Tensorflow day 1

모두의 연구소 Lab. Rubato
소준섭

소준섭

- 모두의연구소
- Lab. Rubato
- Research interests
  - Music Generative Models
  - Music Synthesis

설문조사

Windows

vs

Linux

설문조사

Programming?

설문조사

SW Engineer?

설문조사

Machine Learning?

Deep Learning?

설문조사

Deep Learning theory?

Artificial Intelligence in Real World

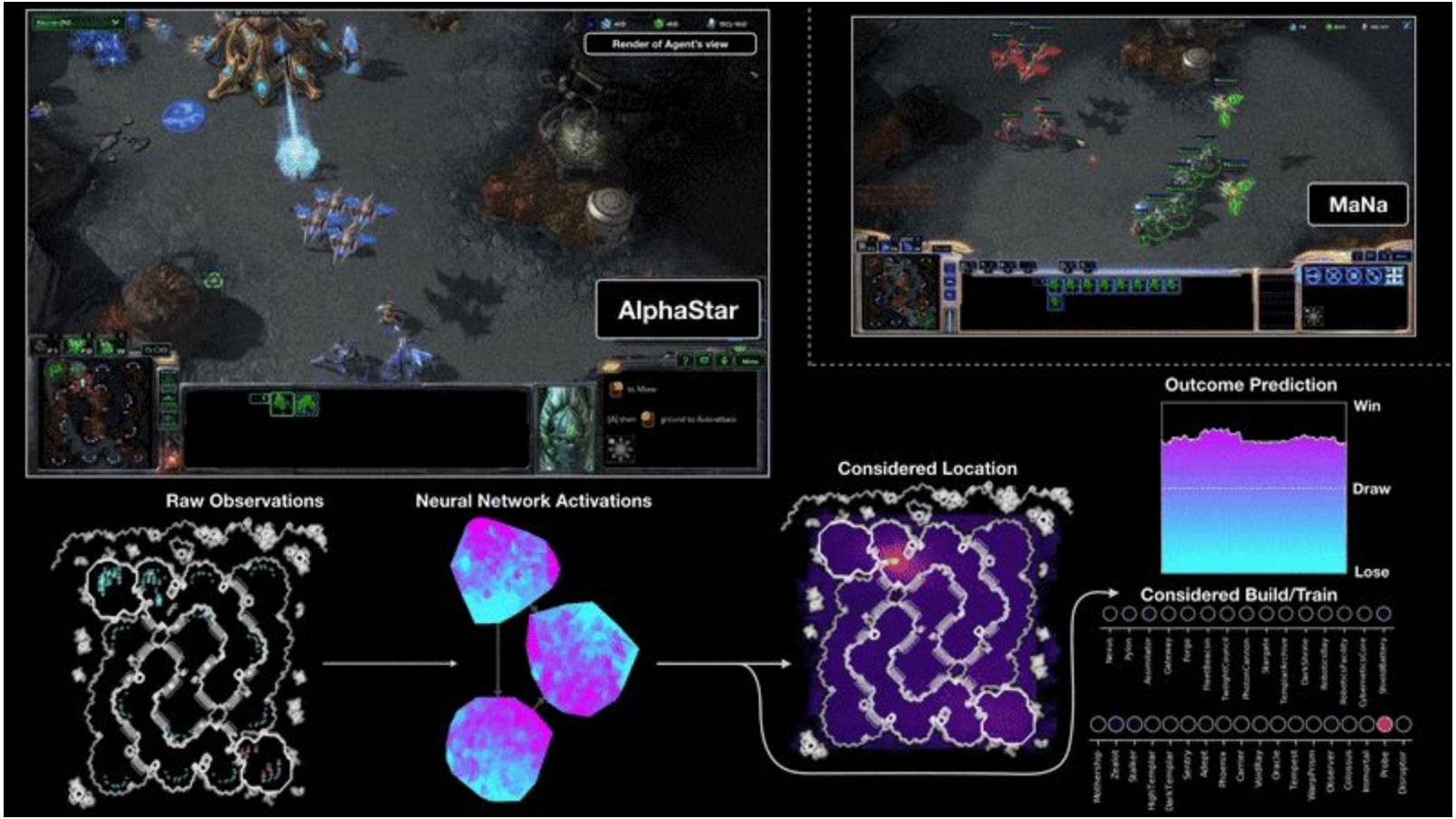
https://ko.wikipedia.org/wiki/%EB%94%A5_%EB%B8%94%EB%A3%A8


Google DeepMind
Challenge Match
8 - 15 March 2016
AlphaGo
ALPHAGO
01:59:54
LEE SEDOL
01:59:02
AlphaGo
Lee Sedol

스타2 AI '알파스타' 프로게이머에게 10대1 압승

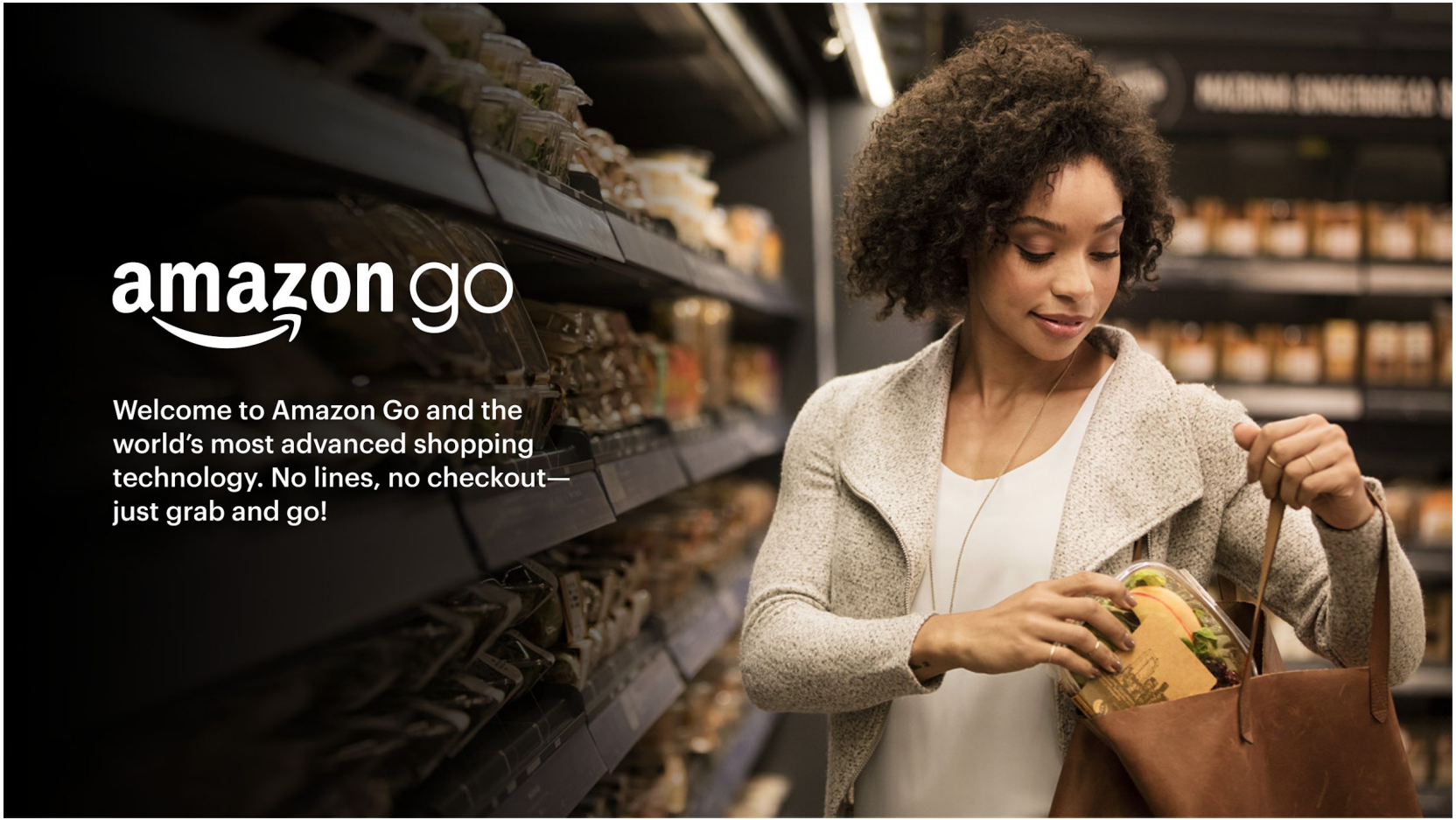
김한준 기자 | 2019.01.25

가- 가+



https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/

https://www.youtube.com/watch?v=tlThdr3O5Qo



https://www.youtube.com/watch?v=NrmMk1Myrxc

SK telecom

NUGU

인공지능 음성인식 디바이스

Clova
Smart Speaker
WAVE

Clova INSIDE

MelNet

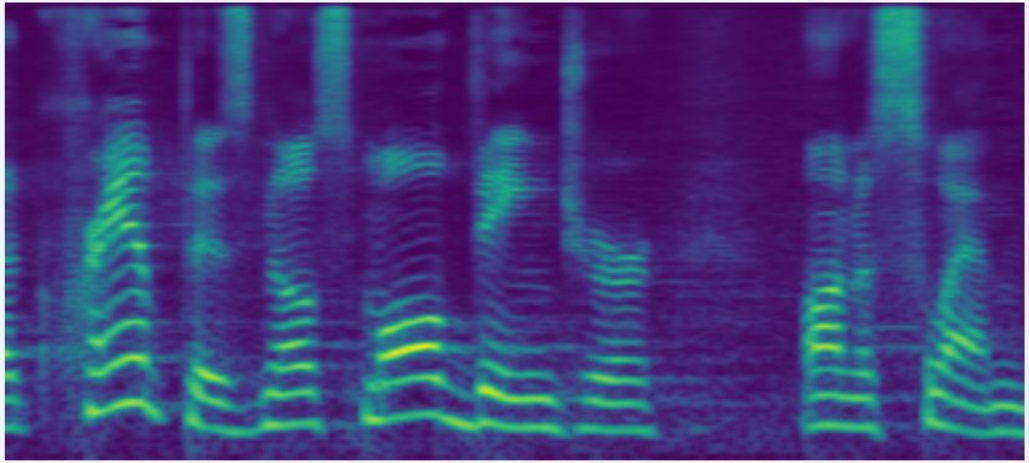
A Generative Model for Audio in the Frequency Domain

June 5, 2019

Existing generative models for audio have predominantly aimed to directly model time-domain waveforms. MelNet instead aims to model the frequency content of an audio signal. MelNet can be used to model audio unconditionally, making it capable of tasks such as music generation. It can also be conditioned on text and speaker, making it applicable to tasks such as text-to-speech and voice conversion. The full paper is available on arXiv: https://arxiv.org/abs/1906.01083.

Demo: Select a speaker and a sentence to view a spectrogram generated by MelNet. Play the audio to visualize how the frequencies change over time.

Speaker: Bill Gates ▾ Text: A cramp is no small danger on a swim. ▾



https://sjvasquez.github.io/blog/melnet/?fbclid=IwAR0ZT_QfmWpOpBj0CK5ReT_Usa229VFnmpppPUBlnmB7Zd545wHgHQjezUs

MuseNet

We've created MuseNet, a deep neural network that can generate 4-minute musical compositions with 10 different instruments, and can combine styles from country to Mozart to the Beatles. MuseNet was not explicitly programmed with our understanding of music, but instead discovered patterns of harmony, rhythm, and style by learning to predict the next token in hundreds of thousands of MIDI files. MuseNet uses the same general-purpose unsupervised technology as GPT-2, a large-scale transformer model trained to predict the next token in a sequence, whether audio or text.

APRIL 25, 2019
6 MINUTE READ, 16 MINUTE LISTEN

https://openai.com/blog/musenet/

How to make a pizza: Learning a compositional layer-based GAN model

Dim P. Papadopoulos
MIT

Youssef Tamaazousti
MIT

Ferda Ofli
QCRI

Ingmar Weber
QCRI

Antonio Torralba
MIT

Massachusetts Institute of Technology

QCRI
معهد قطر لبحوث الحوسبة
Qatar Computing Research Institute
جامعة حمد بن خليفة
HAMAD BIN KHALIFA UNIVERSITY

http://pizzagan.csail.mit.edu/?fbclid=IwAR2ec-MaUqRWClA4iNDpTirpDfHrCQ4nd8hnls1aYOinwEq32UBqa7ygGjA

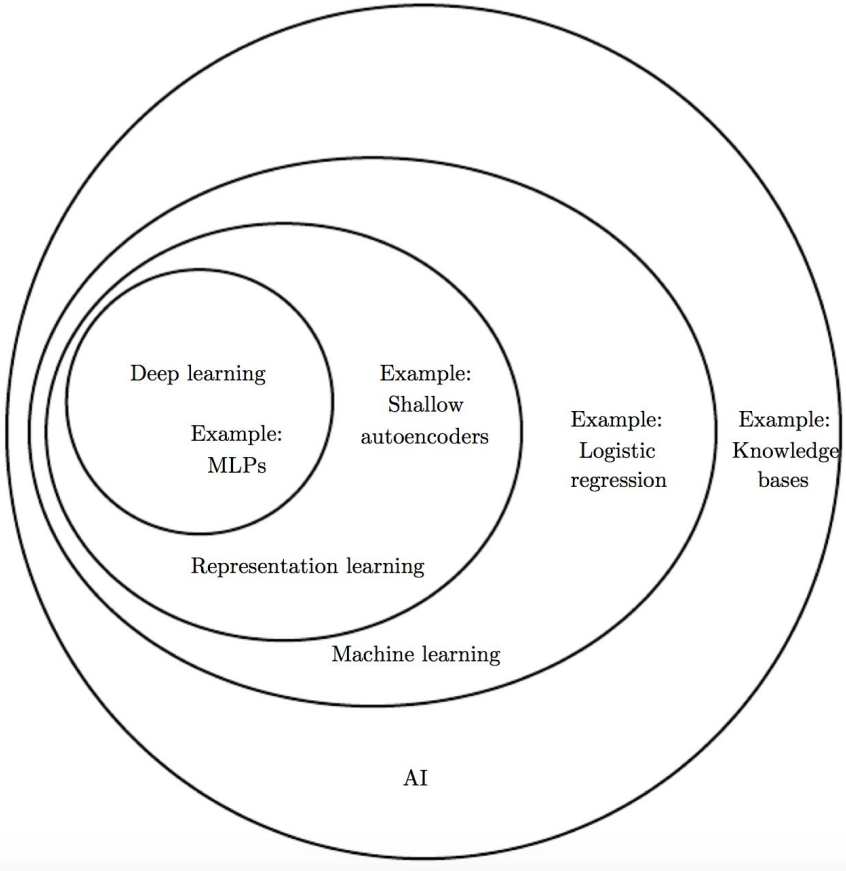


Figure credit: Ian Goodfellow et. al., Deep Learning (2015), chap. 1, p. 9

Machine Learning

- Arthur Samuel (1959). Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.
  - 명시적인 프로그래밍 없이 컴퓨터가 학습하는 능력을 갖추게 하는 연구분야 (heuristic)

- Tom Mitchell (1997). Well-posed Learning Problem: A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.
  - 어떤 작업 T에 대한 컴퓨터 프로그램의 성능을 P로 측정했을 때 경험 E로 인해 성능이 향상됐다면, 이 컴퓨터 프로그램은 작업 T와 성능 측정 P에 대해 경험 E로 학습한 것이다.

Machine Learning

- Spam filter
  - Task T: to flag spam for new emails
  - Experience E: the training data
  - Performance measure P: the ratio of correctly classified emails (accuracy)

Machine Learning

1. 스팸에 주로 나타나는 단어를 살핀다. ‘4U’, ‘신용카드’, ‘무료’, ‘굉장한’ 같은 단어나 구절이 제목에 많이 나타나는 경향을 발견

2. 발견한 각 패턴을 감지하는 알고리즘을 작성하여 프로그램이 해당 패턴을 발견했을 때 그 메일을 스팸으로 분류

3. 프로그램을 테스트하고 충분한 성능이 나올 때까지 1단계 2단계 반복

Machine Learning

- Traditional Programming

- Machine Learning

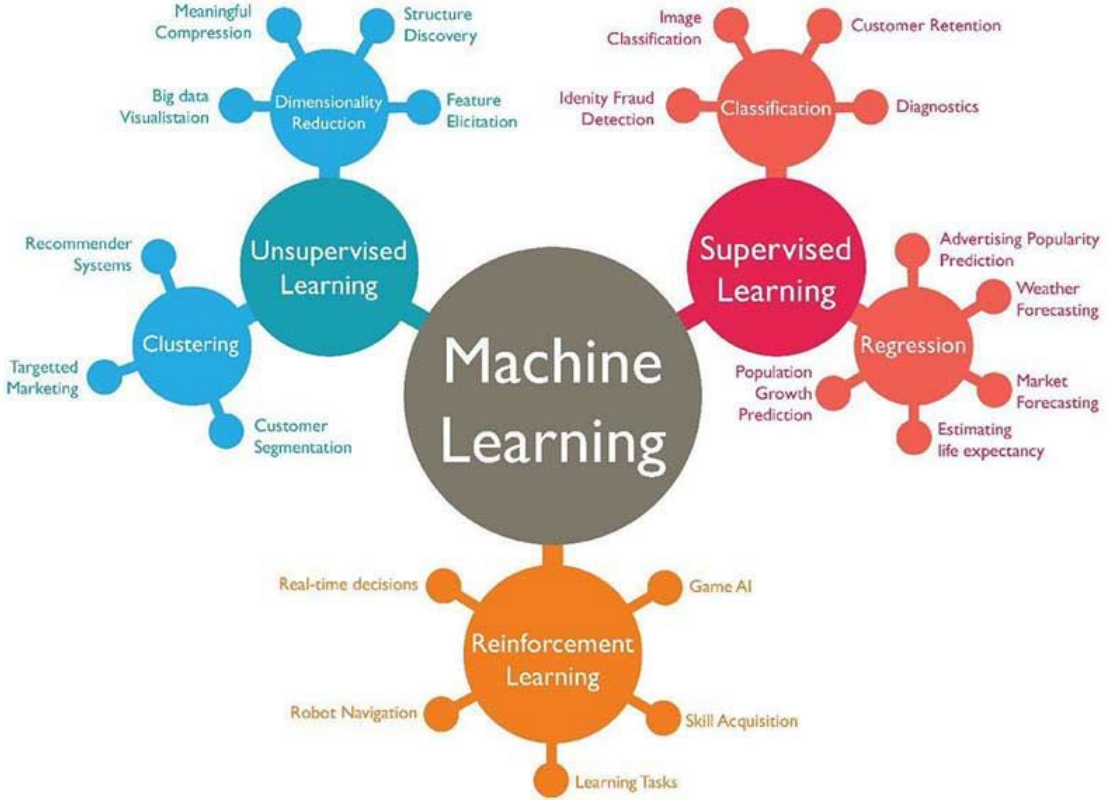
Machine Learning

- 기존 솔루션으로는 많은 수동 조정과 규칙이 필요한 문제
- 전통적인 방식으로는 전혀 해결 방법이 없는 복잡한 문제
- 유동적인 환경: 머신러닝 시스템은 새로운 데이터에 적응 가능
- 복잡한 문제와 대량의 데이터에서 통찰 얻기

Machine Learning

- Go, Chess, Starcraft, games
- Self-driving car
- Speech recognition
- Home-assistant robots
- Computer vision
- Recommend systems
- Natural language processing
- Sentiment analysis
- Web search
- Spam filter
- Fraud detection
- Chatbot
- Bio-informatics
- Healthcare
- Stock market prediction
- Classifying DNA sequences

Machine Learning



https://www.techleer.com/articles/203-machine-learning-algorithm-backbone-of-emerging-technologies/

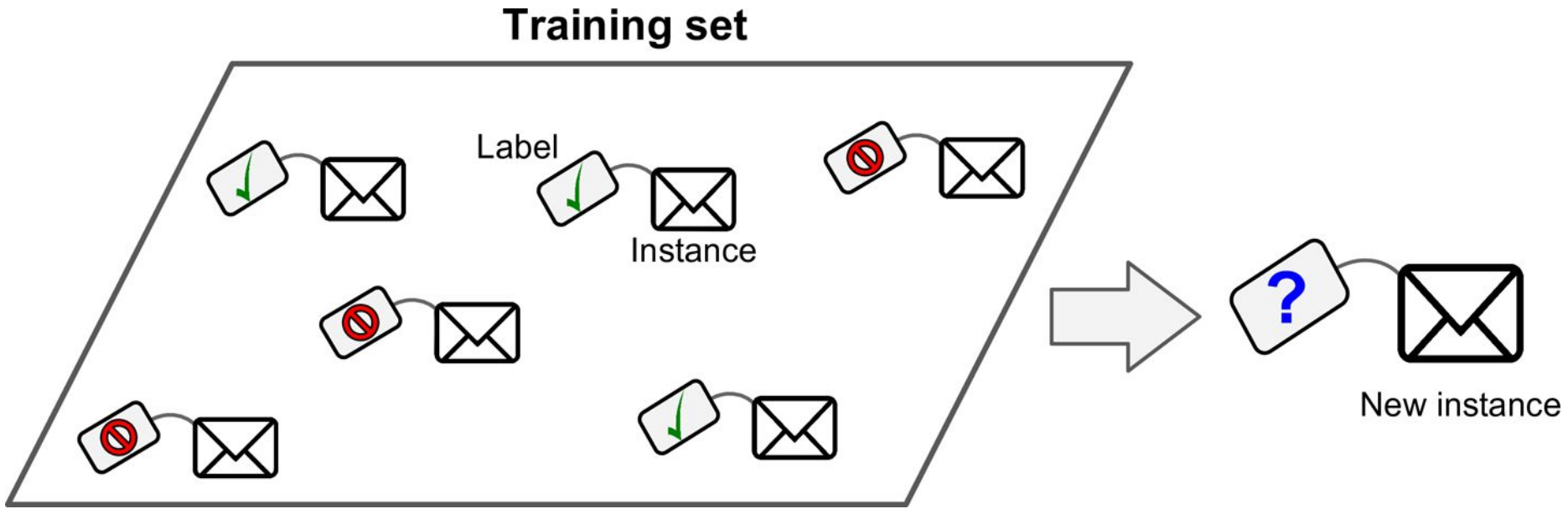
Machine Learning

- Label의 유무
  - Supervised Learning
  - Unsupervised Learning
  - Semi-supervised Learning
- Other types
  - Reinforcement Learning

Supervised Learning

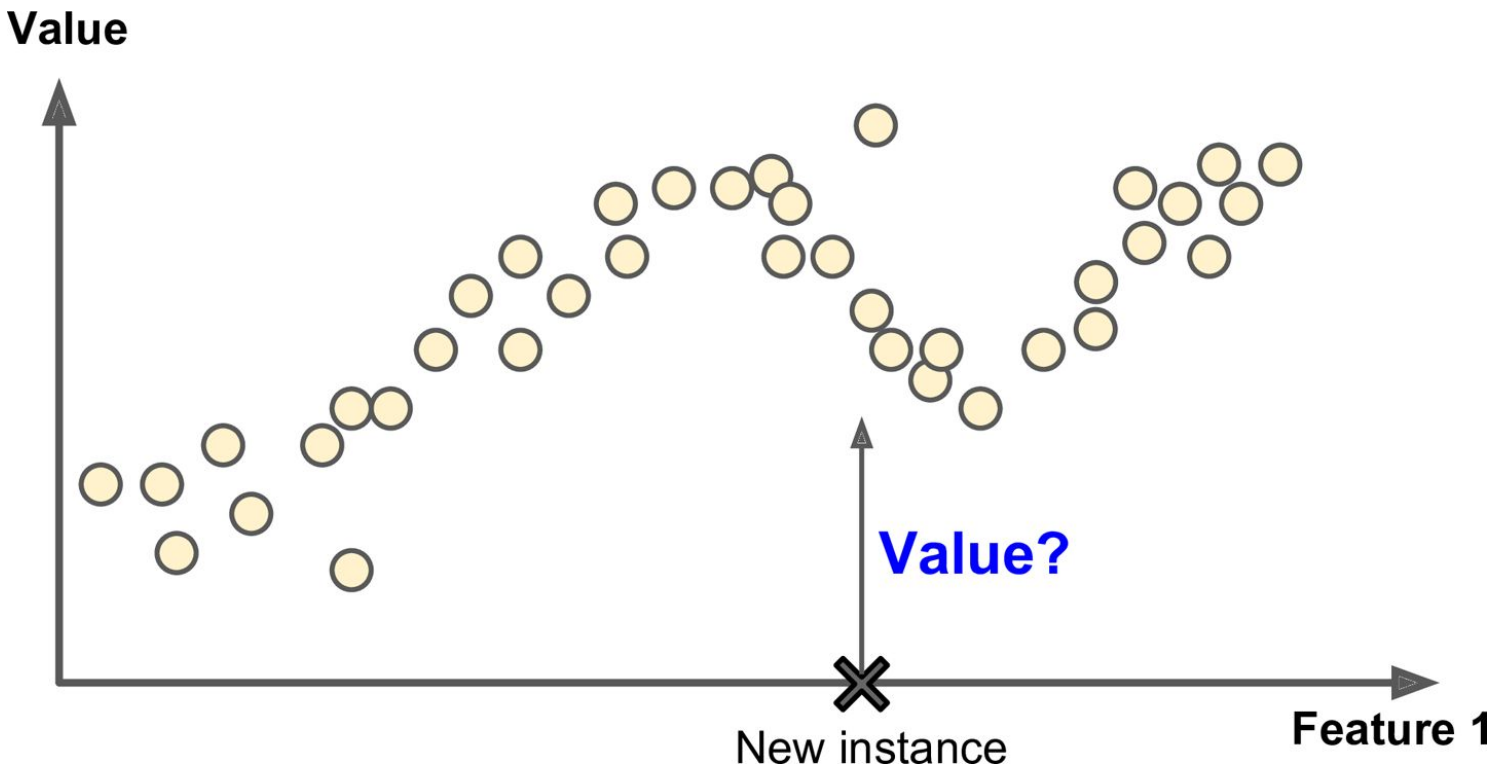
	feature 1	feature 2	feature 3	feature 4	…	…	Label
example 1	172	F	3.8	-2.3			**A**
example 2	180	M	6.3	-1.7			**A**
example 3	167	M	2.9	3.5			**B**
example 4	176	M	1.2	2.1			**C**
example 5	175	F	5.1	-2.9			**C**
…							

Classification



A. Géron, Hands-on Machine Learning with Scikit-Learn & TensorFlow, chap 1, p. 8

Regression

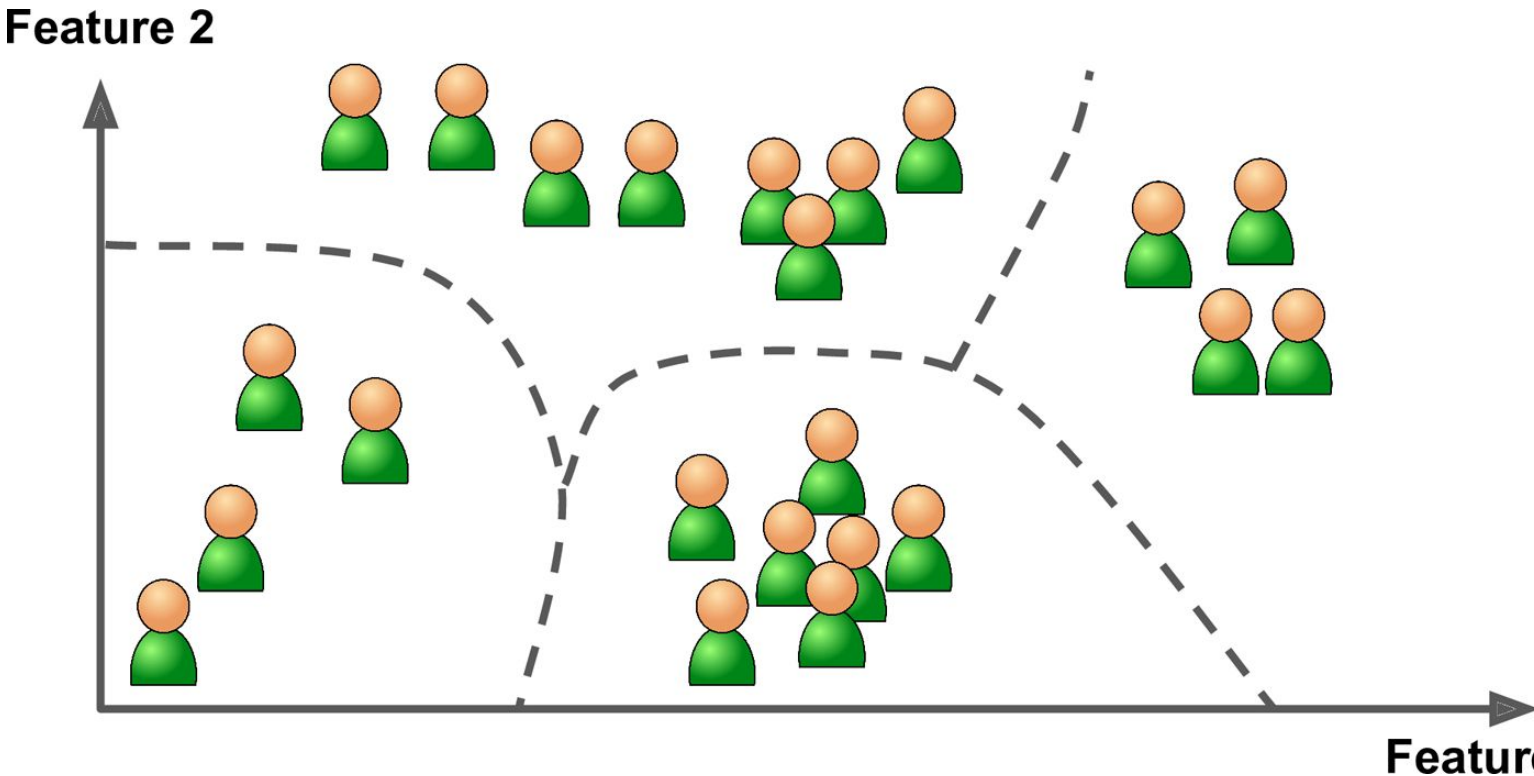


A. Géron, Hands-on Machine Learning with Scikit-Learn & TensorFlow, chap 1, p. 9

Unsupervised Learning

	feature 1	feature 2	feature 3	feature 4	...	...	...
example 1	172	F	3.8	-2.3			
example 2	180	M	6.3	-1.7			
example 3	167	M	2.9	3.5			
example 4	176	M	1.2	2.1			
example 5	175	F	5.1	-2.9			
...							

Clustering



A. Géron, Hands-on Machine Learning with Scikit-Learn & TensorFlow, chap 1, p. 11

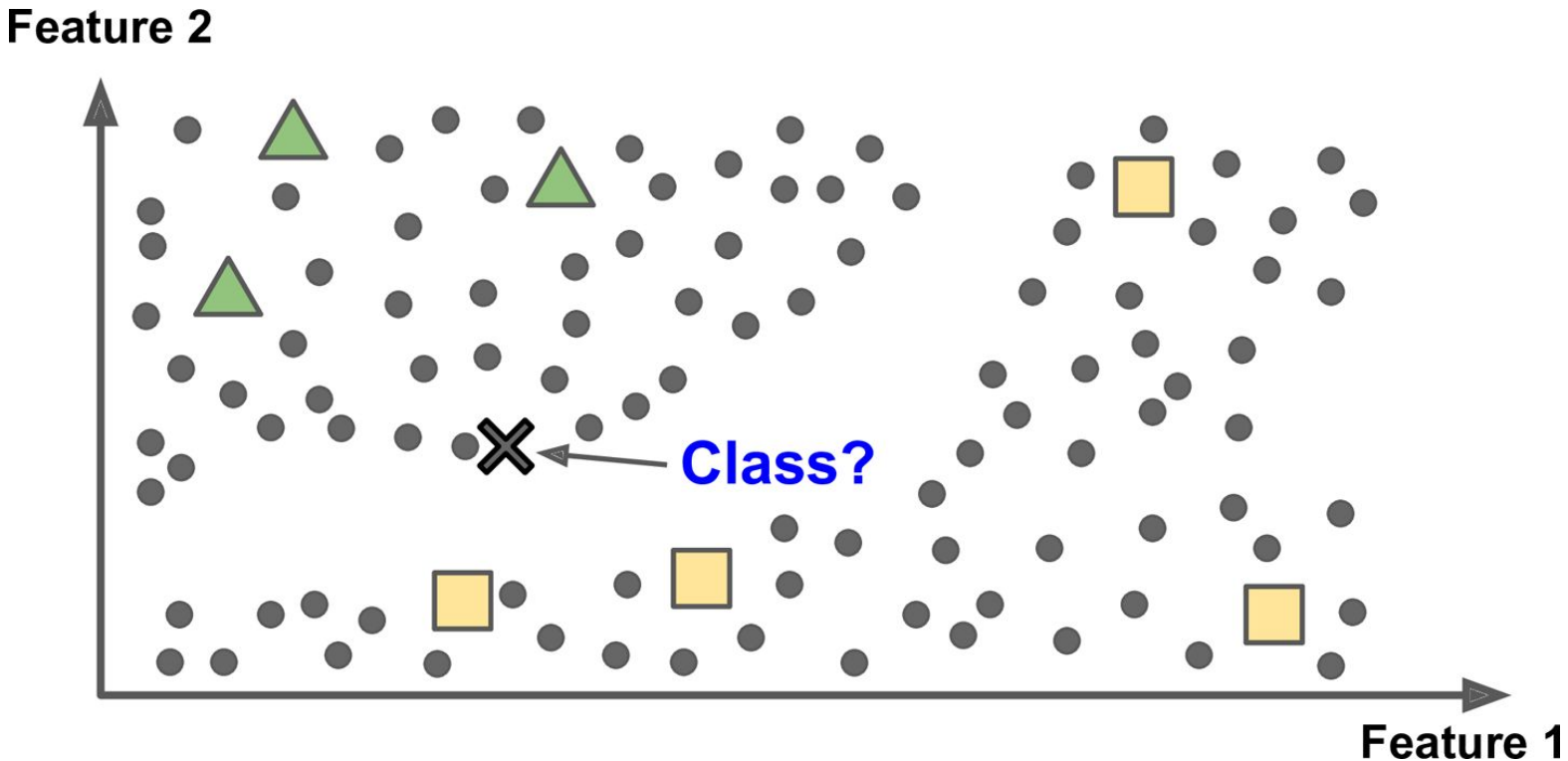
Anomaly Detection

A. Géron, Hands-on Machine Learning with Scikit-Learn & TensorFlow, chap 1, p. 12

Semi-supervised Learning

	feature 1	feature 2	feature 3	feature 4	...	...	Label
example 1	172	F	3.8	-2.3			**A**
example 2	180	M	6.3	-1.7			**N/A**
example 3	167	M	2.9	3.5			**B**
example 4	176	M	1.2	2.1			**C**
example 5	175	F	5.1	-2.9			**N/A**
...							

Semi-supervised Learning



A. Géron, Hands-on Machine Learning with Scikit-Learn & TensorFlow, chap 1, p. 13

Semi-supervised Learning

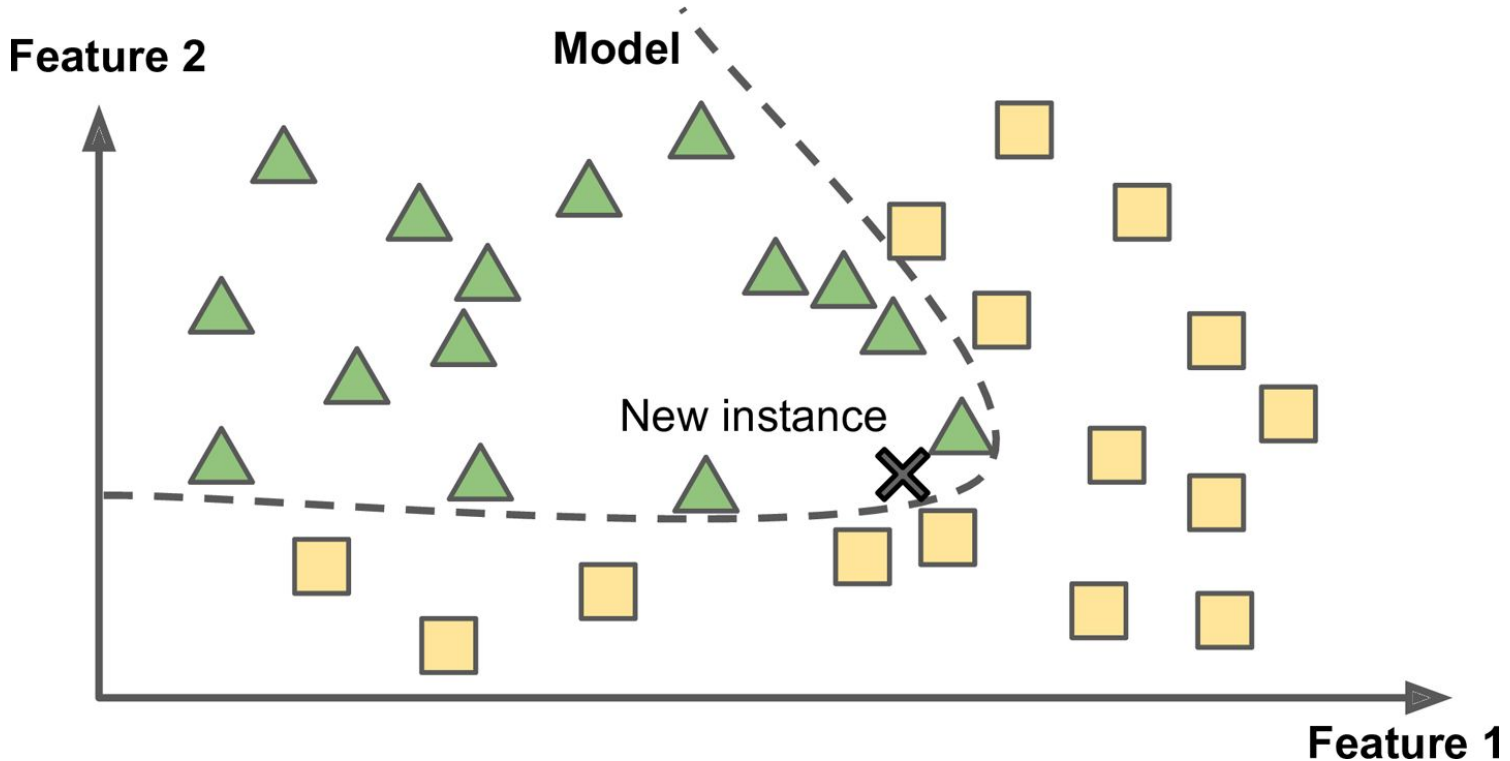
	feature 1	feature 2	feature 3	feature 4	...	...	...
example 1	172	F	3.8	-2.3			A
example 2	180	M	6.3	-1.7			A
example 3	167	M	2.9	3.5			B
example 4	176	M	1.2	2.1			C
example 5	175	F	5.1	-2.9			C
...							

Find mapping function

$$\mathbf{y} = f(\mathbf{x})$$

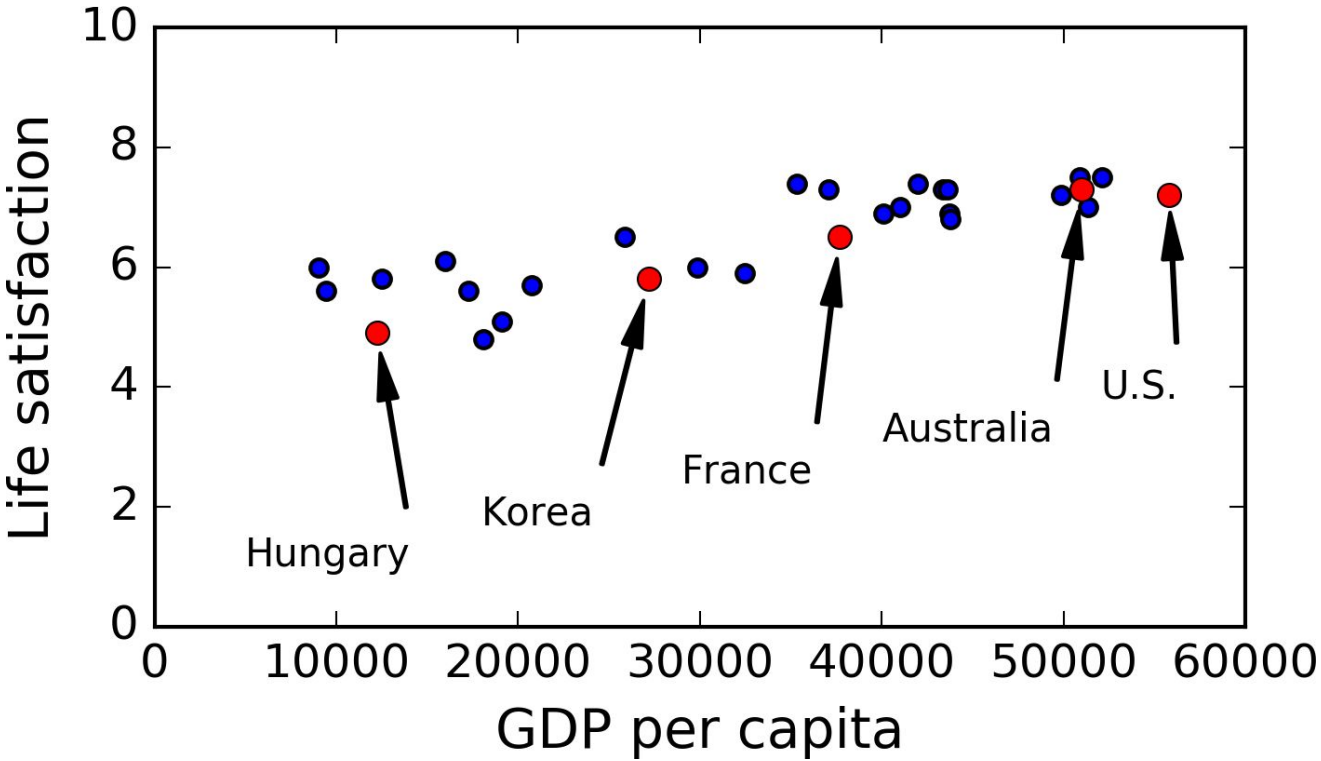
Function Approximator

Model-Based Learning



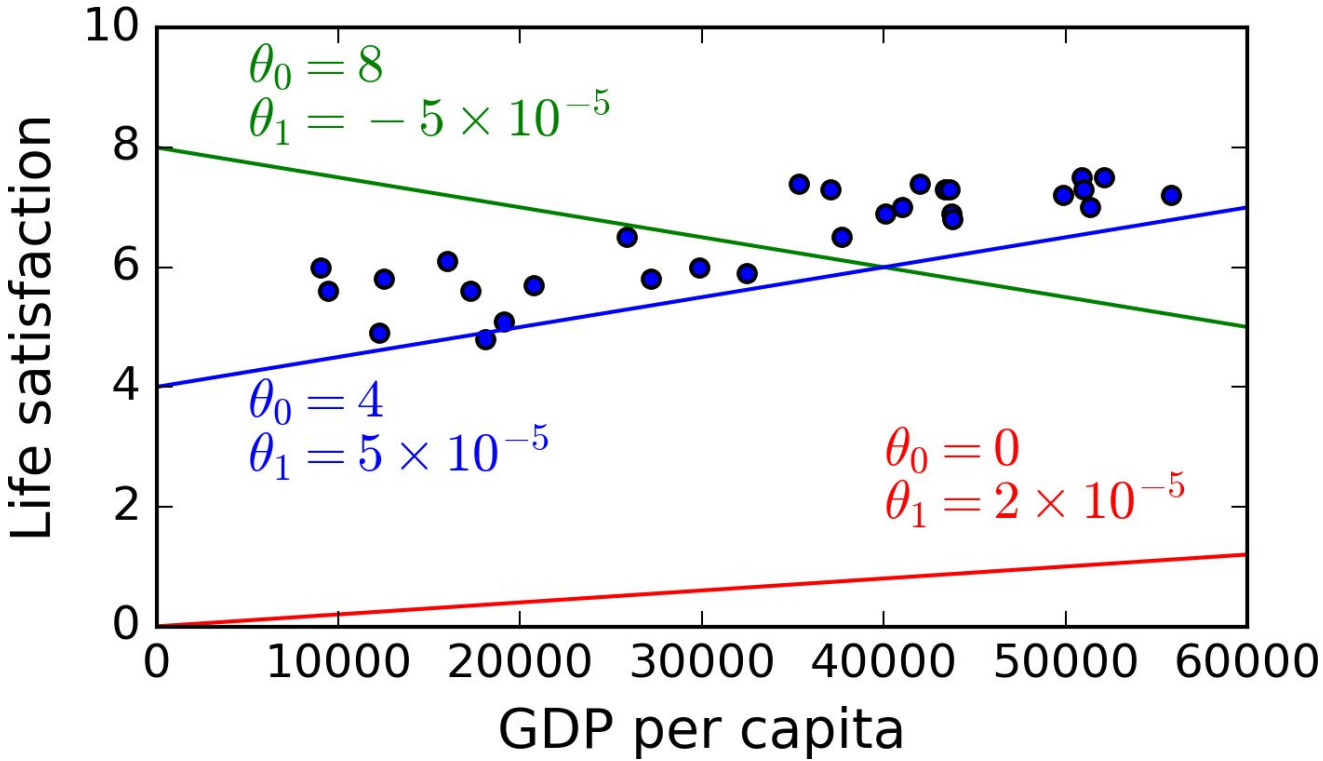
A. Géron, Hands-on Machine Learning with Scikit-Learn & TensorFlow, chap 1, p. 18

Model-Based Learning



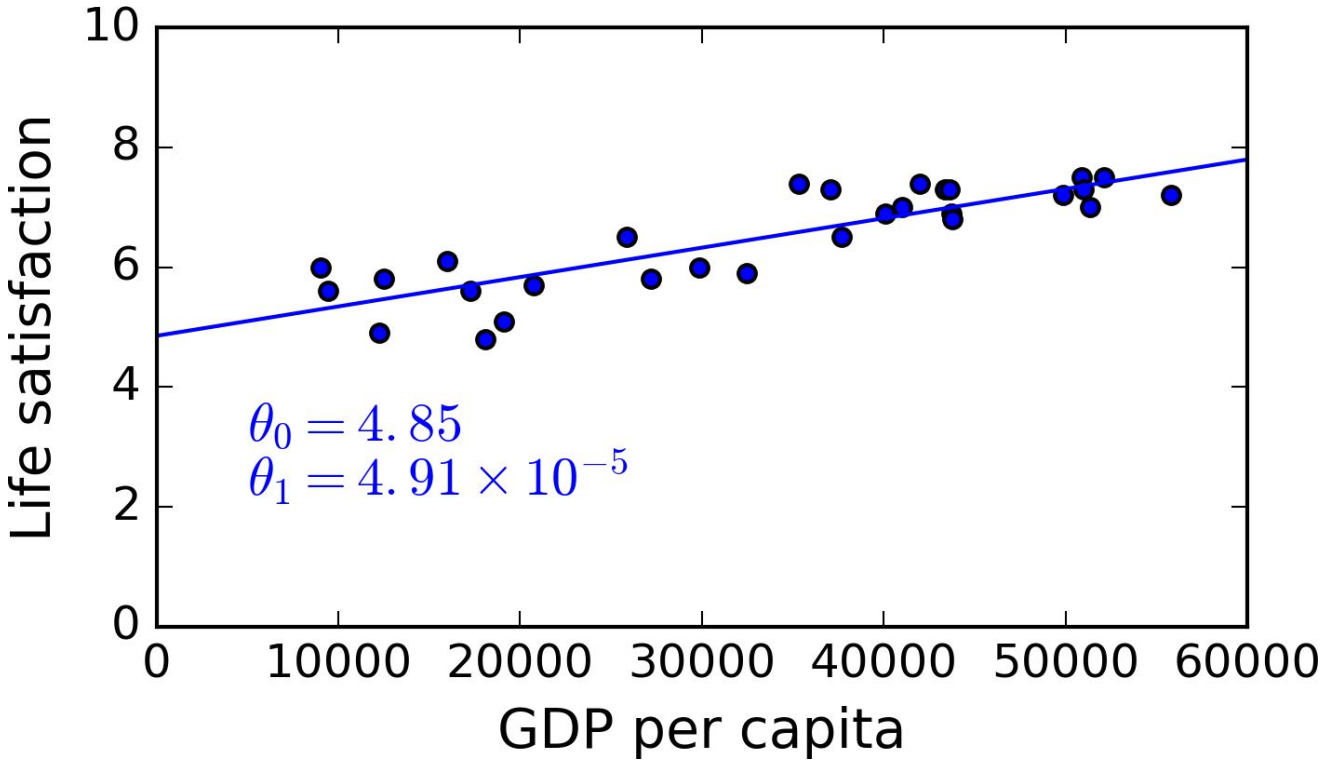
A. Géron, Hands-on Machine Learning with Scikit-Learn & TensorFlow, chap 1, p. 19

Model-Based Learning



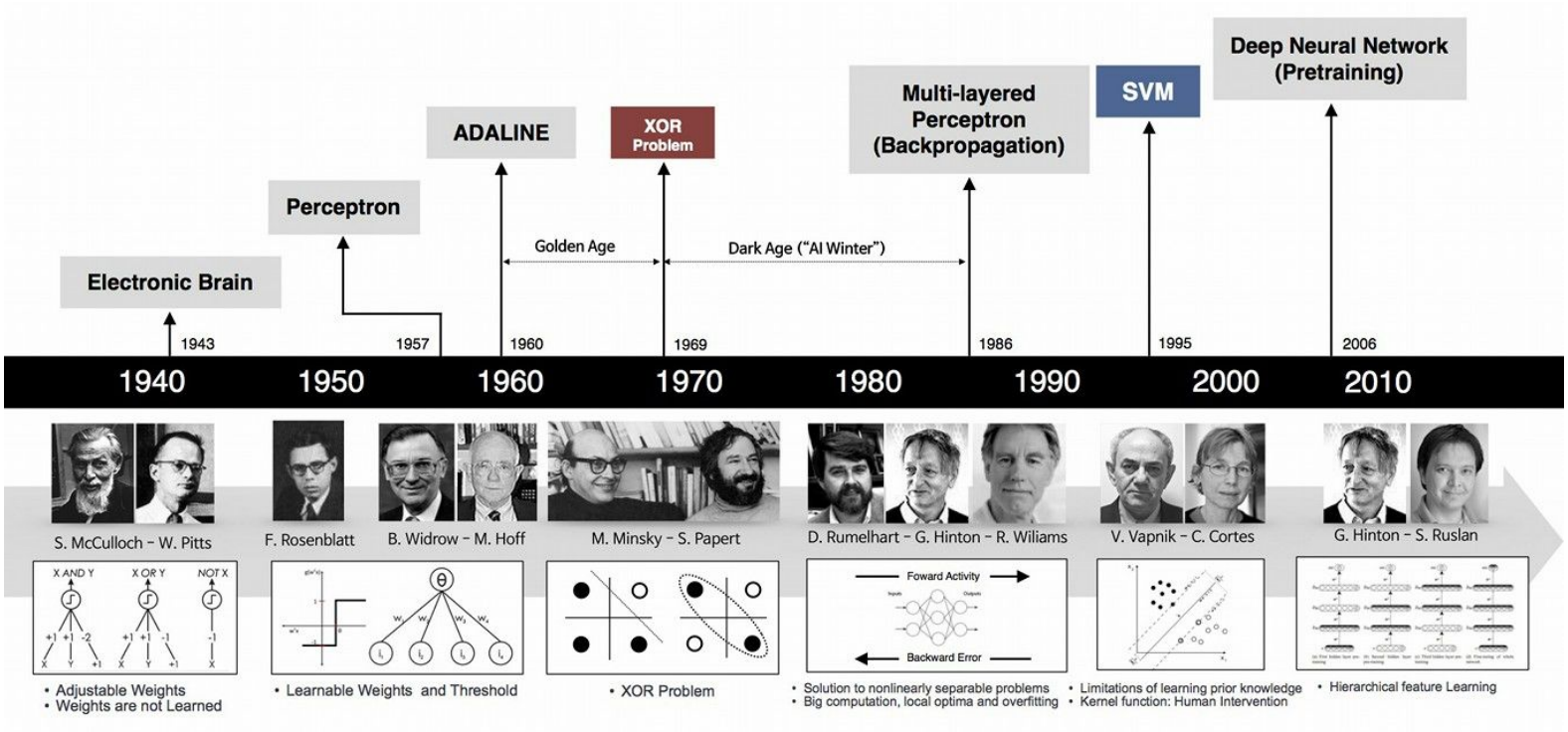
A. Géron, Hands-on Machine Learning with Scikit-Learn & TensorFlow, chap 1, p. 19

Model-Based Learning



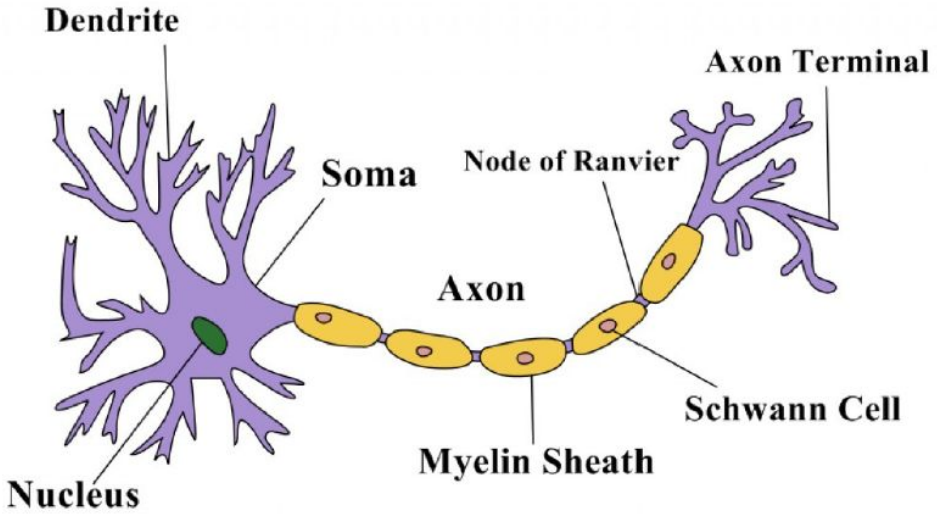
A. Géron, Hands-on Machine Learning with Scikit-Learn & TensorFlow, chap 1, p. 20

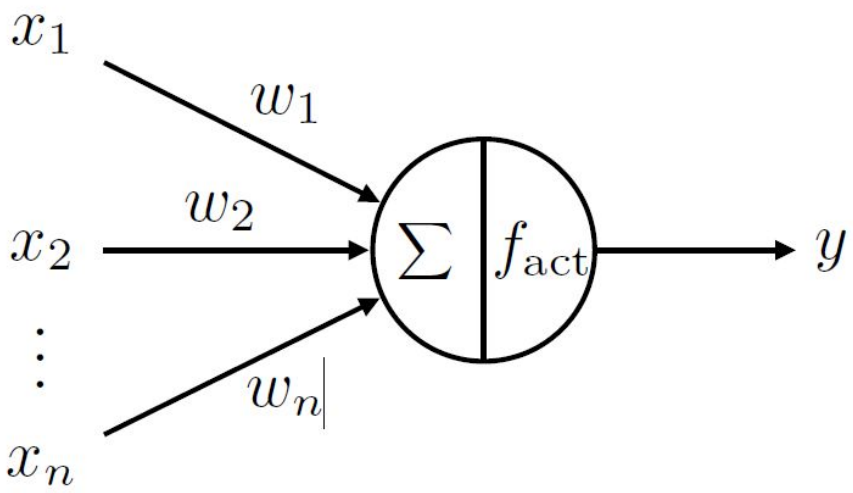
History of Neural Networks



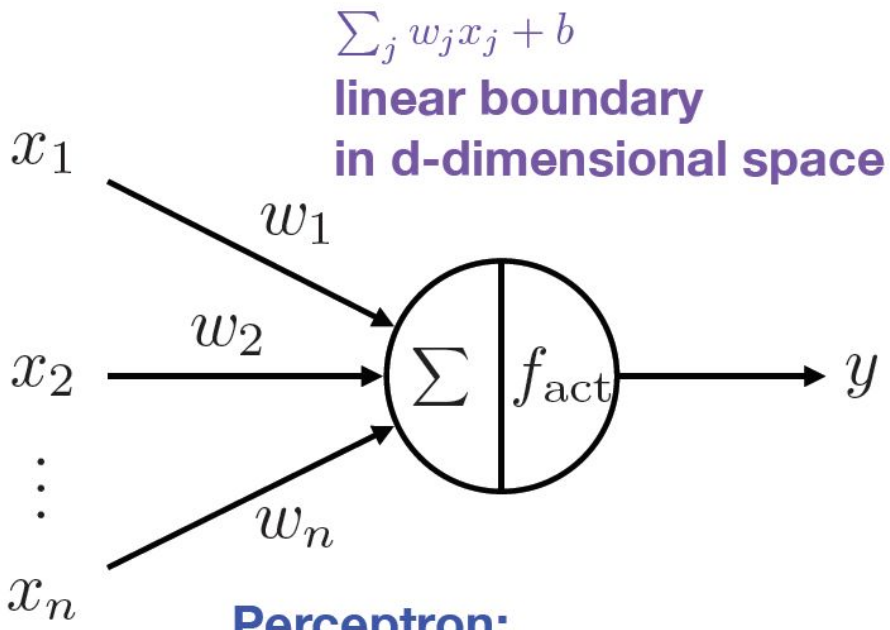
Ian Goodfellow et. al., Deep Learning (2015), chap. 1, p. 9

Perceptron


Dendrite
Axon Terminal
Soma
Node of Ranvier
Axon
Schwann Cell
Myelin Sheath
Nucleus


x_1
w_1
w_2
x_2
Σ
f_{act}
y
w_n
x_n

Perceptron

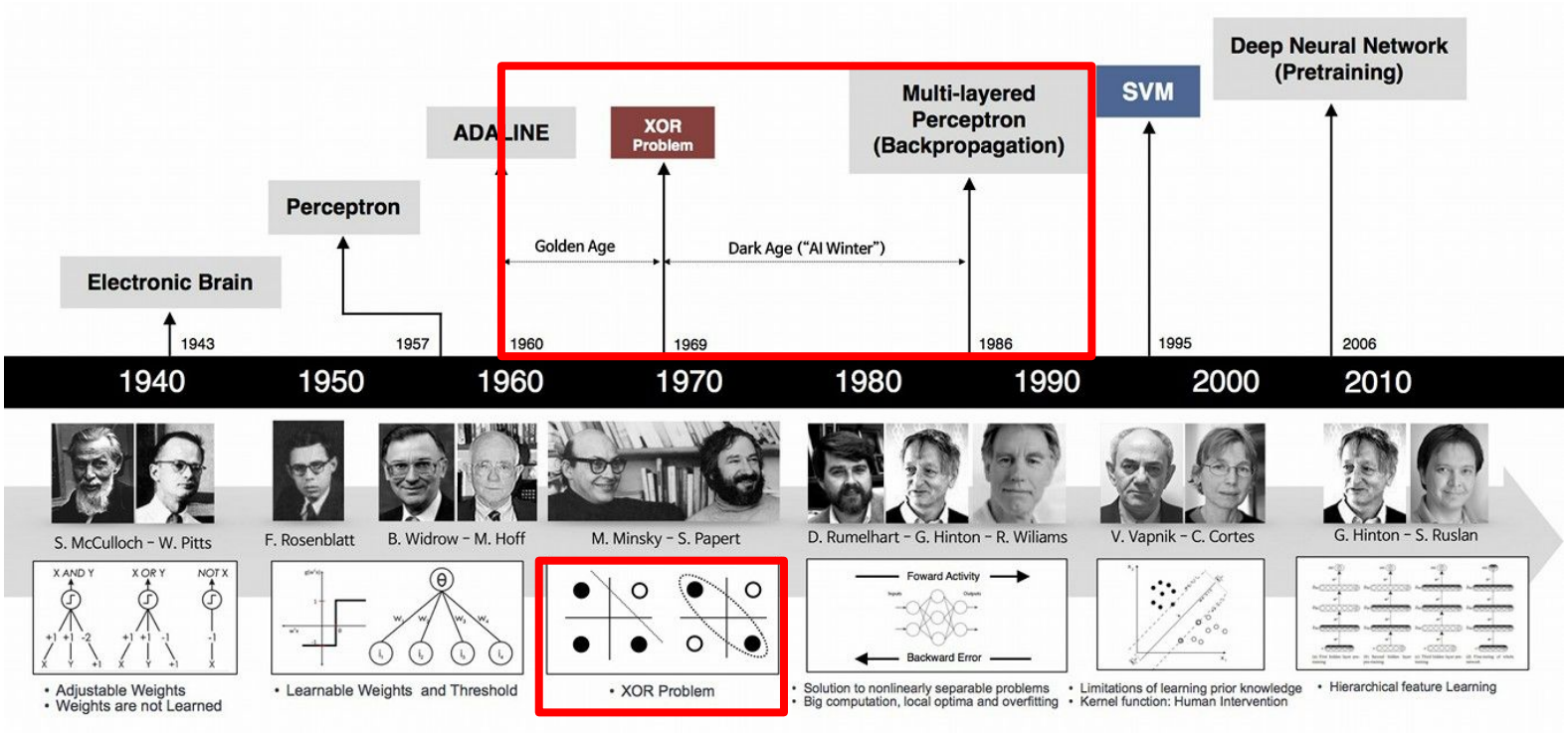


$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

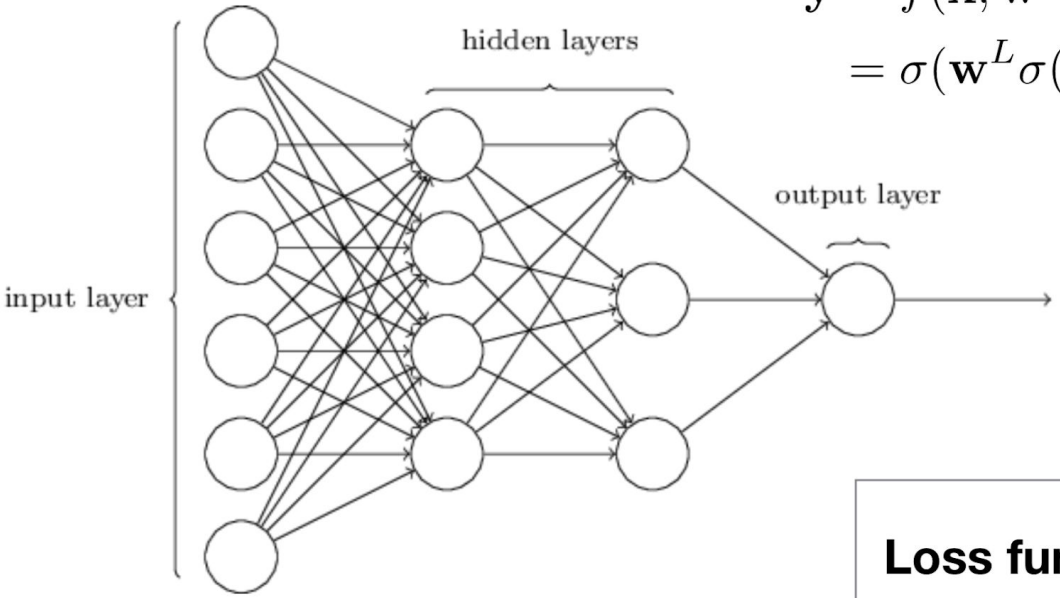
$$\text{output} = f_{\text{act}}\left(\sum_j w_j x_j - \text{threshold}\right)$$

activation function:
Heaviside step function
bias:
-threshold

XOR Problem


Deep Neural Network (Pretraining)
SVM
Multi-layered Perceptron (Backpropagation)
XOR Problem
ADALINE
Perceptron
Electronic Brain
Golden Age
Dark Age ("AI Winter")
1943
1957
1960
1969
1986
1995
2006
1940
1950
1960
1970
1980
1990
2000
2010
S. McCulloch - W. Pitts
F. Rosenblatt
B. Widrow - M. Hoff
M. Minsky - S. Papert
D. Rumelhart - G. Hinton - R. Wiliams
V. Vapnik - C. Cortes
G. Hinton - S. Ruslan
X AND Y
X OR Y
NOT X
Foward Activity
Backward Error
• Adjustable Weights
• Weights are not Learned
• Learnable Weights and Threshold
• XOR Problem
• Solution to nonlinearly separable problems
• Big computation, local optima and overfitting
• Limitations of learning prior knowledge
• Kernel function: Human Intervention
• Hierarchical feature Learning

Neural Networks



$$\hat{\mathbf{y}} = f(\mathbf{x}; \mathbf{w}^1, \cdots, \mathbf{w}^L, \mathbf{b}^1, \cdots, \mathbf{b}^L)$$
$$= \sigma(\mathbf{w}^L \sigma(\mathbf{w}^{L-1} \cdots \sigma(\mathbf{w}^1 \mathbf{x} + \mathbf{b}^1) \cdots) + \mathbf{b}^L)$$

activation function:
sigmoid function

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

Loss function: $\mathcal{L} \equiv \frac{1}{2N} \sum_{n=1}^{N} ||\hat{\mathbf{y}} - \mathbf{y}||^2$

Ian Goodfellow et. al., Deep Learning (2015), chap. 1, p. 9

Learning

- 주어진 loss function $\mathcal{L}$을 **최소화** 하는 parameters Θ를 찾는 것

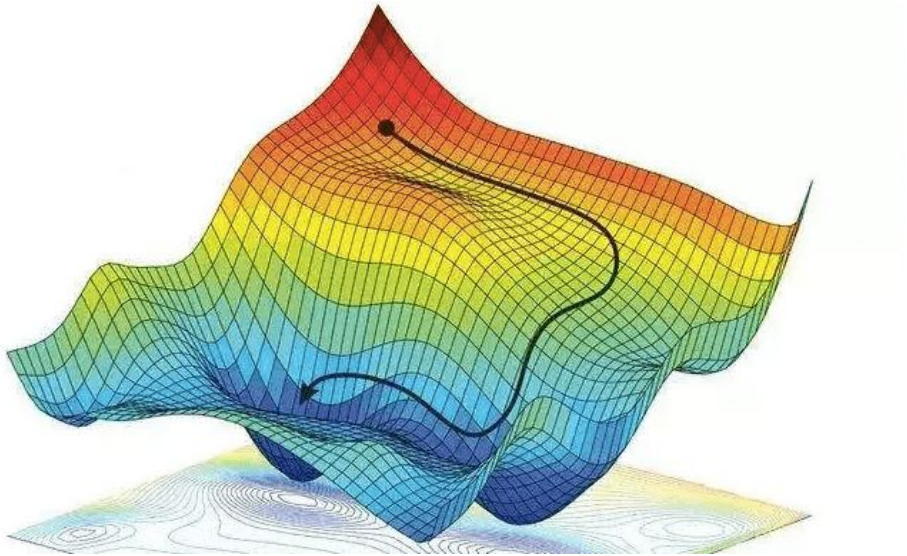
$$\arg\min_{\Theta} \mathcal{L} \qquad \Theta = \{\mathbf{w}_1, \cdots, \mathbf{b}_1, \cdots\}$$

- 간단하게 말하면 굉장히 복잡한 함수를 잘 **fitting** 하는 것
- 학습의 목적 (머신러닝의 목적)
  - 주어진 데이터를 이용하여 loss function을 최소화 함과 동시에 새로운 데이터들에 대해서도 잘 맞추도록 (**generalization**) 하는 것 (avoiding overfitting)

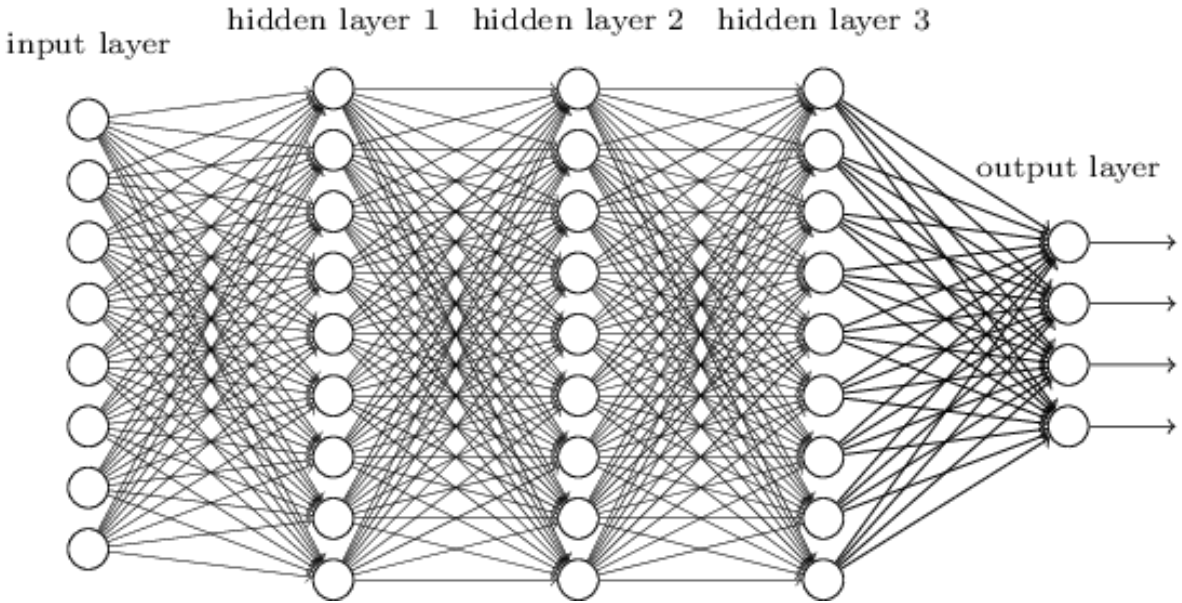
Learning

Gradient Descent

$$w'_k := w_k - \eta \frac{\partial \mathcal{L}}{\partial w_k}$$



Deep Neural Networks



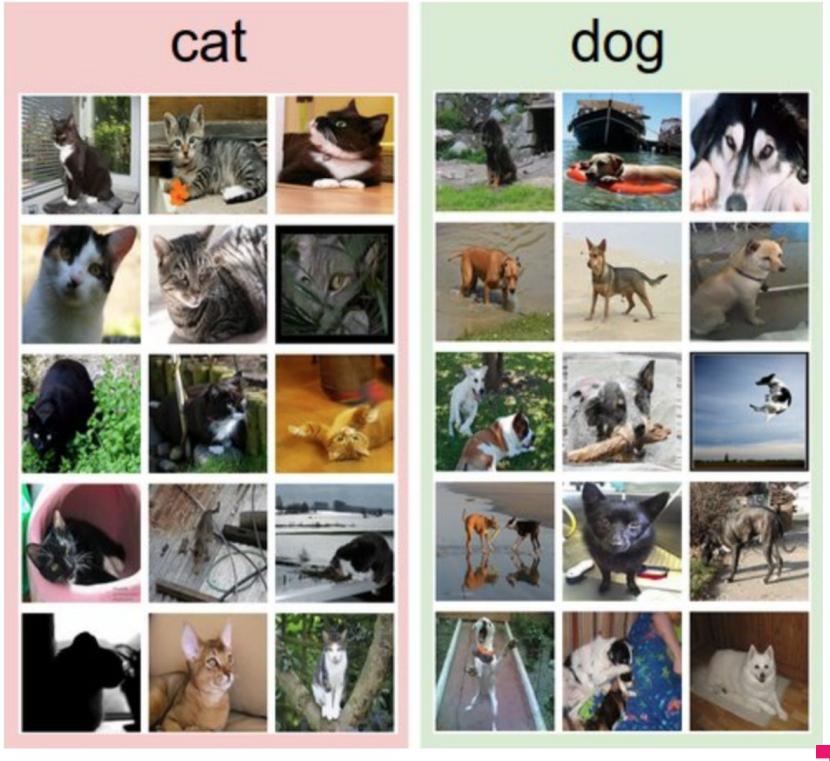
http://neuralnetworksanddeeplearning.com/chap5.html

Deep Neural Networks

- 새로운 architectures (CNN, LSTM, etc)
- Various regularization methods :
- Big Data
- more powerful Computing power : GPUs, TPUs

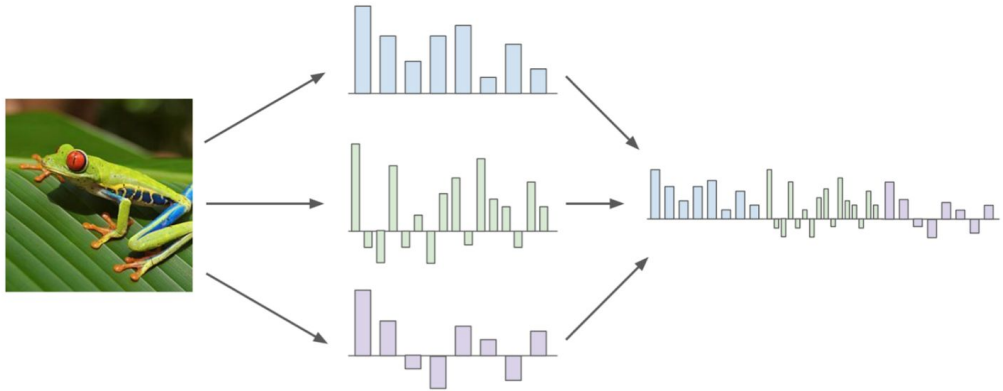
Back to Machine Learning

- Image classification
  - Feature extraction
  - 고양이와 개를 잘 구별할 수 있는 특징들을 뽑아 데이터를 만든다
- 좋은 feature가 ML의 성공여부를 결정
  - Feature들이 서로 독립
  - Label과의 높은 상관관계

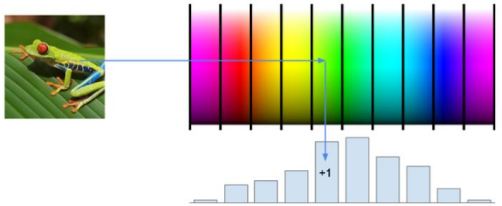


http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture04.pdf

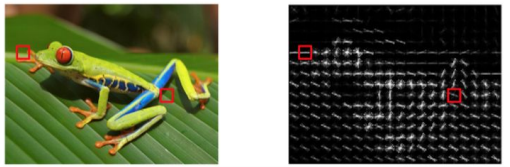
Back to Machine Learning



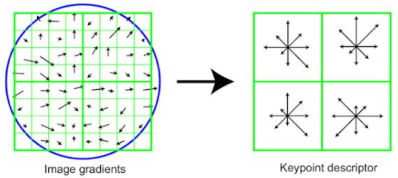
Color Histogram



HoG

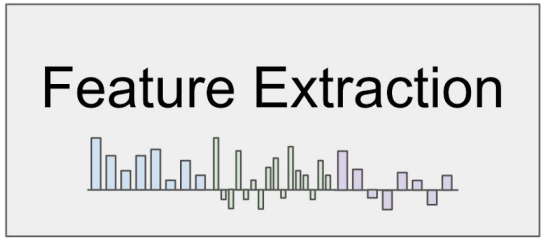


SIFT



http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture04.pdf

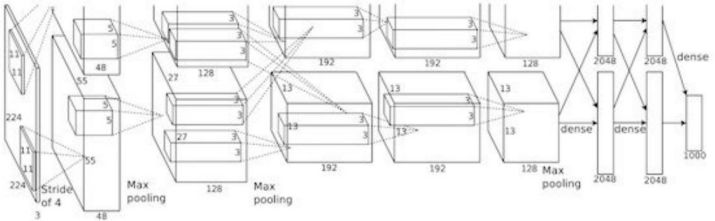
Machine Learning and Deep Learning



f

10 numbers giving scores for classes

training



Krizhevsky, Sutskever, and Hinton, "Imagenet classification with deep convolutional neural networks", NIPS 2012. Figure copyright Krizhevsky, Sutskever, and Hinton, 2012. Reproduced with permission.

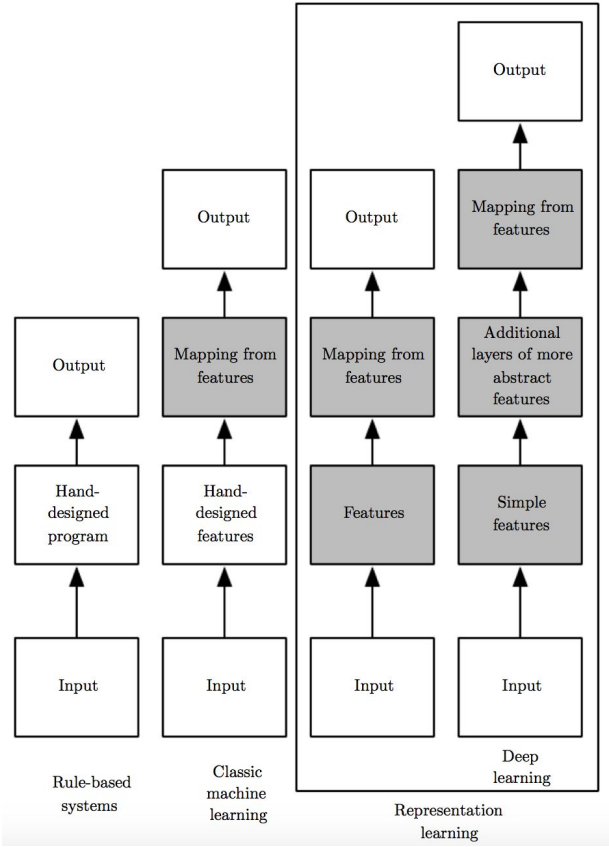
10 numbers giving scores for classes

training

http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture04.pdf

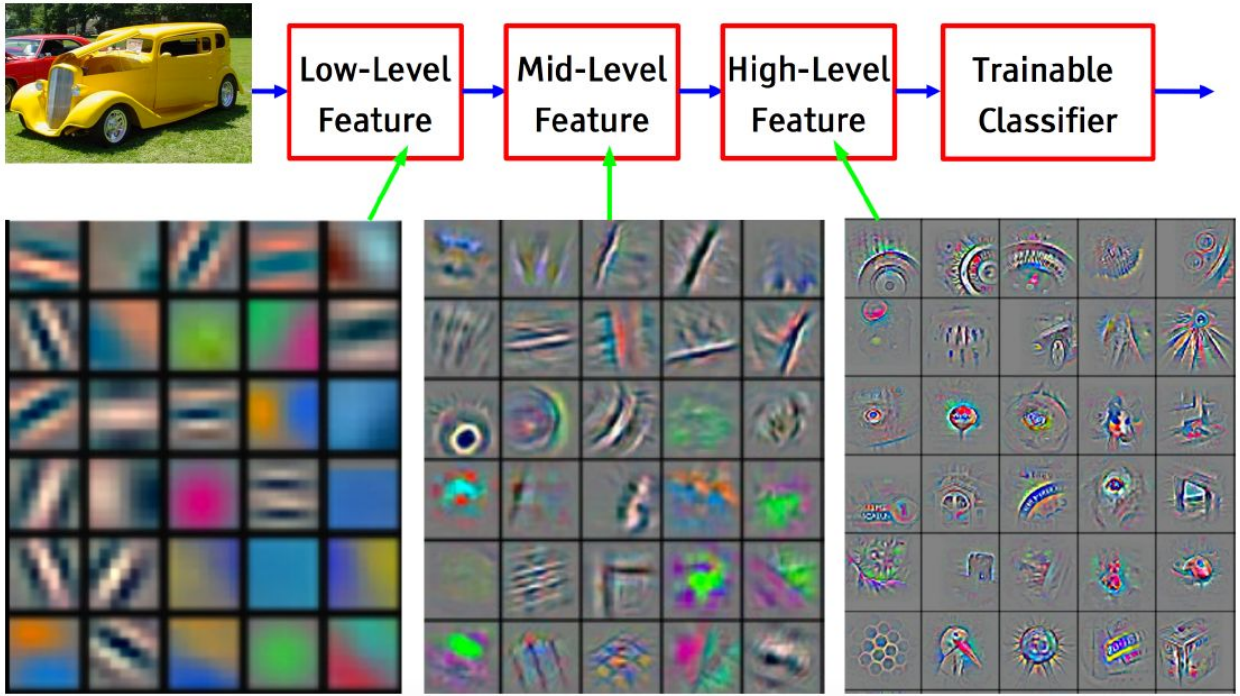
Deep Neural Networks to Deep Learning

- Learning multiple levels of representation
- Representation / Feature learning
- Generally many layers
- From high dimensional vector to low dimensional vector (vice versa)
- F. Chollet's quotation: to map space X to space Y using a continuous geometric transform

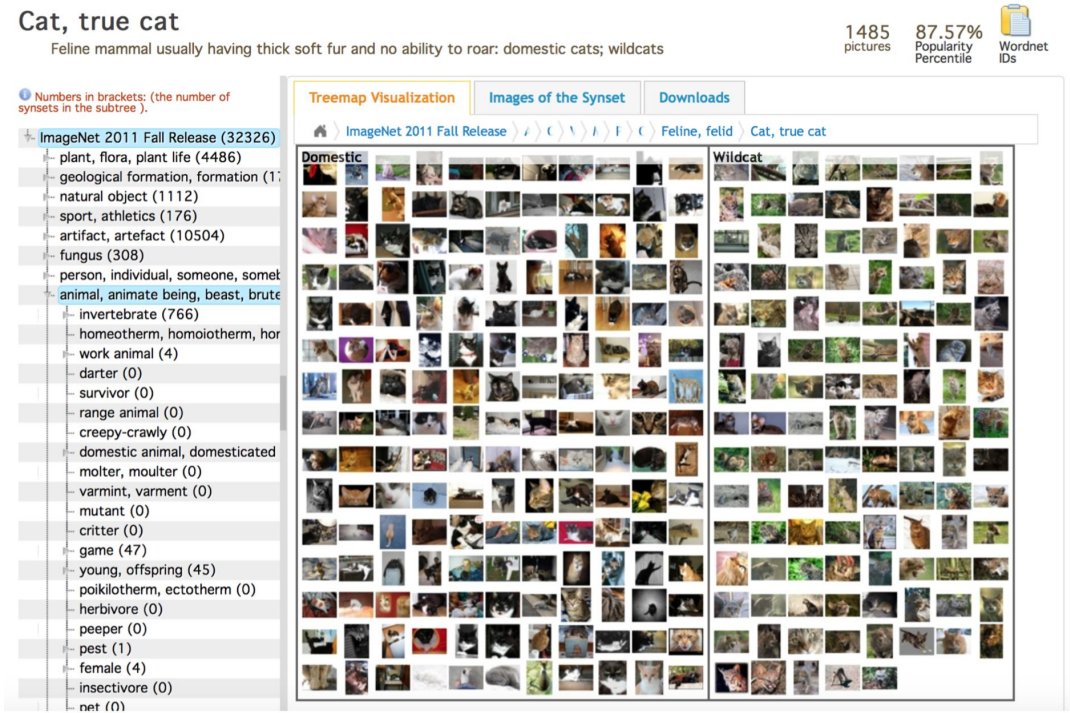


Deep Neural Networks to Deep Learning

= Learning Hierarchical Representations



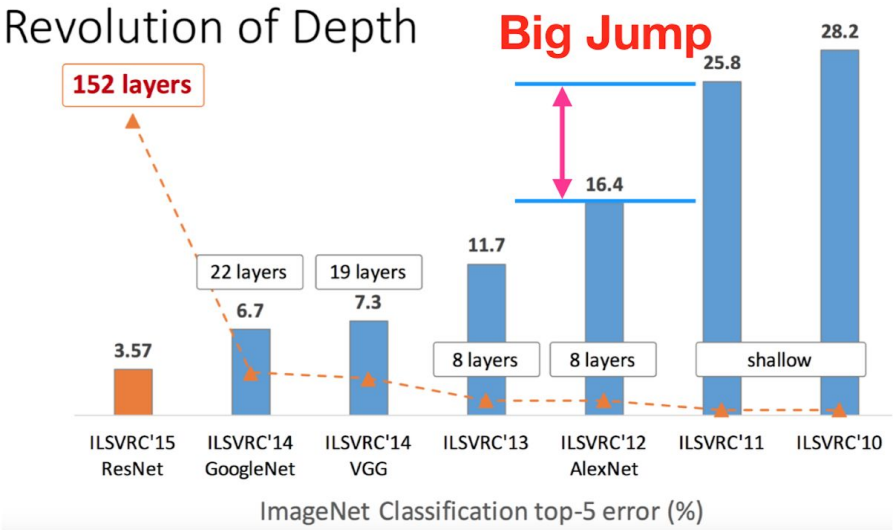
ImageNet Challenge

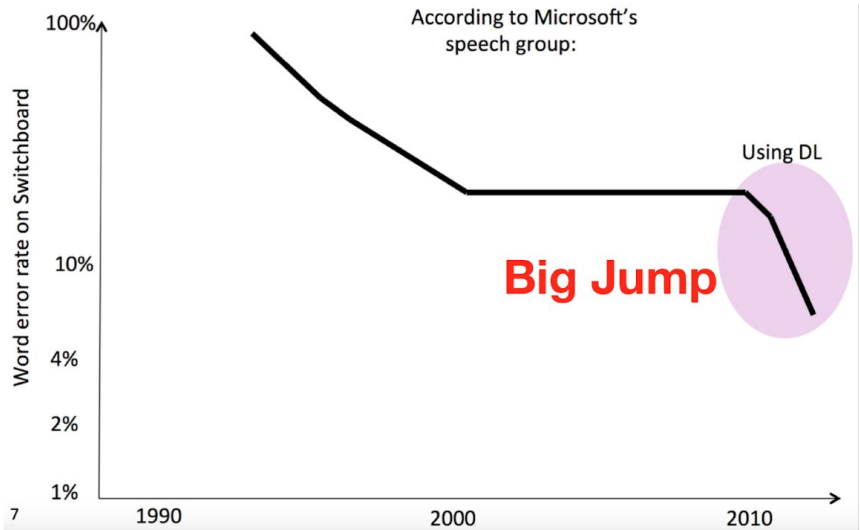


- ImageNet Large Scale Visual Recognition Competition (ILSVRC)
- Image Classification
  - 1000 classes
  - train data : 1.2M
  - validation data : 50K
  - test data : 100K

http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture04.pdf

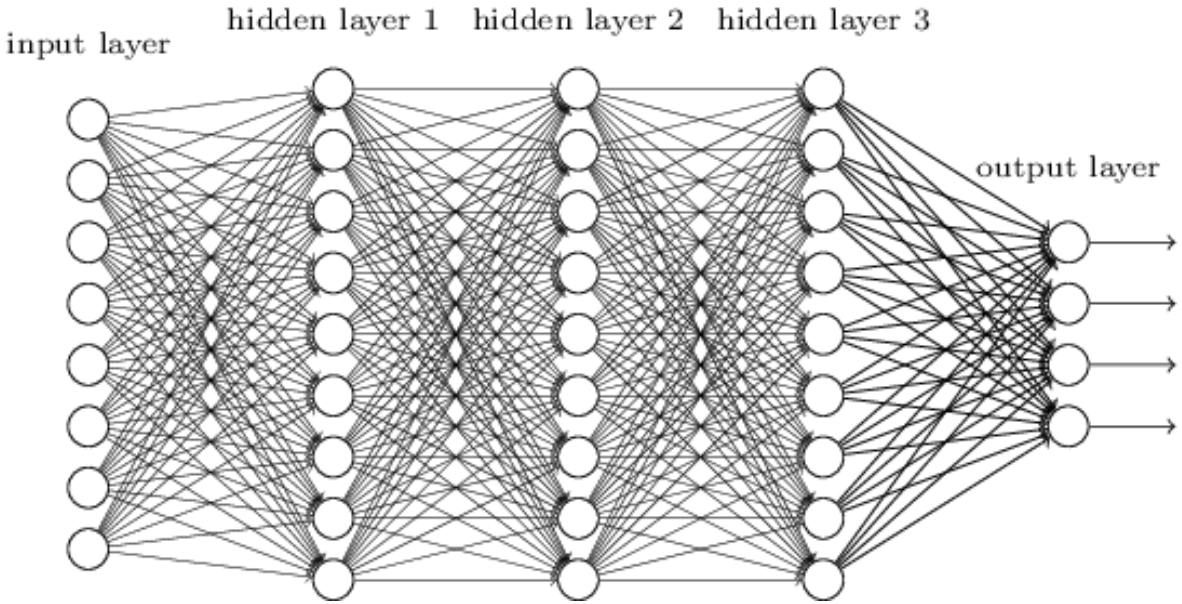
ImageNet Challenge





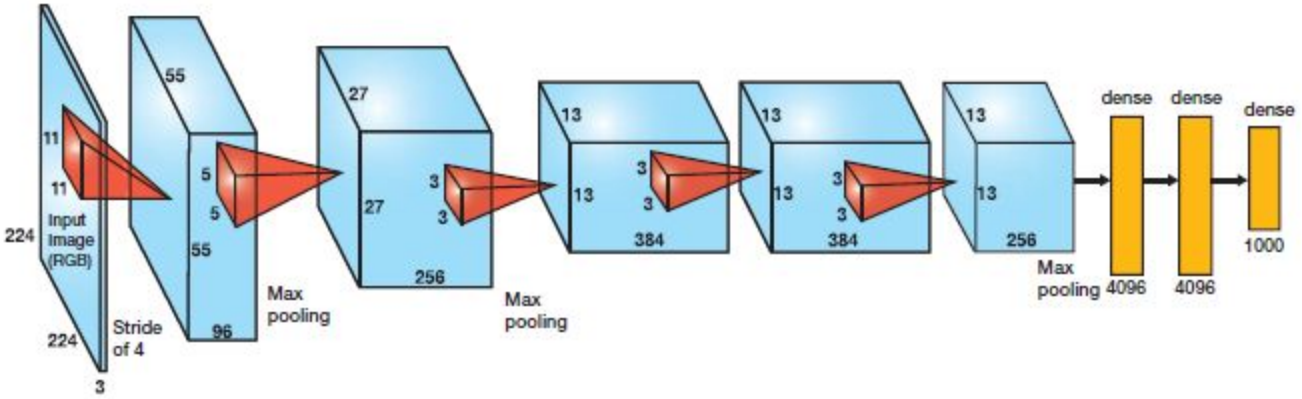
http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture04.pdf

Deep Learning Models



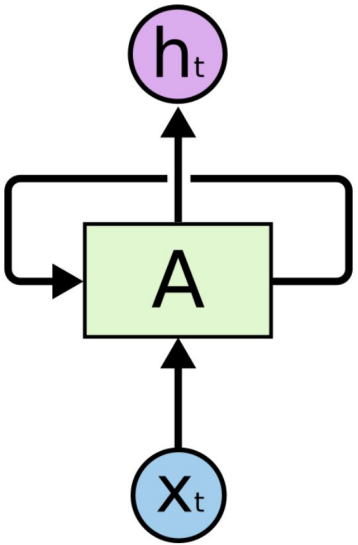
DNN

Deep Learning Models

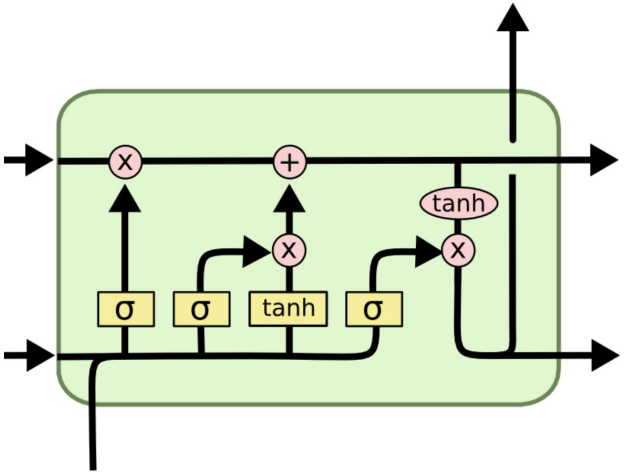


CNN

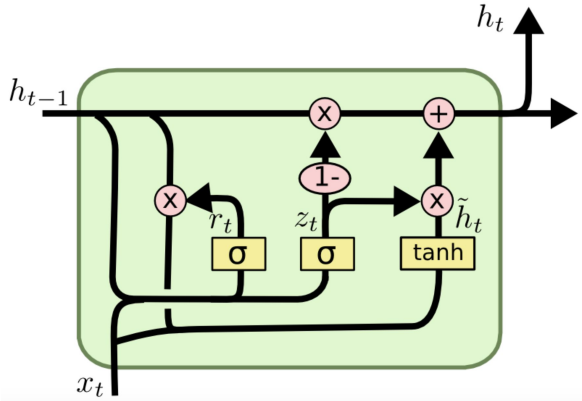
Deep Learning Models



RNN



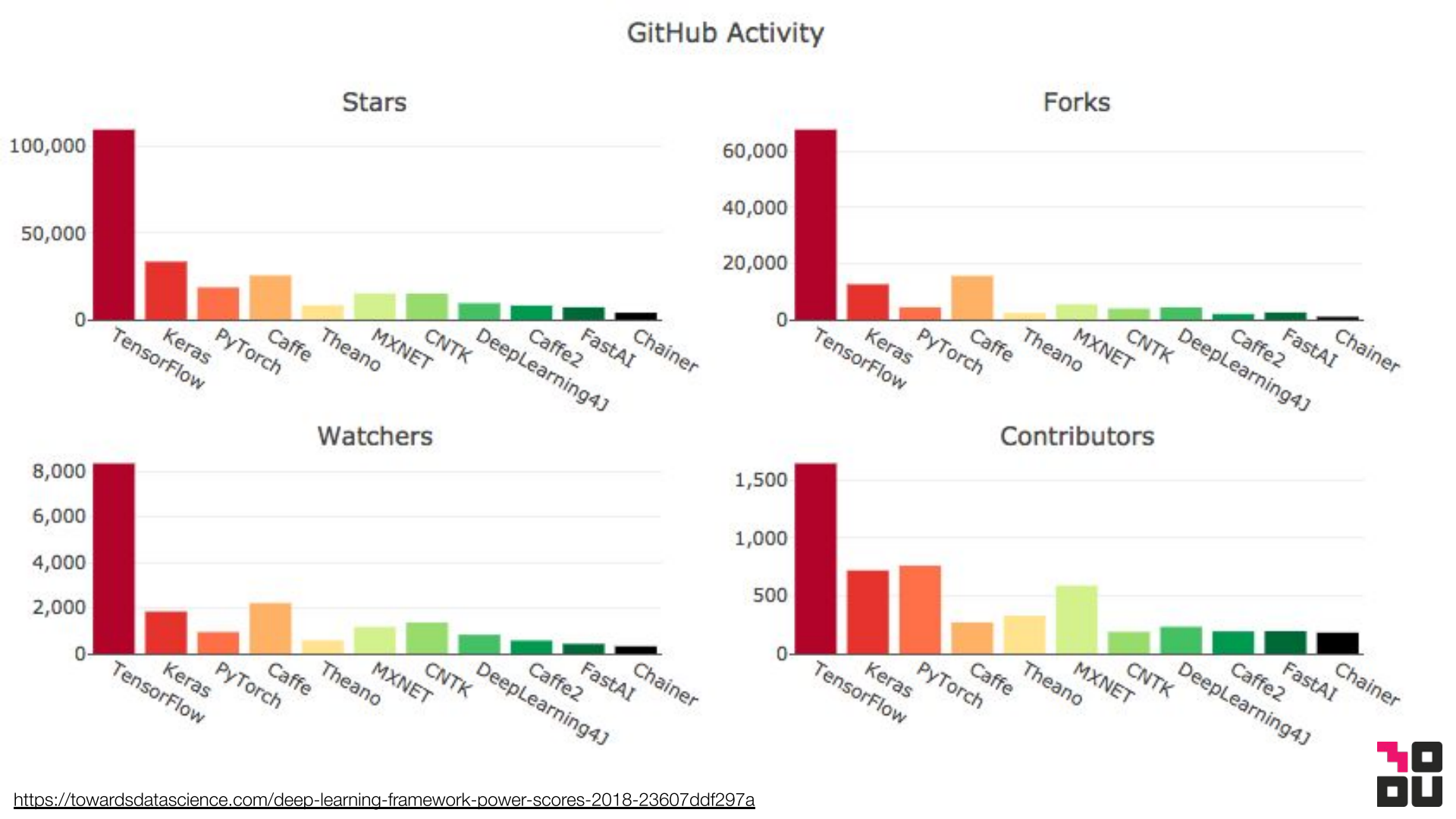
LSTM



GRU

Framework

- Why Tensorflow?



https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a

```python
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Run code now Try in Google's interactive notebook

```python
class MyModel(tf.keras.Model):
  def __init__(self):
    super(MyModel, self).__init__()
    self.conv1 = Conv2D(32, 3, activation='relu')
    self.flatten = Flatten()
    self.d1 = Dense(128, activation='relu')
    self.d2 = Dense(10, activation='softmax')

  def call(self, x):
    x = self.conv1(x)
    x = self.flatten(x)
    x = self.d1(x)
    return self.d2(x)
model = MyModel()

with tf.GradientTape() as tape:
  logits = model(images)
  loss_value = loss(logits, labels)
grads = tape.gradient(loss_value, model.trainable_variable
optimizer.apply_gradients(zip(grads, model.trainable_varia
```

Run code now Try in Google's interactive notebook

그전에

- Tensorflow, Keras and Pytorch
- tf.session and eager execution

다루고자 하는 주제

- Linear Regression
- Logistic Classification
- Multinomial Classification

다루고자 하는 주제

- Hypothesis - 가설 설정
- 어떤 Hypothesis 가 좋을까?
- Cost, Cost Function / Cost 함수
- Minimize Cost 비용 최소화

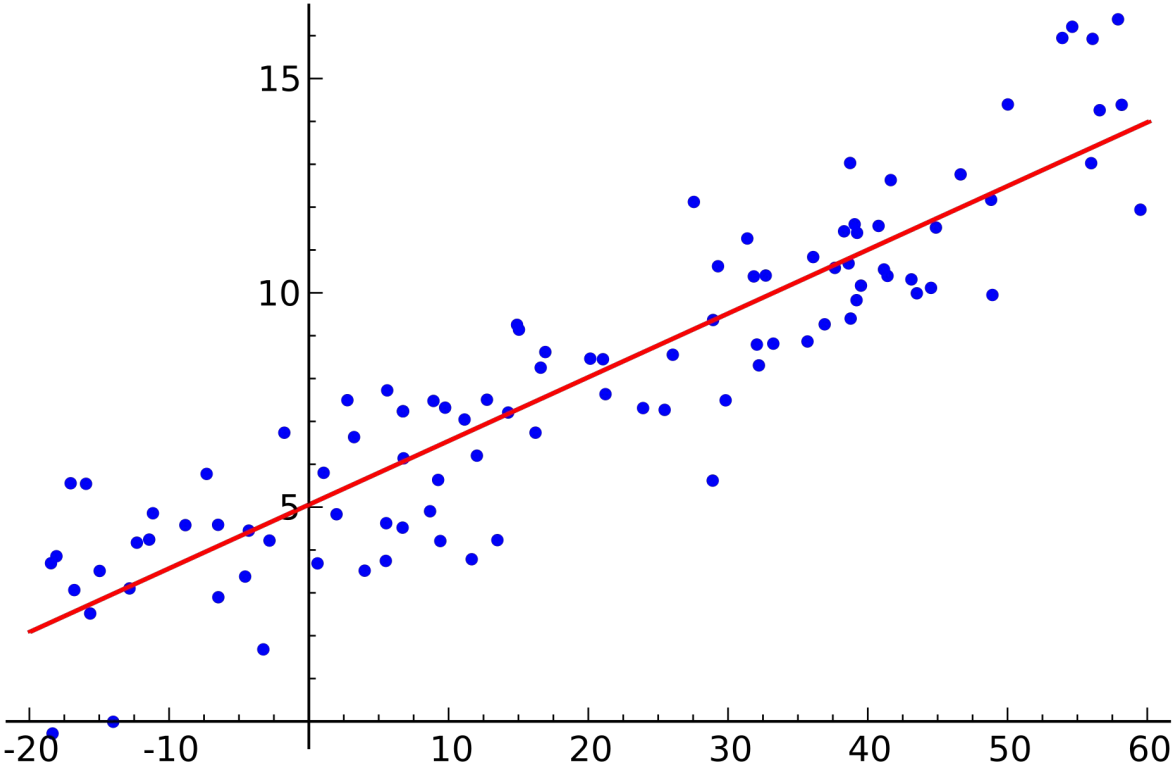
Regression ??

후퇴, 퇴보 되돌아가다?

“Regression toward the mean”

- 전체의 평균으로 되돌아 가는 것
- 어떤 큰 데이터나 작은 데이터가 나와도 전체적으로 봤을때 전체 평균으로 되돌아간다. (통계적 원리)

Linear Regression



$Y = Wx + b$

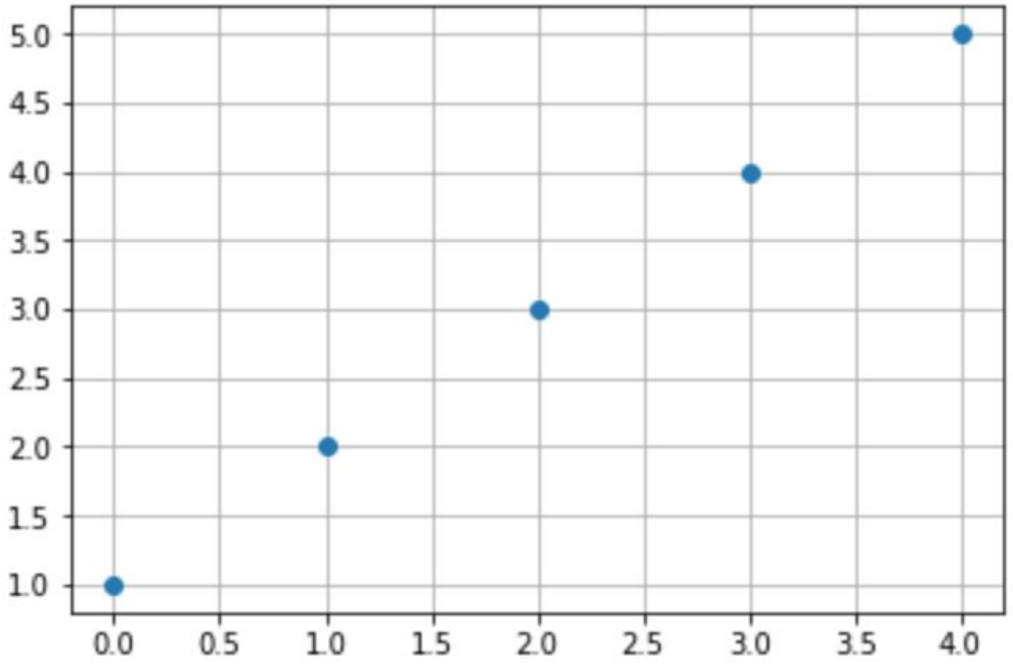
https://en.wikipedia.org/wiki/Linear_regression

Linear Regression

x	y
1	1
2	2
3	3
4	4
5	5

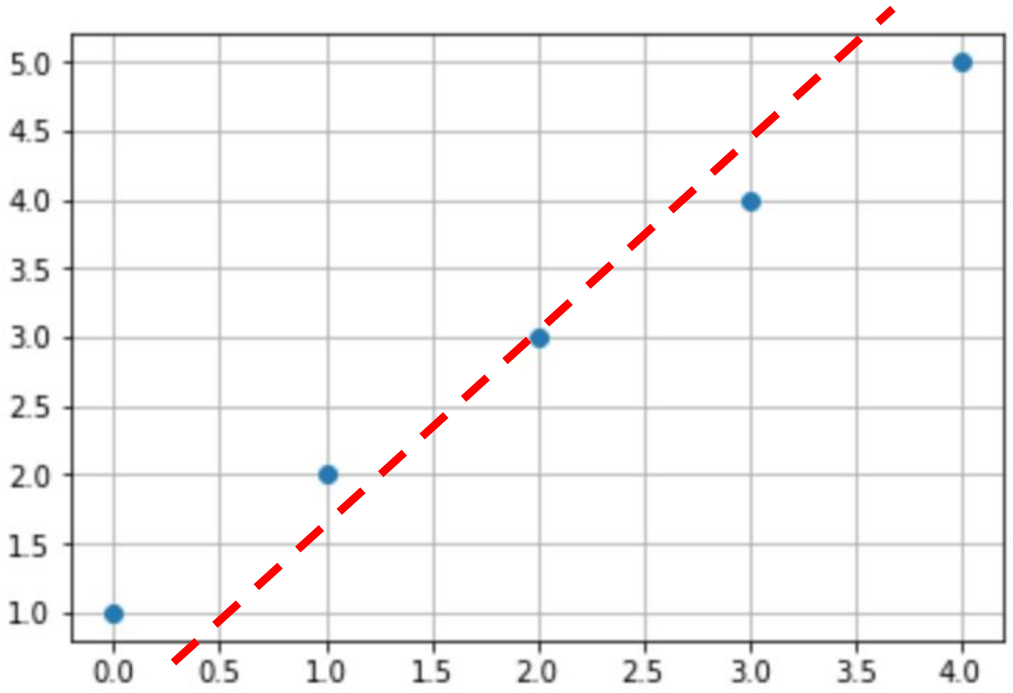
Linear Regression

x	y
1	1
2	2
3	3
4	4
5	5



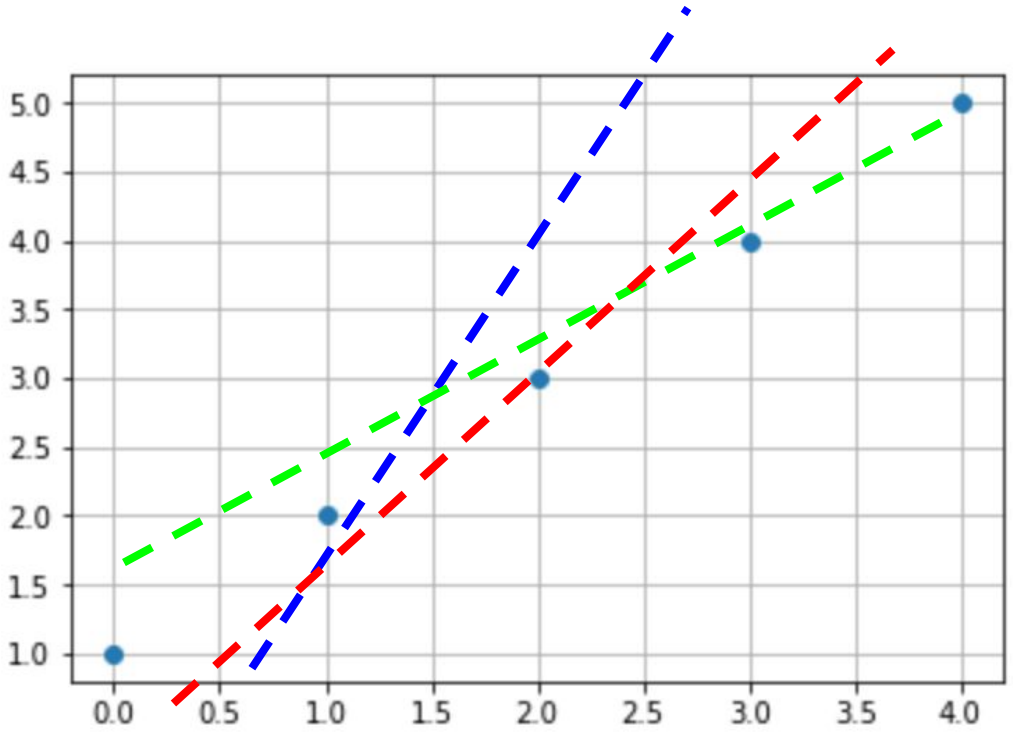
Linear Regression -> Hypothesis (Linear)

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



Hypothesis

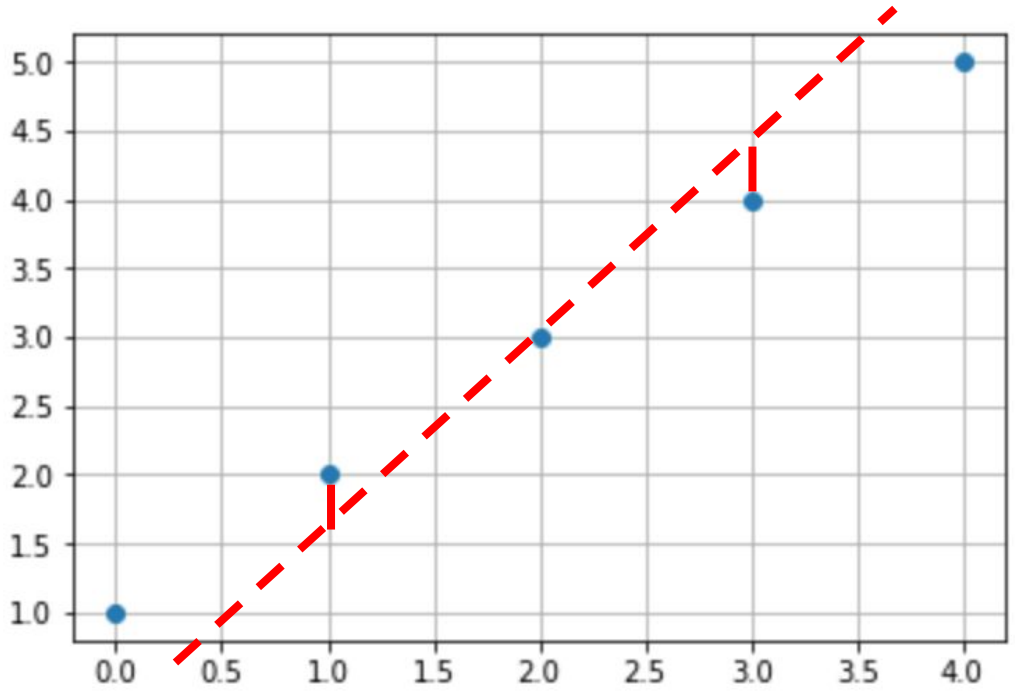
$$H(x) = Wx + b$$



Hypothesis -> Cost

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

$$H(x) - y$$

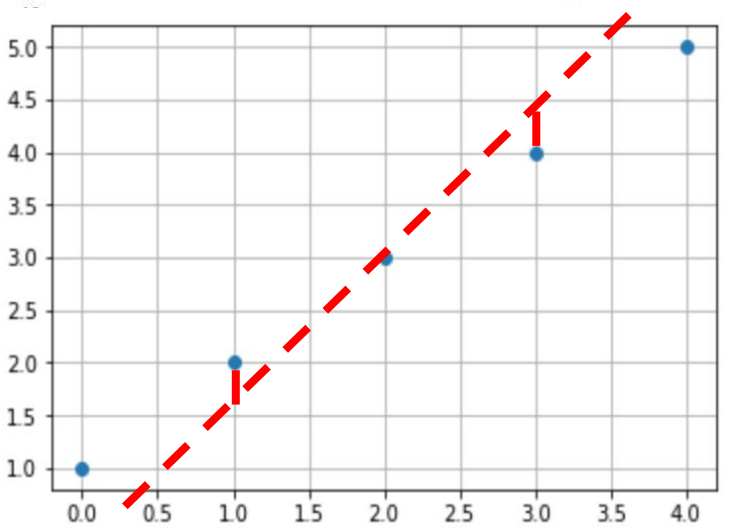


Cost function

$$\frac{(H(x_1)-y_1)^2+(H(x_2)-y_2)^2+(H(x_3)-y_3)^2}{3}$$

$$cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx_i - y_i)^2$$

$$H(x) - y$$



Cost function

$$cost(W) = \frac{1}{m} \sum_{i=1}^{m} (Wx_i - y_i)^2$$

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

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

Hypothesis and Cost

- 이번엔 Cost Function에 집중해봅시다.
- Cost 를 Minimize 한다는 것은 무엇일까요?

Hypothesis and Cost

- 그럼 W의 값을 어떻게 찾아가는지에 대해서 알아봅시다.

Hypothesis $H(x) = Wx + b$

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

Hypothesis $H(x) = Wx$

Cost $cost(W) = \frac{1}{m} \sum_{i=1}^{m} (Wx_i - y_i)^2$

Minimize cost

x	y
1	1
2	2
3	3

$$cost(W) = \frac{1}{m} \sum_{i=1}^{m} (Wx_i - y_i)^2$$

Minimize cost

W = 0, cost(W) = 4.67

$$\frac{1}{3}((0*1-1)^2+(0*2-2)^2+(0*3-3)^2))$$

W = 1, cost(W) = 0

$$\frac{1}{3}((1*1-1)^2+(1*2-2)^2+(1*3-3)^2))$$

W = 2, cost(W) = 4.67

$$\frac{1}{3}((2*1-1)^2+(2*2-2)^2+(2*3-3)^2))$$

W = 3, cost(W) = 18.67

$$\frac{1}{3}((3*1-1)^2+(3*2-2)^2+(3*3-3)^2))$$

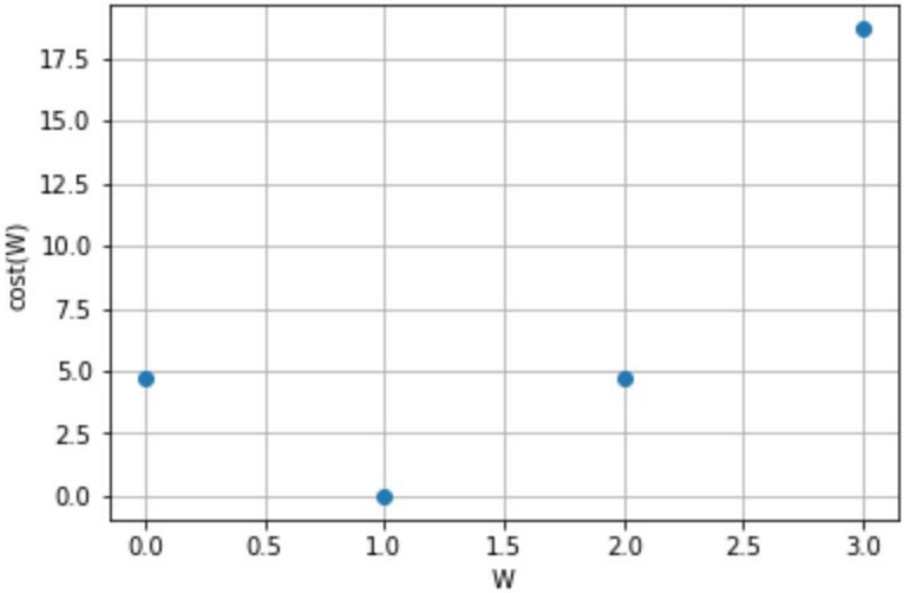
Minimize cost

W = 0, cost(W) = 4.67

W = 1, cost(W) = 0

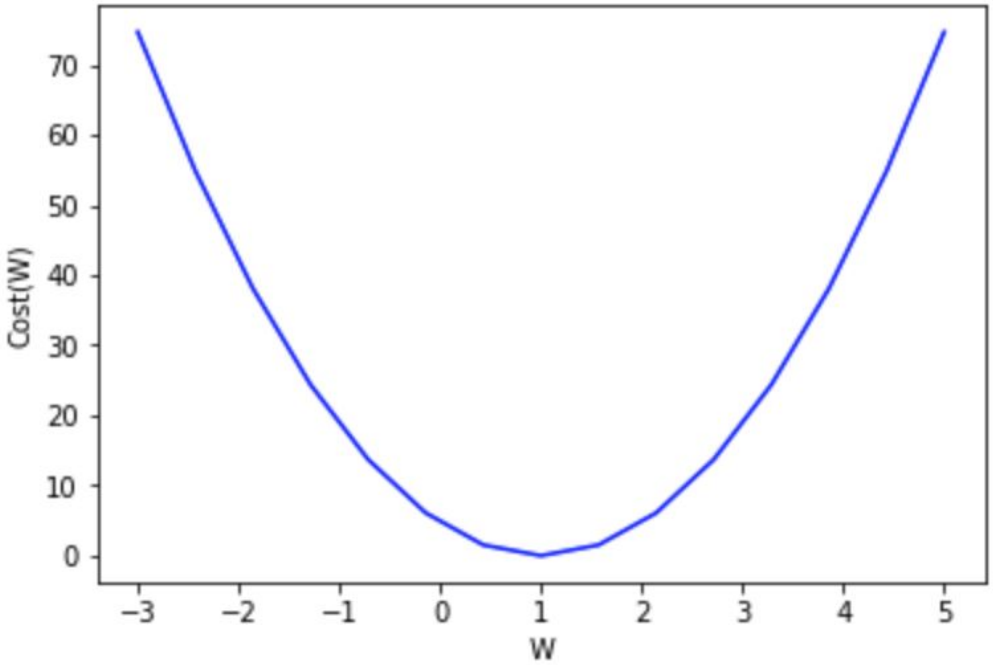
W = 2, cost(W) = 4.67

W = 3, cost(W) = 18.67



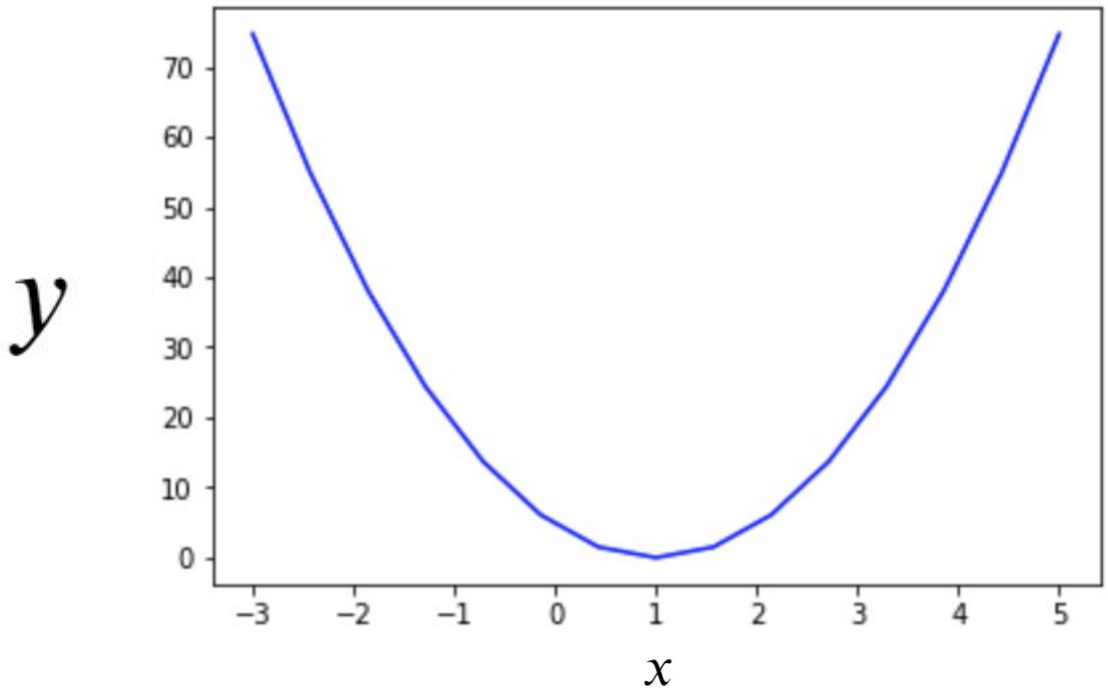
Minimize cost

$$cost(W) = \frac{1}{m} \sum_{i=1}^{m} (Wx_i - y_i)^2$$

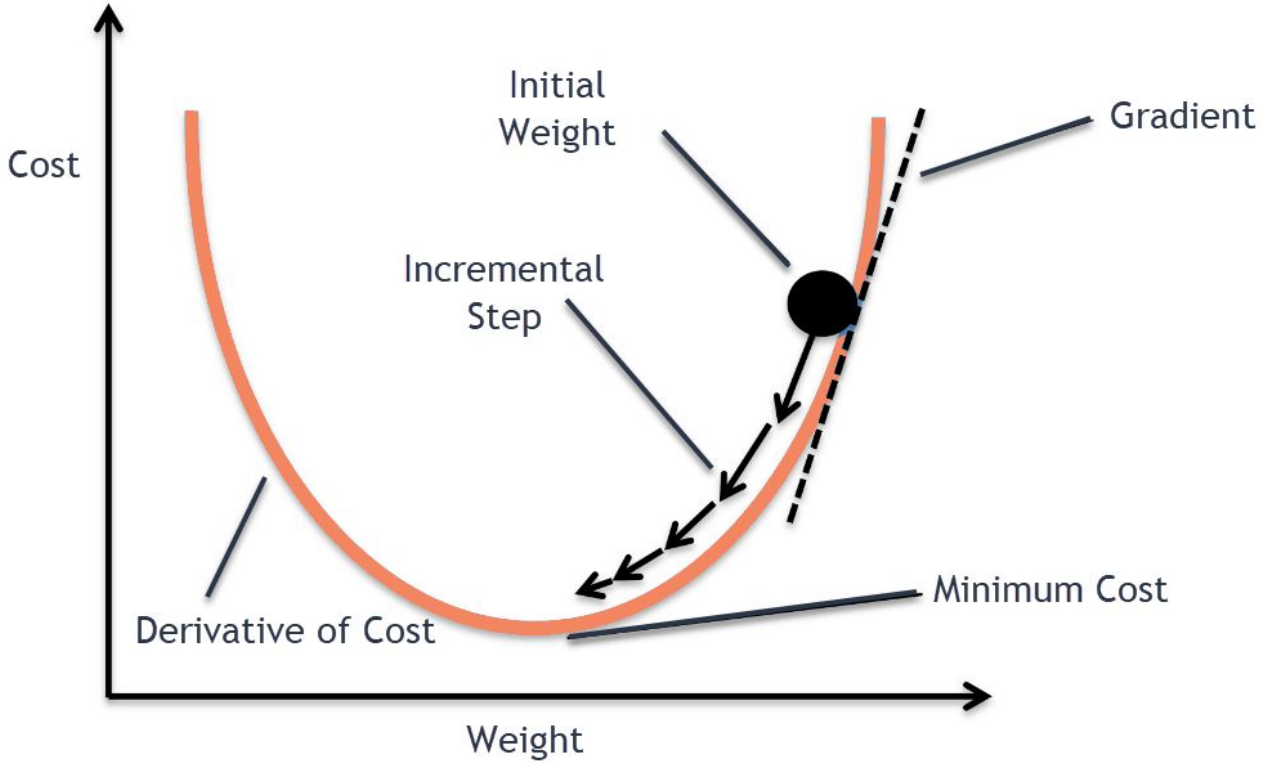


Minimize cost

$$y = (x - 1)^2$$



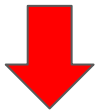
Gradient descent algorithm



https://medium.com/@divakar_239/stochastic-vs-batch-gradient-descent-8820568eada1

Gradient descent algorithm

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



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

Gradient descent algorithm

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

$$W := W - \alpha \frac{1}{2m} \sum_{i=1}^{m} 2(W(x_i) - y_i) x_i$$

$$W := W - \alpha \frac{1}{m} \sum_{i=1}^{m} (W(x_i) - y_i) x_i$$

Gradient descent algorithm

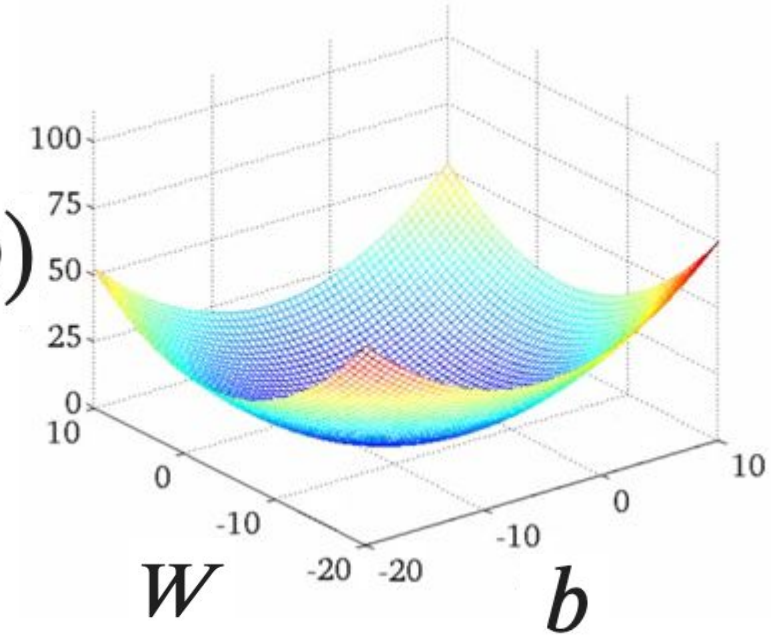
$$cost(W, b) = \frac{1}{2m} \sum_{i=1}^{m} \left(H(x_i) - y_i \right)^2$$

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

Gradient descent algorithm

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

$cost(W, b)$



https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/

Let’s do it

ex) 다음 값을 예측해봅시다.

x	y
10	90
9	80
4	30
3	20

Multi-variable Linear regression

X1	X2	X3	Y
96	91	99	194
88	85	82	181
78	77	73	177
67	66	61	164
55	51	53	157

Hypothesis

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

?

Hypothesis

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

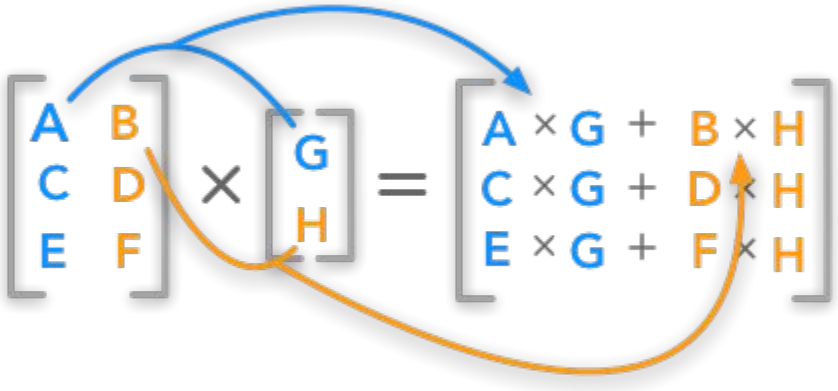
$$H(x_1, x_2, x_3) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

Hypothesis

$$w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots + w_n x_n$$

Hypothesis - Matrix dot product

$$w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots + w_n x_n$$



https://hadrienj.github.io/posts/Deep-Learning-Book-Series-2.2-Multiplying-Matrices-and-Vectors/

Hypothesis - Matrix dot product

$$w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots + w_n x_n$$

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1 w_1 + x_2 w_2 + x_3 w_3 \end{pmatrix}$$

$$H(X) = XW$$

Multi-variable Linear regression

X1	X2	X3	Y
96	91	99	194
88	85	82	181
78	77	73	177
67	66	61	164
55	51	53	157

$$w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots + w_n x_n$$

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1 w_1 + x_2 w_2 + x_3 w_3 \end{pmatrix}$$

Multi-variable Linear regression

X1	X2	X3	Y
96	91	99	194
88	85	82	181
78	77	73	177
67	66	61	164
55	51	53	157

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

$$H(X) = XW$$

Multi-variable Linear regression

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

(5 x 3) (3 x 1) (5 x 1)

Multi-variable Linear regression

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{pmatrix} \cdot \boxed{?} = \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\ x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \end{pmatrix}$$

Multi-variable Linear regression

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\ x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \end{pmatrix}$$

(2 x 3) (3 x 1) (2 x 1)

Let’s do it

ex) 직접 Score Dataset을 이용하여 예측해 봅시다! (총 25개 데이터)

	A	B	C	D
1	73	80	75	152
2	93	88	93	185
3	89	91	90	180
4	96	98	100	196
5	73	66	70	142
6	53	46	55	101
7	69	74	77	149
8	47	56	60	115
9	87	79	90	175
10	79	70	88	164

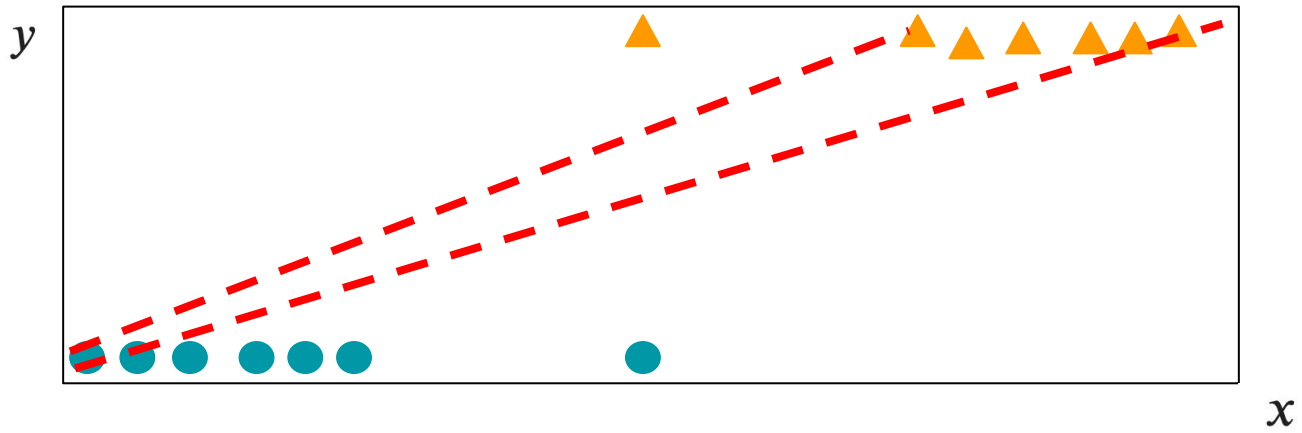
Logistic Regression / Classification

- Classification ?
- Test : Pass / Fail

  위조지폐 : Real / Fake

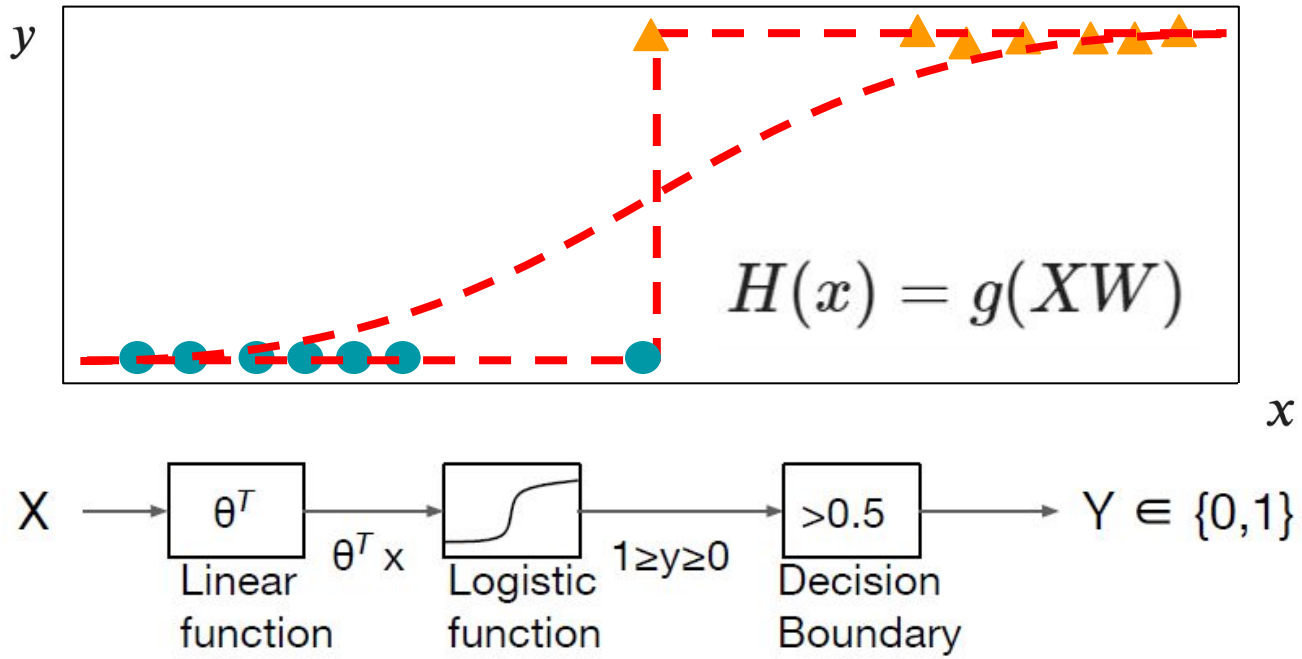
- 위 문제들의 공통점은 무엇일까요?

Hypothesis Representation



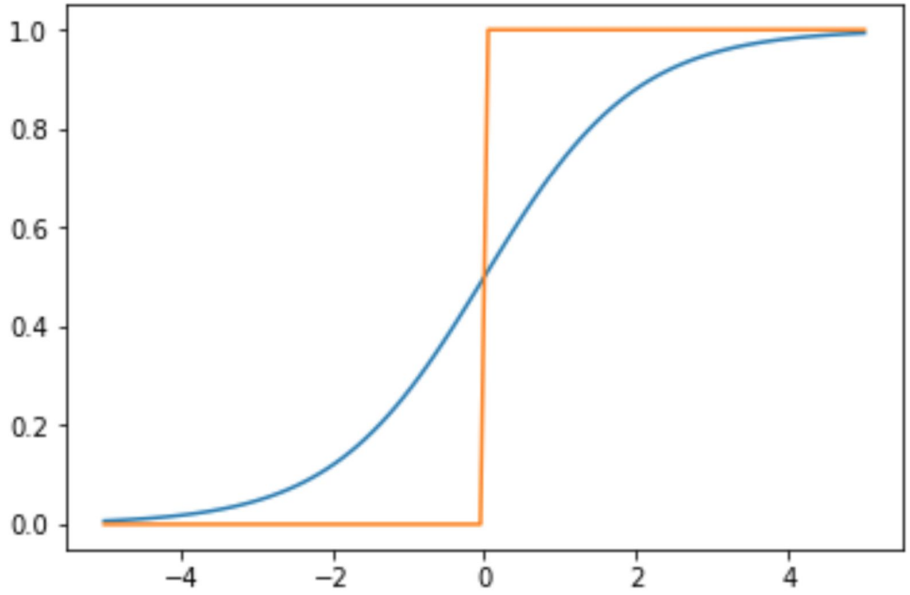
$$H(X) = XW$$

Hypothesis Representation

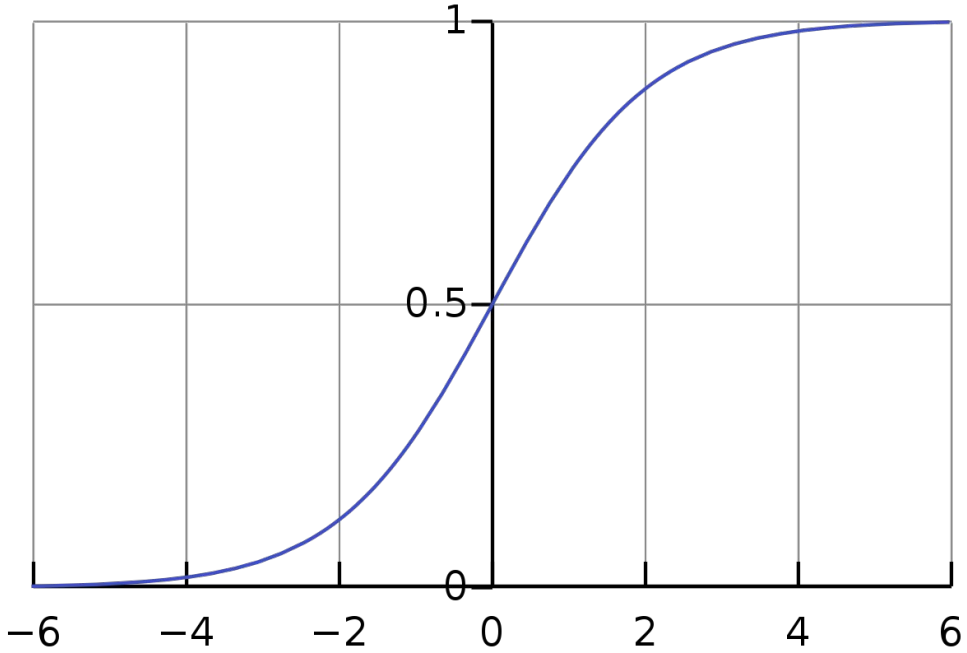


Hypothesis Representation

- 비슷한 점
  - Sigmoid function은 입력값이 크면 1에 가깝고 작으면 0에 가까워져서 step function과 비슷하다
  - 둘 다 [0, 1] 사이의 값을 갖는다
- 차이점
  - Sigmoid function은 부드러운 곡선을 갖는 (smoothed) step function이다
  - Sigmoid의 매끄러움 때문에 작은 가중치 변화로 작은 출력값의 변화가 생긴다



Sigmoid (Logistic) Function



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

$$g(XW) = \frac{1}{1 + e^{-XW}} = \frac{e^{XW}}{e^{XW} + 1}$$

https://en.wikipedia.org/wiki/Sigmoid_function

Cost Function (Loss Function)

- Our Cost Function (이었던 것)

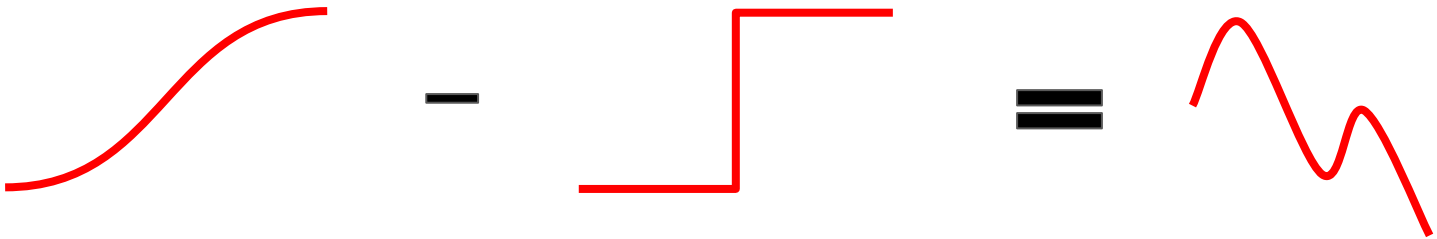
$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} \left(H(x_i) - y_i \right)^2$$

https://en.wikipedia.org/wiki/Sigmoid_function

Cost Function (Loss Function)

- Our Cost Function (이었던 것)

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

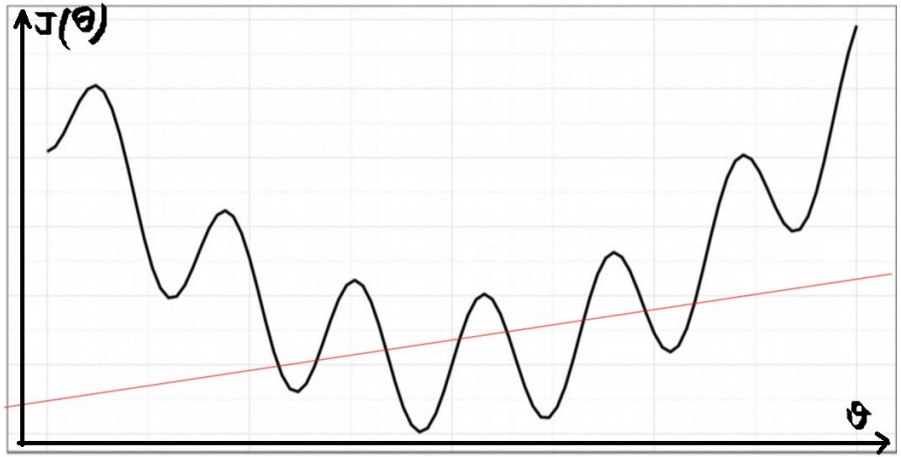


https://en.wikipedia.org/wiki/Sigmoid_function

Cost Function (Loss Function)

- Our Cost Function (이었던 것)

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



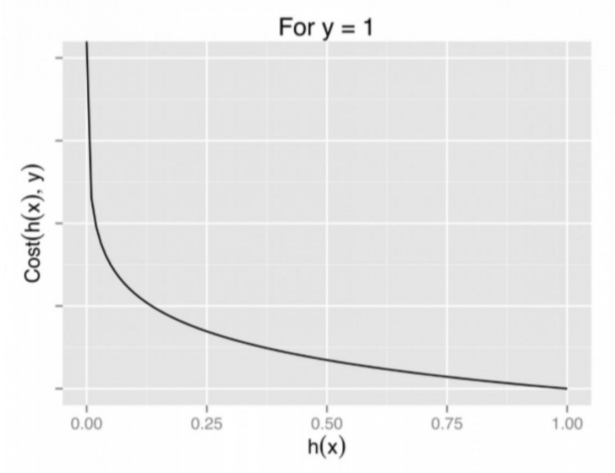
Non-convex function

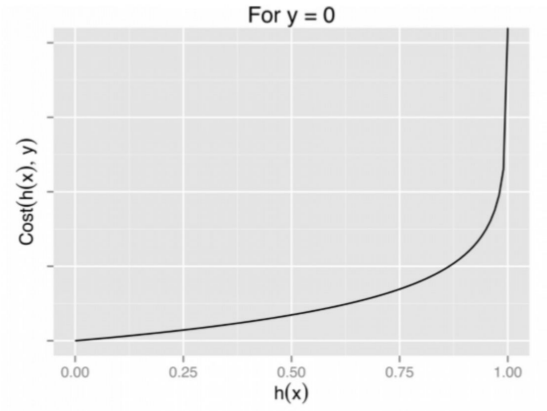
https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

Cost Function (Loss Function)

- 새 Cost Function

$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

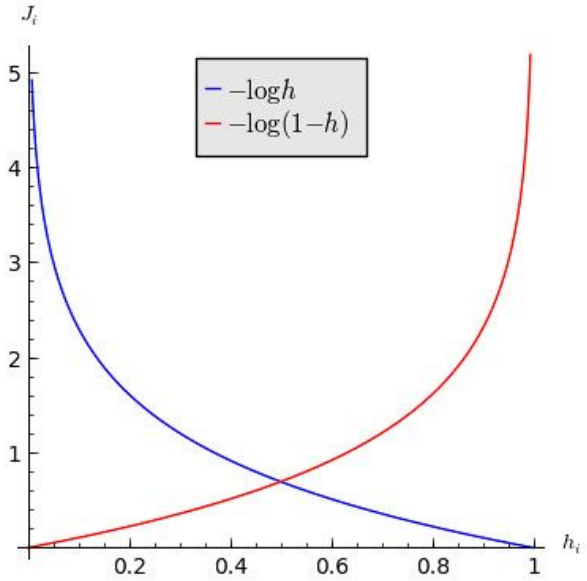




https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

Cost Function (Loss Function)

- 새로운 Cost Function



$$cost(H(x), y) = -ylog(H(x)) - (1 - y)log(1 - H(x))$$

http://www.kaiyin.co.vu/2014/04/logistic-regression-with-gradient_7.html

Gradient Descents

$$cost(H(x), y) = -ylog(H(x)) - (1 - y)log(1 - H(x))$$

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

W 업데이트를 반복

Gradient Descents

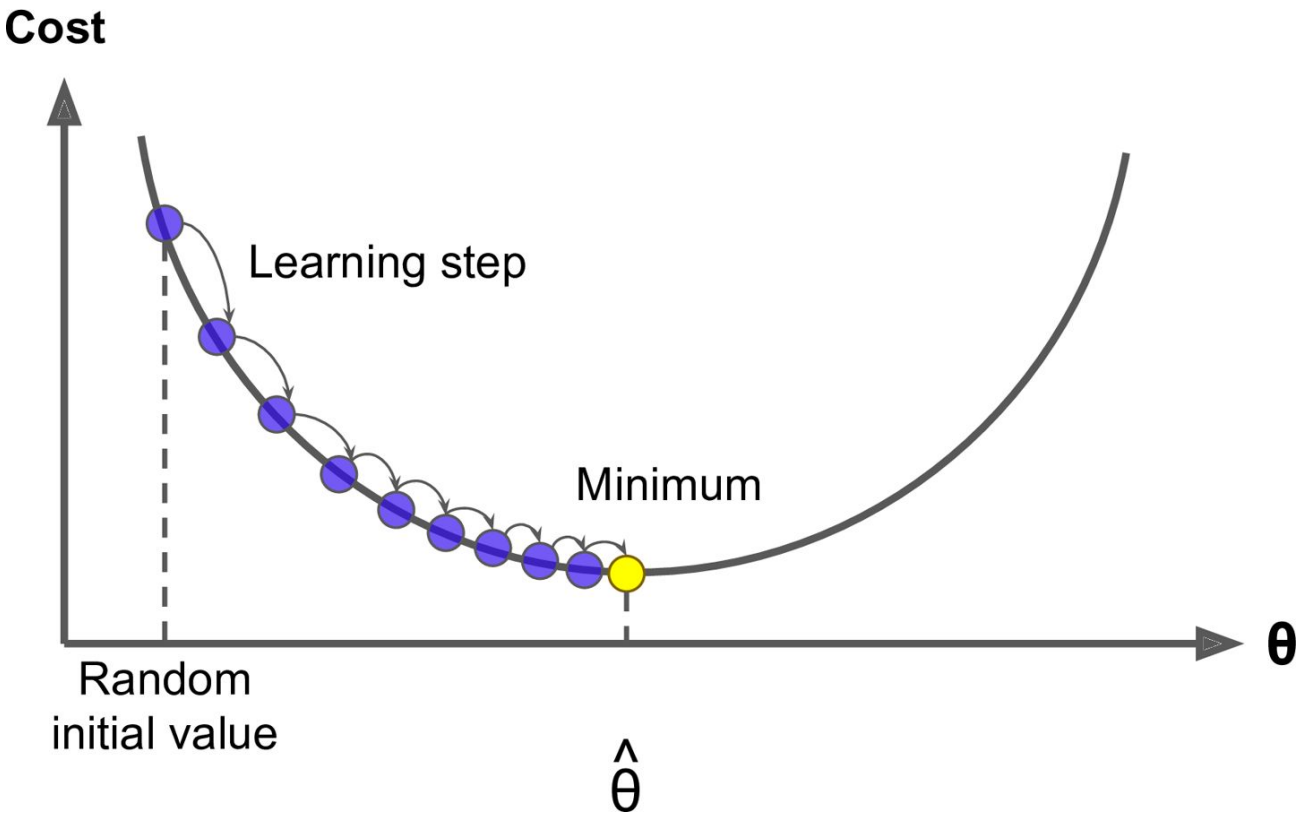
$$cost(H(x), y) = -ylog(H(x)) - (1 - y)log(1 - H(x))$$

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

W 업데이트를 반복

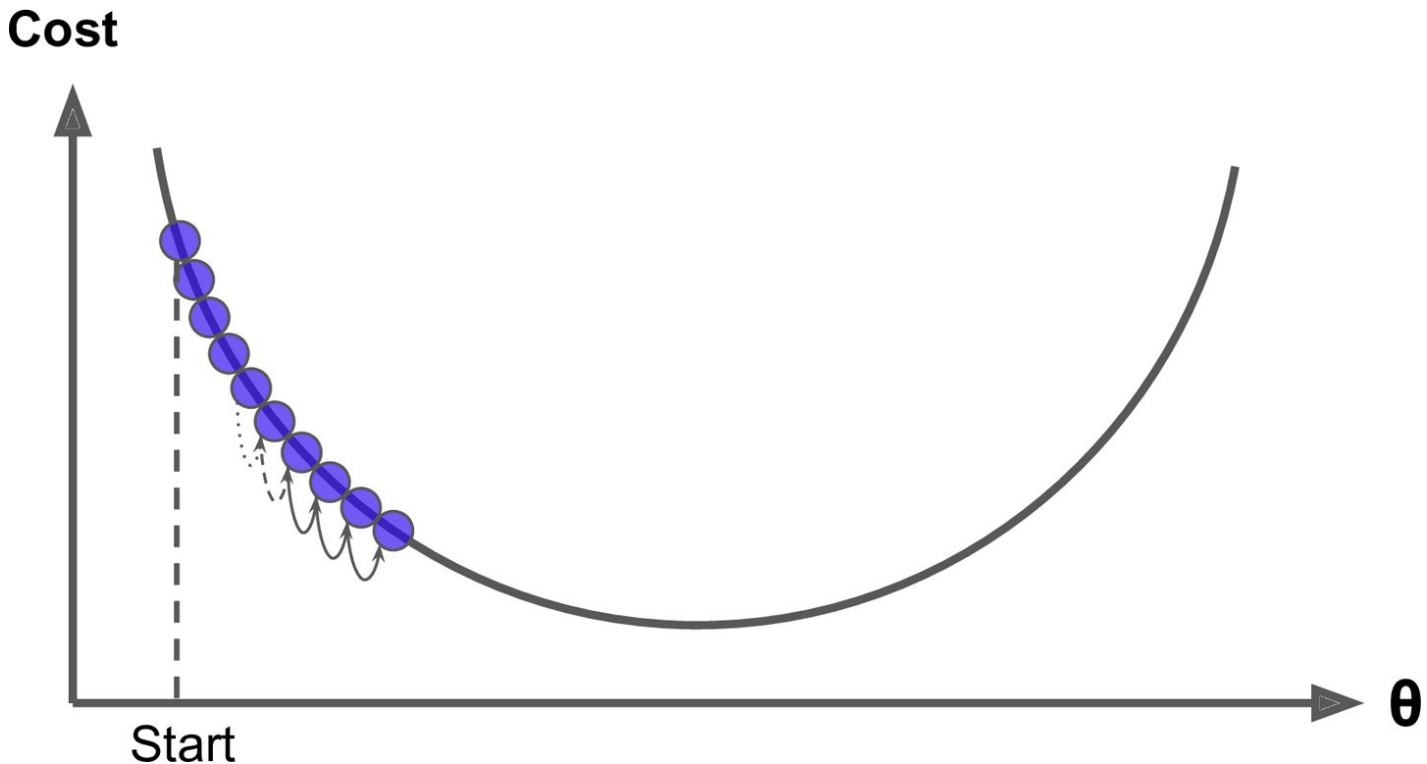
?

Gradient Descents - Good



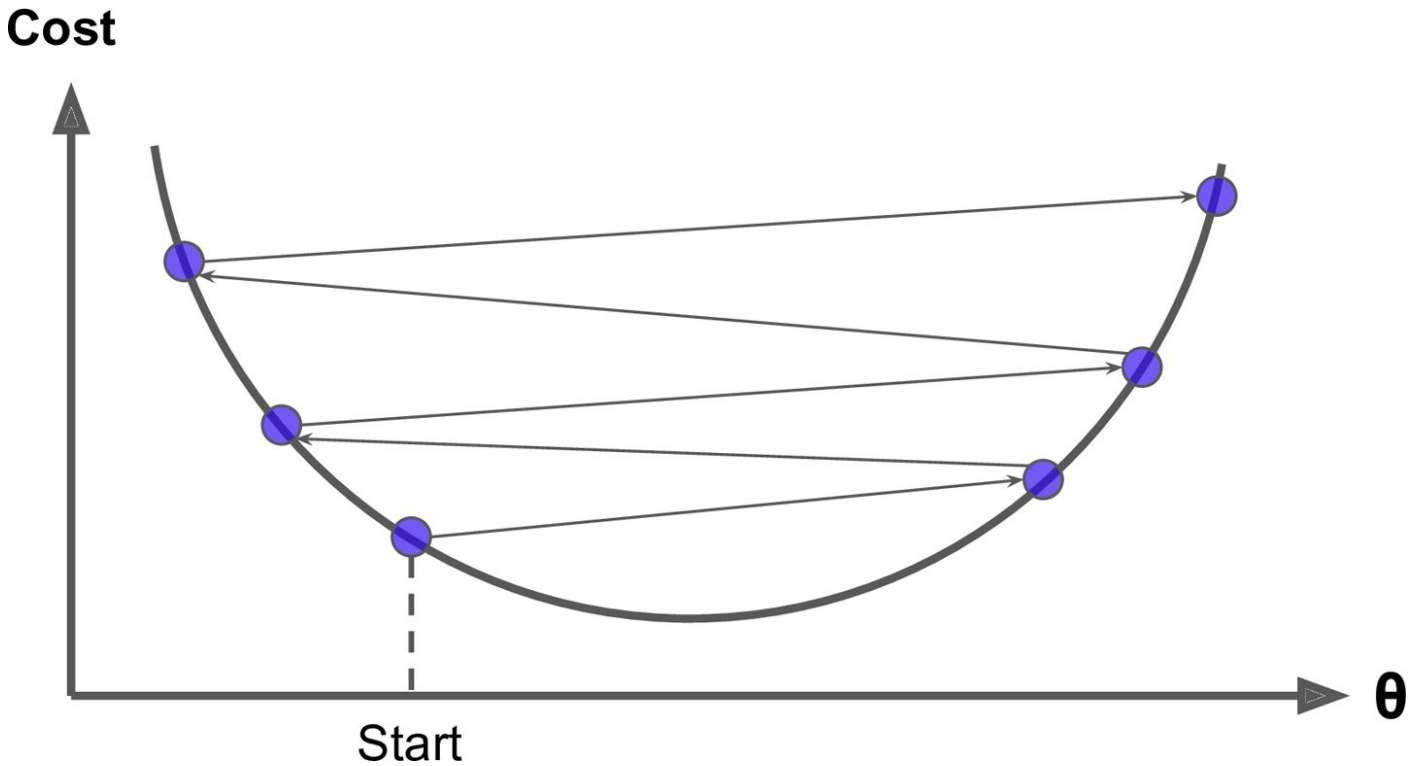
A. Géron, Hands-on Machine Learning with Scikit-Learn & TensorFlow, chap 1, p. 111

Gradient Descents - too small



A. Géron, Hands-on Machine Learning with Scikit-Learn & TensorFlow, chap 1, p. 112

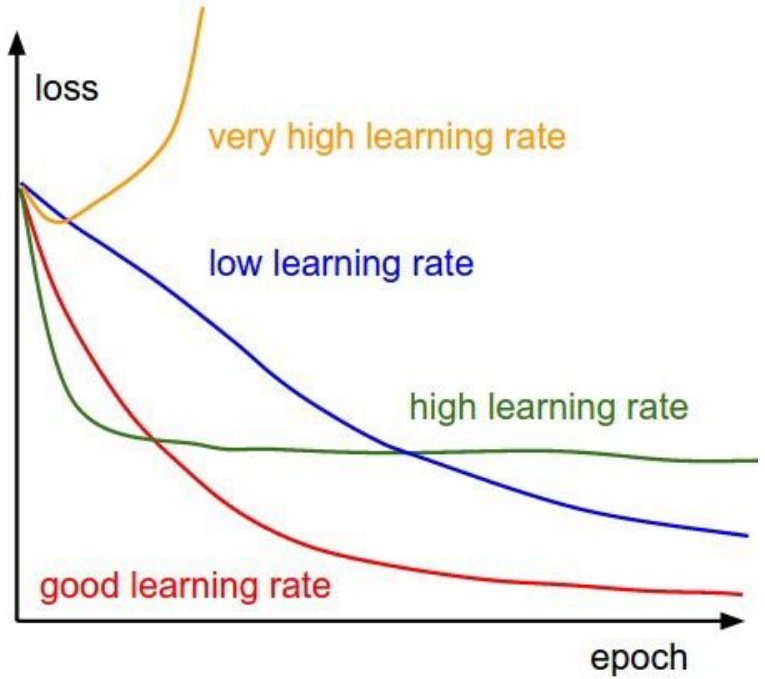
Gradient Descents - too large



A. Géron, Hands-on Machine Learning with Scikit-Learn & TensorFlow, chap 1, p. 112

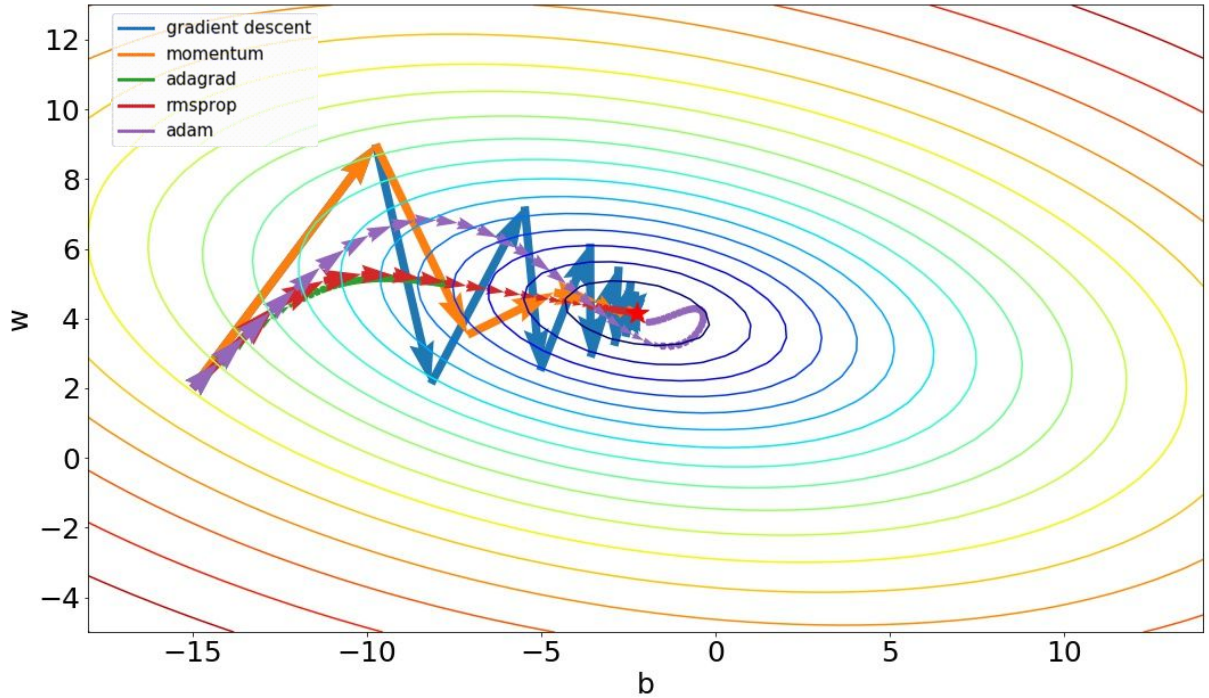
Learning Rate

- 어떤 learning rate가 가장 좋은가?
- learning rate를 계속 유지하는게 좋은가?
  - 적절하게 learning rate를 줄여보자



A. Géron, Hands-on Machine Learning with Scikit-Learn & TensorFlow, chap 1, p. 112

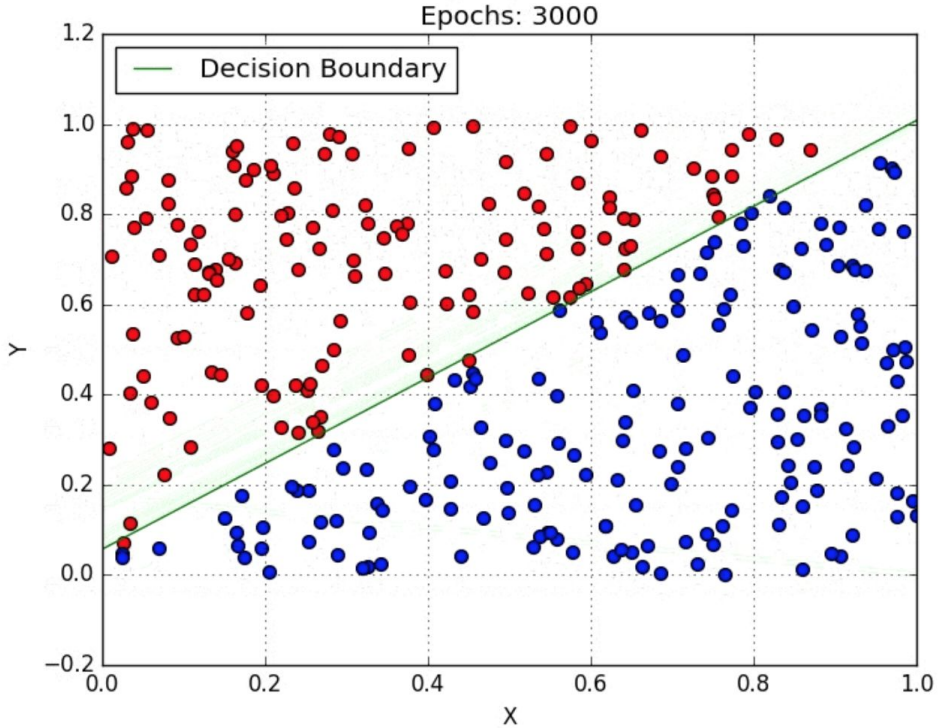
Optimizer



Optimization of Linear Regression using Various Gradient Descent Algorithms

https://github.com/ilguyi/optimizers.numpy

Decision boundary



https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

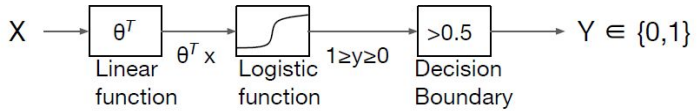
Let’s do it

ex) 직접 Dataset을 이용하여 당뇨 진단을 예측해 봅시다! (총 759개 데이터)

	C	D	E	F	G	H	I
1	0.180328	-0.292929	0	0.00149028	-0.53117	-0.0333333	0
2	0.0819672	-0.414141	0	-0.207153	-0.766866	-0.666667	1
3	0.0491803	0	0	-0.305514	-0.492741	-0.633333	0
4	0.0819672	-0.535354	-0.777778	-0.162444	-0.923997	0	1
5	-0.344262	-0.292929	-0.602837	0.28465	0.887276	-0.6	0
6	0.213115	0	0	-0.23696	-0.894962	-0.7	1
7	-0.180328	-0.353535	-0.791962	-0.0760059	-0.854825	-0.833333	0
8	0	0	0	0.052161	-0.952178	-0.733333	1
9	0.147541	-0.0909091	0.283688	-0.0909091	-0.931682	0.0666667	0
10	0.57377	0	0	0	-0.868488	0.1	0
11	0.508197	0	0	0.120715	-0.903501	-0.7	1
12	0.213115	0	0	0.132638	-0.608027	-0.566667	0
13	0.311475	0	0	-0.19225	0.163962	0.2	1
14	-0.0163934	-0.535354	1	-0.102832	-0.726729	0.266667	0
15	0	0	0	-0.105812	-0.653288	-0.633333	0
16	0.377049	-0.0505051	-0.456265	0.365127	-0.596072	-0.666667	0
17	0.213115	0	0	-0.117735	-0.849701	-0.666667	0

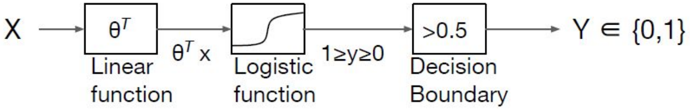
Multinomial classification

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1 w_1 + x_2 w_2 + x_3 w_3 \end{pmatrix}$$

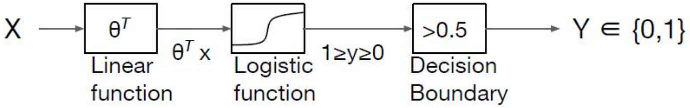


Multinomial classification

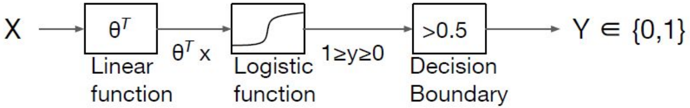
$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1 w_1 + x_2 w_2 + x_3 w_3 \end{pmatrix}$$



$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1 w_1 + x_2 w_2 + x_3 w_3 \end{pmatrix}$$



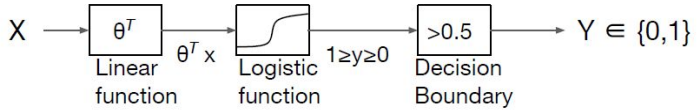
$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1 w_1 + x_2 w_2 + x_3 w_3 \end{pmatrix}$$

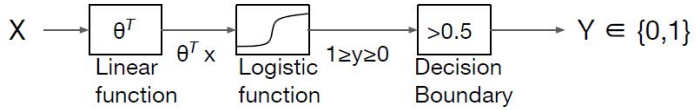


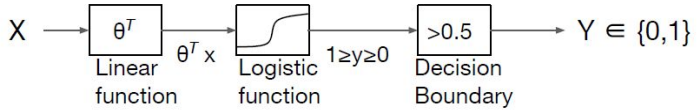
Multinomial classification

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\ x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \\ x_{31}w_1 + x_{32}w_2 + x_{33}w_3 \end{pmatrix}$$

?





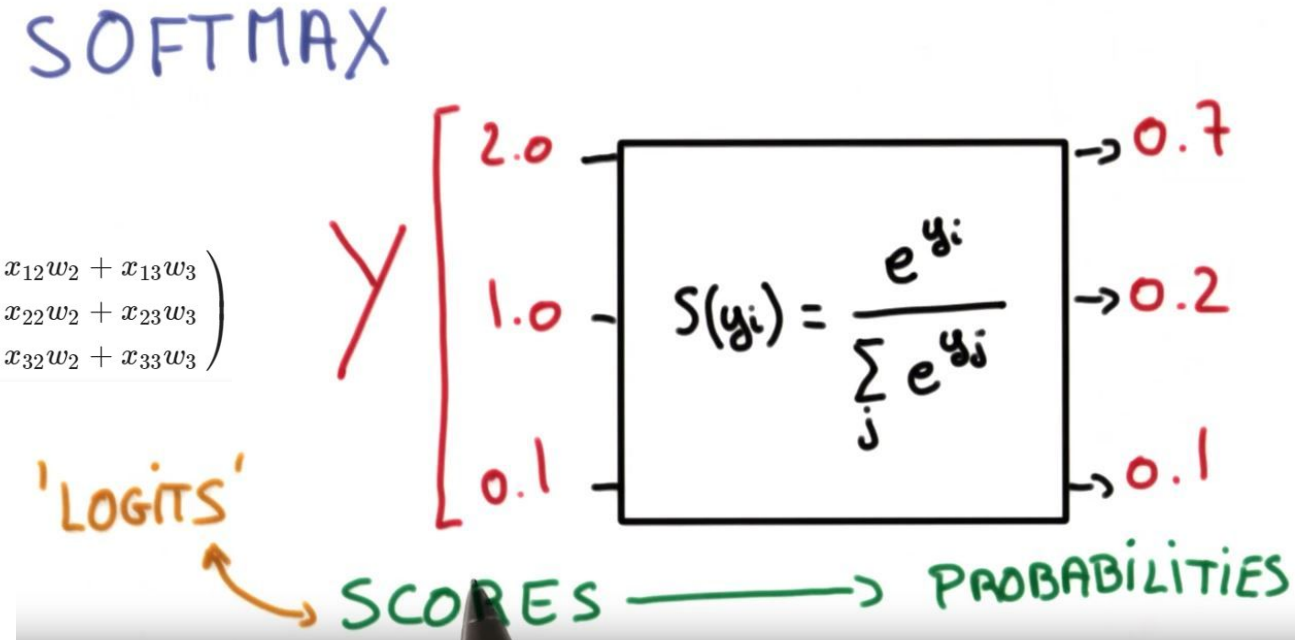


Multinomial classification

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\ x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \\ x_{31}w_1 + x_{32}w_2 + x_{33}w_3 \end{pmatrix} \rightarrow \begin{pmatrix} 2.0 \\ 1.0 \\ 0.1 \end{pmatrix} \rightarrow ? \begin{pmatrix} 0.7 \\ 0.2 \\ 0.1 \end{pmatrix}$$

Softmax function

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\ x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \\ x_{31}w_1 + x_{32}w_2 + x_{33}w_3 \end{pmatrix}$$



https://ireneli.eu/2016/01/29/tensorflow-02-play-with-mnist/

Softmax function

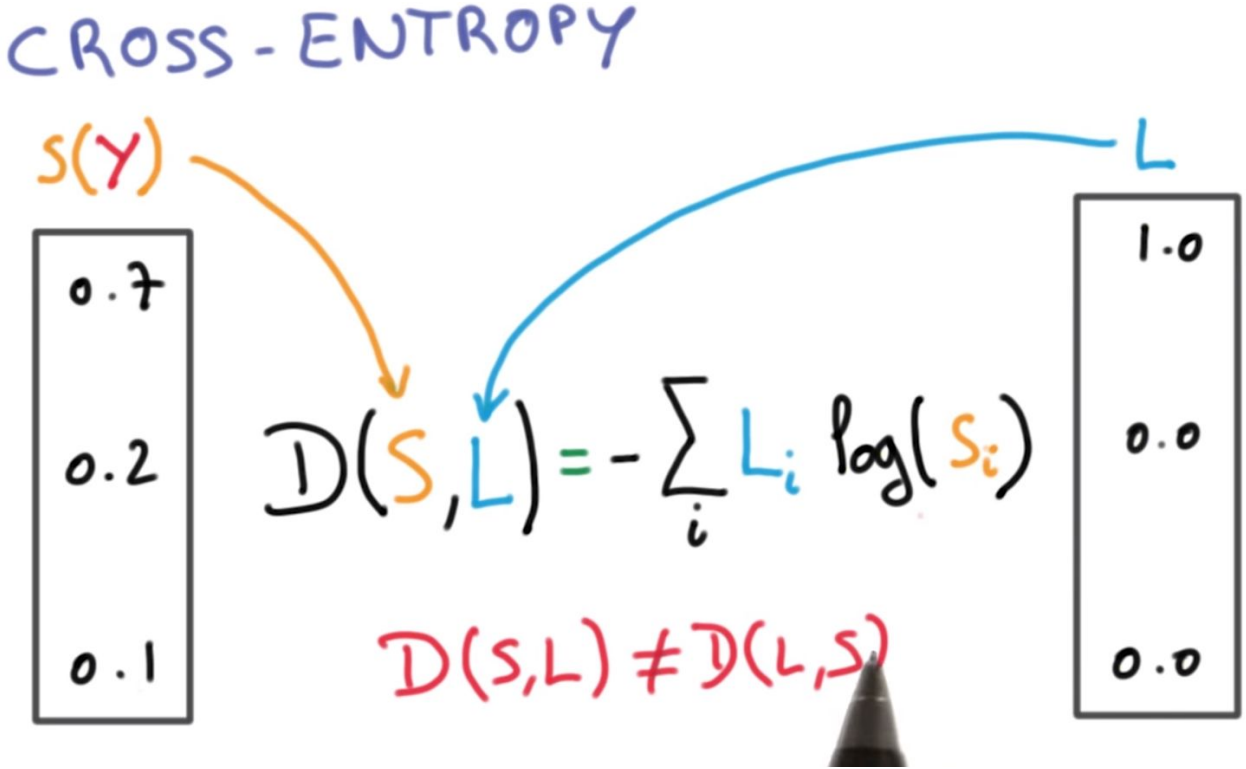
- 일반적으로 분류 문제에서 맨 마지막 레이어는 softmax 함수를 많이 이용한다
- Softmax 수식

$$a_i = \frac{e^{z_i}}{\sum_k e^{z_k}}, \text{ where } z_i = \sum_j W_{ij} x_j + b_i$$

- Softmax 성질
  - Sotfmax 함수의 출력은 확률 분포로 생각할 수 있다. 총합은 1 ($\sum_i a_i = 1$)
  - 원래 값의 대소관계가 유지 ($z_i < z_j \rightarrow a_i < a_j$)

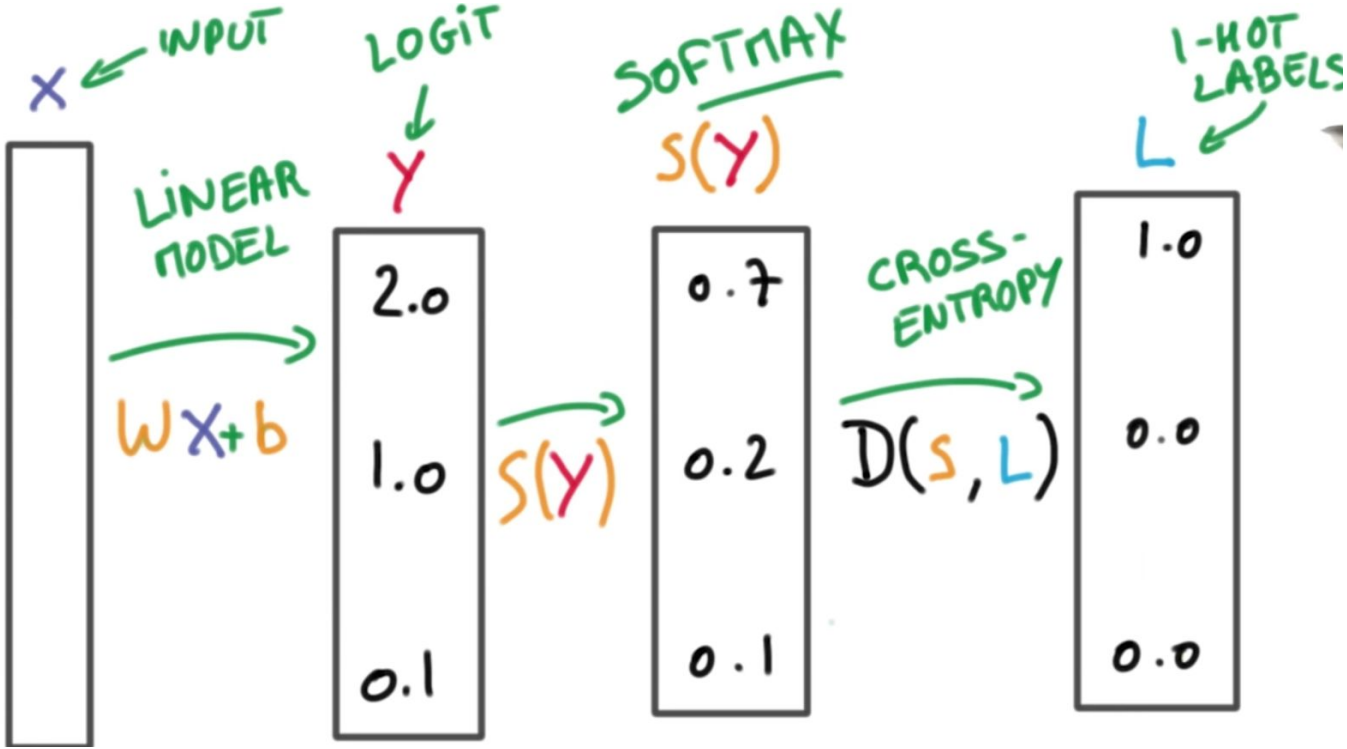
https://ireneli.eu/2016/01/29/tensorflow-02-play-with-mnist/

Cross Entropy



https://ireneli.eu/2016/01/29/tensorflow-02-play-with-mnist/

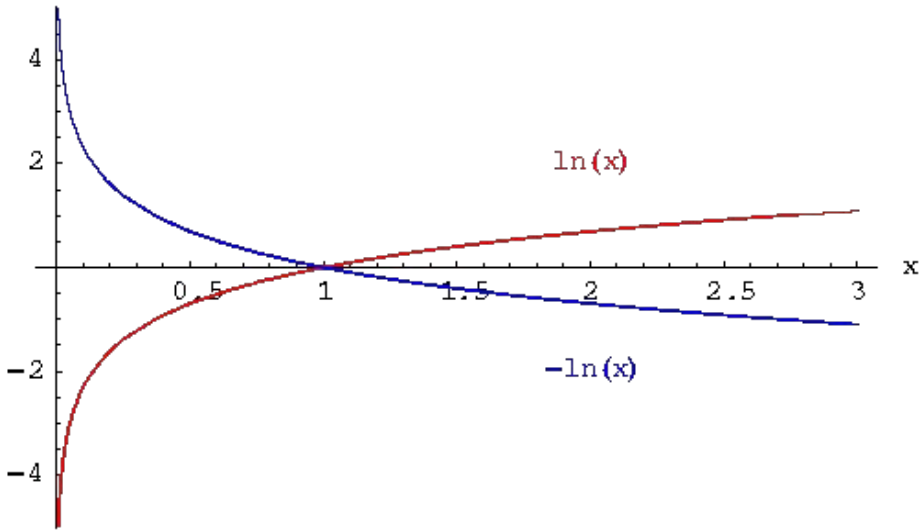
Cross Entropy



https://ireneli.eu/2016/01/29/tensorflow-02-play-with-mnist/

Cross Entropy

$$-\sum_i L_i \log(S_i) \quad => \quad -\sum_i L_i \log(\hat{y}_i) \quad => \quad \sum_i L_i * -\log(\hat{y}_i)$$



https://pythonkim.tistory.com/20

Cross Entropy

$$-\sum_i L_i \log(S_i) \quad => \quad -\sum_i L_i \log(\hat{y}_i) \quad => \quad \sum_i L_i * -\log(\hat{y}_i)$$

$$L = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\hat{Y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} * -log \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} * \begin{bmatrix} \infty \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0, \quad cost=0$$

$$\hat{Y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} * -log \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} * \begin{bmatrix} 0 \\ \infty \end{bmatrix} = \begin{bmatrix} 0 \\ \infty \end{bmatrix} = \infty, \quad cost=\infty$$

https://pythonkim.tistory.com/20

Zoo classification

1. 동물 이름 animal name: (deleted)
2. 털 hair Boolean
3. 깃털 feathers Boolean
4. 알 eggs Boolean
5. 우유 milk Boolean
6. 날 수있는지 airborne Boolean
7. 수중 생물 aquatic Boolean
8. 포식자 predator Boolean
9. 이빨이 있는지 toothed Boolean
10. 척추 동물 backbone Boolean
11. 호흡 방법 breathes Boolean
12. 독 venomous Boolean
13. 물갈퀴 fins Boolean
14. 다리 legs Numeric 0, 2, 4, 6 ...
15. 꼬리 tail Boolean
16. 사육 가능한 지 domestic Boolean
17. 고양이 크기인지 catsize Boolean
18. 동물 타입 type Numeric 0 ~ 6

참고자료

- 밑바닥부터 시작하는 딥러닝 1, 2
  http://www.yes24.com/Product/Goods/34970929?Acode=101
  http://www.yes24.com/Product/Goods/72173703

- 모두를 위한 딥러닝 시즌2
  https://www.edwith.org/boostcourse-dl-tensorflow/joinLectures/22150

- 모두의 연구소 이일구님 강의 자료
  https://github.com/ilguyi