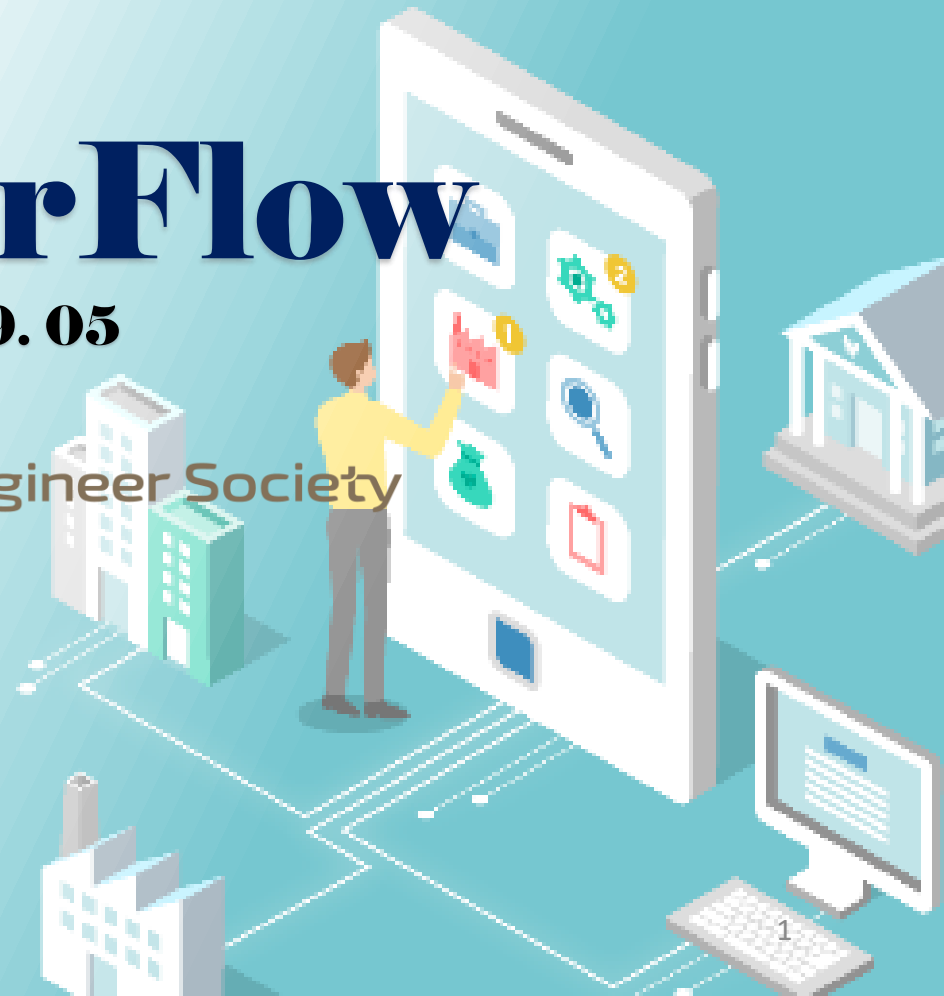


TensorFlow

2019. 05



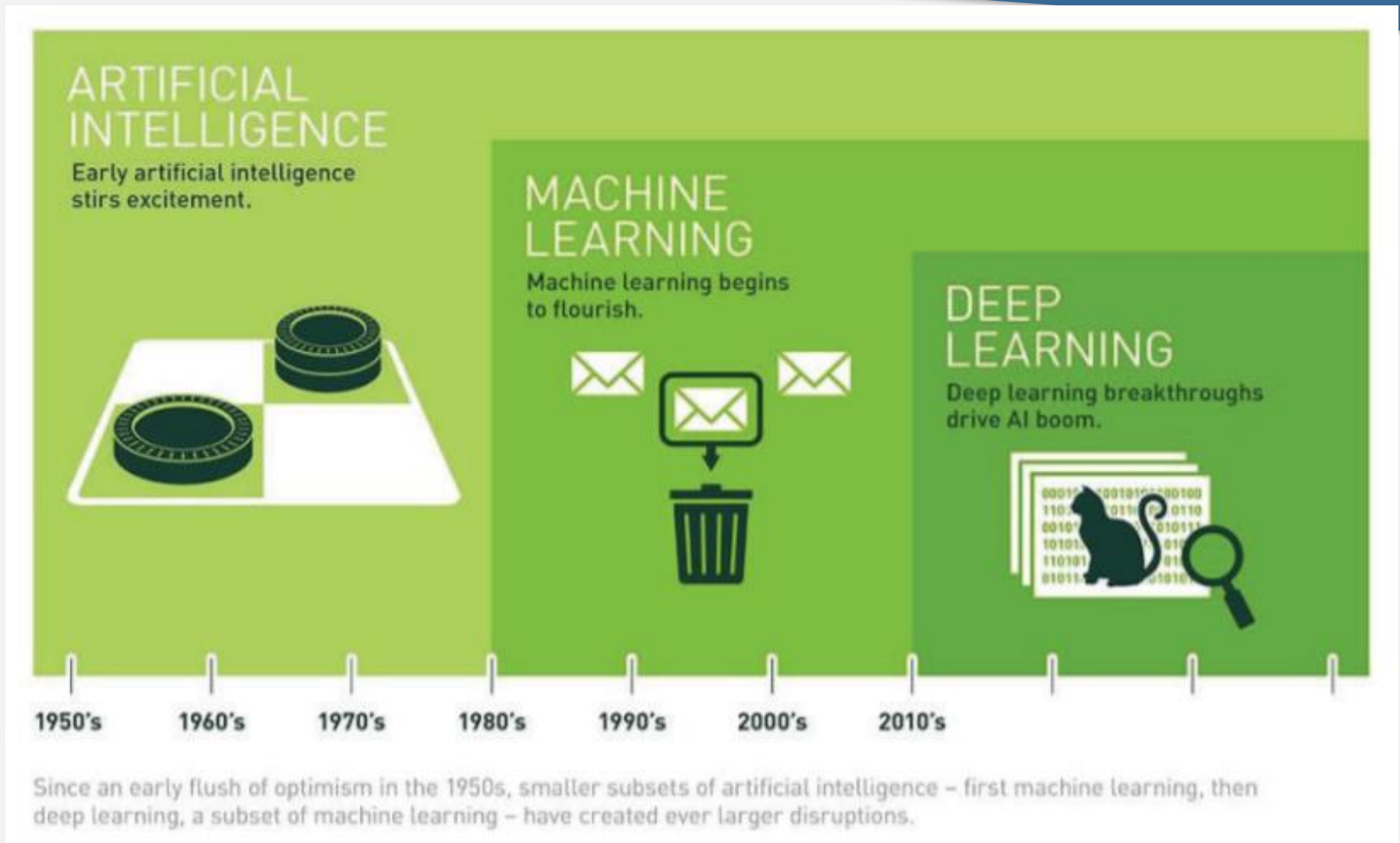
Soft Engineer Society



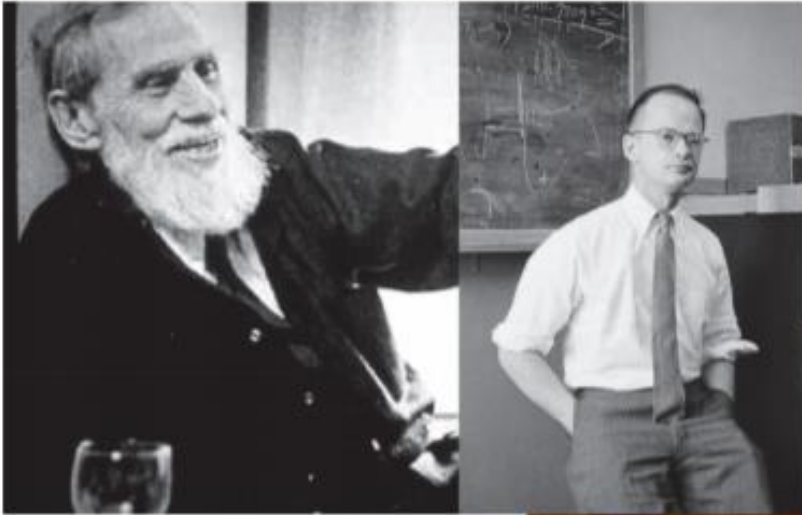


Introduction

AI vs. ML vs. DL



Deep learning의 계보



McCulloch & Pitts
math model



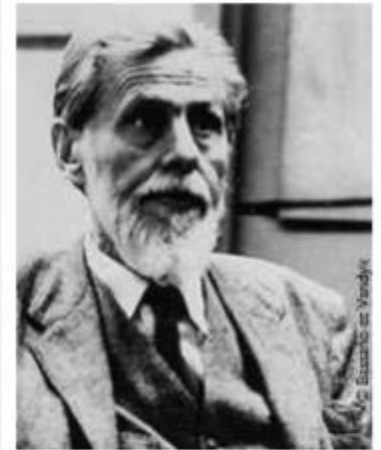
Hinton
DNN, back propagation



Rosenblatt
computer simulation

▶ Neuron Model

- 1943년, 미국 일리노이 의대 정신과 부교수였던 워런 맥컬록(Warren McCulloch)은 "신경 활동에 내재한 개념들의 논리적 계산"이라는 제목의 논문 발표
- 이들은 이 논문에서 신경망을 '이진 스위칭' 소자가 복잡하게 연결된 네트워크로 모형화했다.
- 인공신경망을 개념화한 최초의 논문이다.



▶ Restricted Boltzmann Machine(RBM)

- 2004년 제프리 힌튼(Geoffrey Hinton) 교수가 RBM이라는 새로운 딥러닝 기반의 학습 알고리즘을 제안하면서 주목을 받기 시작했다.

▶ TensorFlow란?

- TensorFlow ™는 Data Flow Graph를 사용하여 수치 계산을 하는 오픈 소스 소프트웨어 라이브러리
- Google에서 개발
- Machine Intelligence를 위한 Open Source software library : Data Flow Graphic 사용
- Python을 기반으로 한다.

▶ 개요

- Tensorflow는 뉴럴네트워크에 최적화되어 있는 개발 프레임워크이기 때문에, 그 자료형과 실행 방식이 약간 일반적인 프로그래밍 방식과 다르다.
- 연산과 데이터에 대한 모든 정보는 그래프 구조 안에 저장된다.
- 그래프 구조로 표현된 정보를 이용해서 트랜잭션 간의 의존성을 인식하고 노드에 입력 데이터가 들어올 Tensor가 준비될 때 디바이스에 비동기적으로 병렬적으로 연산을 할당한다.

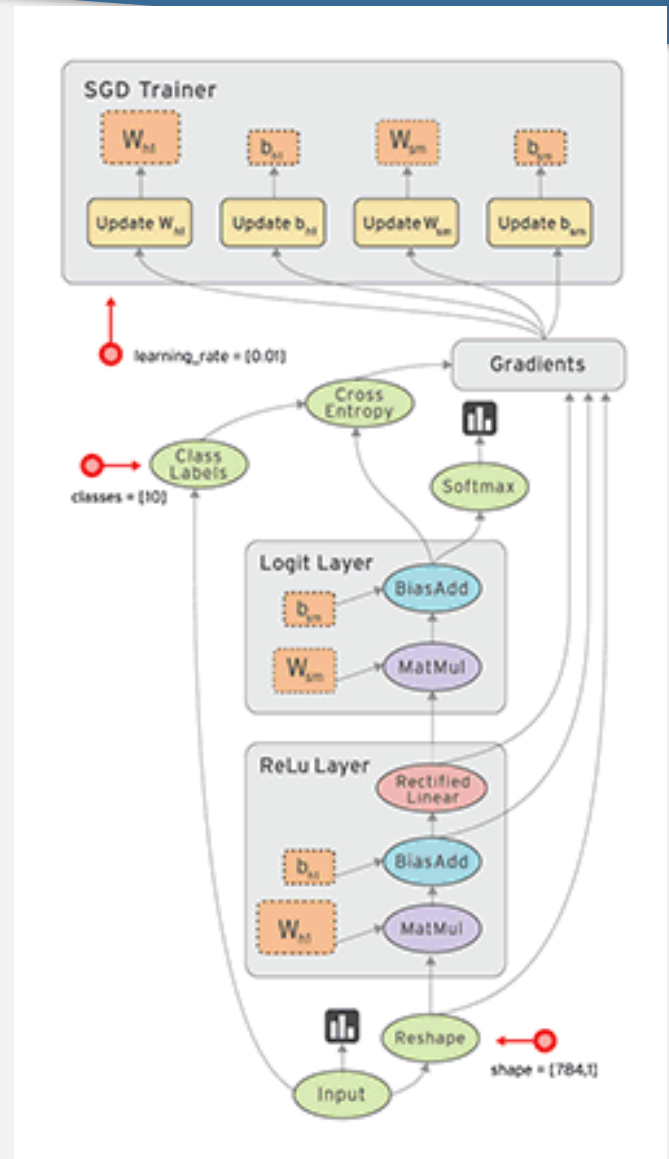
Data Flow Graph?

► Data Flow Graph?

- Node : 수학 연산을 나타낸다
- Edge : Node 전달된 다차원 데이터 배열 (tensor)를 나타낸다.

• Nirmal Jith:

“why tensorflow uses a graph based approach?
For example, it will help to split to sub graphs and hence parallel processing will be easier.
If you can include that it will be great.”



▶ Anaconda 다운로드

- <https://www.anaconda.com/distribution/>



Windows



macOS



Linux

Anaconda 2019.03 for macOS Installer

Python 3.7 version

Download

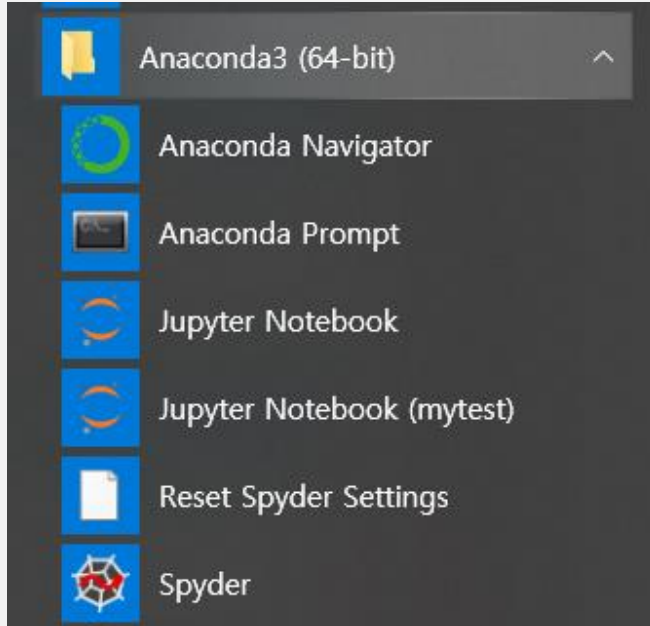
64-Bit Graphical Installer (637 MB)
64-Bit Command Line Installer (542 MB)

Python 2.7 version

Download

64-Bit Graphical Installer (624 MB)
64-Bit Command Line Installer (530 MB)

▶ Anaconda 설치 완료 창



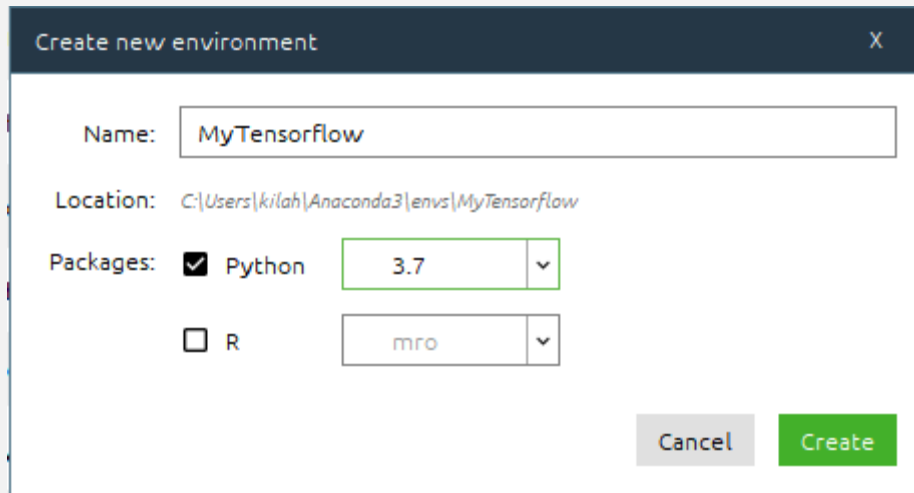
▶ 추가 작업

- 위 창에서 Anaconda Navigator 클릭
- Navigator 창에서 Anaconda Navigator-‘Environments’-‘create’ 선택

▶ 추가 작업(계속)

- ‘Create new environment’ 창에서

- ① Name : MyTensorflow 입력
- ② Packages : Python 3.7 선택 후 create 클릭



Create new environment

Name: MyTensorflow

Location: C:\Users\kilah\Anaconda3\envs\MyTensorflow

Packages: ☒ Python 3.7 ☐ R mro

Cancel Create

▶ 추가 작업(계속)

- Anaconda에서 package 추가 설치

- ① tensorflow
- ② jupyter
- ③ jupyter-client
- ④ numpy
- ⑤ matplotlib
- ⑥ pandas

※ 주의 : 로컬에 파이썬, 넘파이, 텐서플로가 설치되어 있으면 anaconda와 설치 버전이 맞아야 한다. 그렇지 않을 경우 오류발생

```
> pip install numpy # numpy
```

```
> pip install tensorflow
```

```
> pip install --upgrade numpy # 오류 발생시
```

▶ Tensorflow의 Type

- ① `tf.constant` : 상수
- ② `tf.Variable` : synaptic weights
- ③ `tf.placeholder` : input / output 뉴런

Tensorflow 프로그래밍 기초

▶ tensorflow import

```
import tensorflow as tf
```

▶ constant 사용

```
hello = tf.constant("Hello, TensorFlow")
```

▶ Session을 이용한 실행

- 심벌릭 표현으로 된 수식을 계산하기 위해 세션을 생성한 뒤 실행한다

```
sess = tf.Session()  
sess.run(hello)
```

▶ constant

- 상수를 저장하는 데이터 형이다.
- value : 상수의 값이다.

```
tf.constant(value, dtype=None, shape=None, name='Const',  
verify_shape=False)
```

- dtype : 상수의 데이터형이다. tf.float32와 같이 실수, 정수 등의 데이터 타입을 정의한다.
- shape : 행렬의 차원을 정의한다. shape=[3,3]으로 정의해주면, 이 상수는 3x3 행렬을 저장하게 된다.
- name : name은 이 상수의 이름을 정의

constant

```
import tensorflow as tf

print(tf.__version__)

# constant 생성
hello = tf.constant("Hello, TensorFlow")
sess = tf.Session()
print(sess.run(hello)) # b'Hello, TensorFlow'
```



```
import tensorflow as tf
```

```
a = tf.constant([5], dtype=tf.float32)  
b = tf.constant([10], dtype=tf.float32)  
c = tf.constant([6], dtype=tf.float32)
```

```
d = a * b + c
```

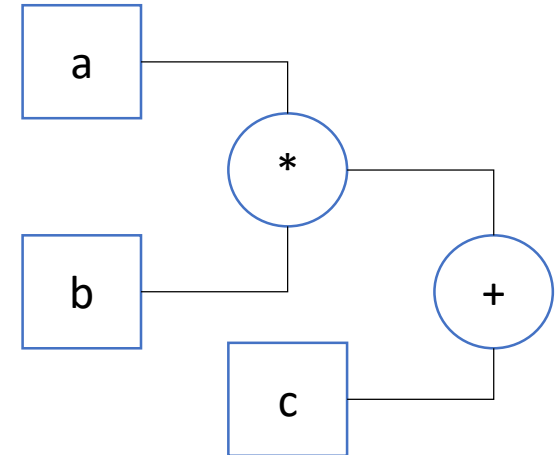
```
print(d)
```

```
'''Tensor("add:0", shape=(1,), dtype=float32)'''
```

```
sess = tf.Session()  
result = sess.run(d)
```

```
print(result)
```

```
'''[56.]'''
```



▶ placeholder

- 선언과 동시에 초기화 하는 것이 아니라 선언 이후에 값을 전달하는 방식을 사용
- 반드시 실행 시 데이터가 제공되어야 한다. "feed_dict" 이용

```
tf.placeholder(dtype, shape, name)
```

- dtype : 데이터 타입을 의미(필수)
- shape : 입력 데이터의 형태. 상수나 다차원 배열 등 (디폴트 : None)
- name : 해당 placeholder의 이름 (디폴트 : None)

placeholder

```
import tensorflow as tf

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

y = tf.multiply(a, b)
sess = tf.Session()
print(sess.run(y, feed_dict={a:3, b:3}))
```

placeholder

```
import tensorflow as tf

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

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

▶ Variable

- Tensorflow에서는 `Variable()` 이라는 생성자를 사용해서 변수를 생성할 수 있다.
- 이 변수는 생성되는 순간에 데이터의 타입과 크기가 결정되며, 그 래프를 실행하기 전에 초기화를 해줘야 그 값이 변수에 지정이 된다.

```
x = tf.Variable(2.)  
tf.global_variables_initializer()
```

Variables

```
import tensorflow as tf

x = tf.Variable(2.)
print(x)
'''<tf.Variable 'Variable_4:0' shape=() dtype=float32_ref>'''

init_op = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init_op)
print(sess.run(x))
'''2.0'''
```

Variables

```
import tensorflow as tf

input_data = [1, 2, 3, 4, 5]
x = tf.placeholder(dtype=tf.float32)
W = tf.Variable([2],dtype=tf.float32)
y = W * x

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

result = sess.run(y,feed_dict={x:input_data})

print( result)
'''[ 2.  4.  6.  8. 10.]'''
```

카테고리별 연산

카테고리	Shape
수학	add, sub, mul, div, exp, log, greater, less, equal
배열	concat, slice, split, constant, rank, shape, shuffle
행렬	matmul, matrixinverse, matrixdeterminant
신경망	softmax, sigmoid, relu, convolution2d, maxpool
세션	save, restore
큐잉과 동기화	enqueue, dequeue, mutexacquire, mutexrelease
흐름제어	merge, switch, enter, leave, nextiteration

Rank, Shapes and Types

Rank	Shape	Dimension number	Example
0	[]	0-D	0-D tensor. Scalar
1	[D0]	1-D	1-D tensor, [3]
2	[D0, D1]	2-D	2-D tensor, [2, 3]
3	[D0, D1, D2]	3-D	3-D tensor, [1, 2, 3]
n	[D0, D1, D2, ... , Dn-1]	n-D	Tensor, [D0, D1, D2, ..., Dn-1]

dType의 종류

Type	Description
tf.float32	32 bits floating point
tf.float64	64 bits floating point
tf.int8	8 bits signed integer
tf.int16	16 bits signed integer
tf.int32	32 bits signed integer
tf.int64	64 bits signed integer

Tensorflow 수확함수

함수	Description
<code>tf.add(...)</code>	덧셈
<code>tf.subtract(...)</code>	뺄셈
<code>tf.multiply(...)</code>	곱셈
<code>tf.truediv(...)</code>	나눗셈의 몫 (파이썬 3버전)
<code>tf.mod(...)</code>	나머지
<code>tf.abs(...)</code>	절댓값
<code>tf.negative(...)</code>	음수
<code>tf.sign(...)</code>	부호
<code>tf.reciprocal(...)</code>	역수
<code>tf.square(...)</code>	제곱

Tensorflow 수확함수

함수	Description
<code>tf.round(...)</code>	반올림
<code>tf.sqrt(...)</code>	제곱근
<code>tf.pow(...)</code>	거듭제곱
<code>tf.exp(...)</code>	지수 값
<code>tf.log(...)</code>	로그 값
<code>tf.maximum(...)</code>	최댓값
<code>tf.minimum(...)</code>	최솟값
<code>tf.cos(...)</code>	코사인 함수 값
<code>tf.sin(...)</code>	사인 함수 값

※ 참고: https://www.tensorflow.org/api_docs/python/tf/math

Tensorflow 행렬 연산 함수

함수	Description
<code>tf.diag(...)</code>	대각 행렬
<code>tf.transpose(...)</code>	전치 행렬
<code>tf.matmul(...)</code>	두 텐서를 행렬 곱한 결과 (텐서 리턴)
<code>tf.matrix_determinant(...)</code>	정방행렬의 행렬식 값
<code>tf.matrix_inverse(...)</code>	정방행렬의 역행렬



Machine Learning

▶ Machine Learning이란?

- Explicit Programming의 한계
 - 현재 우리가 사용하는 프로그램 방식
 - 문제를 해결하기 위한 기능을 개발자가 개발하는 방식
- Machine Learning
 - 문제가 주어지면 컴퓨터 스스로 학습하여 해결책(모델)을 작성

“Field of study that gives computers the ability to learn without being explicitly programmed”

명시적으로 프로그램 되지 않고 컴퓨터가 학습을 하는 분야

- Arthur Samuel(1959)

인공 신경망 주 활용분야

▶ 머신러닝 / 딥러닝의 활용분야

- 이미지 안에서 물체를 인식, 이미지의 내용을 요약
(CNN : 합성곱 신경망)
 - 덴스캡(DenseCap): 2015년 스탠퍼드 대학교 비전랩에서 개발
이미지의 장면을 글로 요약
 - 2015년 구글 딥드림(DeepDream): 서로 다른 이미지의 패턴을 인식하
고 합성하는 프로그램
 - 구글마젠타 : 순환 신경망을 이용한 작곡 (텐서플로를 이용)
 - 페이스북 딥텍스트: 텍스트 분석 엔진
합성곱 신경망 + 순환신경망(CNN)
- 언어 번역
- 음성 인식
- 자율주행 자동차 개발, 예술분야

▶ ML의 종류

① supervised

- 학습할 데이터(label)가 주어지고 이를 통해 기계학습 실시
- Training Data Set에 의존

(예)

- 암환자의 데이터를 이용해 학습을 한 후 다른 암환자를 진단
- AlphaGo : 주어진 바둑기보를 이용해 학습을 실시 한 후 대결
- Image labeling: tag된 이미지를 이용해 학습
- Email spam filter: label된(spam or ham) email을 통해 학습

② unsupervised

- 학습 데이터가 없이 주어지는 데이터를 통해 스스로 학습 실시

(예)

- Word Clustering 등

▶ Regression / Classification

① Regression(회귀)

- 일반적으로 연속적인 숫자, 즉 예측 값이 float 형태인 문제들을 해결하는데 사용
- 출력에 연속성이 있다.
- 예를 들어, 지하철 역과의 거리, 일정 거리 안의 관공서, 마트, 학군의 수 등등 여러 feature들로 어떤 지역의 땅값을 예측하는 문제

② Classification(분류)

- class를 예측하는 것 : 예측해야 할 대상이 정해져 있음
- Binary classification
 - Yes/No로 대답할 수 있는 문제
(예 : Spam이냐? Ham이냐?) or (예 : Pass냐? Fail이냐?)
- Multi-class classification
 - 예측할 클래스가 여러 종류
(예 : A, B, C, D, F)



Linear Regression

Linear Regression

▶ Linear Regression(선형회귀)

- 선형으로 예측되는 모델을 구현하기 위한 알고리즘
- 일반적인 예측 알고리즘에서 선형회귀 방식은 매우 유용

▶ 용어

① Training data set

- 학습을 시키기 위해 필요한 label이 존재하는 데이터

② Iteration, step, epoch

- 여러 번 학습을 반복. 얼마나 반복할 것이냐?

③ Learning, training

- 원하는 기준(Model)을 찾아냄

④ Test set, evaluation set

- label이 없는 데이터. 학습에 의해 찾아낸 Model이 얼마나 정확한지 테스트 하기 위한 데이터

⑤ feature(특질)

- 입력 값, 주어진 데이터 (input 이라고 하지 않는다.)
예를 들어 2개의 feature를 사용했으면 입력벡터가 2차원

⑥ weight(w_1 , w_2)

- 여러 개의 feature중 어느 것에 더 가중치를 줄 것인가?

▶ 학습의 목표

- 오차를 최소로 혹은 오차 함수의 값이 최소가 되는 w 와 b 를 찾는 것이다.
- 선을 찾는 것 = 기준을 찾는 것 = 모델을 찾는 것
- $w_1x_1 + w_2x_2 = b$ 일 때 w_1, w_2, b 를 찾는 것

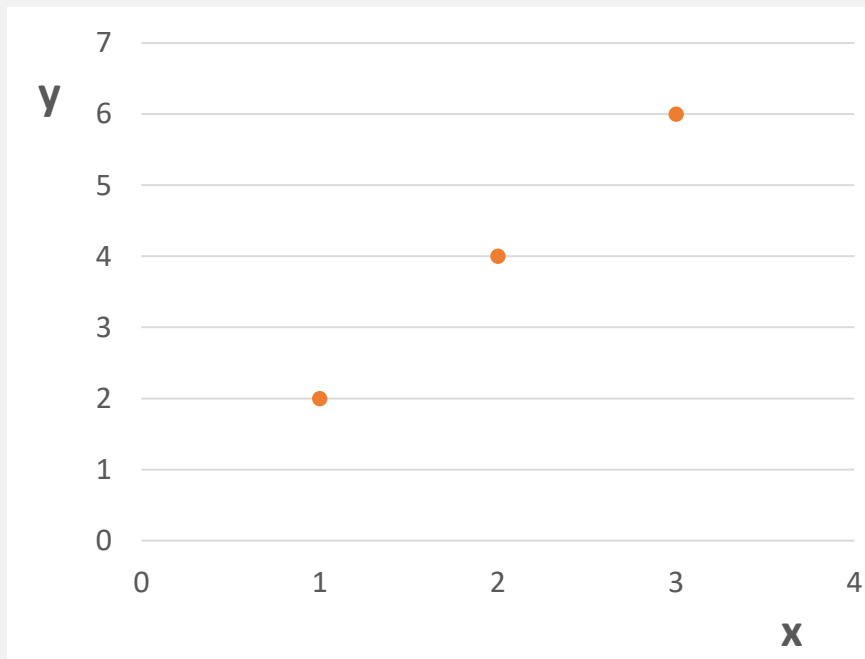
▶ 경사 하강법(Gradient Descent) :

- 오차평면에서 공을 올려 놓았을 때 공이 굴러가는 방향(오차가 적어지는 방향)으로 w 를 조정하여 기울기가 0인 w 를 찾는 것

Linear Regression

▶ Training Data Set

- 예) 제시된 데이터(Training Data Set)가 아래와 같다고 가정

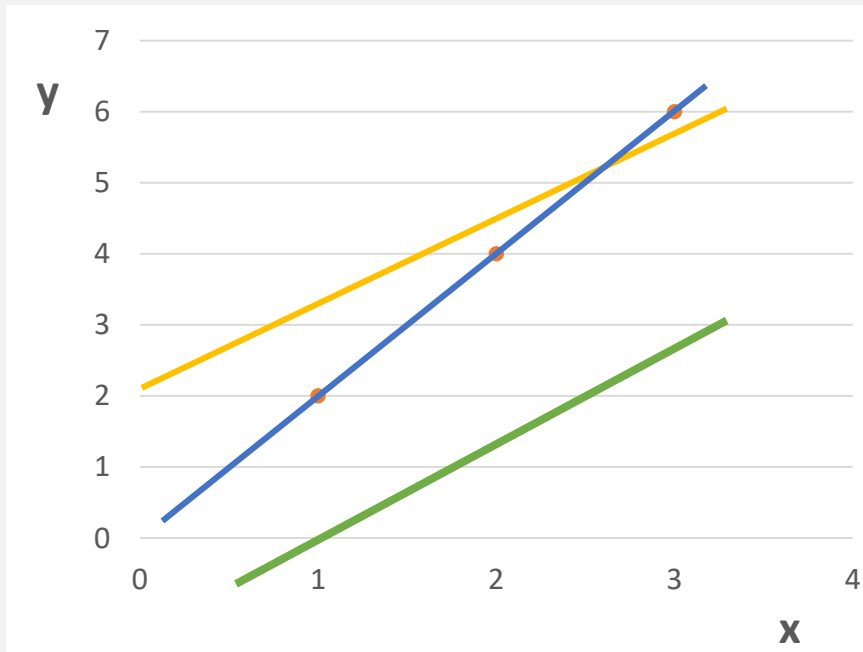


x	y
2	2
4	4
6	6

- 위의 데이터를 이용해 학습 프로그램을 만들어보자

► Hypothesis

- 예) 주어진 데이터가 linear하다고 가정하고 학습을 시킴
- 학습 프로그램은 여러 개의 선을 그려 그 중 주어진 데이터와 가장 오차가 적은 선을 찾아낸다.



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

Cost Function

▶ 어떤 가설(Hypothesis)이 좋은가?

- 가설로 만든 선과 실제 데이터 사이의 거리를 측정해서 그 오차 값이 적어야 함 (H값이 바로 예측값(prediction)이다.)

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

▶ Cost function

- 오차를 측정하는 것(거리)
- 아래와 같은 공식은 좋지 않다

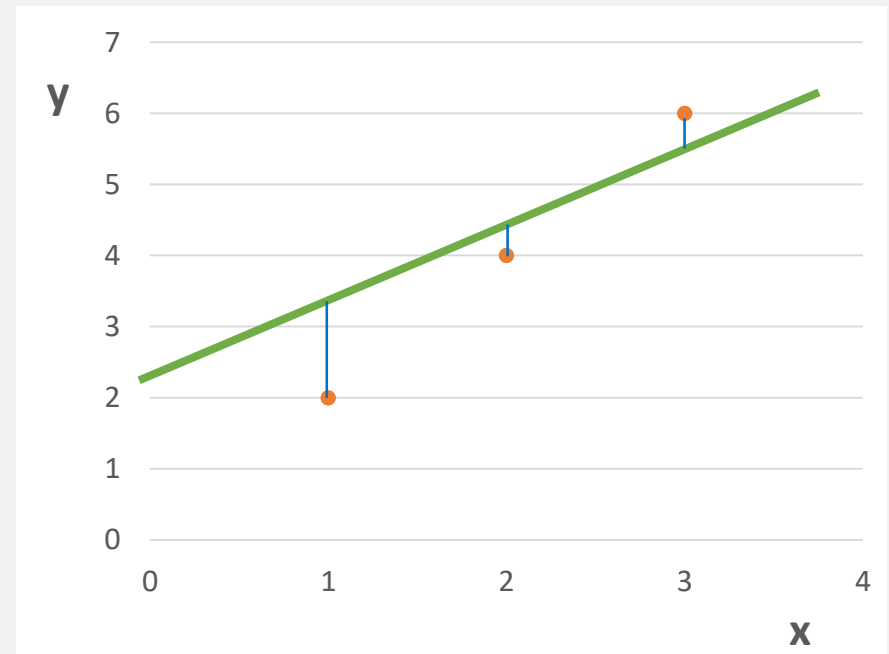
$$H(x) - y$$



예측값



Label(true값)



► Cost function

- 거리를 측정하여 그 오차가 최소인 Cost 함수를 찾아야 함
- Cost가 최소화(minimize)하는 w , b 를 구하는 것이 학습의 목표

$$Cost(W, b) = \frac{(H(x^{(1)}) - y^{(1)})^2 + (Hx^{(2)} - y^{(2)})^2 + (Hx^{(3)} - y^{(3)})^2}{3}$$



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

m : 총 데이터 개수

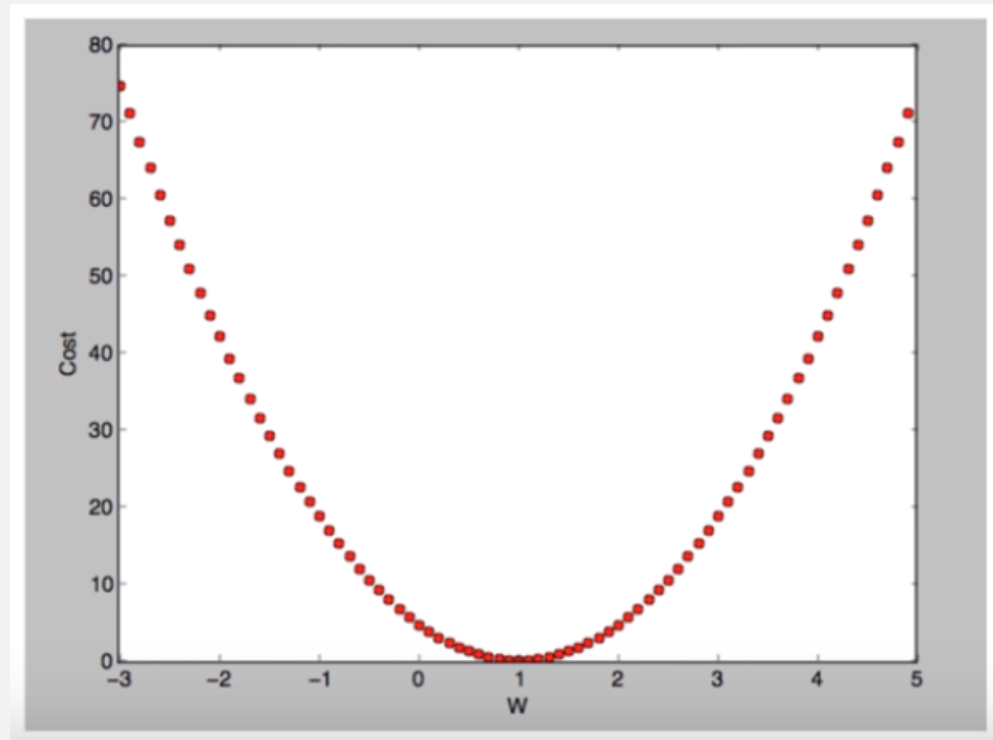
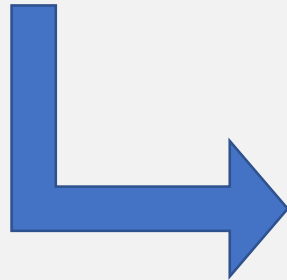
예측값

Label(true값)

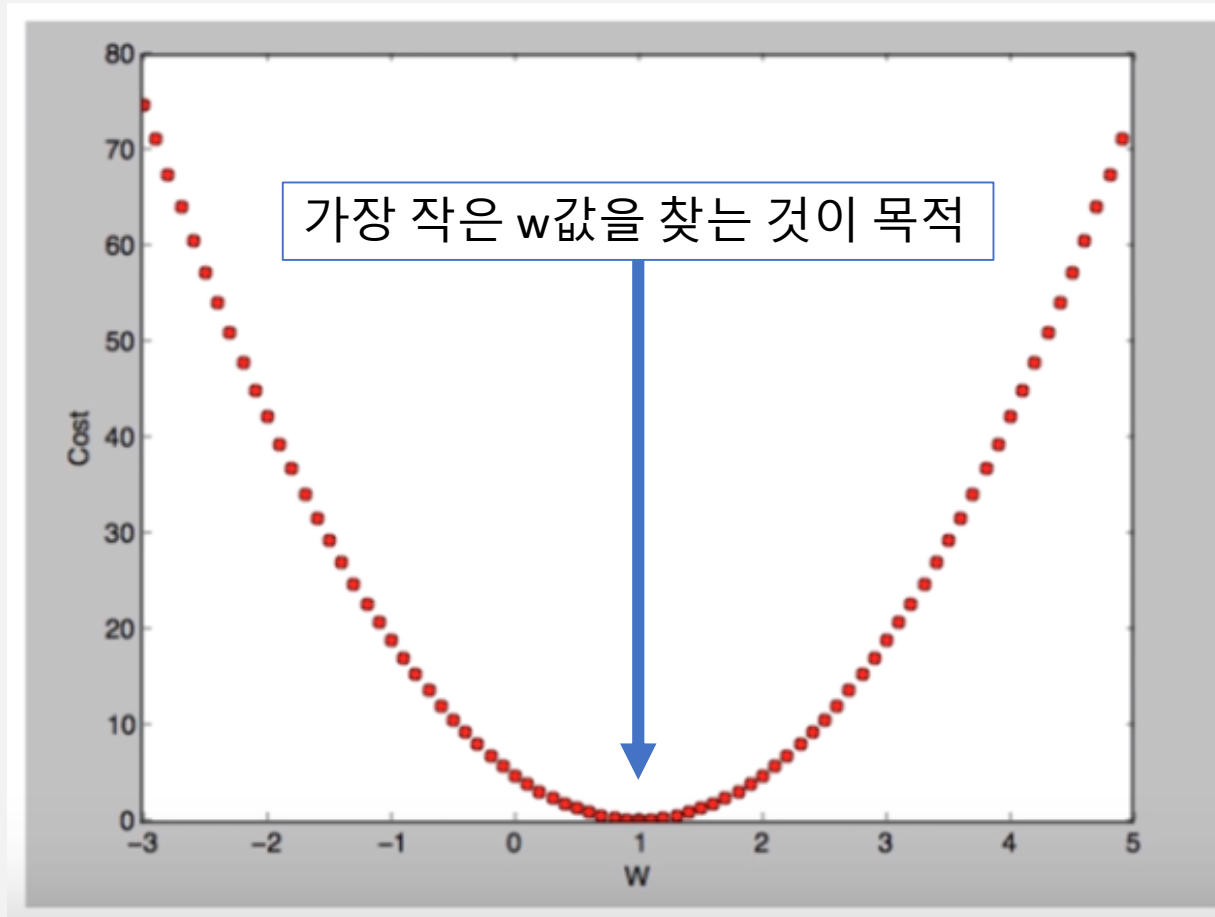
Cost function

▶ Cost function의 원리

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



▶ Cost function의 원리



Gradient Descent algorithm

▶ Gradient Descent algorithm

- Cost function에서 최소화된 w 값을 찾기 위한 알고리즘
- 최소값을 찾을 때 많이 사용함
- $W(\text{feature})$ 가 많아도 원하는 값을 찾을 수 있다.
- 그래프에 임의의 w 을 주고 해당 값을 이용한 경사도 확인
- 기울기가 0이 아니면 w 값을 수정하고 다시 경사도를 확인
- 위 동작을 반복
- 값을 얼마만큼씩 수정할 것인가? learning rate
- 이렇게 w 값을 수정하여 기울기가 0인 값을 찾으면 학습이 끝난다.
- 기울기를 구하기 위해서? “미분”

▶ 기울기 구하기(미분)

- 미분공식 : cost를 W 에 대해 미분하면 기울기를 구할 수 있다.

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

▶ 참고 (미분)

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

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

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

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

- ▶ Hypothesis

$$H(X) = WX$$

- ▶ Cost function

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

- ▶ Gradient Descent

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

▶ Cost Function의 종류

- 1) MSE(Mean Squared Error)
- 2) CE (Cross Entropy)
- 3) KL-Divergence
- 4) MLE(Maximum Likelihood Estimation)

▶ Optimizer 종류

- 오차에 대해서 w 를 업데이트 시키는 알고리즘

- 1) Gradient Descent
- 2) Batch Gradient Descent
- 3) Mini-Batch Gradient Descent
- 4) SGD(Stochastic GD)
- 5) Momentum
- 6) AdaGrad
- 7) AdaDelta
- 8) Adam
- 9) RMSprop

※ Learning rate

- 학습률
- w 의 변경되는 정도

Activation Function 의 종류

▶ Activation Function

- 1) sigmoid(=logistics)
- 2) Tanh
- 3) ReLU
- 4) Leaky ReLU

Multi variable의 처리

▶ Multi variable

- Feature가 여러 개인 데이터를 처리하고자 할 때

Instance가 매우
많아질 수 있다.



x1	x2	x3	y
73	80	75	105
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

▶ Multi variable

- Feature가 여러 개인 데이터를 처리하고자 할 때
- 데이터가 많아도 동일하다.

$$H(x_1, x_2, x_3) = w_1x_1 + w_2x_2 + w_3x_3 + b$$

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

Matrix를 이용한 Hypothesis

- ▶ Matrix 곱으로 표현

$$x_1w_1 + x_2w_2 + x_3w_3 + \dots + b$$

$$\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} x_1w_1 \\ x_2w_2 \\ x_3w_3 \end{bmatrix}$$

- ▶ Many instance : 3 → 데이터 샘플 수, 3 → feature의 수

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

$3 \times 3 \qquad \qquad 3 \times 1 \qquad \qquad 3 \times 1$

Matrix를 이용한 Hypothesis

- ▶ w 의 크기를 어떻게 구할까?
 - 데이터의 개수(n)와 feature(3)은 주어진다.
 - 결과 ($n \times 1$) 도 주어진다.
 - 그러므로 w 의 크기는 $[n \times 3] \cdot [? \times ?] = [n \times 1]$
 - 행 크기 : x 의 열 크기(n)
 - 열 크기 : 결과값의 열 크기(n)

Matrix를 이용한 Hypothesis

▶ W의 크기를 어떻게 구할까?

- 예 : 아래와 같은 데이터가 존재한다면
데이터가 5개, feature 3, y의 크기가 5x1이면

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

$5 \times 3 \qquad \qquad 3 \times 1 \qquad \qquad 5 \times 1$

Matrix를 이용한 Hypothesis

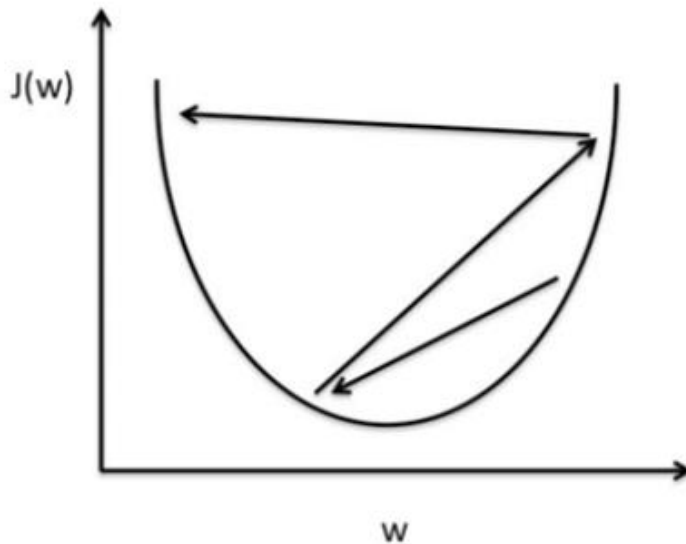
- ▶ W 의 크기는 얼마일까?
 - 출력이 여러 개인 경우

X	W	$H(X)$
$[n, 3]$	$?$	$[n, 2]$

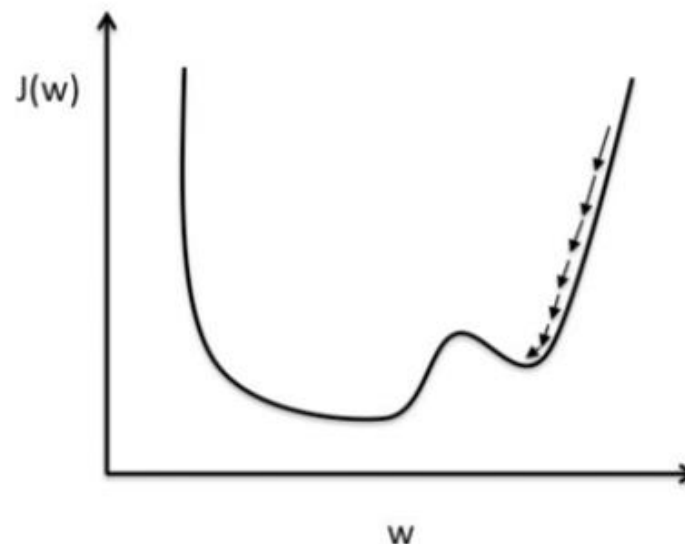
$$H(X) = XW$$

Learning_rate overshooting

Learning rate: NaN!



Large learning rate: Overshooting.



Small learning rate: Many iterations until convergence and trapping in local minima.

http://sebastianraschka.com/Articles/2015_singlelayer_neurons.html



Logistic Classification

▶ 분류

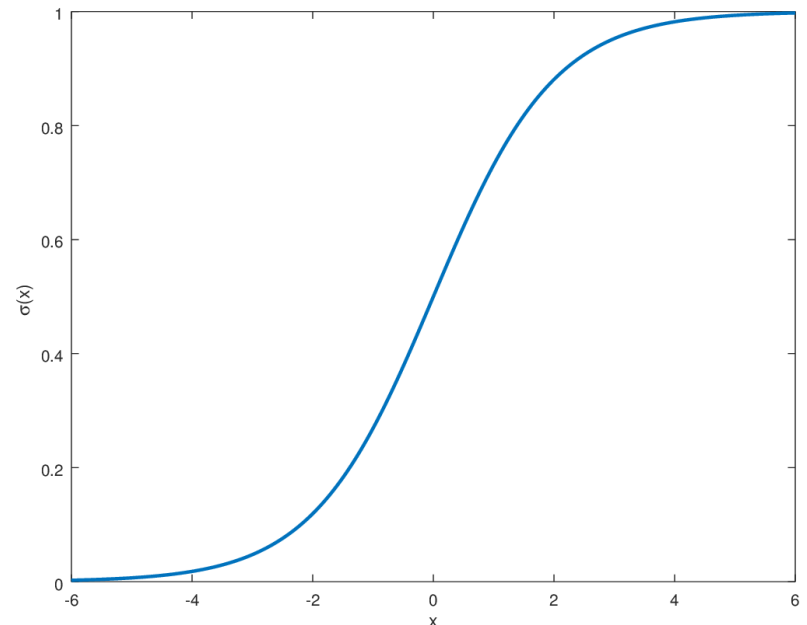
- <https://www.youtube.com/watch?v=tYxkIOTdeu8>
- <https://www.youtube.com/embed/V2MMyk7bdwY>

Logistic Classification

▶ Binary classification

- 기존의 가설 함수 $H(x)$ 는 0, 1 의 단순 값으로 분류하기 어려움 (실수를 리턴하기 때문)
- 그래서 만들어 낸 함수가 s자 모양을 한 sigmoid 함수
데이터가 아무리 커져도 0, 1 로 만든다. → logistic function

$$H(X) = \frac{1}{1 + e^{W^T X}}$$



Sigmoid의 cost function

▶ Hypothesis

$$H(X) = \frac{1}{1 + e^{w^T X}}$$

▶ Cost function의 그래프 :

- 매끄러운 그래프를 그릴 수 없다. so, 새로운 cost 함수 필요

$$C(H(x), y) = \begin{cases} -\log(H(x)) & : y = 1 \\ -\log(1 - H(x)) & : y = 0 \end{cases}$$



$$C(H(x), y) = -\log(H(x)) - (1 - y) \log(1 - H(x))$$

Sigmoid의 GradientDescent

- ▶ Cost function의 minimize

$$cost(W) = -\frac{1}{m} \sum_{i=0}^m y \log(H(x)) + (1 - y) \log(1 - H(x))$$

```
#cost function
```

```
cost = tf.reduce_mean(-tf.reduce_sum(Y*tf.log(hypothesis)  
    + (1-Y) * tf.log(1-hypothesis)))
```

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

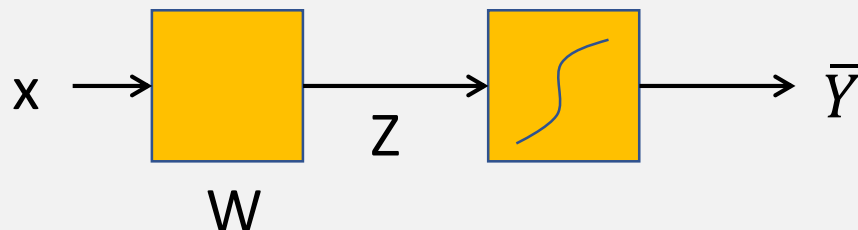
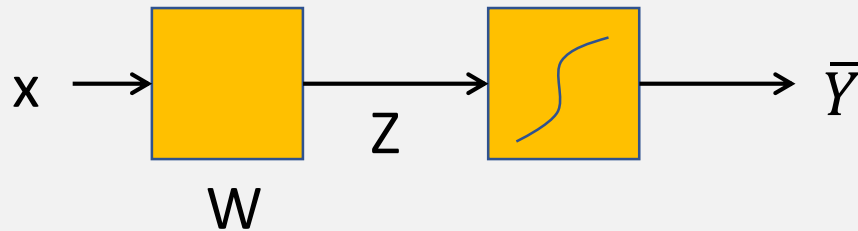
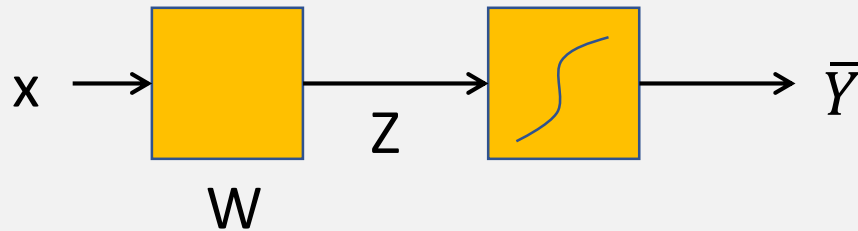
```
# Minimize
```

```
learning_rate = tf.Variable(0.1)  
optimizer = tf.train.GradientDescentOptimizer(learning_rate)  
train = optimizer.minimize(cost)
```

Multinomial Classification

► Softmax Algorithm

- 만약 3개의 클래스로 구별한다면, 0, 1, 2가 나오는 것이 아니라 0~1사이의 값으로 출력 -> Softmax



Softmax Algorithm

▶ Softmax

- 3개의 출력 값의 합은 **1**이다.
- 즉 확률로 출력함!

▶ One-hot encoding

- Softmax로 출력된 확률을 하나의 값으로 만드는 함수

▶ Cost function

- Cross entropy function

$$-\frac{1}{m} \sum_{i=0}^m Y \times \log(H(x))$$

```
cost = tf.reduce_mean(-tf.reduce_sum(  
    Y * tf.log(hypothesis), axis=1))
```

Softmax Algorithm

► Cost function

- tf에서 제공하는 `softmax_cross_entropy_with_logits()`을 사용
- 첫 번째 전달인자 : `logit`값
- 두 번째 전달인자 : one hot encoding된 `Y` 레이블값

```
logits = tf.matmul(X, W) + b
hypothesis = tf.nn.softmax(logits)

cost_i = tf.nn.softmax_cross_entropy_with_logits(logits=logits,
                                                  labels=Y_one_hot)
cost = tf.reduce_mean(cost_i)
```