Machine Learning

<Contents>

1) Basic understanding of machine learning algorithms

   - Linear regression, Logistic regression (classification)

     :Something that you need to learn before learning “Deep Learning”

     & make you understand it much easier

   - **Deep Learning** : Neural networks, Convolutional Neural Network, Recurrent Neural Network

2) Solve your problems using machine learning tools

   - ex) Tensorflow & Python

<Basic Concepts>

1. Machine Learning :

→Limitations of explicit programming (The reason why ML appeared on the world)

1) Spam filter: many rules
2) Automatic driving: TOO many rules

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

2. Supervised learning:

→Learning with labeled examples : training set (training set = labeled example)

ex) Q. How to distinguish a cat from many dogs?

A. Learning with training set(=showing a plenty of cat pictures and making computer learn about the cat) →Easy way: Google Teachable Machine …

<Most Common Problem Type in Machine Learning>

1) Image labeling : learning from tagged images
2) Email spam filter : learning from labeled "spam or ham" email
3) Predicting exam score : learning from previous exam score and time spent

<Types of Supervised Learning>

1) Regression

   : ex) Predicting final exam score based on time spent

2) Binary Classification

   : ex) Pass/Fail based on time spent

3) Multi-Label Classification

   : ex) Letter grade (A,B,C,…) based on time spent

3. Unsupervised learning : un-labeled data

→ When we cannot provide labeled examples

ex) Google news grouping OR Word clustering

<Linear Regression>

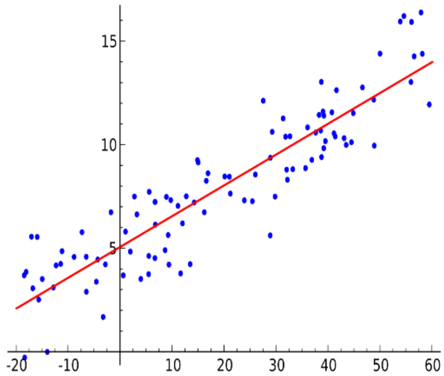
Ex) 10 hours → score: 95

9 hours → score: 85

7 hours → score: 65

3 hours → score: 35

Predict: We could predict that if a child studied for 8 hours, then he would get 75 on the exam.



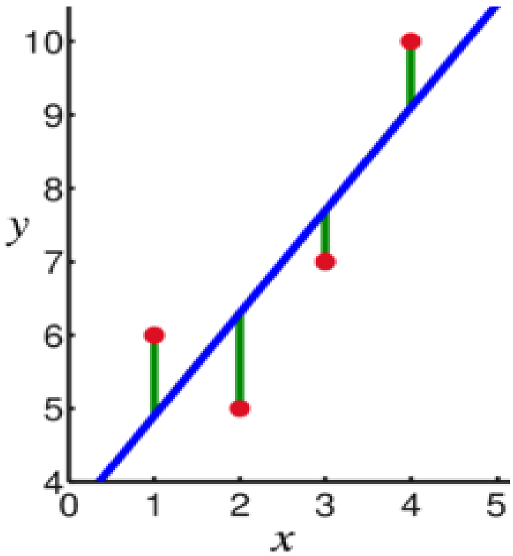
To predict a result with features:

ex) hours → "x" & scores → "y"

Make a Linear Hypothesis

: H(x) = Wx + b

<Image from: https://en.wikipedia.org/wiki/Linear_regression>



How to know what's good hypothesis?

→Use "Cost Function"(Lost Function)

: $(H(x) - y)^2$

Totally

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

→H(x) = Wx + b

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

Goal:

Minimize cost(W,b)

= Find "W & b" that minimizes the cost!!

<How to minimize cost>

Simplified hypothesis

H(x) = Wx

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

If) x = 1 → y = 1

x = 2 → y = 2

x = 3 → y = 3

If.... W = 1 : H(x) = x

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

If.... W = 2 : H(x) = 2x

Cost = $\frac{(2*1-1)^2 + (2*2-2)^2 + (2*3-3)^2}{3}$ = 4.67

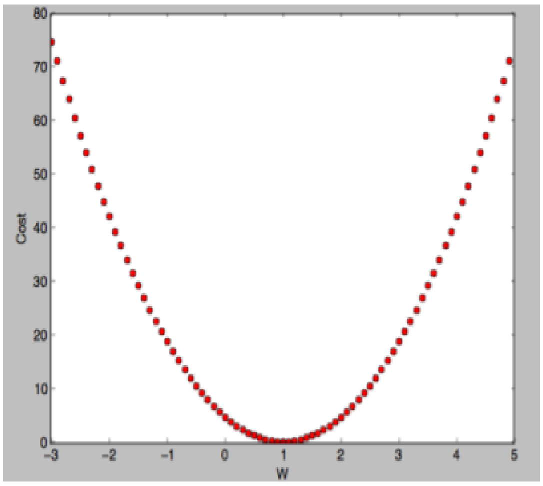
If.... W = 0 : H(x) = 0

Cost = $\frac{(0*1-1)^2 + (0*2-2)^2 + (0*3-3)^2}{3}$ = 4.67

If.... W = -1 : H(x) = -x

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

Then, what "cost(W)" looks like??



Our goal is to find the minimum "W" that minimizes the "Cost"

But how??

→ "Gradient descent algorithm"

<Gradient Descent algorithm>

1) Minimize cost function

2) Gradient descent is used many minimization problems

3) For a given cost function, cost(W,b), it will find W,b to minimize cost

4) It can be applied to more general function:

→cost(w1,w2, …) : Even if with the cost function that has many "W"

< How it works?>

1) Start with initial guesses

   1. Start at any value   ex) (0,0)
   2. Keep changing "W" and "b" a little bit to try and reduce "Cost(W,b)"

2) Each time you change the parameters, you select the gradient which reduces cost(W,b) the most possible

3) Repeat

4)Do so until you converge to a local minimum

5) Has an interesting property

< How tensorflow knows the gradient?>

[For simple proof → cost(W) = $\frac{1}{2m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$]

Formal definition

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

> α : learning rate
> → constant number(At this time α = 0.1)

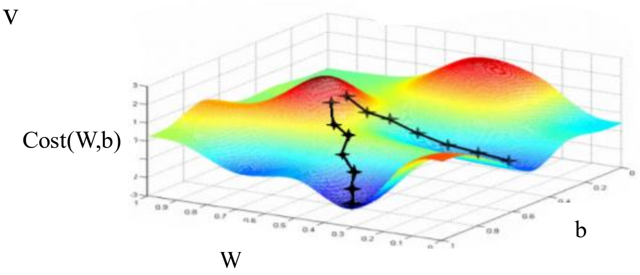
> By this equation,
> 1) If you guess a bigger number than the minimum W
> : Tensorflow will make you try a <u>smaller</u> number
> →Because if you guessed a bigger number, $(-\alpha\frac{\partial}{\partial W}cost(W))$ will be a negative number so that the next W will be smaller than the number you guessed.
> 2) If you guess a smaller number than the minimum W
> : Tensorflow will make you try a <u>bigger</u> number
> →Because if you guessed a smaller number, $(-\alpha\frac{\partial}{\partial W}cost(W))$ will be a positive number

After derivative

Gradient Descent Algorithm

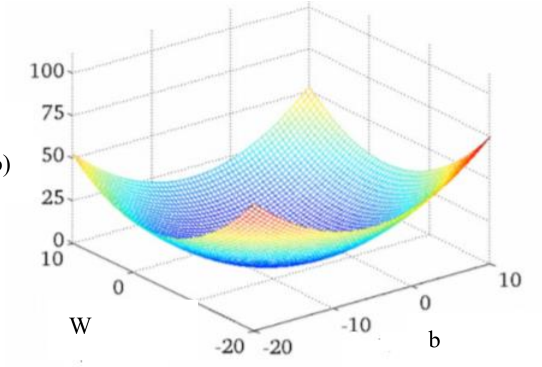
W ≒ W − $\alpha\frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})x^{(i)}$

v



> <u>You should be careful not to make your cost function like this convex function</u>
> →Because you can get different "W & b" depending on where you start (= where you start to guess)

Cost(W,b)



W

b

<u>This function is the right form you need to get</u>

→Because even if you start from any value, you will get the right “W&b”

<Multi-variable Linear Regression>

ex)

	Quiz 1(X_1)	Quiz2(X_2)	Quiz3(X_3)	Final(Y)
Instance 1	70	80	70	150
Instance 2	90	85	90	185
Instance 3	90	90	90	180
Instance 4	95	98	100	195
Instance 5	70	65	75	140

:

:

Hypothesis for multi-variable linear regression

$H(X_1, X_2, X_3) = W_1X_1 + W_2X_2 + W_3X_3 + \dots W_nX_n + b$

Cost function for multi-variable linear regression

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

Hypothesis using Matrix

$W_1X_1 + W_2X_2 + W_3X_3 + \dots W_nX_n$

$$\left.(x_1 \quad x_2 \quad x_3) \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = (x_1w_1 + x_2w_2 + x_3w_3)\right\} \text{Only one instance}$$

➔ H(x) = XW

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

[5 , 3] [3 , 1] [5 , 1] (It should be the same number)

(_ = instance number, _ = variable number, _ = Y number)

(_ is usually stated as (n) or (-1) in Numpy, (None) in tensorflow)

➔ H(x) = XW

Hypothesis in theory and tensorflow

--- Theory

: H(x) = Wx + b

--- Implementation (Tensorflow)

: H(X) = XW (Because of matrix)

<Logistic (regression) classification >

1. Binary Classification = 0 or 1 encoding

- Spam Email Detection: Spam (1)or Ham (0)
- Facebook feed: Show (1) or Hide (0)
- Credit Card Fraudulent Transaction detection: Legitimate (0) or Fraud (1)

2. Why we do not use Linear regression on "0 or 1" encoding?

- We know that Y is "0 or 1"

  But, H(x) = Wx + b
- Hypothesis can give values larger than 1 or less than 0

  ex) If "W = 0.5" and "x = 100" → hypothesis = 50 >> 1

3. So, we use "Logistic Hypothesis"

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

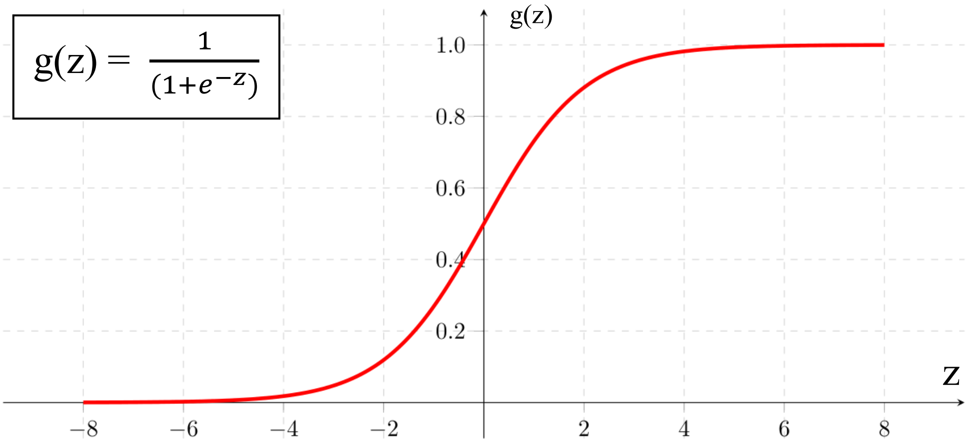


Image from: https://commons.wikimedia.org/wiki/File:Sigmoid-function-2.svg

→ Logistic function

→ Sigmoid function (Sigmoid: Curved in two direction)

Logistic Hypothesis
z = Wx & H(x) = g(z)
→ $H(x) = \frac{1}{(1+e^{-W^T X})}$

<Cost function & Gradient decent in logistic classification>

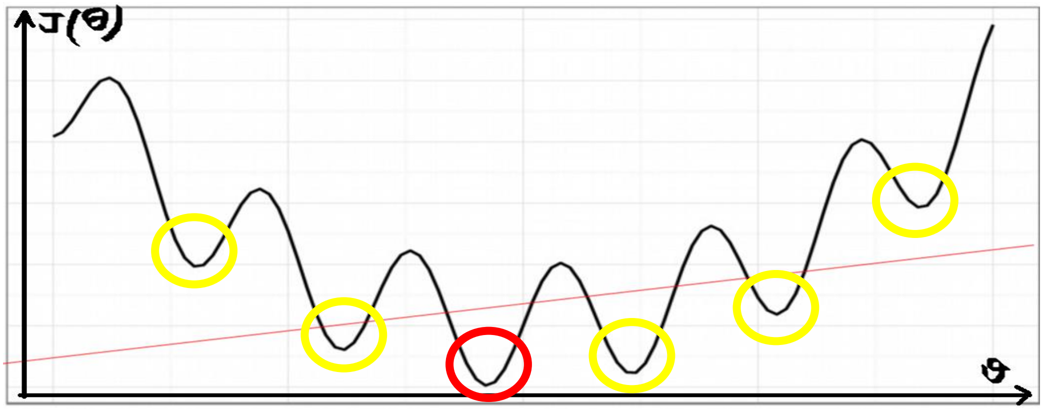
Cost function in linear regression = $\frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$

if) H(x) = Wx + b

→ Wherever you start, you will meet the minimum value

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

→You cannot meet the global minimum because there are lots of local minimum



<Image from https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148>

So, we need to use "another cost function"

- New cost function for logistic regression

cost(W) = $\frac{1}{m}\sum_{i=1}^{m} \quad c(H(x), y)$

$$c(H(x),y) \begin{cases} -\log(H(x)) & \rightarrow \text{When "y = 1"} \\ -\log(1 - H(x)) & \rightarrow \text{When "y = 0"} \end{cases}$$

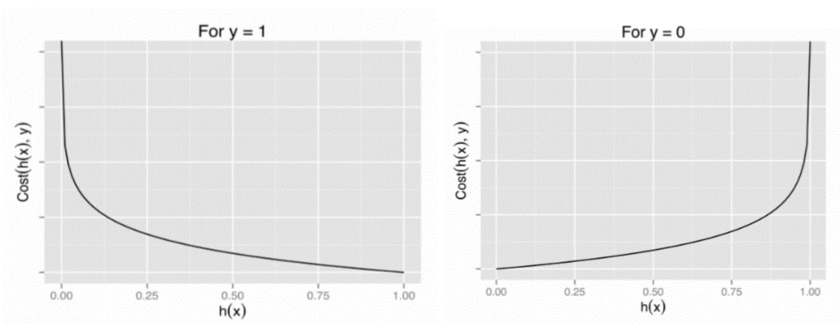


Image from : https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148>

<If we made c(H(x),y) in one sentence…>

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

So, minimize cost with Gradient decent algorithm

$$\text{Cost}(W) = \frac{1}{m}\sum_{i=1}^{m}[-\ y\log(H(x))\ -\ (1-y)\log(1\ -\ H(x))]$$

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

<Problem that I met>

1) How to install tensorflow

   <Simplest way>

   1. If you have installed a python → Uninstall it
   2. Go to anaconda.com → install "**64-Bit Graphical Installer(466 MB)**" (Because I'm using Windows)
   3. Open anaconda command prompt → type "**pip install tensorflow**"

   !!!!COMPLETE!!!!

   Now, I will use "Jupyter notebook"

   1. Type "jupyter notebook" in the anaconda command prompt
   2. Click "**New → Python3**" when the page pop up (See the right part)
   3. If you are using Tensorflow 2.0

      →type "import tensorflow.compat.v1 as tf" & "tf.disable_v2_behavior()"

   !!!!Now Ready to Start!!!!

2) How to use TensorFlow (= TensorFlow Mechanics)

   1. Build graph using TensorFlow operations
   2. Feed data and run graph (operation)

      →**sess.run(op,feed_dict={x:x_data})**

   3. Update variables in the graph (then, return values)

<Materials from>

-1. Andre Ng's ML class

1) https:\\class.coursera.org/ml-003/lecture

2) holehouse.org → mlclass

-2. Sung Kim (YouTube channel)

Code: https://github.com/hunkim/DeepLearningZeroToAll/

-3. Convolutional Neural Networks for Visual Recognition

1)http://cs23In.stanford.edu/

-4. TensorFlow

1)https://www.tensorflow.org

2)https://github.com/aymericdamien/TensorFlow-Examples