  This will in turn lead to **overfitting** or **underfitting**.

- **k-fold 방식**: 대표적인 교차검증 방법
  - Training set을 k개의 subsets으로 분할한다.
  - Holdout validation 방식을 k회 적용한다.
  - 각 round 마다 순서대로 하나 subset을 validation set으로 지정한다. (나머지는 training set)
  - 각 round 마다 error를 산출하고, 모든 error들을 k로 나눈 평균값을 모델의 최종 성능으로 사용한다.

k-Fold Cross-Validation

						Validation Accuracy
Round 1	Validation	Test	Test	Test	Test	93%
Round 2	Test	Validation	Test	Test	Test	91%
Round 3	Test	Test	Validation	Test	Test	95%
Round 4	Test	Test	Test	Validation	Test	97%
Round 5	Test	Test	Test	Test	Validation	92%

Final Accuracy: (93 + 91 + 95 + 97 + 92) / 5 = 93.6 %

How to get MNIST Data


```
import numpy as np
import tensorflow as tf
import matplotlib.pypolot as pat
tf.logging.set_verbosity(tf.logging.ERROR)

from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

print("# of train      : “, mnist.train.num_examples)      # = len(mnist.train.images)      = 55000
print("# of validation: “, mnist.validation.num_examples) # = len(mnist.validation.images) = 5000
print("# of test       : ", mnist.test.num_examples)       # = len(mnist.test.images))      = 10000

print(mnist.train.images.shape)      # (55000, 784)
print(mnist.validation.images.shape) # ( 5000, 784)
print(mnist.test.images.shape)       # (10000, 784)

print(mnist.train.labels.shape)      # (55000, 10)
print(mnist.validation.labels.shape) # ( 5000, 10)
print(mnist.test.labels.shape)       # (10000, 10)

print(mnist.train.images.dtype, mnist.train.labels.dtype) # float32 float64
print(mnist.test.images.dtype, mnist.test.labels.dtype)   # float32 float64
```

How to get MNIST Data

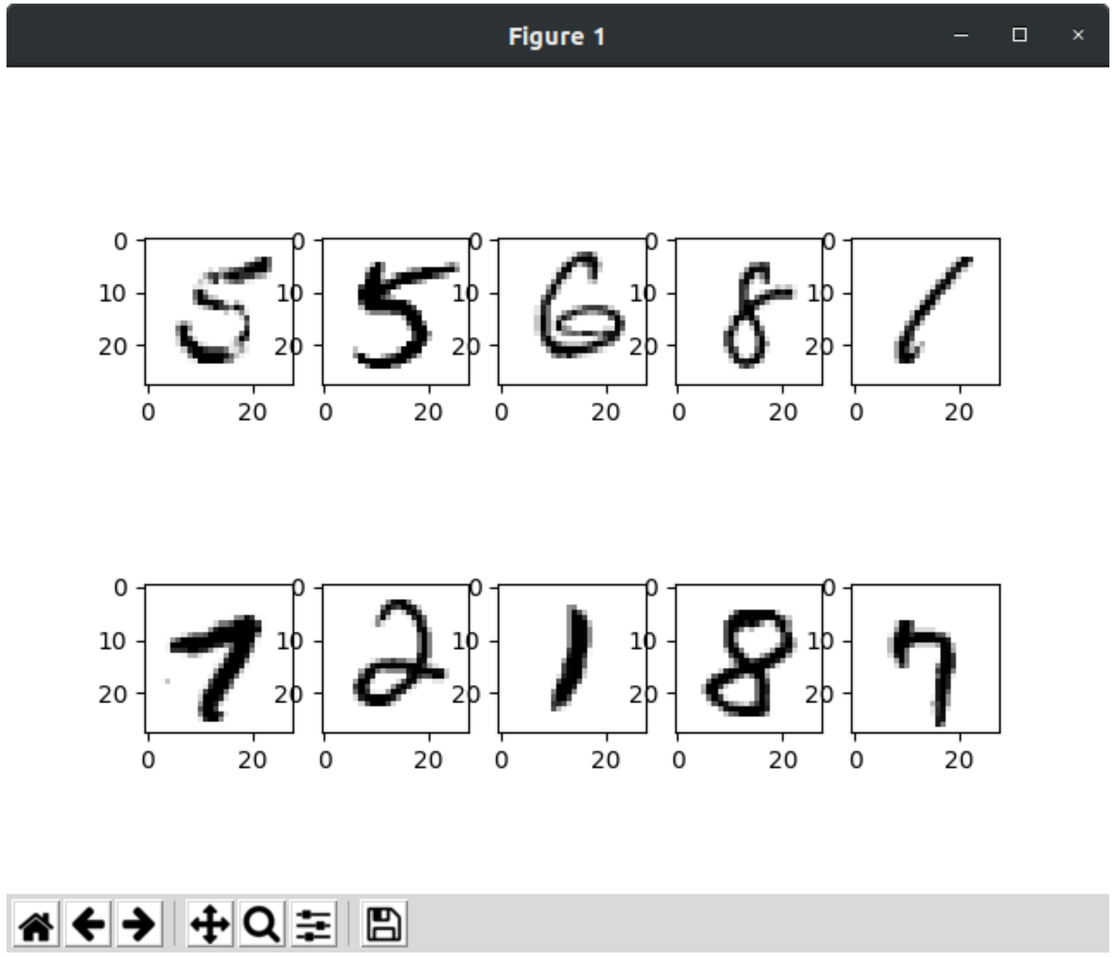

```
print(mnist.train.images[0])       # pixel values representing 7 (float32: 0.0 ~ 1.0)
print(mnist.train.labels[0])       # one-hot encoded label ex) [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
print(mnist.train.labels[0].shape) # (10,)

fig = plt.figure()

batch_size = 5

# Return the next five images and labels from the data set.
pixels, labels = mnist.train.next_batch(batch_size, shuffle=True)
for i in range(batch_size):
    print(np.argmax(labels[i]), ': ", labels[i])
    subplot = fig.add_subplot(2, 5, i+1)
    subplot.imshow(pixels[i].reshape(28,28), cmap=plt.cm.gray_r)

# Return the next five images and labels from the data set.
pixels, labels = mnist.train.next_batch(batch_size, shuffle=True)
for i in range(batch_size):
    print(np.argmax(labels[i]), ': ", labels[i])
    subplot = fig.add_subplot(2, 5, i+1)
    subplot.imshow(pixels[i].reshape(28,28), cmap=plt.cm.gray_r)

plt()
```

실행할 때마다 결과가 달라진다.

실행할 때마다 결과가 달라진다.

How to get MNIST Data

```
result:

5 : [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
5 : [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
6 : [0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
8 : [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
1 : [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
7 : [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
2 : [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
1 : [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
8 : [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
7 : [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
```



Minibatch / Epoch

- **미니배치 (minibatch)**
  - 데이터를 하나씩 학습시키는 것 보다 여러 개를 한꺼번에 학습시키는 것이 효율적임
  - 하지만 사용 가능한 리소스(resource)의 한계 때문에 데이터를 적당한 크기로 나누어서 학습시킴
  - batch size: 한 번에 처리하는 데이터의 수

- **에포크 (epoch)**
  - Training data 전체를 한 바퀴 도는 것
  - Training data를 여러 번 반복해서 재사용하는 경우가 많은데, 이 때 전체 데이터를 한 번 사용했을 때를 한 세대(epoch)라고 한다. 즉, 동일한 training data를 n번 반복하여 학습시킨다면 총 n세대의 iteration 이 수행된다.

MNIST Classification


```
import tensorflow as tf
tf.logging.set_verbosity(tf.logging.ERROR)

from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

# None: undetermined # of rows
X = tf.placeholder(tf.float32, [None, 784]) # inputs: 784 = 28x28
Y = tf.placeholder(tf.float32, [None, 10])  # outputs: 10 = 0~9

W1 = tf.Variable(tf.random_normal([784, 128], stddev=0.001))
L1 = tf.nn.relu(tf.matmul(X, W1))

W2 = tf.Variable(tf.random_normal([128, 128], stddev=0.001))
L2 = tf.nn.relu(tf.matmul(L1, W2))

W3 = tf.Variable(tf.random_normal([128, 10], stddev=0.001))
L3 = tf.matmul(L2, W3)

model = L3
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits=model, labels=Y))
optimizer = tf.train.AdamOptimizer(learning_rate=0.001).minimize(cost)
```

MNIST Classification

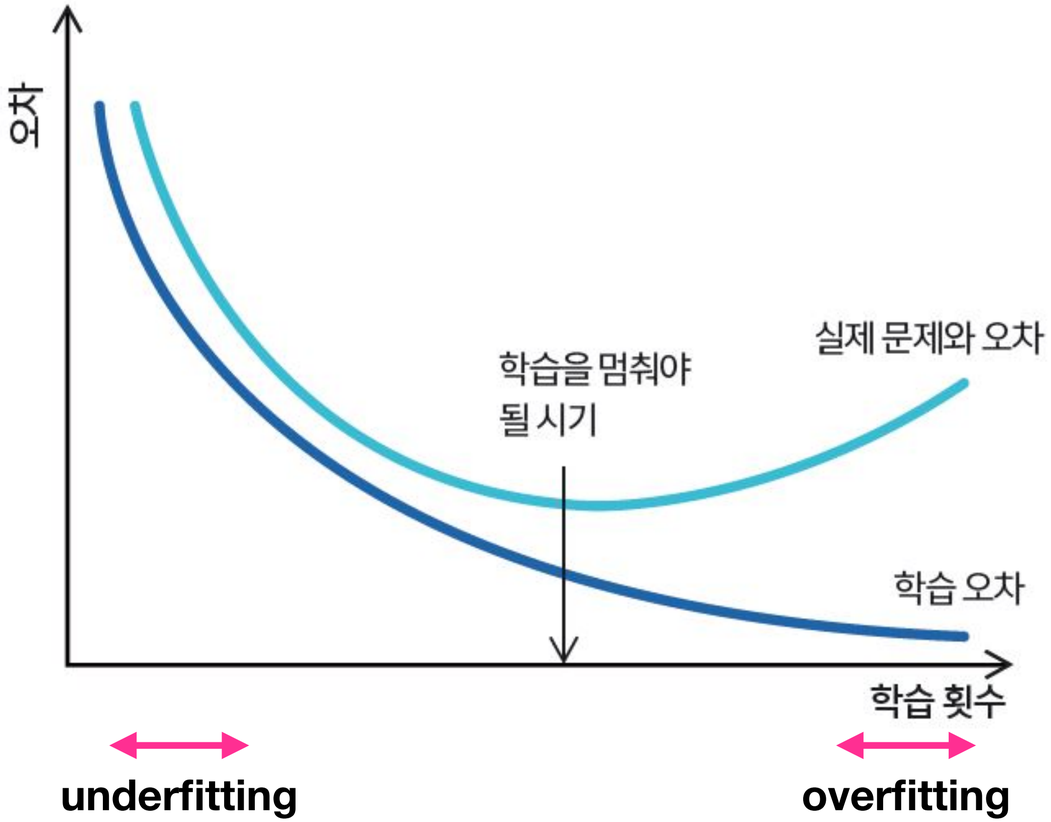

```
comparison = tf.equal(tf.argmax(model, 1), tf.argmax(Y, 1)) # 1: max. index in each row (each image)
accuracy = tf.reduce_mean(tf.cast(comparison, tf.float32))

batch_size = 1000 # the size of a batch
num_batches = int(mnist.train.num_examples / batch_size) # how many batches?

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

# training
for iEpoch in range(100): # reapeating training 100 times

    # without batch
    # x_data.shape=(55000, 784), y_data.shape=(55000, 10)
    if batch_size >= mnist.train.num_examples:
        x_data, y_data = mnist.train.images, mnist.train.labels
        sess.run(optimizer, feed_dict={X: x_data, Y: y_data})

    # with batch
    # x_data.shape=(batch_size, 784), y_data.shape=(batch_size, 10)
    else:
        for iBatch in range(num_batches):
            x_data, y_data = mnist.train.next_batch(batch_size)
            sess.run(optimizer, feed_dict={X: x_data, Y: y_data})

    train_accuracy = sess.run(accuracy, feed_dict={X: x_data, Y: y_data})
    print('Epoch:', iEpoch, ', Accuracy for Training Data:', train_accuracy)
```

MNIST Classification


```
# testing
#l = mnist.test.labels
#print(l.shape)                                          # (10000, 10)
#print(sess.run(tf.argmax(l,1)))                         # [7 2 1 ... 4 5 6]
#print(sess.run(tf.equal(tf.argmax(l,1), tf.argmax(l,1)))) # [ True  True True ...  True  True  True]

x_data, y_data = mnist.test.next_batch(batch_size)
test_accuracy = sess.run(accuracy, feed_dict={X: x_data, Y: y_data})
print("Accuracy for Test Data:", test_accuracy)

sess.close()
```

과적합 (Overfitting)

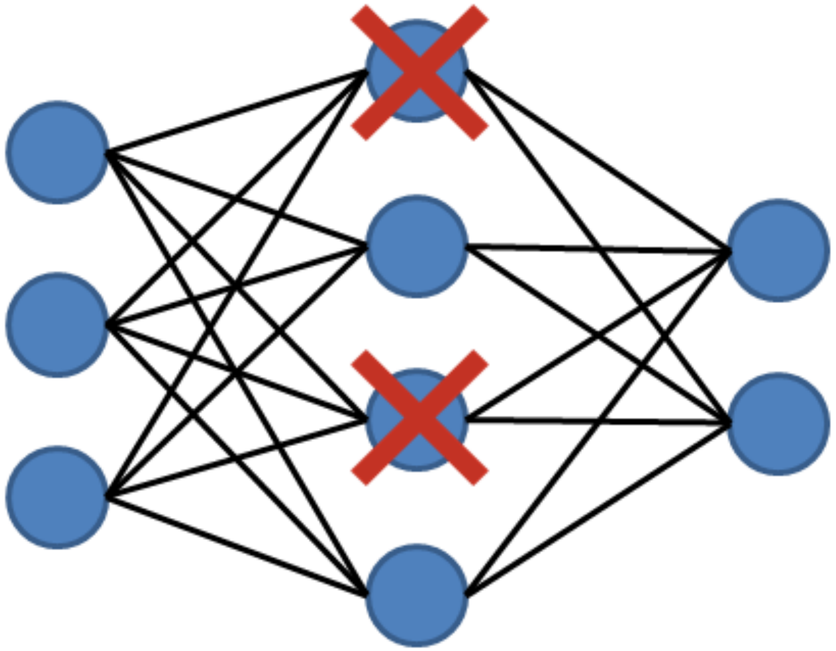
- 주어진 학습 데이터에만 너무 딱 맞게 학습되어서 그 외의 데이터에는 잘 맞지 않는 현상



http://dongascience.donga.com/news.php?idx=11225

Dropout

- **드롭아웃 (dropout)**
  - 과적합 문제를 해결하기 위한 방법 중 하나
  - 학습시 각 단계 마다 일부 뉴런을 사용하지 않도록 제거하여 전체 신경망 중 일부분만을 사용하도록 하는 것
  - 다만 드롭아웃 기법을 사용해서 학습하더라도 학습이 끝난 후 예측 단계에서는 전체 신경망을 사용해야 함



https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/

MNIST Classification


```
import tensorflow as tf

tf.logging.set_verbosity(tf.logging.ERROR)
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

# None: undetermined # of rows
X = tf.placeholder(tf.float32, [None, 784]) # inputs: 784 = 28x28
Y = tf.placeholder(tf.float32, [None, 10])  # outputs: 10 = 0~9

keep_ratio = tf.placeholder(tf.float32) # dropout factor

W1 = tf.Variable(tf.random_normal([784, 128], stddev=0.1))
L1 = tf.nn.relu(tf.matmul(X, W1))
L1 = tf.nn.dropout(L1, keep_ratio)

W2 = tf.Variable(tf.random_normal([128, 128], stddev=0.1))
L2 = tf.nn.relu(tf.matmul(L1, W2))
L2 = tf.nn.dropout(L2, keep_ratio)

W3 = tf.Variable(tf.random_normal([128, 10], stddev=0.1))
L3 = tf.matmul(L2, W3)

model = L3
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits=model, labels=Y))
optimizer = tf.train.AdamOptimizer(0.01).minimize(cost)
```

MNIST Classification

2/2

```
comparison = tf.equal(tf.argmax(model, 1), tf.argmax(Y, 1)) # 1: max. index in each row (each image)
accuracy = tf.reduce_mean(tf.cast(comparison, tf.float32))

batch_size = 1000 # the size of a batch
num_batches = int(mnist.train.num_examples / batch_size) # how many batches?

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

# training
for iEpoch in range(100): # reapeating training 100 times
    for iBatch in range(num_batches):
        x_data, y_data = mnist.train.next_batch(batch_size)
        sess.run(optimizer, feed_dict={X: x_data, Y: y_data, keep_ratio: 0.8})

    train_accuracy = sess.run(accuracy, feed_dict={X: x_data, Y: y_data, keep_ratio: 1.0})
    print('Epoch:', iEpoch, ', Accuracy for Training Data:', train_accuracy)

# testing
x_data, y_data = mnist.test.next_batch(batch_size)
test_accuracy = sess.run(accuracy, feed_dict={X: x_data, Y: y_data, keep_ratio: 1.0})
print("Accuracy for Test Data:", test_accuracy)

sess.close()
```

How to load MNIST Data using Keras

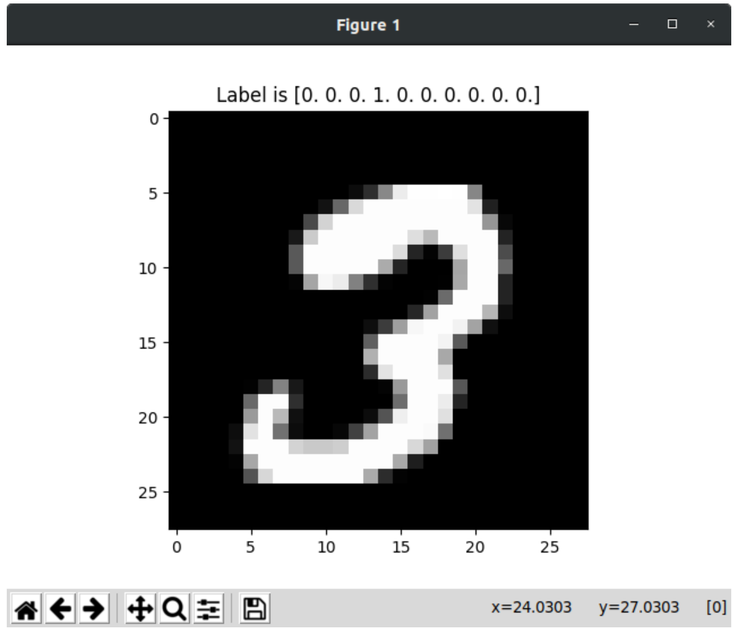
- Warning while loading MNIST data using **read_data_sets()**
  - read_data_sets (from tensorflow.contrib.learn.python.learn.datasets.mnist) is **deprecated** and will be removed in a future version.
  - Please write **your own downloading logic**.

- 이러한 warning을 나타나지 않게 하기 위해서는 직접 MNIST data loader를 작성해야 하는데 여러 가지 방법이 있을 수 있지만 **tf.keras.datasets.mnist.load_data()**를 이용해서 작성해보자.

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

##############
# TensorFlow #
tf.logging.set_verbosity(tf.logging.ERROR)
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", reshape=True, validation_size=5000, one_hot=True)

# training data
print(mnist.train.images.dtype == “float32”) # True
print(mnist.train.labels.dtype == “float64") # True
print(mnist.train.images.shape == tf.TensorShape([55000,784])) # True
print(mnist.train.labels.shape == tf.TensorShape([55000,10])) # True

# validation data
print(mnist.validation.images.dtype == “float32") # True
print(mnist.validation.labels.dtype == “float64") # True
print(mnist.validation.images.shape == tf.TensorShape([5000,784])) # True
print(mnist.validation.labels.shape == tf.TensorShape([5000,10])) # True

# test data
print(mnist.test.images.dtype == “float32") # True
print(mnist.test.labels.dtype == “float64") # True
print(mnist.test.images.shape == tf.TensorShape([10000,784])) # True
print(mnist.test.labels.shape == tf.TensorShape([10000,10]) # True
```

```
#########
# Keras #

# x: image, y: label
data_train, data_test = tf.keras.datasets.mnist.load_data()
(x_train, y_train) = data_train
(x_test, y_test) = data_test

# simpler code:
#(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

print(x_train.dtype == “uint8") # True
print(len(x_train)==60000 & len(y_train)==60000) # True

print(x_test.dtype == “uint8") # True
print(len(x_test)==10000 & len(y_test)==10000) # True

# uint8 (0~255) -> float32 (0.0~1.0)
# array dimension: 28x28 -> 784
x_train = x_train.reshape(x_train.shape[0], 784).astype('float32') / 255
x_test  = x_test.reshape(x_test.shape[0], 784).astype('float32') / 255

# uint8 -> float32 -> float64
y_train = tf.keras.utils.to_categorical(y_train, 10).astype('float64')
y_test = tf.keras.utils.to_categorical(y_test, 10).astype('float64')
```

```
# split: train data set -> validation data set
x_validation = x_train[-5000:]
y_validation = y_train[-5000:]
x_train = x_train[:-5000]
y_train = y_train[:-5000]

def next_batch(images, labels, nBatches):
    indices = np.arange(0, len(images))
    np.random.shuffle(indices)
    indices = indices[:nBatches]
    return [images[i] for i in indices], [labels[i] for i in indices]

images, labels = next_batch(x_train, y_train, 1000)

plt.title('Label: {label}'.format(label=labels[0]))
plt.imshow(images[0].reshape((28,28)), cmap='gray')
plt.show()
```



Irreducible / Reducible Error

- **Irreducible error**
  - 시스템 고유의 노이즈 등에 의해 발생하는 에러
  - 원인을 알 수도 없고 이를 해결할 수도 없다.

- **Reducible error**
  - 모델을 수정함으로서 줄일 수 있는 에러
  - “error due to **bias**”와 “error due to **variance**”로 나눌 수 있다.

https://medium.com/wwblog/the-bias-variance-tradeoff-4774c5231c88

Error due to Bias / Variance

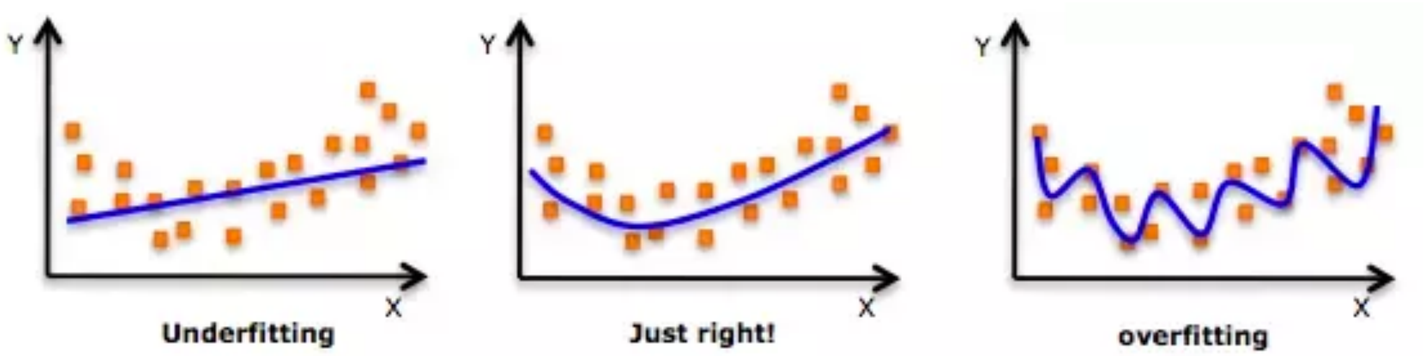
- **Error due to bias**
  - 모델이 너무 **rigid**해서 training set의 모든 features를 표현하지 못하는 경우 (**under-fitting**)
  - Training data가 말해주는 정보와는 달리 특정 결과에 편향되어 있기 때문에 붙여진 이름

- **Error due to variance**
  - 모델이 너무 **sensitive**해서 고려하지 말아야 할 에러들까지 features로 판단하여 반영하는 경우 (**over-fitting**)
  - 주어진 training set에 따라서 너무나도 다른 결과들을 보여주기 때문에 붙여진 이름

Bias-Variance Tradeoff

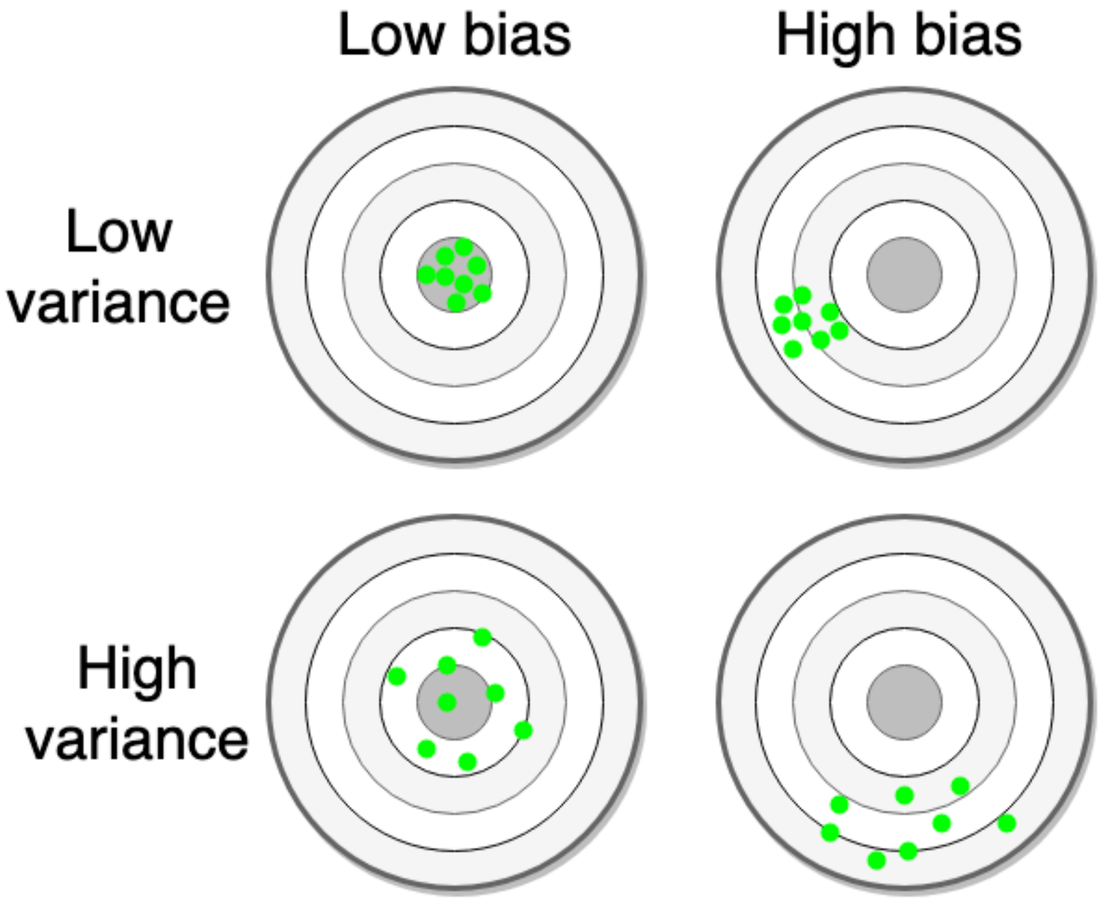
https://medium.com/wwblog/the-bias-variance-tradeoff-4774c5231c88

low-order polynomial ⟷ **high-order polynomial**



https://www.quora.com/Whats-the-difference-between-overfitting-and-underfitting

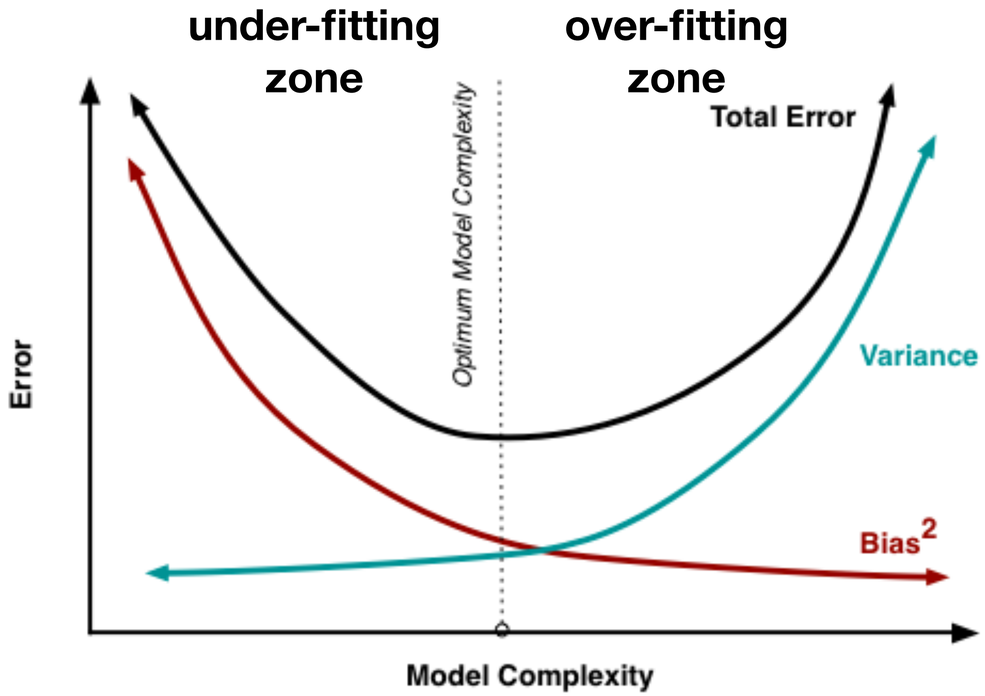
Error due to Bias / Variance



http://www.machinelearningtutorial.net/2017/01/26/the-bias-variance-tradeoff/

Bias-Variance Tradeoff

https://medium.com/wwblog/the-bias-variance-tradeoff-4774c5231c88



https://stats.stackexchange.com/questions/336433/bias-variance-tradeoff-math

Total Error = Bias^2 + Variance + Irreducible Error

Regularization

- **과적합(over-fitting)** 문제를 해결하기 위한 또다른 방법으로 **정규화**라는 것이 있다.

- 복잡한 모델에 일종의 **penalty**를 부과하는 방법

- 학습 과정에서 **특정 가중치 값들이 커지면서 over-fitting**이 되는 경우가 발생할 수 있다.
- 이렇게 특정 가중치가 크면 예측값이 정상적인 규칙을 벗어나 있는 경우도 예측이 가능해지지만 **비정상적인 데이터**를 **예측**할 수는 있지만 이것을 맞다고 판단하는 것은 옳지 않다.
  (예외적인 상황일 가능성이 큼)

- **"오컴의 면도날(Occam's razor)" 이론**: 같은 현상을 설명하는 두 개의 주장이 있다면 간단한 쪽을 선택

Regularization

- Regularization term을 추가하여 W값이 클 경우에 penalty를 부여한다.
- 따라서, **가중치** 역시 **최소**가 되도록 업데이트 한다.

L1 Regularization

원래의 cost function

$$\text{Cost} = \boxed{\sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2} + \lambda \sum_{j=0}^{M} |W_j|$$

L2 Regularization

원래의 cost function

$$\text{Cost} = \boxed{\sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2} + \underbrace{\lambda \sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}$$

http://laid.delanover.com/difference-between-l1-and-l2-regularization-implementation-and-visualization-in-tensorflow/

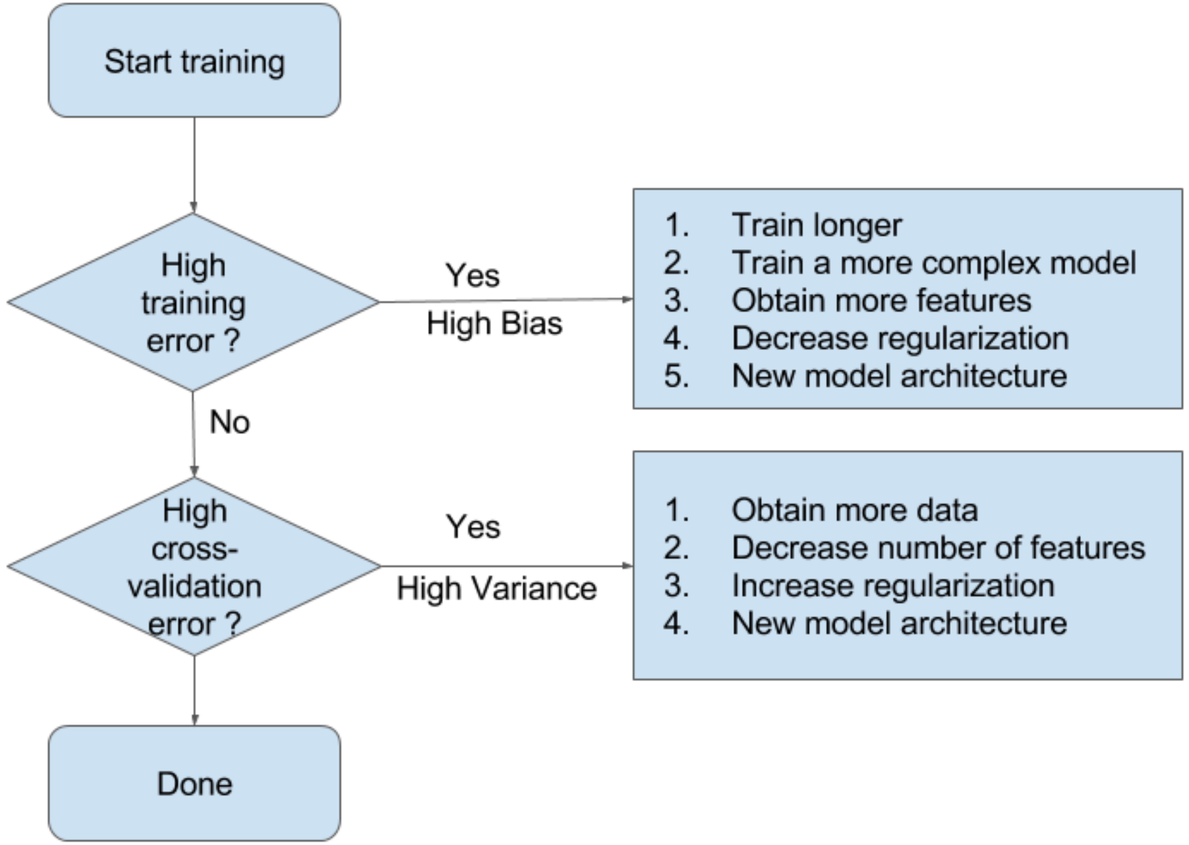
How to Detect Problems

- **High bias**
  - **Training** set에 대한 에러가 **크다**.
  - **Validation** set에 대한 에러 크기가 training set의 경우와 **비슷**하다.

- **High variance**
  - **Training** set에 대한 에러가 **작다**.
  - **Validation** set에 대한 에러 크기가 **매우 크다**.

https://www.quora.com/How-does-an-engineer-find-out-whether-his-supervised-machine-learning-model-suffers-from-more-bias-or-more-variance

How to Solve Problems



https://www.learnopencv.com/bias-variance-tradeoff-in-machine-learning/

How to fix a high bias problem?

- **Train longer**: Many machine learning algorithms are set up as iterative optimization problems where the training error (or a function of the training error) is minimized. Just letting the algorithm run for more hours or days can help reduce the bias. In Neural Networks, you can change a parameter called the “learning rate” which will help the training error go down faster.

- **Train a more complex model**: A more complex model will fit the training data better. In the problem N00b was looking at, he could increase the degree of the polynomial to get a more complex model. In the case of Neural Networks, one can add more layers. Finally, in the case of an SVM, you can use a non-linear SVMs instead of a linear one.

- **Obtain more features**: Sometimes you just do not have enough information to train a model. For example, if you are trying to train a model that can predict the gender of a person based on the color of their hair, the problem might be impossible to solve. But if you add a new feature — the length of the hair — the problem becomes more tractable.

- **Decrease regularization**: Recall that N00b was trying to fit a polynomial to his data. In his naive implementation the parameters could take any values. This makes the model very flexible. However, we may want to constrain the flexibility of the model to prevent overfitting. So usually a regularization term ( ) is added to the cost function we are trying to minimize. But in this case, we do not have an overfitting problem. We have an under-fitting problem, and we can reduce it by reducing the value of .

- **New model architecture**: This is just another way of saying that if nothing works, start over.

https://www.learnopencv.com/bias-variance-tradeoff-in-machine-learning/

How to fix a high variance problem?

- **Obtain more data**: Because the validation error is large, it means that the training set and the validation set that were randomly chosen from the same dataset, somehow have different characteristics. This usually means that you do not have enough data and you need to collect more.

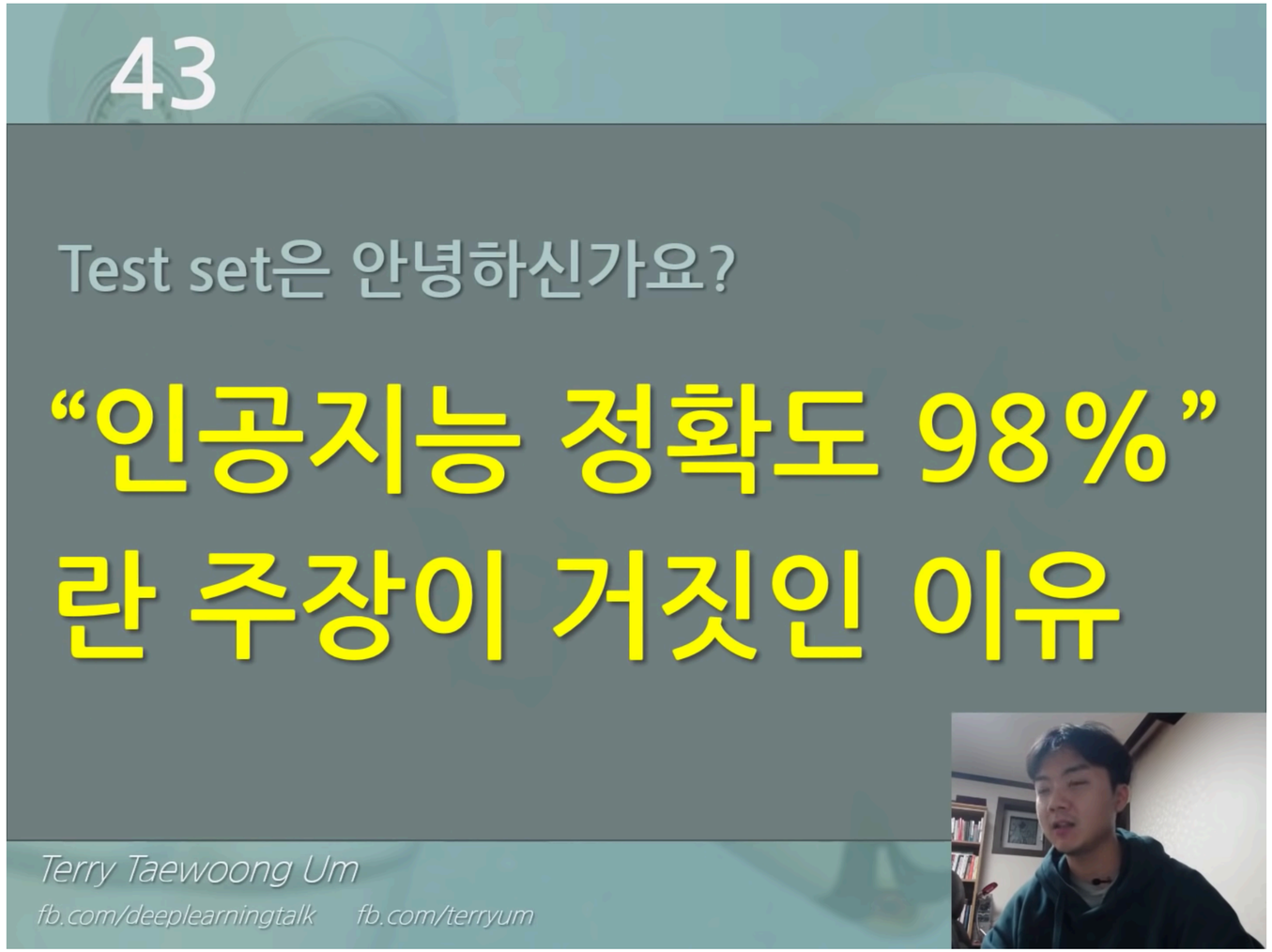
- **Decrease number of features**: Sometimes collecting more data is not an option. In that case, you can reduce the number of features. You may have to remove features manually. For example, in our previous example of identifying the gender of a person based on hair color and hair length, you may decide to drop hair color and keep hair length.

- **Increase regularization**: When we have a high variance problem the model is fitting the training data. In fact, the model is probably fitting even the noise in training set and therefore not performing as well on the validation set. We can reduce the flexibility of the model by using regularization that puts constraints on the magnitude of the parameters. This is done by adding a regularization term to the cost function. When we are fitting a polynomial model, the regularization term is of the form . So we are not only trying to reduce the training set error, we are also trying to make sure the magnitudes of the parameters are small, thus limiting the flexibility of the model. Increasing increases regularization.

- **New model architecture**: Try something else. Better luck next time!

https://www.learnopencv.com/bias-variance-tradeoff-in-machine-learning/

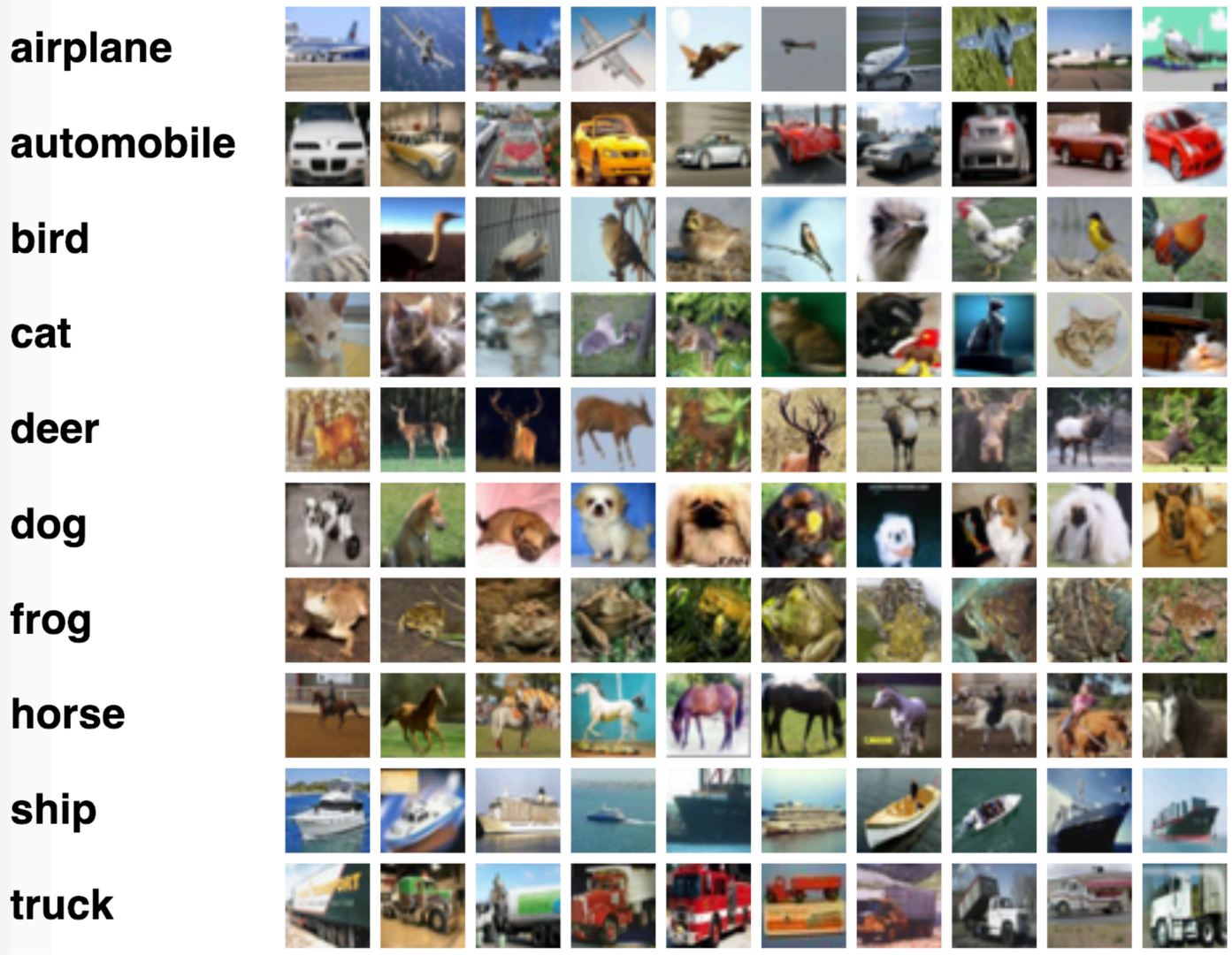
“인공지능 정확도 98%” 주장이 거짓인 이유



https://www.youtube.com/watch?v=neAJ9eldUWM

CIFAR-10 / 100

CIFAR-10 Data Set



https://www.cs.toronto.edu/~kriz/cifar.html

CIFAR-10 Data Set

- **10 classes**: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck
- **60,000** images
  - Training set: 5개×10,000장 (5개 파일마다 개수의 차이는 조금 있음), 10종류×5,000장
  - Test set: 1개×10,000장, 10종류×1,000장
- **32×32** with **3 channels**
- **1 byte** per each channel (**0~255 unsigned char**)
- The classes are completely **mutually exclusive**. (**no overlap** between two different classes)
- The archive contains the files (data_batch_1 ~ data_batch_2, test_batch) that are Python **“pickled” objects** produced with **cPickle**.

MNIST vs CIFAR-10/100

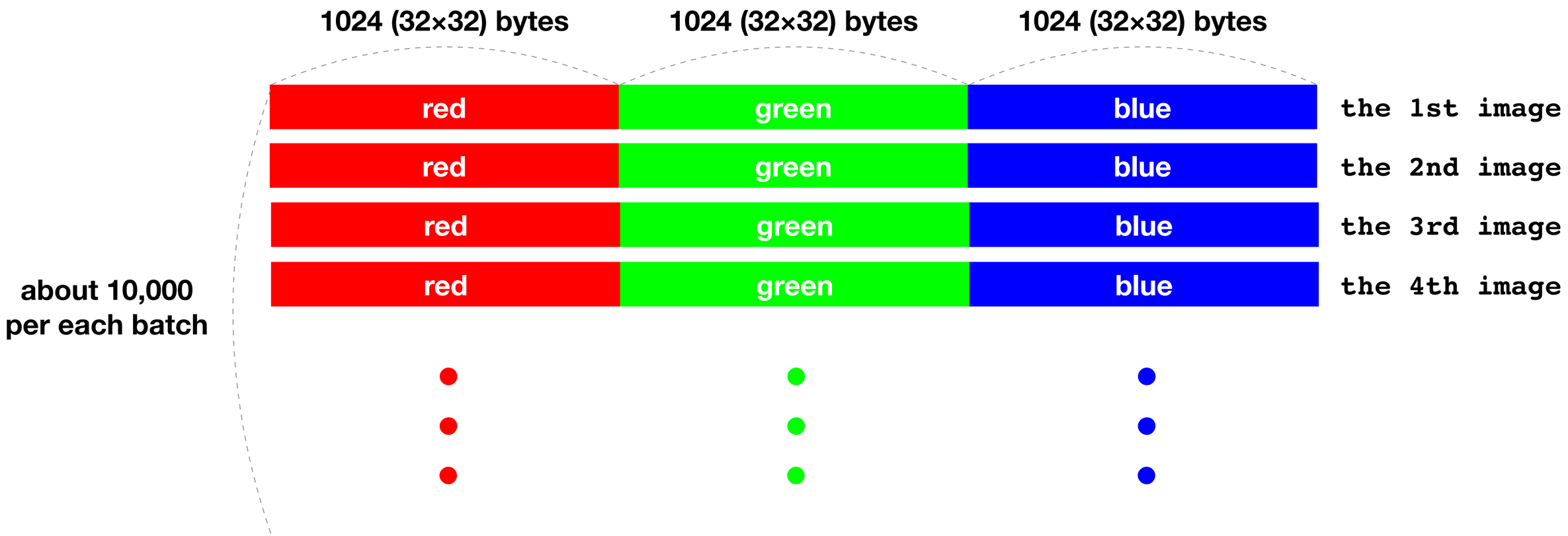
- **MNIST**
  - 60,000 training image/labels, 10,000 test images/labels
  - Pixel data: 28×28×1 (grayscale image), 0.0~1.0 (float32)
  - Label data: 10 digits (one-hot encoded 0~9), float64

- **CIFAR-10 & CIFAR-100**
  - 50,000 training images/labels, 10,000 test images/labels
  - Pixel data: 32×32×3 (RGB image), 0~255 (uint8)
  - Label data: 1 digit (label index), uint8 (for train data) / int64 (for test data)

https://keras.io/datasets/

CIFAR-10/100 Image Format

- **Row-major order 2D array**



How to get CIFAR 10 Data


```
import tensorflow as tf
import matplotlib.pyplot as plt
import tensorflow.keras.datasets.cifar10 as cifar10

# download location: ~/.keras/datasets/cifar-10-batches-py
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

print('# of training data: ', len(x_train)) # 50000
print('# of test     data: ', len(x_test))  # 10000

print(x_train.shape, y_train.shape) # (50000, 32, 32, 3) (50000, 1)
print(x_test.shape, y_test.shape)   # (10000, 32, 32, 3) (10000, 1)

print(x_train.dtype, y_train.dtype) # uint8 uint8
print(x_test.dtype, y_test.dtype)   # uint8 int64

print(x_train[0])       # pixel values (uint8: 0~255)
print(x_train[0].shape) # (32, 32, 3)

print(y_train[0])       # label index: [6]
print(y_train[0].shape) # (1,)
```

How to get CIFAR 10 Data


```
label = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

fig = plt.figure()
fit.set_size_inches(10, 5)

for i in range(15):

    subplot = fig.add_subplot(3, 5, i+1)  # i+1:the i-th subplot index

    subplot.set_xticks([])                # no scale in x-axis
    subplot.set_yticks([])                # no scale in y-axis

    # label
    label_index = y_train[i][0]
    subplot.set_title(label[label_index])

    # image
    image = x_train[i]
    subplot.imshow(image[i])

plt.show()
```

How to get CIFAR 100 Data


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

import tensorflow.keras.datasets.cifar100 as cifar100

# download location: ~/.keras/datasets/cifar-100-python
(x_train, y_train), (x_test, y_test) = cifar100.load_data()

print('# of training data: ', len(x_train)) # 50000
print('# of test     data: ', len(x_test))  # 10000

print(x_train.shape, y_train.shape) # (50000, 32, 32, 3) (50000, 1)
print(x_test.shape, y_test.shape)   # (10000, 32, 32, 3) (10000, 1)

print(x_train.dtype, y_train.dtype) # uint8 int64
print(x_test.dtype, y_test.dtype)   # uint8 int64

print(x_train[0])       # pixel values (uint8: 0~255)
print(x_train[0].shape) # (32, 32, 3)

print(y_train[0])       # label index: [19]
print(y_train[0].shape) # (1,)
```

How to get CIFAR 100 Data

2/2

```
label = ['apple', 'aquarium_fish', 'baby', 'bear', 'beaver', 'bed', 'bee', 'beetle', 'bicycle', 'bottle',
'bowl', 'boy', 'bridge', 'bus', 'butterfly', 'camel', 'can', 'castle', 'caterpillar', 'cattle', 'chair',
'chimpanzee', 'clock', 'cloud', 'cockroach', 'couch', 'crab', 'crocodile', 'cup', 'dinosaur', 'dolphin',
'elephant', 'flatfish', 'forest', 'fox', 'girl', 'hamster', 'house', 'kangaroo', 'keyboard', ‘lamp',
'lawn_mower', 'leopard', 'lion', 'lizard', 'lobster', 'man', 'maple_tree', 'motorcycle', ‘mountain', 'mouse',
'mushroom', 'oak_tree', 'orange', 'orchid', 'otter', 'palm_tree', 'pear', ‘pickup_truck', 'pine_tree',
'plain', 'plate', 'poppy', 'porcupine', 'possum', 'rabbit', 'raccoon', 'ray', ‘road', 'rocket', 'rose',
'sea', 'seal', 'shark', 'shrew', 'skunk', 'skyscraper', 'snail', 'snake', ‘spider', 'squirrel', 'streetcar',
'sunflower', 'sweet_pepper', 'table', 'tank', 'telephone', ‘television', 'tiger', 'tractor', 'train',
'trout', 'tulip', 'turtle', 'wardrobe', 'whale', 'willow_tree', ‘wolf', 'woman', 'worm']

fig = plt.figure()
fit.set_size_inches(10, 5)

for i in range(15):
    subplot = fig.add_subplot(3, 5, i+1)  # i+1:the i-th subplot index
    subplot.set_xticks([])                # no scale in x-axis
    subplot.set_yticks([])                # no scale in y-axis

    label_index = y_train[i][0]           # label
    subplot.set_title(label[label_index])
    image = x_train[i]                    # image
    subplot.imshow(image[i])

plt.show()
```

How to get CIFAR Labels


```
import os
import pickle

def unpickle(file_path, file_name, mode):
    file_path_name = os.path.join(file_path, file_name)
    fo = open(file_path_name, ‘rb')
    dict = pickle.load(fo, encoding=mode)
    fo.close()
    return dict

home_path = os.getenv(‘HOME')

# CIFAR-10
file_path = os.path.join(home_path, ‘.keras/datasets/cifar-10-batches-py')
info_dict = unpickle(file_path, 'batches.meta', ‘ASCII')
label_names = info_dict[‘label_names']
print(label_names) # ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship',
'truck']
```

How to get CIFAR Labels


```
# CIFAR-100
file_path = os.path.join(home_path, '.keras/datasets/cifar-100-python')
info_dict = unpickle(file_path, 'meta', 'bytes')

label_names = info_dict[b'coarse_label_names'] # b: binary
print(label_names) #  [b'aquatic_mammals', b'fish', b'flowers', b'food_containers', b'fruit_and_vegetables',
b'household_electrical_devices', b'household_furniture', b'insects', b'large_carnivores', b'large_man-
made_outdoor_things', b'large_natural_outdoor_scenes', b'large_omnivores_and_herbivores', b'medium_mammals',
b'non-insect_invertebrates', b'people', b'reptiles', b'small_mammals', b'trees', b'vehicles_1',
b'vehicles_2']

label_names = info_dict[b'fine_label_names'] # b: binary
print(label_names) # [b'apple', b'aquarium_fish', b'baby', b'bear', b'beaver', b'bed', b'bee', b'beetle',
b'bicycle', b'bottle', b'bowl', b'boy', b'bridge', b'bus', b'butterfly', b'camel', b'can', b'castle',
b'caterpillar', b'cattle', b'chair', b'chimpanzee', b'clock', b'cloud', b'cockroach', b'couch', b'crab',
b'crocodile', b'cup', b'dinosaur', b'dolphin', b'elephant', b'flatfish', b'forest', b'fox', b'girl',
b'hamster', b'house', b'kangaroo', b'keyboard', b'lamp', b'lawn_mower', b'leopard', b'lion', b'lizard',
b'lobster', b'man', b'maple_tree', b'motorcycle', b'mountain', b'mouse', b'mushroom', b'oak_tree', b'orange',
b'orchid', b'otter', b'palm_tree', b'pear', b'pickup_truck', b'pine_tree', b'plain', b'plate', b'poppy',
b'porcupine', b'possum', b'rabbit', b'raccoon', b'ray', b'road', b'rocket', b'rose', b'sea', b'seal',
b'shark', b'shrew', b'skunk', b'skyscraper', b'snail', b'snake', b'spider', b'squirrel', b'streetcar',
b'sunflower', b'sweet_pepper', b'table', b'tank', b'telephone', b'television', b'tiger', b'tractor',
b'train', b'trout', b'tulip', b'turtle', b'wardrobe', b'whale', b'willow_tree', b'wolf', b'woman', b'worm']
```

CIFAR-10 Classification


```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import tensorflow.keras.datasets.cifar10 as cifar10
tf.logging.set_verbosity(tf.logging.ERROR)

# download location: ~/.keras/datasets/cifar-10-batches-py
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# flatten (32x32x3 to 3072)
# uint8 -> float32
# 0.0~1.0 -> 0~255
x_train = x_train.reshape(-1, 3072).astype('float32')/255.0
x_test  = x_test.reshape(-1, 3072).astype('float32')/255.0

# to the one-hot encoding
y_train = keras.utils.to_categorical(y_train, num_classes=10, dtype='float32')
y_test  = keras.utils.to_categorical(y_test, num_classes=10, dtype='float32')

def next_batch(images, labels, batch_size):
    indices = np.arange(0, len(images))
    np.random.shuffle(indices)
    indices = indices[:batch_size]
    images = [images[i] for i in indices]
    labels = [labels[i] for i in indices]
    return images, labels
```

CIFAR-10 Classification


```
# None: undetermined # of rows
X = tf.placeholder(tf.float32, [None, 3072]) # inputs: 3072 = 32x32x3
Y = tf.placeholder(tf.float32, [None, 10])   # outputs: 10 = 0~9

W1 = tf.Variable(tf.random_normal([3072, 256], stddev=0.001))
L1 = tf.nn.relu(tf.matmul(X, W1))

W2 = tf.Variable(tf.random_normal([256, 256], stddev=0.001))
L2 = tf.nn.relu(tf.matmul(L1, W2))

W3 = tf.Variable(tf.random_normal([256, 10], stddev=0.001))
L3 = tf.matmul(L2, W3)

model = L3
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits=model, labels=Y))
optimizer = tf.train.AdamOptimizer(learning_rate=0.001).minimize(cost)

comparison = tf.equal(tf.argmax(model, 1), tf.argmax(Y, 1)) # 1: max. index in each row (each image)
accuracy = tf.reduce_mean(tf.cast(comparison, tf.float32))

batch_size = 1000 # the size of a batch
num_batches = int(len(x_train) / batch_size) # how many batches?
```

CIFAR-10 Classification


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

# training
for iEpoch in range(100): # reapeating training 100 times

    # without batch
    if batch_size >= len(x_train):
        x_data, y_data = next_batch(x_train, y_train, batch_size)
        sess.run(optimizer, feed_dict={X: x_data, Y: y_data})

    # with batch
    else:
        for iBatch in range(num_batches):
            x_data, y_data = x_train, y_train
            sess.run(optimizer, feed_dict={X: x_data, Y: y_data})

    train_accuracy = sess.run(accuracy, feed_dict={X: x_data, Y: y_data})
    print('Epoch:', iEpoch, ', Accuracy for Training Data:', train_accuracy)

x_data, y_data = mnist.test.next_batch(batch_size)
test_accuracy = sess.run(accuracy, feed_dict={X: x_data, Y: y_data})
print("Accuracy for Test Data:", test_accuracy)

sess.close()
```

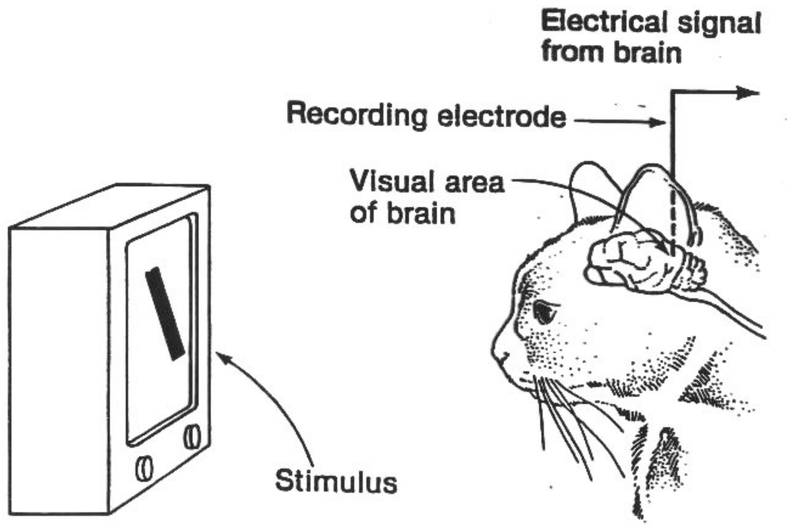
CNN

CIFAR-10 Classification 코드의 문제점

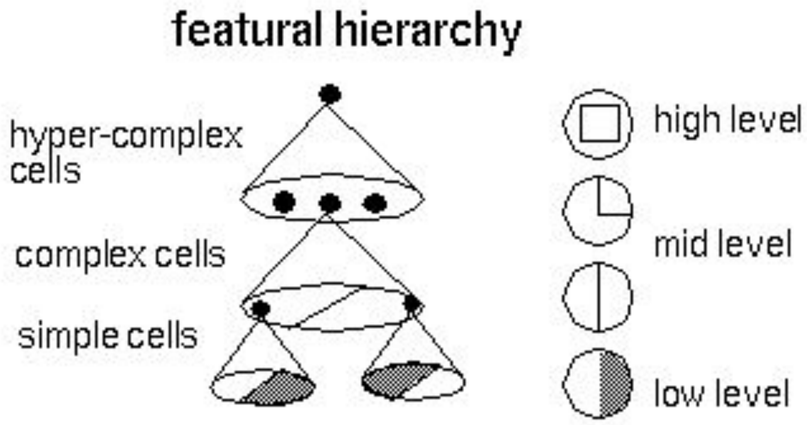
- 너무 **느리고** (계산량이 많고), **부정확**하다.
- 지금까지 다룬 신경망은 현재 계층의 모든 뉴런이 이전 계층의 모든 뉴런과 연결되어 있는 **FCNN**이다.
  (FCNN를 **Affine Layer**라고 부르기도 한다.)
- FCNN은 3차원 RGB 데이터를 **평탄화(flatten)**시킨 **1차원 데이터**를 **입력**으로 제공해야 한다.
- 이러한 평탄화(flatten) 과정에서 3차원 데이터의 **공간 정보**가 **상실**된다.
  (공간적으로 가까이 있는 픽셀값이 1차원 array 상에서는 멀리 떨어져 있다.)
- 또한, 인간이 **물체를 인식하는 방식**을 전혀 반영하고 있지 못하다.

Image Classification (이미지 분류)

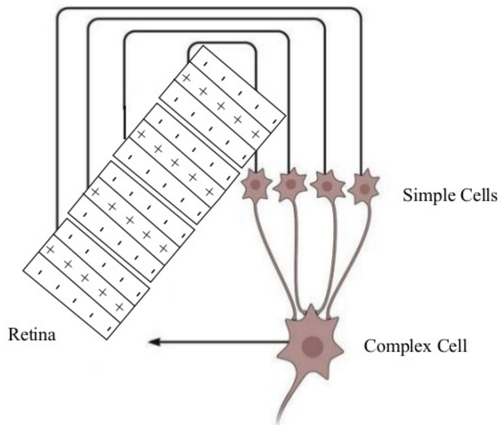
- 입력 이미지(input image)로 부터 개, 고양이 등과 같은 클래스(class)를 출력(output)하는 일련의 과정
- 인간의 뇌는 어떻게 사물을 인지하는가?
- **Hubel & Wiesel**은 고양이 실험을 통해 **신경 세포 마다 반응하는 시각 정보 특징이 다름**을 발견함 (가장자리, 수직선, 수평선, 대각선 등)



https://goodpsychology.wordpress.com/2013/03/13/235/



http://cns-alumni.bu.edu/~slehar/webstuff/pcave/hubel.html



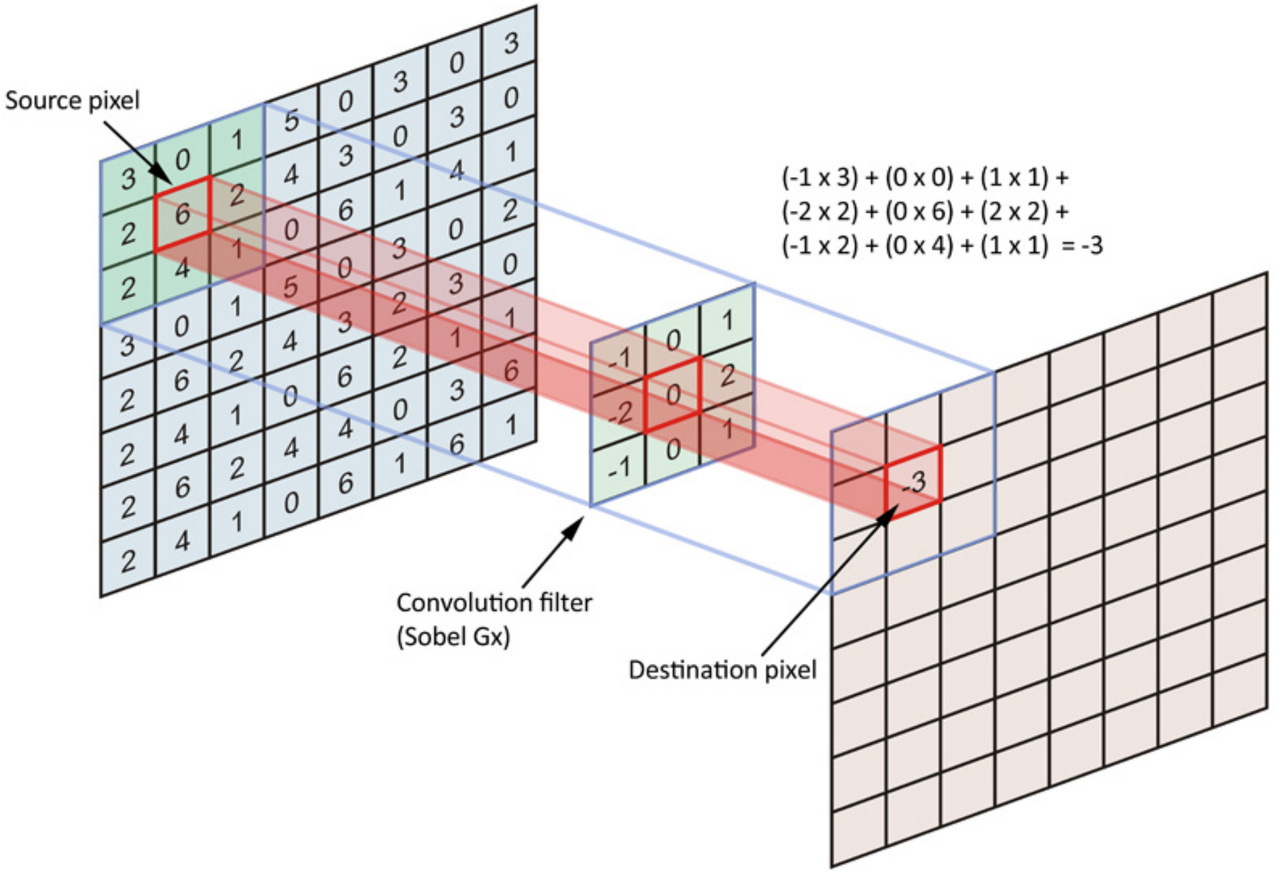
https://www.slideshare.net/starfish57/cnshubelandwiesel

Vision is hierarchical phenomenon.

CNN (합성곱 신경망)

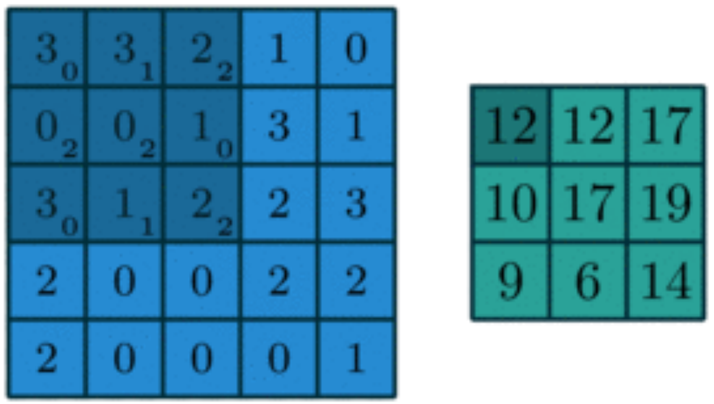
- **CNN**: Convolutional Neural Network
- 지금까지 살펴본 **FNN**(Fully-connected Neural Network) 외에도 신경망을 구성하는 방식은 다양함
- 지금까지의 예제들은 하나의 이미지를 구성하는 픽셀값들을 **1차원 배열**로 변환하여 학습을 하였음
- CNN은 이미지의 **2차원 공간 정보**를 유지하면서 **각 부분별 특징**을 효과적으로 인식하는 방식으로 작동함
- **Convolution (합성곱)**: signal 또는 image processing에서 특징을 추출하기 위해 사용하는 알고리즘

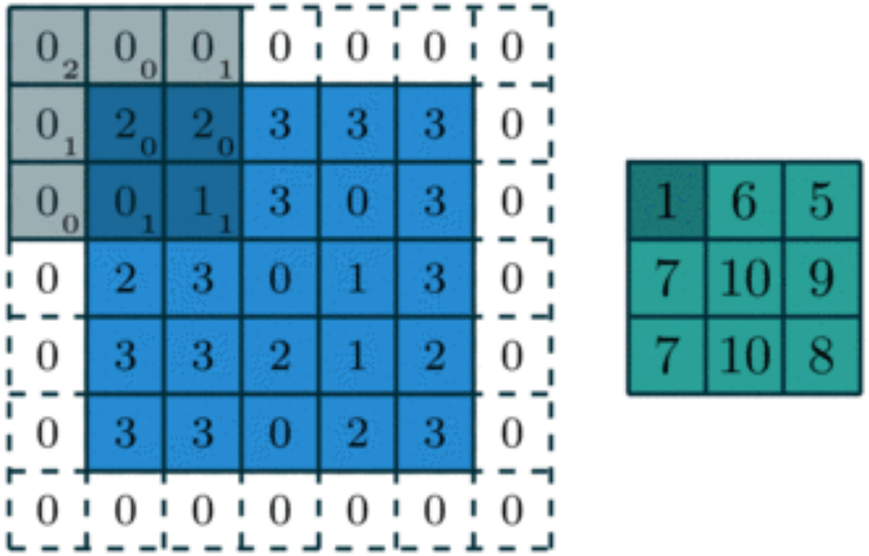
Convolution (합성곱)



http://openresearch.ai/t/network-in-network/39

Convolution (합성곱)





input image size	5×5×1
kernel size	3×3×1
output image size	3×3×1
stride	1
padding	0

input image size	5×5×1
kernel size	3×3×1
output image size	3×3×1
stride	2
padding	1

http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html

Convolution (합성곱)

- Input Image Size: (W_{input}, H_{input})
- Output Image Size: (W_{output}, H_{output})
- Filter Size: (W_{filter}, H_{filter})
- Padding: P
- Stride: S

$$W_{output} = \frac{W_{input} + 2P - W_{filter}}{S} + 1$$

$$H_{output} = \frac{H_{input} + 2P - H_{filter}}{S} + 1$$

Convolution Examples

original image

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

simple box blur

0	0	0	5	0	0	0
0	5	18	32	18	5	0
0	18	64	100	64	18	0
5	32	100	100	100	32	5
0	18	64	100	64	18	0
0	5	18	32	18	5	0
0	0	0	5	0	0	0

Gaussian blur

-1	-1	-1
2	2	2
-1	-1	-1

Horizontal lines

-1	2	-1
-1	2	-1
-1	2	-1

Vertical lines

-1	-1	2
-1	2	-1
2	-1	-1

45 degree lines

2	-1	-1
-1	2	-1
-1	-1	2

135 degree lines

line detection

-1	-1	-1
-1	8	-1
-1	-1	-1

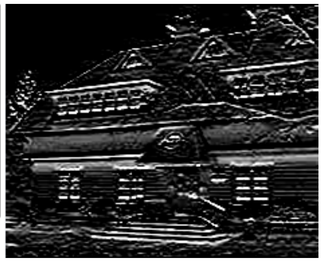
edge detection

-1	-2	-1
0	0	0
1	2	1

Horizontal

-1	0	1
-2	0	2
-1	0	1

Vertical



Sobel edge detection

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

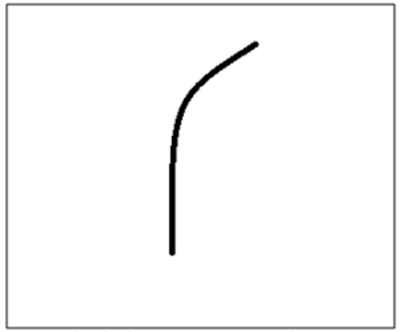
Laplacian edge detection

http://aishack.in/tutorials/image-convolution-examples/

왜 Convolution인가?

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

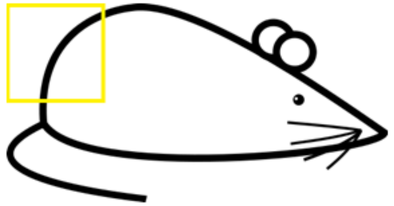
Pixel representation of filter



Visualization of a curve detector filter

kernel

Original image



Visualization of the filter on the image

Visualization of the receptive field

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	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

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

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	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

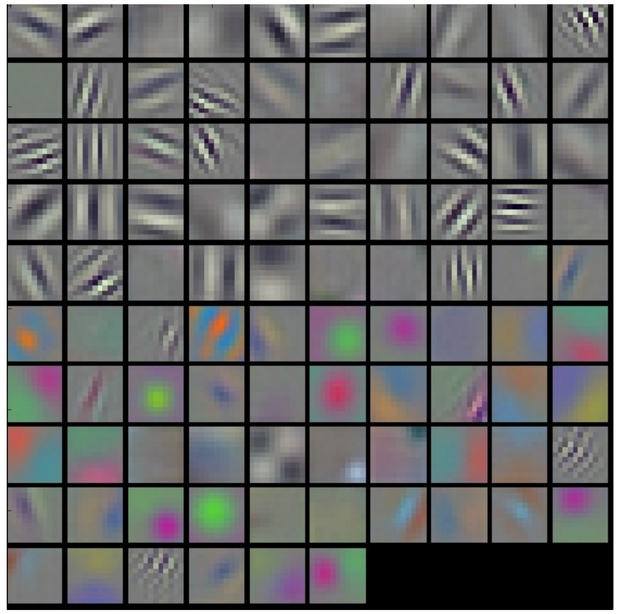
Pixel representation of filter

Multiplication and Summation = 0

https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

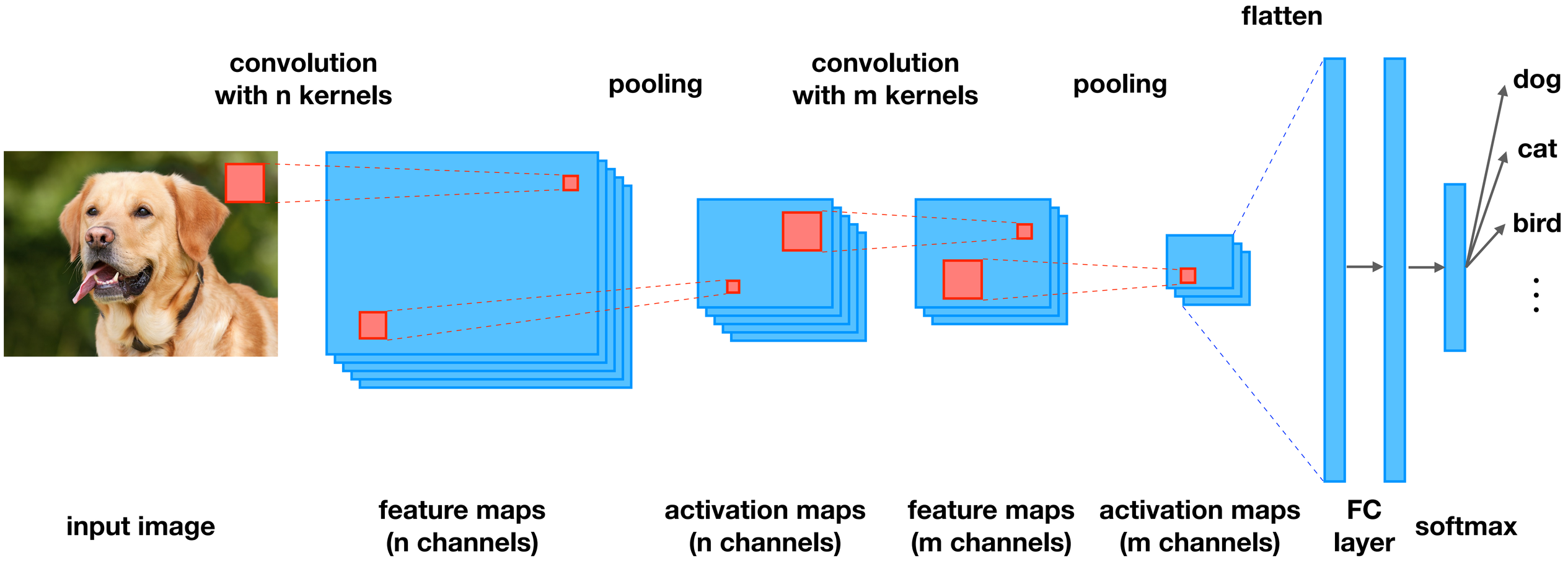
Looking Filters (Kernels)

- CNN은 이미지의 **특징(feature)**를 추출하기 위해 **filter** (또는 **kernel**)을 사용한다.
- CNN이 결국 어떠한 필터를 사용하는가는 CNN을 통해 학습된 결과를 살펴보면 알 수 있다.
- 다음은 AlexNet에서 CNN으로 학습한 결과로 얻어진 첫 번째 레이어의 필터 이미지들이다.
- 많은 필터들이 **edge detector** 역할을 하는 것을 확인할 수 있다.



https://umbum.tistory.com/223

CNN Example


convolution
with n kernels
pooling
convolution
with m kernels
pooling
flatten
dog
cat
bird
input image
feature maps
(n channels)
activation maps
(n channels)
feature maps
(m channels)
activation maps
(m channels)
FC
layer
softmax

Google's Inception Architecture

convolution layer

pooling layer

softmax

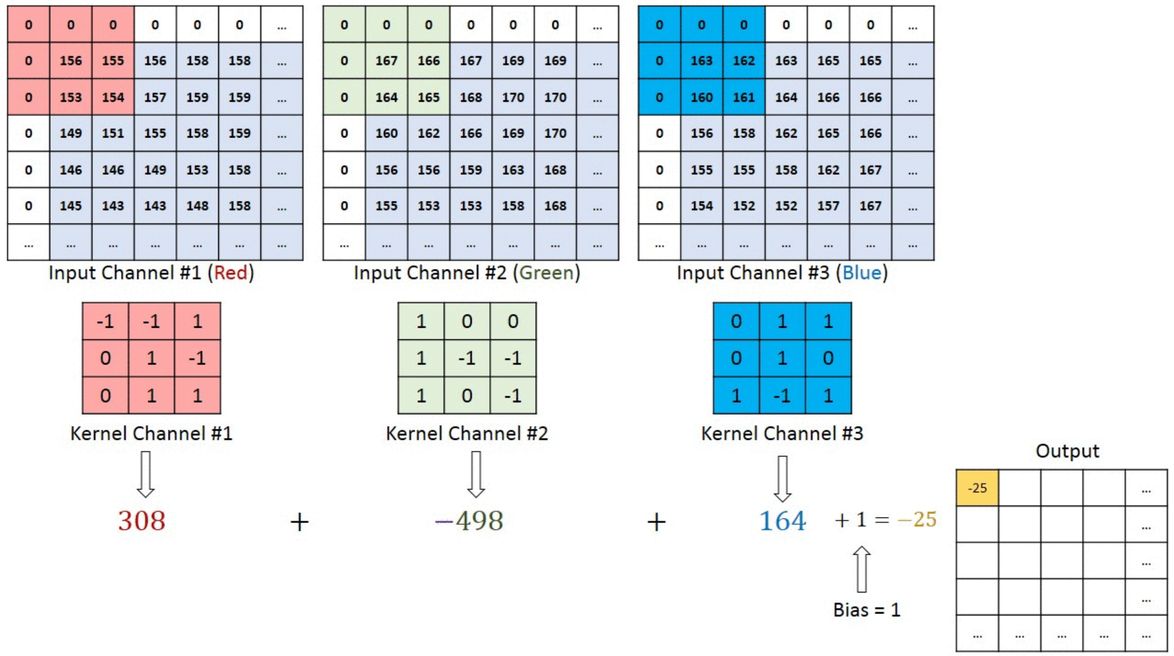
https://indico.io/exploring-computer-vision-transfer-learning/

CNN 용어

- **convolution layer**: kenrel을 순회시키면서 convolution을 계산하여 feature를 추출하는 layer.
  보통 서로 다른 여러 개의 kernel을 사용함. Output의 개수 = 사용한 kernel의 수.
- **kernel**: convolution 계산시 사용하는 행렬. CNN에서의 학습 대상. 필터(filter)라고도 부름.
- **feature map**: input data에 convolution을 적용한 결과.
- **activation map**: feature map에 ReLU와 같은 activation function을 적용한 결과, 즉 convolution layer의 최종 output.
- **channel**: convolution layer에서 각 kernel별로 만들어 지는 output. 즉, 사용한 kernel의 수 = channel의 개수.
- **padding**: convolution 적용시 output의 크기가 input과 같아지도록 input의 가장자리에 0과 같은 값을 채워 넣는 pixel 층의 두께.
- **stride**: convolution 계산시 kernel을 이동시키는 양. 즉, 다음 convolution 계산을 위해 kernel이 건너 뛰는 pixel의 개수.
- **pooling layer**: convolution layer의 출력 이미지를 입력으로 받아서 이보다 크기가 작은 출력 이미지를 생성함으로써 특정 데이터를 강조하는 역할을 함. Max, min, average 방식이 있음.

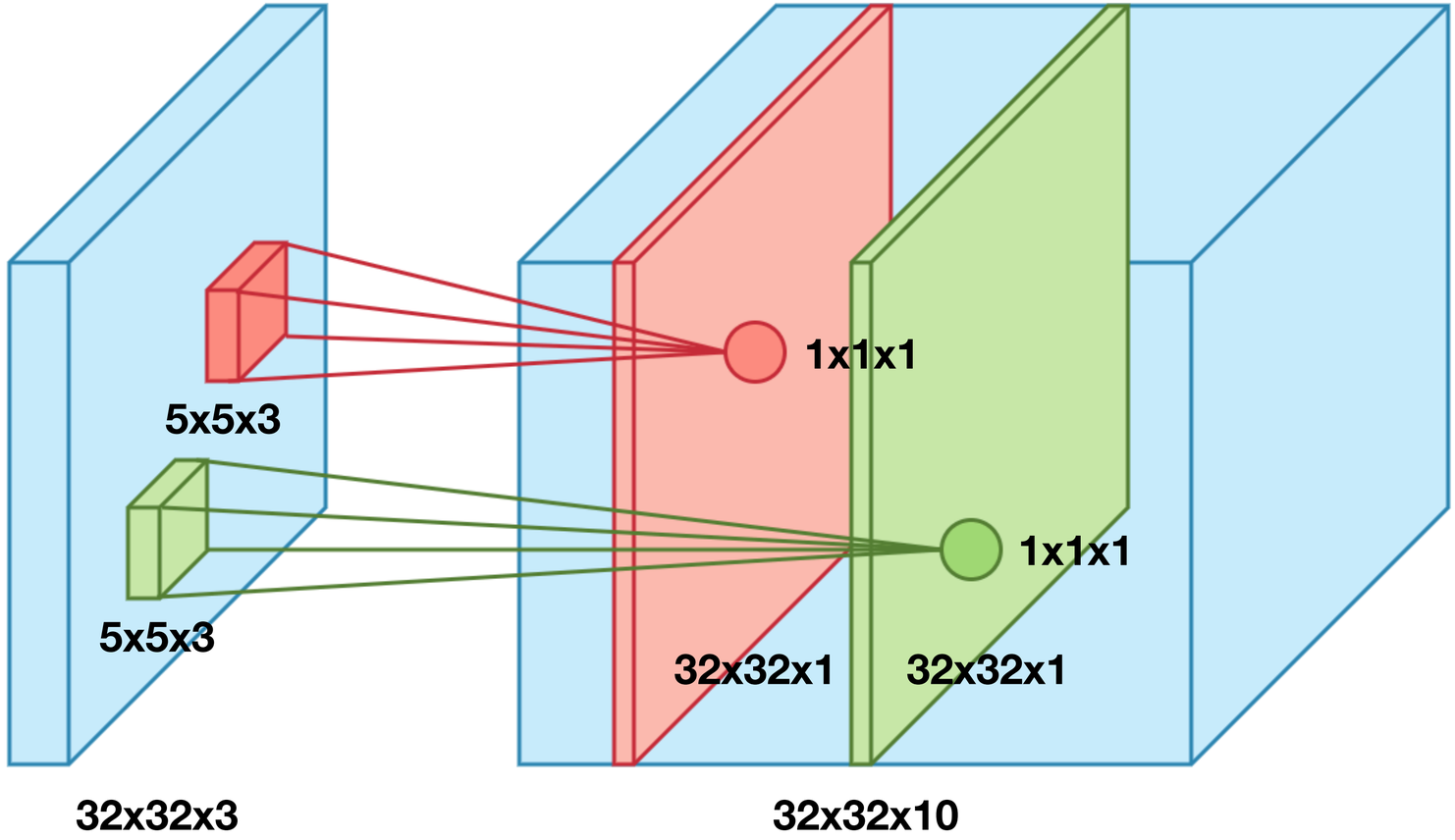
Multi-channel Input Convolution

- 입력 이미지가 RGB 등 여러 채널을 갖는 경우
  각 채널을 순회하며 합성곱을 계산하여 각 채널별로 피처맵을 만들고,
  모든 피처맵을 합하여 최종 피처맵을 만듬
- 따라서, kernel size = (kernel width, kernel height, **kernel depth = # of channels**)



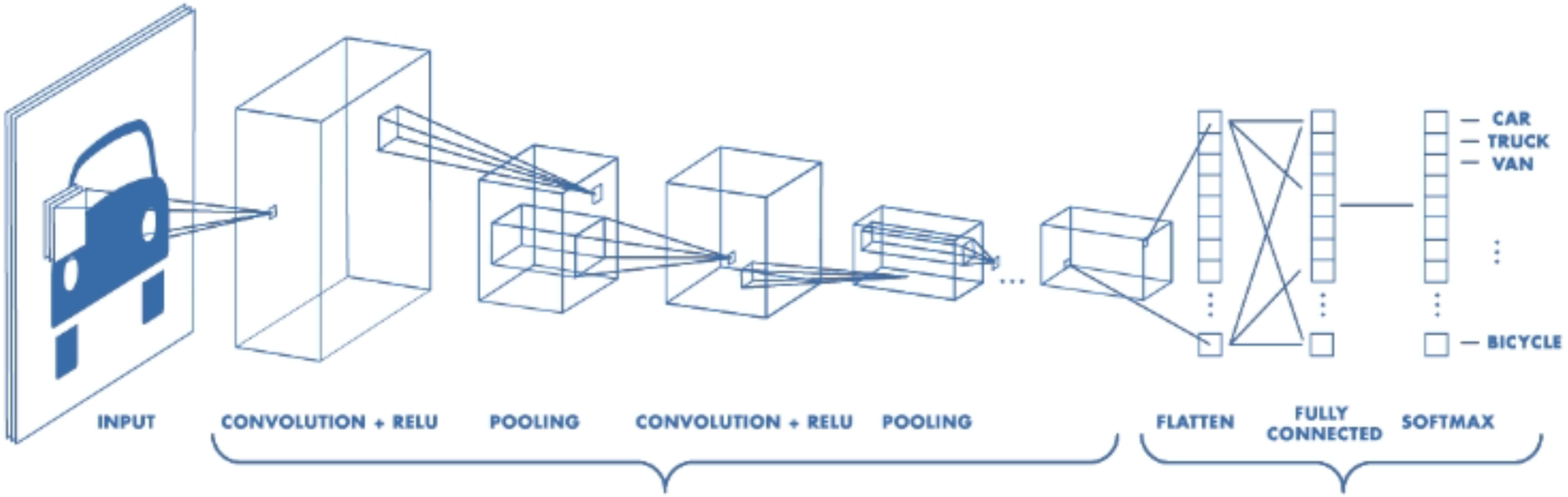
http://machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html

Multi-channel Input Convolution



https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050

Multi-channel Input Convolution



Feature Extraction **Classification**

https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050

Pooling

3	7	8	4
4	2	2	6
3	1	6	1
7	5	8	9

7	8
7	9

max pooling

2	2
1	1

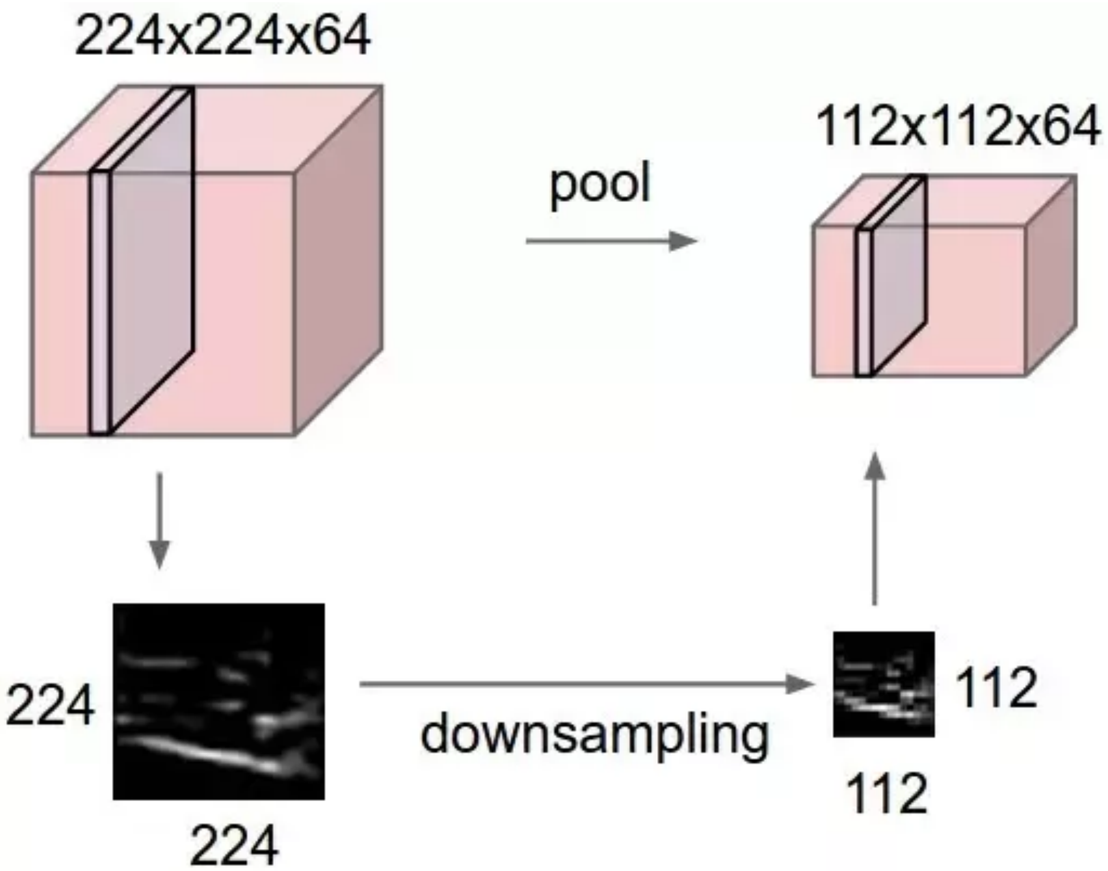
min pooling

4	5
4	6

avg pooling

Pooling for Multiple Channels

- Pooling 전후로 depth는 변하지 않는다.



https://www.quora.com/What-is-max-pooling-in-convolutional-neural-networks

CNN for MNIST


```
import tensorflow as tf
tf.logging.set_verbosity(tf.logging.ERROR)
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

X = tf.placeholder(tf.float32, [None, 28, 28, 1]) # [count, height, width, channels]
Y = tf.placeholder(tf.float32, [None, 10])        # [count, outputs]

# the 1st convolution layer
W1 = tf.Variable(tf.random_normal([3,3,1,32], stddev=0.1)) # kernel size: 3x3x1, kernel count: 32
L1 = tf.nn.conv2d(X, W1, strides=[1,1,1,1], padding="SAME") # [1,stride,stride,1], SAME: input size = output
size
L1 = tf.nn.relu(L1) # add non-linearity not to vanish gradient
L1 = tf.nn.max_pool(L1, ksize=[1,2,2,1], strides=[1,2,2,1], padding="SAME") # kernel size: 2x2

# the 2nd convolution layer
W2 = tf.Variable(tf.random_normal([3,3,32,64], stddev=0.1)) # kernel size: 3x3x32, kernel count: 64
L2 = tf.nn.conv2d(L1, W2, strides=[1,1,1,1], padding="SAME") # [1,stride,stride,1], SAME: input size = output
size
L2 = tf.nn.relu(L2) # add non-linearity not to vanish gradient
L2 = tf.nn.max_pool(L2, ksize=[1,2,2,1], strides=[1,2,2,1], padding="SAME") # kernel size: 2x2

# fully connected layer
W3 = tf.Variable(tf.random_normal([7*7*64,256], stddev=0.1)) # 28x28 -> 14x14 -> 7x7 through two pooling
layers
L3 = tf.reshape(L2, [-1,7*7*64]) # shape: (256, 3136) (here, 3136=7x7x64)
L3 = tf.matmul(L3, W3)
L3 = tf.nn.relu(L3)
```

CNN for MNIST


```
# output layer
W4 = tf.Variable(tf.random_normal([256,10], stddev=0.1))
L4 = tf.matmul(L3, W4)

model = L4
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits=model, labels=Y))
optimizer = tf.train.AdamOptimizer(0.01).minimize(cost)

batch_size = 1000 # the size of a batch
num_batches = int(mnist.train.num_examples / batch_size) # how many batches?

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

# training
for iEpoch in range(15): # reapeating training 15 times
    for iBatch in range(num_batches):
        x_data, y_data = mnist.train.next_batch(batch_size)
        x_data = x_data.reshape(-1,28,28,1) # reshape 1D array to (28,28,1)
        sess.run([optimizer, cost], feed_dict={X: x_data, Y: y_data})

# testing
comparison = tf.equal(tf.argmax(model, 1), tf.argmax(Y, 1))
accuracy = tf.reduce_mean(tf.cast(comparison, tf.float32))
print("Accuracy: ", sess.run(accuracy, feed_dict={X: mnist.test.images.reshape(-1,28,28,1), Y:
mnist.test.labels}))

sess.close()
```

CNN for CIFAR-10


```
import numpy as np
import tensorflow as tf
from tensorflow import keras
import tensorflow.keras.datasets.cifar10 as cifar10
tf.logging.set_verbosity(tf.logging.ERROR)

(x_train, y_train), (x_test, y_test) = cifar10.load_data()

y_train = keras.utils.to_categorical(y_train, num_classes=10, dtype='float32')
y_test  = keras.utils.to_categorical(y_test, num_classes=10, dtype='float32')

def next_batch(images, labels, batch_size):
  indices = np.arange(0 , len(images))
  np.random.shuffle(indices)
  indices = indices[:batch_size]
  images = [images[i] for i in indices]
  labels = [labels[i] for i in indices]
  return images, labels

X = tf.placeholder(tf.float32, shape=[None,32,32,3])
Y = tf.placeholder(tf.float32, shape=[None,10])
```

CNN for CIFAR-10

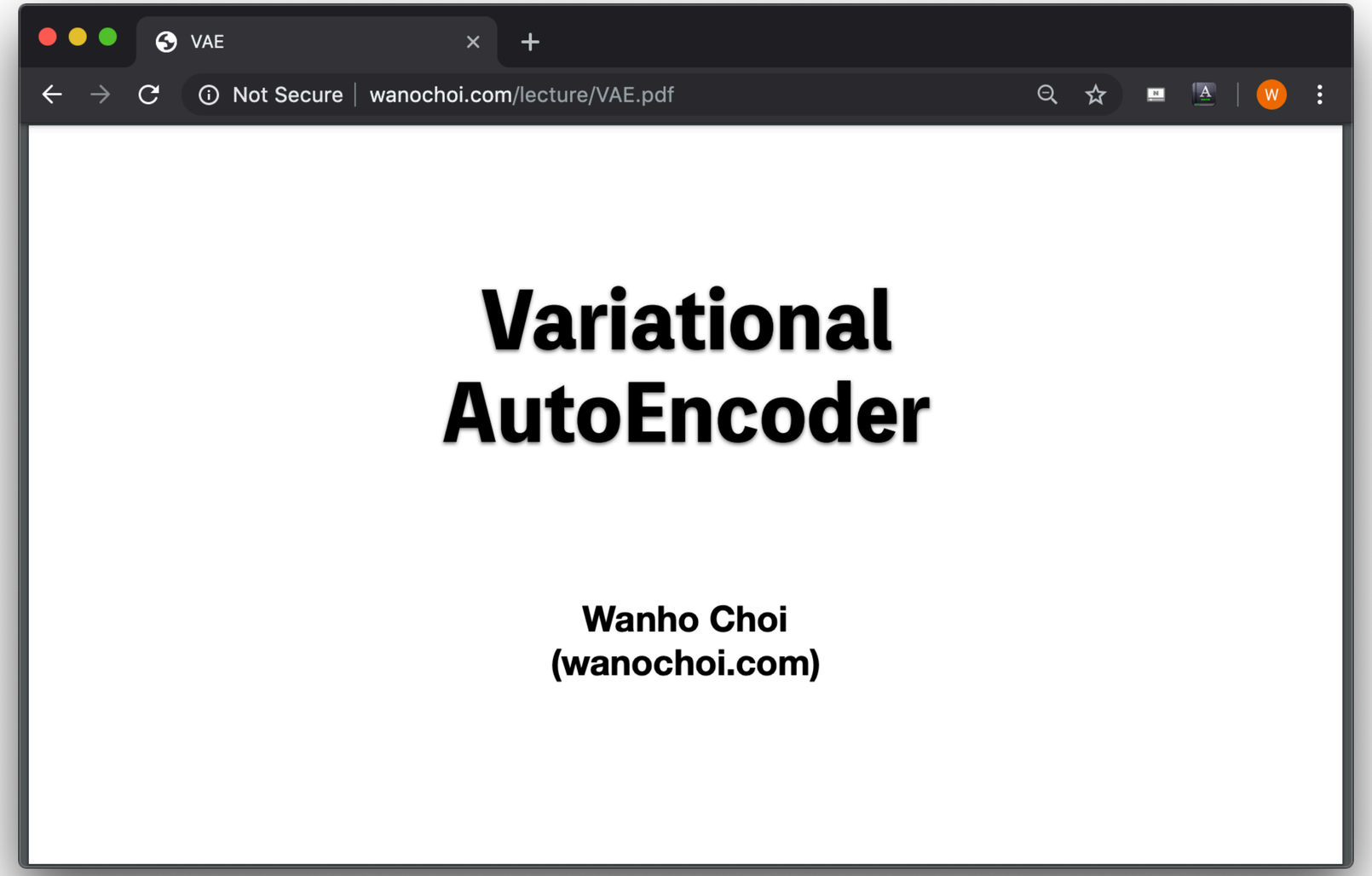

```
W1 = tf.Variable(tf.random_normal(shape=[5,5,3,64], stddev=0.001))
L1 = tf.nn.relu(tf.nn.conv2d(X, W1, strides=[1,1,1,1], padding='SAME'))
L1 = tf.nn.max_pool(L1, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

W2 = tf.Variable(tf.random_normal(shape=[5,5,64,32], stddev=0.001))
L2 = tf.nn.relu(tf.nn.conv2d(L1, W2, strides=[1,1,1,1], padding='SAME'))
L2 = tf.nn.max_pool(L2, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

W3 = tf.Variable(tf.random_normal(shape=[3,3,32,128], stddev=0.001))
L3 = tf.nn.relu(tf.nn.conv2d(L2, W3, strides=[1,1,1,1], padding='SAME'))
L3 = tf.reshape(L3, [-1,8*8*128]) # flatten to feed into FCNN

W4 = tf.Variable(tf.random_normal(shape=[8*8*128,256], stddev=0.001))
L4 = tf.nn.relu(tf.matmul(L3, W4))

W5 = tf.Variable(tf.random_normal(shape=[256,10], stddev=0.001))
L5 = tf.matmul(L4, W5)

model = L5
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=model, labels=Y))
optimizer = tf.train.AdamOptimizer(0.001).minimize(cost)
```

CNN for CIFAR-10


```
comparison = tf.equal(tf.argmax(model, 1), tf.argmax(Y, 1))
accuracy = tf.reduce_mean(tf.cast(comparison, tf.float32))

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

for iEpoch in range(2000):
    x_data, y_data = next_batch(x_train, y_train, 1000)
    sess.run(optimizer, feed_dict={X: x_data, Y: y_data})
    train_accuracy = sess.run(accuracy, feed_dict={X: x_data, Y: y_data})
    print('Epoch:', iEpoch, ', Accuracy for Training Data:', train_accuracy)

x_data, y_data = next_batch(x_test, y_test, 1000)
test_accuracy = sess.run(accuracy, feed_dict={X: x_data, Y: y_data})
print('Accuracy for Test Data:', test_accuracy)

sess.close()
```

Autoencoder

Please refer to the following.



```
import tensor flow as tf
import numpy as np
import matplotlib.pyplot as plt

tf.logging.set_verbosity(tf.logging.ERROR)
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

X = tf.placeholder(tf.float32, [None, 784]) # 784=28x28
# We don't need Y.

# encoder
W1 = tf.Variable(tf.random_normal([784, 256]))
b1 = tf.Variable(tf.random_normal([256]))
L1 = tf.nn.sigmoid(tf.add(tf.matmul(X, W1), b1))

W2 = tf.Variable(tf.random_normal([256, 32]))
b2 = tf.Variable(tf.random_normal([32]))
L2 = tf.nn.sigmoid(tf.add(tf.matmul(L1, W2), b2))

# decoder
W3 = tf.Variable(tf.random_normal([32, 256]))
b3 = tf.Variable(tf.random_normal([256]))
L3 = tf.nn.sigmoid(tf.add(tf.matmul(L2, W3), b3))

W4 = tf.Variable(tf.random_normal([256, 784]))
b4 = tf.Variable(tf.random_normal([784]))
L4 = tf.nn.sigmoid(tf.add(tf.matmul(L3, W4), b4))
```

```
model = L4
cost = tf.reduce_mean(tf.pow(X - model, 2)) # Y = X
optimizer = tf.train.RMSPropOptimizer(learning_rate=0.01).minimize(cost)

batch_size = 1000 # the size of a batch
num_batches = int(mnist.train.num_examples / batch_size) # how many batches?

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

# training
for iEpoch in range(1000): # reapeating training 1000 times
    for iBatch in range(num_batches):
        x_data, y_data = mnist.train.next_batch(batch_size)
        sess.run([optimizer], feed_dict={X: x_data})

# visualization
sample_size = 10
samples = sess.run(model, feed_dict={X: mnist.test.images[:sample_size]})
print(samples.shape) # (10, 784)
fig, ax = plt.subplots(2, sample_size, figsize=(sample_size, 2))

for i in range(sample_size):
    ax[0][i].set_axis_off()
    ax[0][i].imshow(np.reshape(mnist.test.images[i], (28, 28)))
    ax[1][i].set_axis_off()
    ax[1][i].imshow(np.reshape(samples[i], (28, 28)))

plt.show()
sess.close()
```

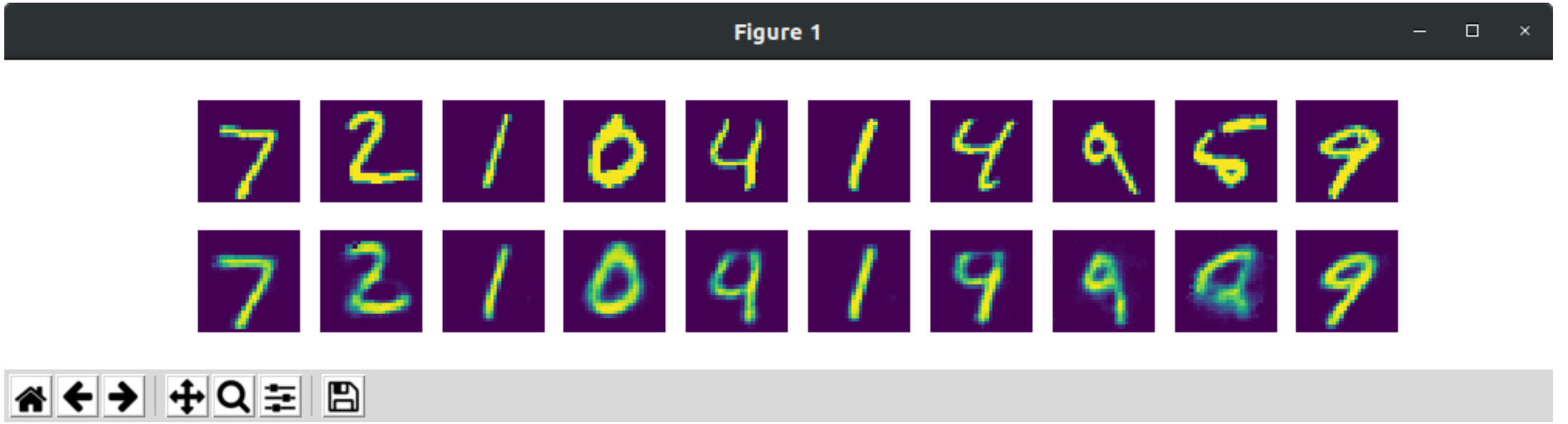
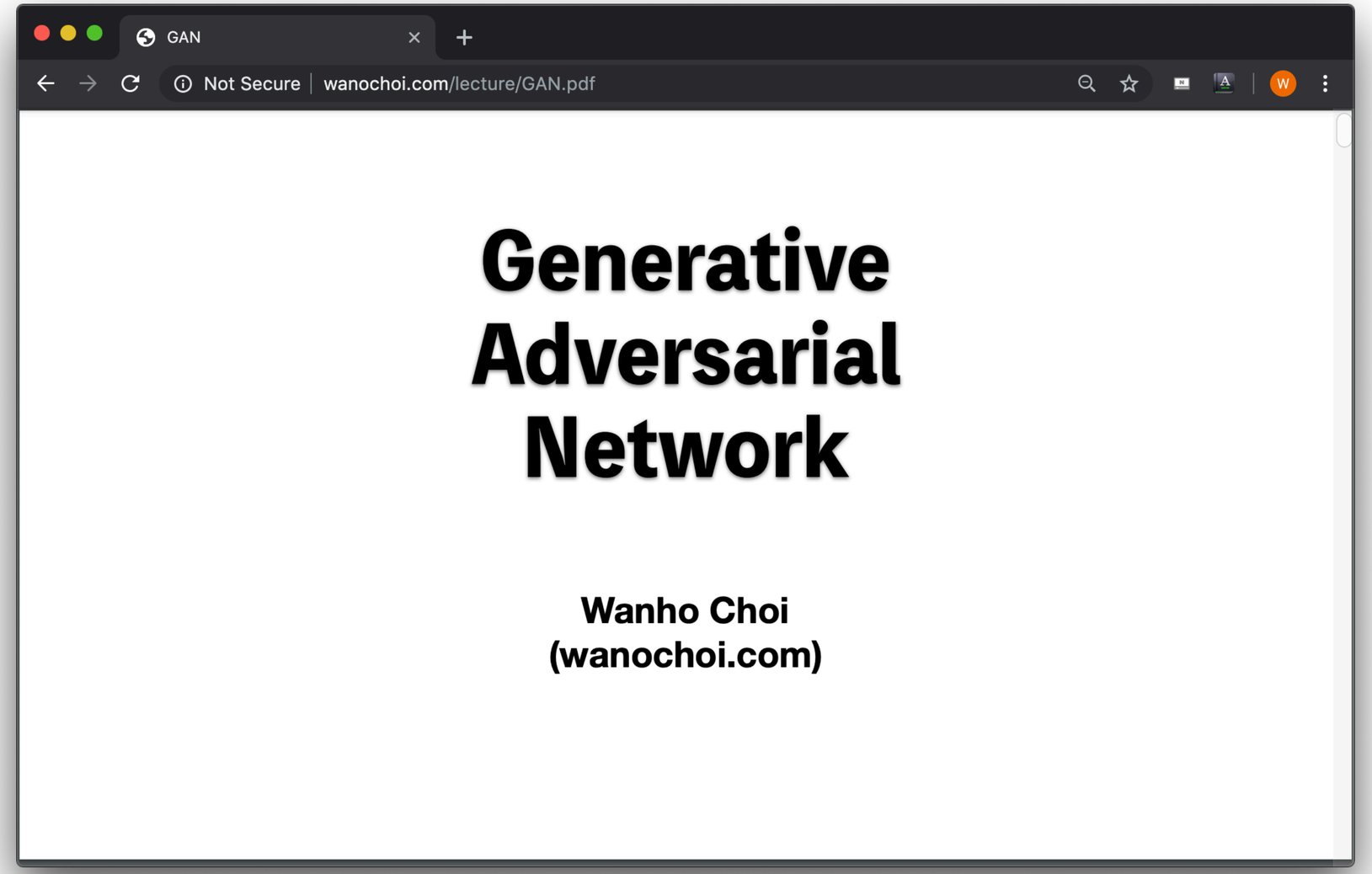


Figure 1

GAN

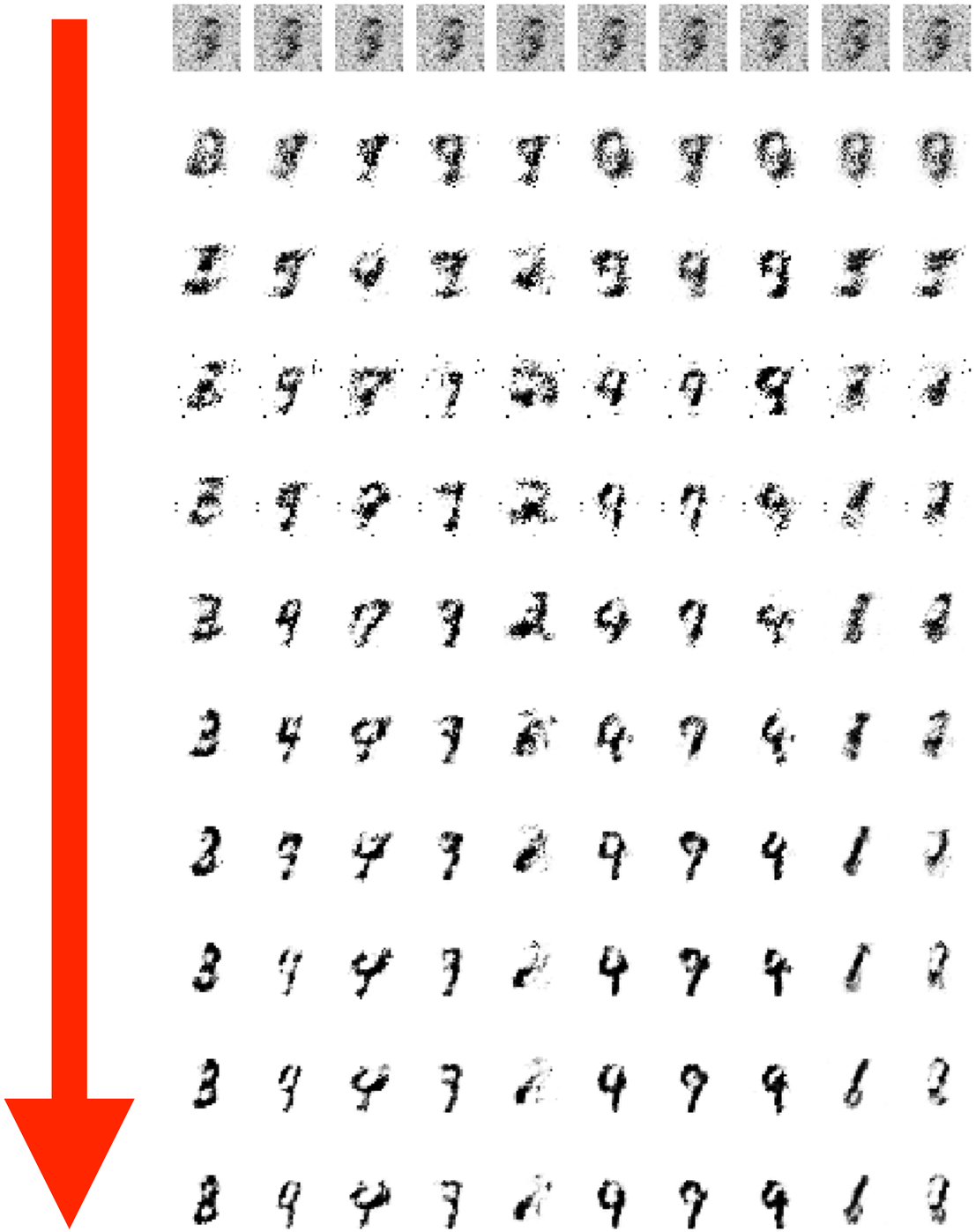
Please refer to the following.



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

tf.logging.set_verbosity(tf.logging.ERROR)
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

n_input  = 28*28 # MNIST image size
n_hidden = 256   # hidden layer node count
n_noise  = 128   # noise dimension

X = tf.placeholder(tf.float32, [None, n_input])
Z = tf.placeholder(tf.float32, [None, n_noise])

# G-Network
G_W1 = tf.Variable(tf.random_normal([n_noise, n_hidden], stddev=0.01))
G_b1 = tf.Variable(tf.zeros([n_hidden]))
G_W2 = tf.Variable(tf.random_normal([n_hidden, n_input], stddev=0.01))
G_b2 = tf.Variable(tf.zeros([n_input]))

def generator(z): # 128 -> 256 -> 28*28
    L1 = tf.nn.relu(tf.matmul(z, G_W1) + G_b1)
    L2 = tf.nn.sigmoid(tf.matmul(L1, G_W2) + G_b2)
    return L2

# D-Network
D_W1 = tf.Variable(tf.random_normal([n_input, n_hidden], stddev=0.01))
D_b1 = tf.Variable(tf.zeros([n_hidden]))
D_W2 = tf.Variable(tf.random_normal([n_hidden, 1], stddev=0.01))
D_b2 = tf.Variable(tf.zeros([1]))
```

```
def discriminator(inputs): # 28*28 -> 256 -> 1
    L1 = tf.nn.relu(tf.matmul(inputs, D_W1) + D_b1)
    L2 = tf.nn.sigmoid(tf.matmul(L1, D_W2) + D_b2)
    return L2

def noise_gen(num_images, noise_dimension):
    return np.random.normal(size=(num_images, noise_dimension))

G = generator(Z)

D_fake = discriminator(G)
D_real = discriminator(X)

D_cost = -tf.reduce_mean(tf.log(D_real) + tf.log(1-D_fake))
G_cost = -tf.reduce_mean(tf.log(D_fake))

# var_list: optimization only for the specified variables
D_opt = tf.train.AdamOptimizer(learning_rate=0.0001).minimize(D_cost, var_list=[D_W1, D_b1, D_W2, D_b2])
G_opt = tf.train.AdamOptimizer(learning_rate=0.0001).minimize(G_cost, var_list=[G_W1, G_b1, G_W2, G_b2])

batch_size = 1000 # the size of a batch
num_batches = int(mnist.train.num_examples / batch_size) # how many batches?
n_sample = 10 # sample size

z_test = noise_gen(n_sample, n_noise)
```

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

for iEpoch in range(1000): # reapeating training 1000 times

    # training
    for iBatch in range(num_batches):
        x_data, _ = mnist.train.next_batch(batch_size)
        z_data = noise_gen(batch_size, n_noise)

        sess.run(D_opt, feed_dict={X: x_data, Z: z_data})
        sess.run(G_opt, feed_dict={Z: z_data})

    # image generation & save
    if iEpoch==0 or (iEpoch+1)%100==0: # 100 = saving period
        samples = sess.run(G, feed_dict={Z: z_test})

        fig, ax = plt.subplots(1, n_sample, figsize=(n_sample, 1))

        for i in range(n_sample):
            ax[i].set_axis_off()
            ax[i].imshow(np.reshape(samples[i], (28,28)), cmap=plt.cm.gray_r)

        file_path_name = ‘samples/{}.png'.format(str(iEpoch).zfill(3))
        print(file_path_name)

        plt.savefig(file_path_name, bbox_inches=‘tight')
        plt.close(fig)

sess.close()
```



000.png

099.png

199.png

299.png

399.png

499.png

599.png

699.png

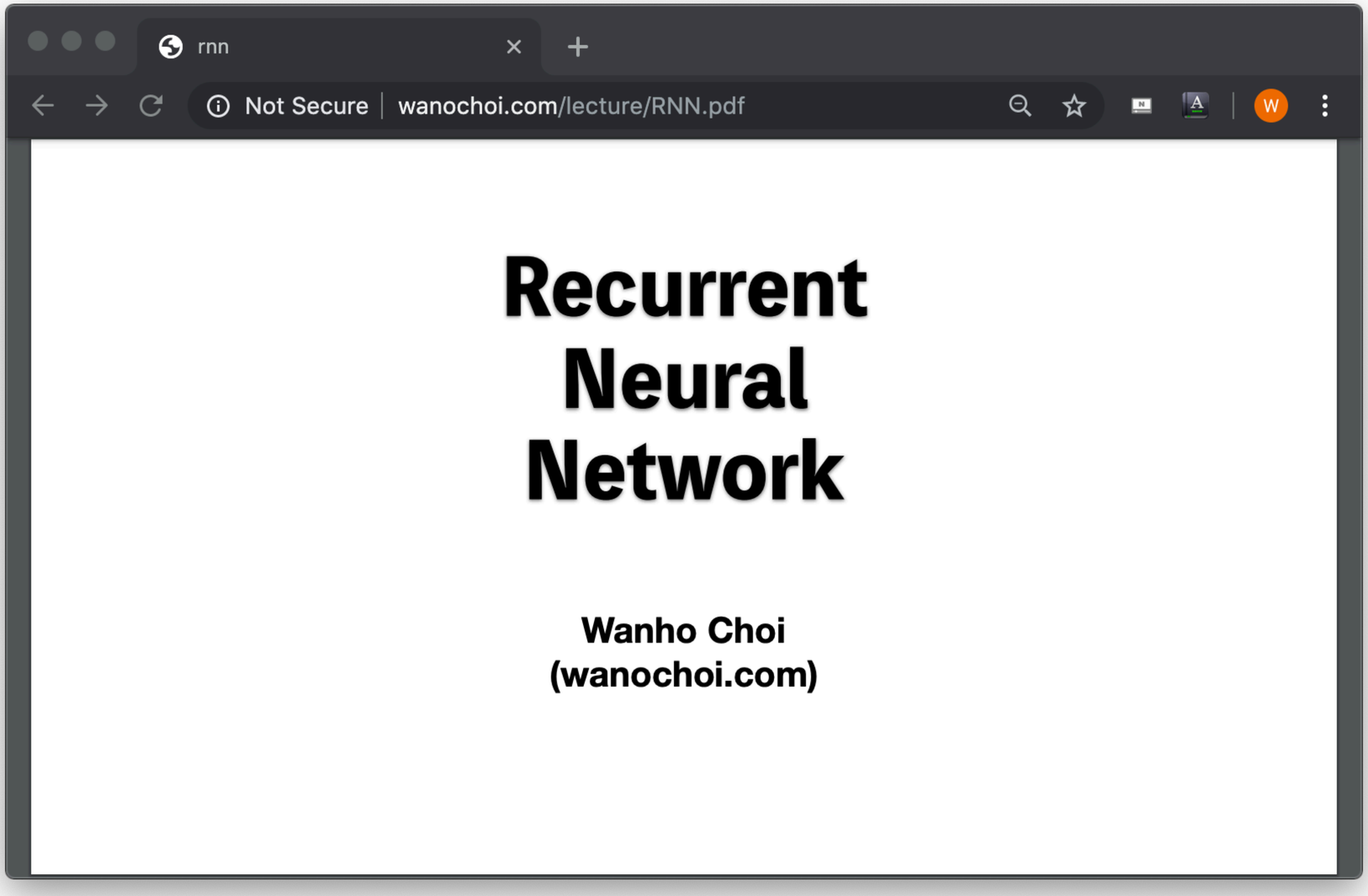
799.png

899.png

999.png

RNN

Please refer to the following.

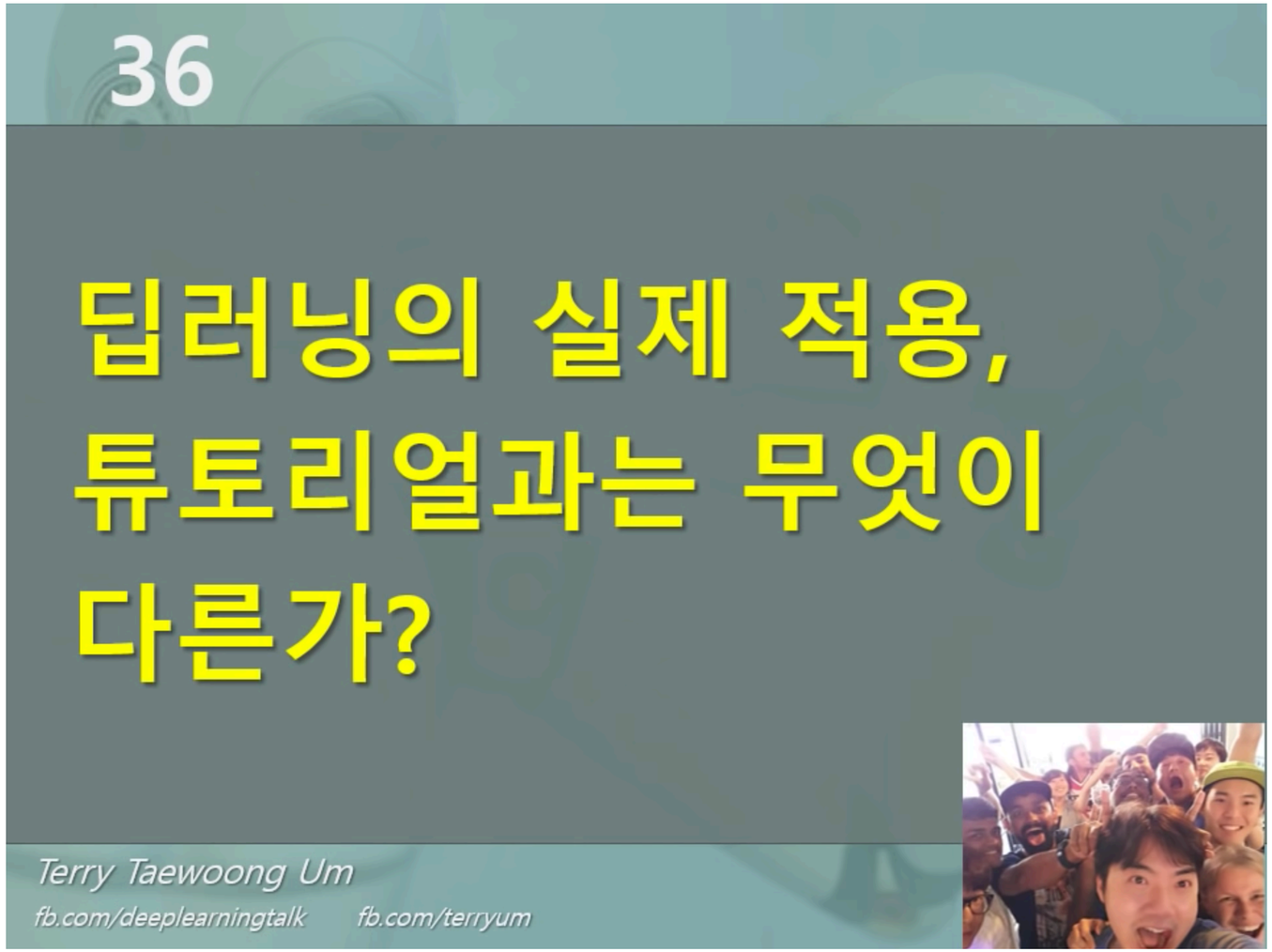


DQN

TensorBoard

Supplement

실제 적용, 튜토리얼과는 무엇이 다른가?



https://www.youtube.com/watch?v=daTqajvdeTw

머신 러닝 엔지니어, 이제는 곧 멸종될 타이틀

얼마 전 눈이 번쩍 뜨이는 경험을 했다. 회사의 프론트 엔드를 담당하고 있는 주니어 엔지니어가, 최근 관심 갖기 시작한 머신 러닝에 대해 좀 더 배우고자 해커톤에 참여하기로 결정한 것이다. 그 엔지니어는 '인공지능을 다시 따분한 기술로 만들겠다'는 모토로 운영되는 fast.ai 온라인 강좌 등을 통해 인공지능을 빠르게 익혀가더니, 금방 텐서플로(TensorFlow) 모델을 사용하기 시작했다.

물론 처음부터 신기한 결과물을 낸 건 아니다. 처음에는 인물 사진에 수염을 칠해주는 인공지능을 만들고 좋아했다. 그런데 그게 며칠 지나니까 생산 시스템에 도입할 수 있을만한 의미 있는 결과물이 나오기 시작했다. 몇 주가 지나니 그 엔지니어가 만들어 시험해 본 것이 사업 운영에 유의미한 차이를 만드는 걸 모두가 볼 수 있었다.

필자는 대학에서 머신 러닝을 전공한 사람으로, 실제 IT 분야에서의 경력도 바로 인공지능에서부터 쌓기 시작했다. 초창기에 비해 지금은 머신 러닝을 사용하고 여러 분야에 적용하는 게 무척 쉽고 간단해졌다. 머신 러닝이라는 시각에서 지금 시대는 축복이 넘치는 때다. 그래서 저 주니어 엔지니어는 5일 만에 필자가 5년 동안 이룩한 걸 이뤄낼 수 있을 것으로 예상된다. 그만큼 툴과 교육 재료가 좋아진 것이다.

그래서 이제는 머신 러닝이나 인공지능과 관련이 있는 대학 졸업증이 있어야만 인공지능을 사용할 수 있는 건 아니게 되었다. 기초적인 머신 러닝 응용 툴들은 요즘 나오는 기초 개발자 툴박스에 들어 있는 경우가 많다. 90년대만 해도 신경망을 실험해보고 싶은 개발자라면, 가장 간단한 콘셉트를 개발한 후 점진적으로 발전시키는 방향을 택했다. 매 단계 발전할 때마다 인공지능의 근간이 되는 수학과 여러 가지 이론들을 충분히 이해해야만 따라갈 수 있었다.

하지만 지금은 어떤가? 인공지능 초보라고 하더라도 구글 클라우드의 AutoML을 사용해 인공지능 모델을 개발하고, 그로 인해 흥미로운 결과물을 '자동으로' 만들어낼 수 있게 되었다. 복잡하고 어려운 부분은 추상화로 해결이 되고 있고, 추상화를 해결하기 위한 도구들은 갈수록 강력해지고 있다. 어셈블리에 대해 배우고 싶어 하는 코더를 만난 지가 언제인지 우리는 기억도 나지 않는다.

현대의 개발자들이 인공지능 배경에 깔린 수학적 모델을 깊이 있게 이야기할 수 있는 건 아니다. 그렇지만 그렇게 할 수 있었던 과거의 선배들보다 더 가시적이고 실질적인 결과를 내고 있다. fast.ai의 창립자인 제레미 하워드(Jeremy Howard)는 최근 트윗을 통해 "한 번도 인공지능과 관련된 정규 교육을 받아본 적이 없다. 왜 굳이 강연장에 가야만 하는 걸까? 처음부터 그런 교육 과정은 시간 낭비일 뿐이라고 생각했다"고 말했다.

이는 사실 놀라운 말이다. 여태까지 조직들이 엔지니어들을 어떤 식으로 고용해왔는지 생각해보라. 특히 인공지능 분야의 전문가를 뽑을 때 말이다. 100이면 100, 대학 졸업장을 요구할 것이다. 컴퓨터 과학이나 수학, 프로그래밍 등의 학위를 소지한 사람들을 찾기 위해서다. 그게 틀렸다는 건 아니지만, '그런 배경이 없는 사람도 엔지니어로서의 결과를 낼 수 있는 때'라면 다른 방향에서 구인 구직에 접근할 수 있지 않을까?

고객의 필요를 충족시켜 주고, 조직의 사업 행위를 IT 기술로 뒷받침할 수 있다면, 굳이 수학적 배경과 원리, 학문적 역사를 줄줄이 꿰고 있지 않더라도 충분히 머신 러닝 엔지니어라고 불려도 될 것이다. 실제 필자가 가장 기분 좋게, 효율적으로 협업 했던 머신 러닝 엔지니어들은 대부분 '독학으로' 머신 러닝을 배운 사람들이었다.

심지어 5년 이하 경력자들도 많았다. 그래서 필자는 이렇게 인공지능과 머신 러닝에 대해 배우는 것이 용이한 시대에, 굳이 오래되고 학위가 좋은 개발자를 뽑을 필요가 없다는 결론을 내리기에 이르렀다. 마치 우리가 포토샵 전문 학위를 만들고, 포토샵 전문가를 엄선해서 디자인 작업을 의뢰하지 않는 것과 같다. 이제 머신 러닝은 하나의 도구가 되어가고 있는 것이다. 누구나 조금만 익히면 포토샵으로 사진을 보정할 수 있는 것처럼, 머신 러닝도 어지간한 코더들이라면 웬만큼 쓸 수 있는 툴이 될 것이다.

오픈소스 신경망 라이브러리인 케라스(Keras)를 만든 프랑수와 숄레(Francois Chollet)는 "최고의 사람들은 90% 이상 독학한 사람들"이라고 표현한다. "좋은 대학을 나와서도 독학을 할 줄 모르는 사람은 결국 도태되더군요. 출신 대학이 무엇이고 교육적 배경이 무엇이든 결국 배울 줄 아는 사람이 가장 뛰어난 사람이었습니다."

물론 이런 생각이 지나치게 급진적이라고 보는 사람들도 있을 것이다. 난 그런 사람들을 지금 당장 설득할 생각으로 글을 쓰는 게 아니다. 그저 도구가 강력해짐에 따라 배우기가 쉬워지고, 따라서 굳이 가방 끈이 길지 않더라도 유의미한 결과를 내는 시대가 됐다는 걸 상기시켜 주고 싶었다. 그런 때는 스스로를 가르칠 수 있는 능력이 매우 중요해진다는 건 조금만 생각해도 알 수 있을 것이다.

머신 러닝이 붐이지만, 그 붐이 어떤 방향으로 나아가는지 검토해볼 필요가 있다. 필자가 보기에 '머신 러닝 엔지니어'나 '인공지능 전문가'는 곧 사라질 타이틀이다.

https://www.boannews.com/media/view.asp?idx=76810#

https://www.informationweek.com/strategic-cio/team-building-and-staffing/the-title-machine-learning-engineer-will-start-to-disappear/a/d-id/1333816

현 시점, 딥 러닝에 대해 꼭 알아야 할 것 9

- 딥 러닝은 복잡한 문제를 해결하는 데에만 적당하다.
- 딥 러닝은 특성 공학(feature engineering) 문제를 해결한다.
- 사람들은 ‘블랙박스’를 두려워 한다.
- 딥 러닝 기술은 전이 학습(transfer learning)이 가능하다.
- 딥 러닝은 로지스틱 회귀(logistic regression)에 뛰어나다.
- 딥 러닝 프로젝트 대부분 기울기 하강(gradient descent)과 관련이 있다.
- 딥 러닝은 다양한 형태로 활용되고 존재한다.
- 콘볼루션 신경망이 가장 보편적인 신경망이다.
- 딥 러닝은 다양한 부분에서 인간보다 뛰어난 면모를 보인다.

https://www.boannews.com/media/view.asp?idx=77276

Reference

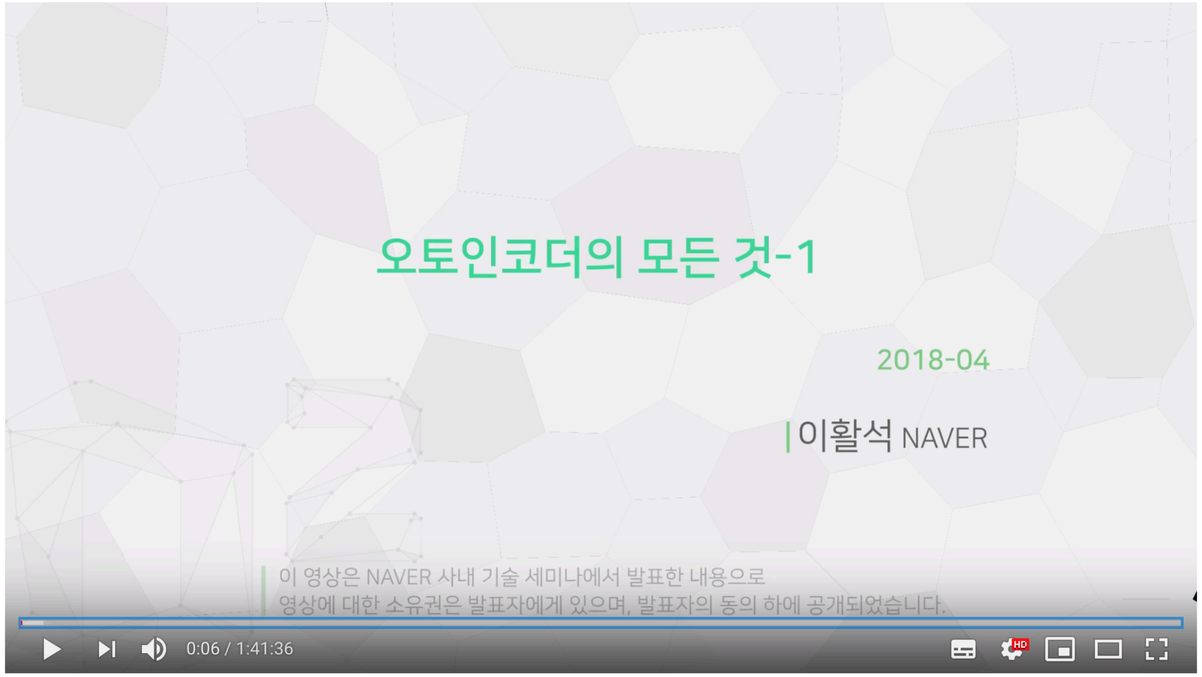
골빈해커의
3분 딥러닝
텐서플로맛
김진중 지음
텐서플로 코드로 맛보는
CNN, AE, GAN, RNN, DQN
(+ Inception)
한빛미디어

오토인코더의 모든 것 - 1/3 - YouTu | 오토인코더의 모든 것 - 2/3 - YouTu | 오토인코더의 모든 것 - 3/3 - YouTu

https://www.youtube.com/watch?v=o_peo6U7IRM

YouTube

검색



오토인코더의 모든 것 - 1/3

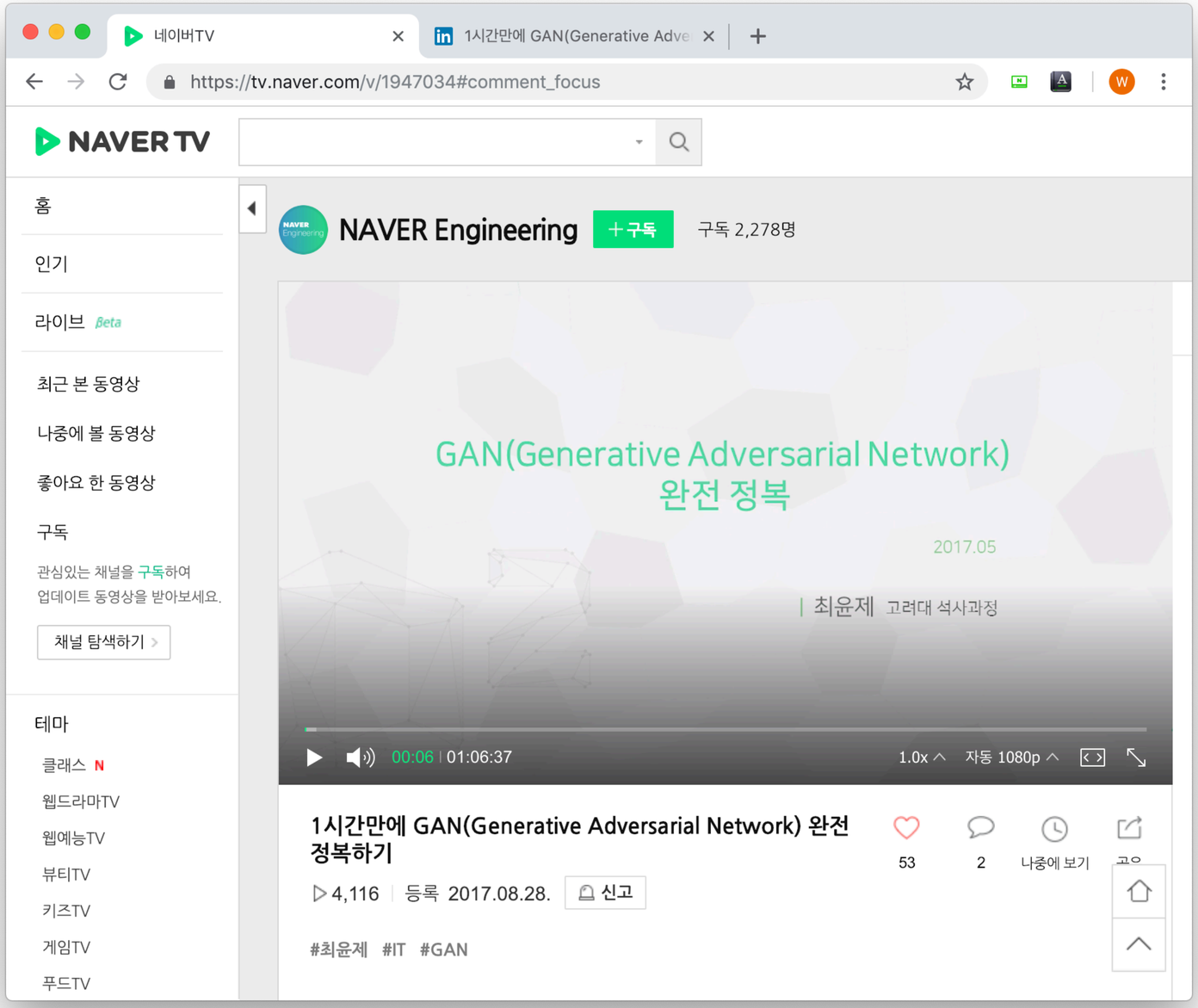
조회수 5,780회

101 0 공유 저장

naver d2
게시일: 2018. 5. 9.

구독 3천

발표자: 이활석(NAVER)
발표일: 2017.11.


네이버TV
1시간만에 GAN(Generative Adve
https://tv.naver.com/v/1947034#comment_focus
NAVER TV
홈
인기
라이브 βeta
최근 본 동영상
나중에 볼 동영상
좋아요 한 동영상
구독
관심있는 채널을 구독하여
업데이트 동영상을 받아보세요.
채널 탐색하기
테마
클래스 N
웹드라마TV
웹예능TV
뷰티TV
키즈TV
게임TV
푸드TV
NAVER Engineering
+구독
구독 2,278명
GAN(Generative Adversarial Network)
완전 정복
2017.05
| 최윤제 고려대 석사과정
00:06 | 01:06:37
1.0x
자동 1080p
1시간만에 GAN(Generative Adversarial Network) 완전 정복하기
53
2
나중에 보기
4,116
등록 2017.08.28.
신고
#최윤제 #IT #GAN

모두를 위한 머신러닝/딥러닝 강의

Not Secure | hunkim.github.io/ml/

모두를 위한 머신러닝/딥러닝 강의

모두를 위한 머신러닝과 딥러닝의 강의

알파고와 이세돌의 경기를 보면서 이제 머신 러닝이 인간이 잘 한다고 여겨진 직관과 의사 결정능력에서도 충분한 데이타가 있으면 어느정도 또는 우리보다 더 잘할수도 있다는 생각을 많이 하게 되었습니다. Andrew Ng 교수님이 말씀하신것 처럼 이런 시대에 머신 러닝을 잘 이해하고 잘 다룰수 있다면 그야말로 "Super Power"를 가지게 되는 것이 아닌가 생각합니다.

더 많은 분들이 머신 러닝과 딥러닝에 대해 더 이해하고 본인들의 문제를 이 멋진 도구를 이용해서 풀수 있게 하기위해 비디오 강의를 준비하였습니다. 더 나아가 이론에만 그치지 않고 최근 구글이 공개한 머신러닝을 위한 오픈소스인 TensorFlow를 이용해서 이론을 구현해 볼수 있도록 하였습니다.

수학이나 컴퓨터 공학적인 지식이 없이도 쉽게 볼수 있도록 만들려고 노력하였습니다.

NEW 시즌 RL - Deep Reinforcement Learning

비디오 리스트 (천천이 업데이트 예정입니다. 시즌 1 먼저 들으신 다음 들으시면 좋습니다.)

- Lecture 1: 수업의 개요 비디오 강의 슬라이드
- Lecture 2: OpenAI GYM 게임해보기 비디오 강의 슬라이드
  - Lab 2: OpenAI GYM 게임해보기 실습 비디오 실습슬라이드
- Lecture 3: Dummy Q-learning (table) 비디오 강의 슬라이드

Q & A