딥러닝 Tensorflow 기본

신경망과 딥러닝 *history*

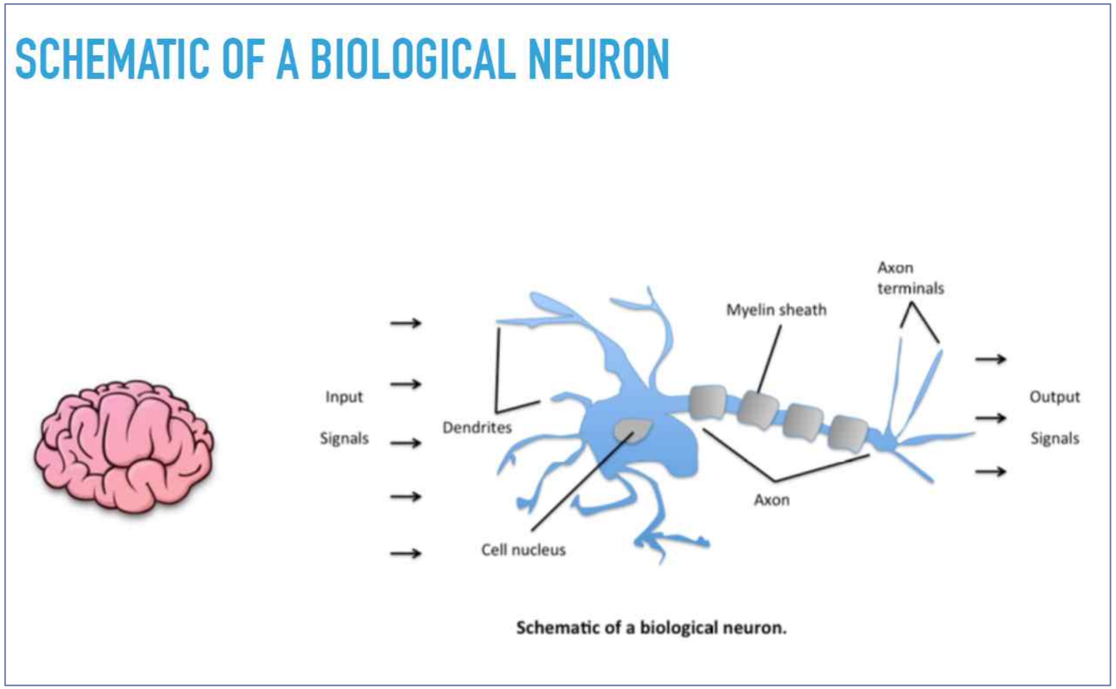
퍼셉트론

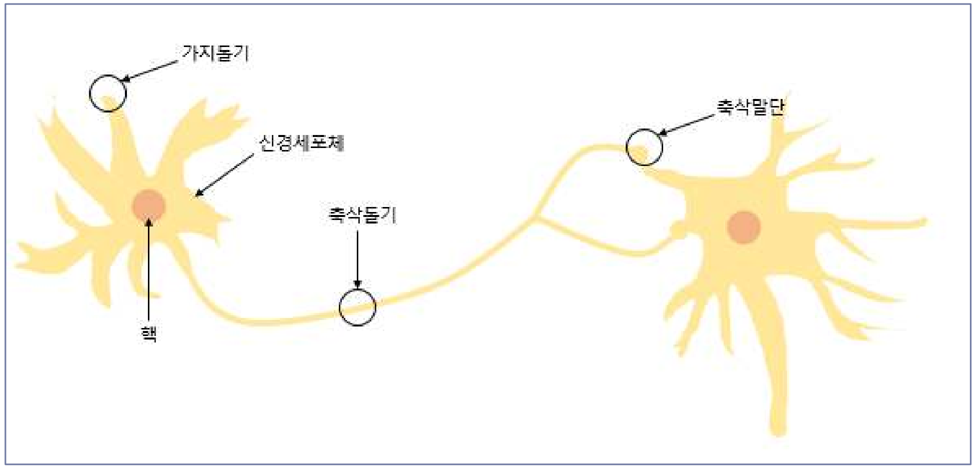
위키백과, 우리 모두의 백과사전.

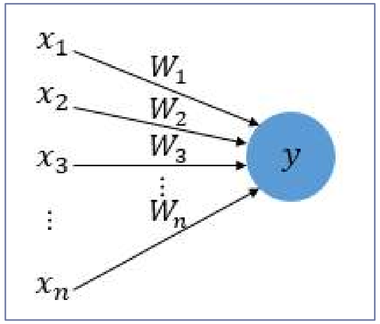
퍼셉트론(perceptron)은 인공신경망의 한 종류로서, 1957년에 코넬 항공 연구소(Cornell Aeronautical Lab)의 프랑크 로젠블라트 (Frank Rosenblatt)에 의해 고안되었다. 이것은 가장 간단한 형태의 피드포워드(Feedforward) 네트워크 - 선형분류기- 로도 볼 수 있다.

퍼셉트론이 동작하는 방식은 다음과 같다. 각 노드의 가중치와 입력치를 곱한 것을 모두 합한 값이 활성함수에 의해 판단되는데, 그 값이 임계치(보통 0)보다 크면 뉴런이 활성화되고 결과값으로 1을 출력한다. 뉴런이 활성화되지 않으면 결과값으로 -1을 출력한다.

마빈 민스키와 시모어 페퍼트는 저서 "퍼셉트론"에서 단층 퍼셉트론은 XOR 연산이 불가능하지만, 다층 퍼셉트론으로는 XOR 연산이 가능함을 보였다.

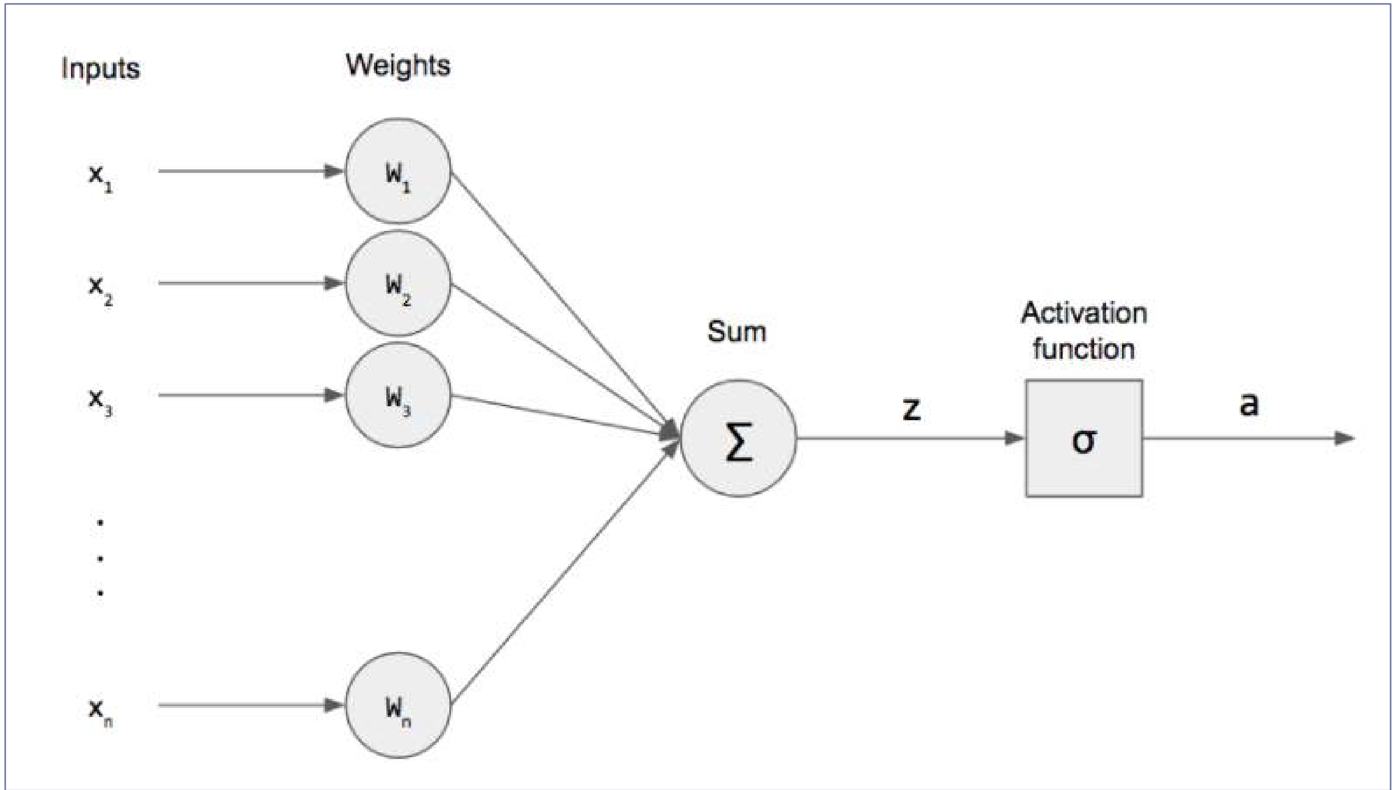






신경 세포 뉴런의 입력 신호와 출력 신호가 퍼셉트론에서 각각 입력 값과 출력 값에 해당된다.

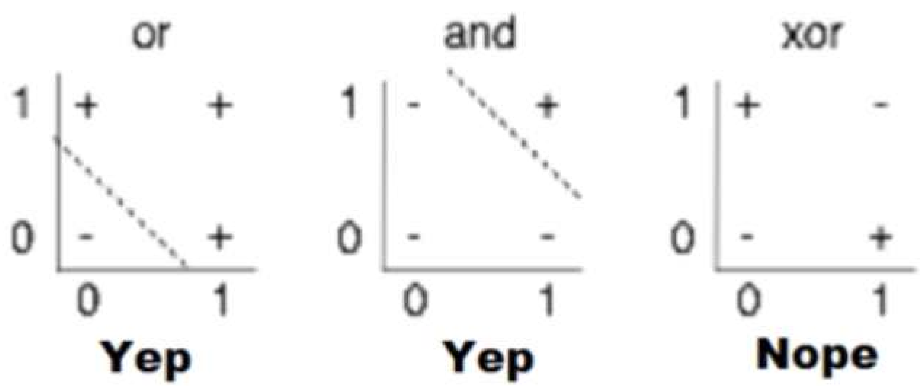
Perceptron은 neuron의 구조를 그대로 본따서 만들어졌다



https://roboreport.co.kr/%EC%8B%A0%EA%B2%BD%EB%A7%9D-%EC%9D%B4%EB%A1%A0-%EC%9D%B4%ED%95%B4%ED%95%98%EA%B8%B0-1-perceptron/

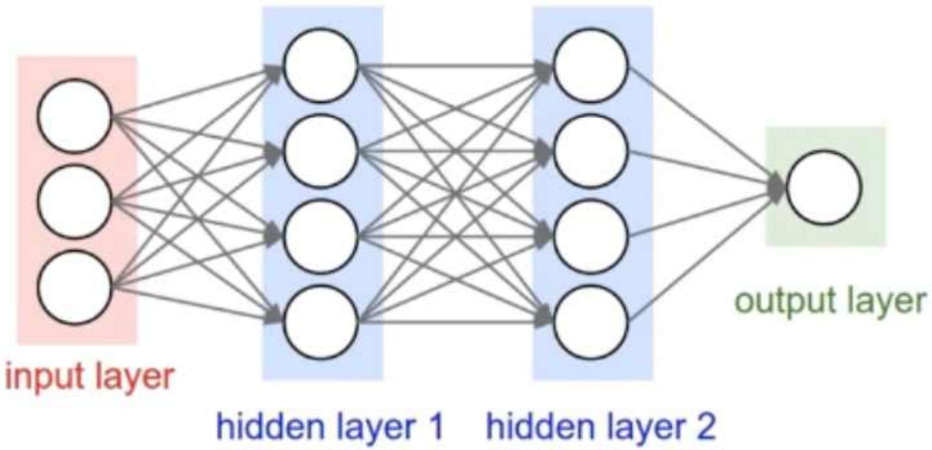
XOR PROBLEM

(Simple) XOR problem: linearly separable?



MARVIN MINSKY, 1969

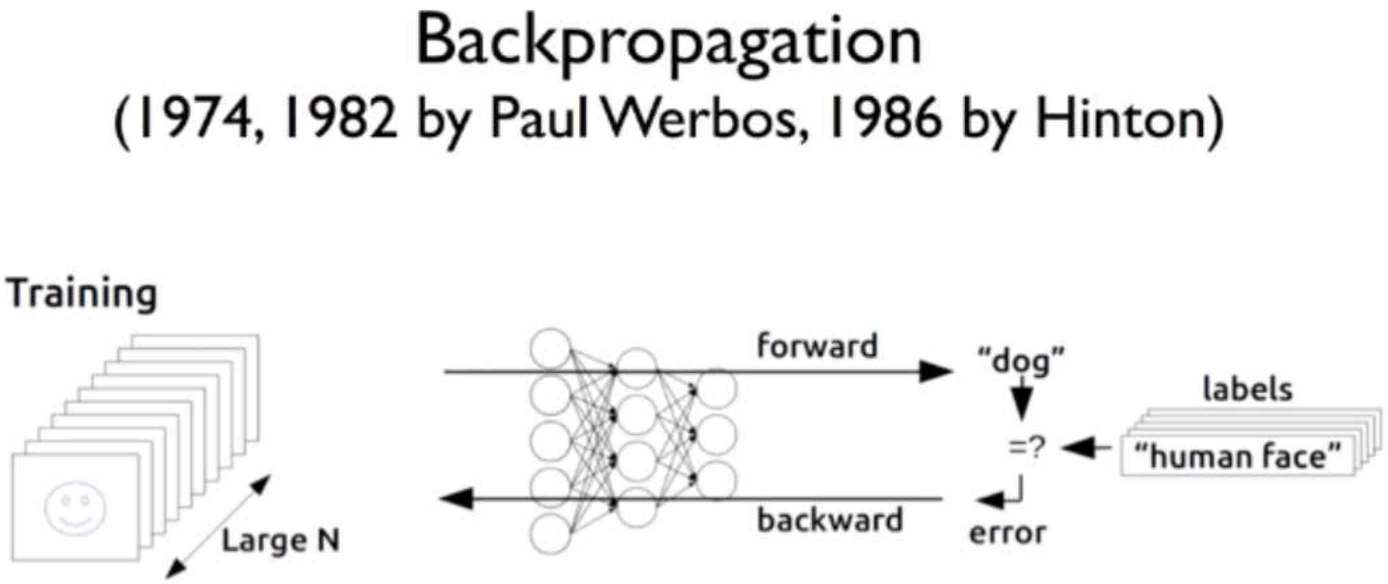
"No one on earth had found a viable way to train*"



*Marvin Minsky, 1969

http://cs231n.github.io/convolutional-networks/

BACKPROPAGATION



https://devblogs.nvidia.com/parallelforall/inference-next-step-gpu-accelerated-deep-learning/

TERMINATOR 2 (1991)

Terminator 2 (1991)

JOHN: Can you learn? So you can be... you know. More human. Not such a dork all the time.

TERMINATOR: My CPU is a neural-net processor... a learning computer. But **Skynet** presets the switch to "read-only" when we are sent out alone.

We'll learn how to **set** the neural net

...

TERMINATOR Basically. (starting the engine, backing out) The **Skynet** funding bill is passed. The system goes on-line August 4th, 1997. Human decisions are removed from strategic defense. **Skynet** begins to learn, at a geometric rate. It becomes **self-aware** at 2:14 a.m. eastern time, August 29. In a panic, they try to pull the plug.

SARAH: And **Skynet** fights back.

TERMINATOR: Yes. It launches its ICBMs against their targets in Russia.

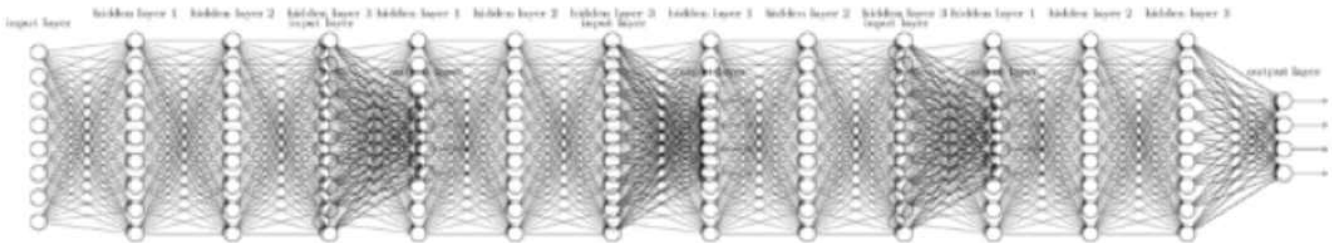
SARAH: Why attack Russia?

TERMINATOR: Because **Skynet** knows the Russian counter-strike will remove its enemies here.

http://pages.cs.wisc.edu/~jerryzhu/cs540/handouts/neural.pdf

A BIG PROBLEM

- Backpropagation just did not work well for normal neural nets with many layers
- Other rising machine learning algorithms: SVM, RandomForest, etc.
- **1995** "Comparison of Learning Algorithms For Handwritten Digit Recognition" by LeCun et al. found that this new approach worked better



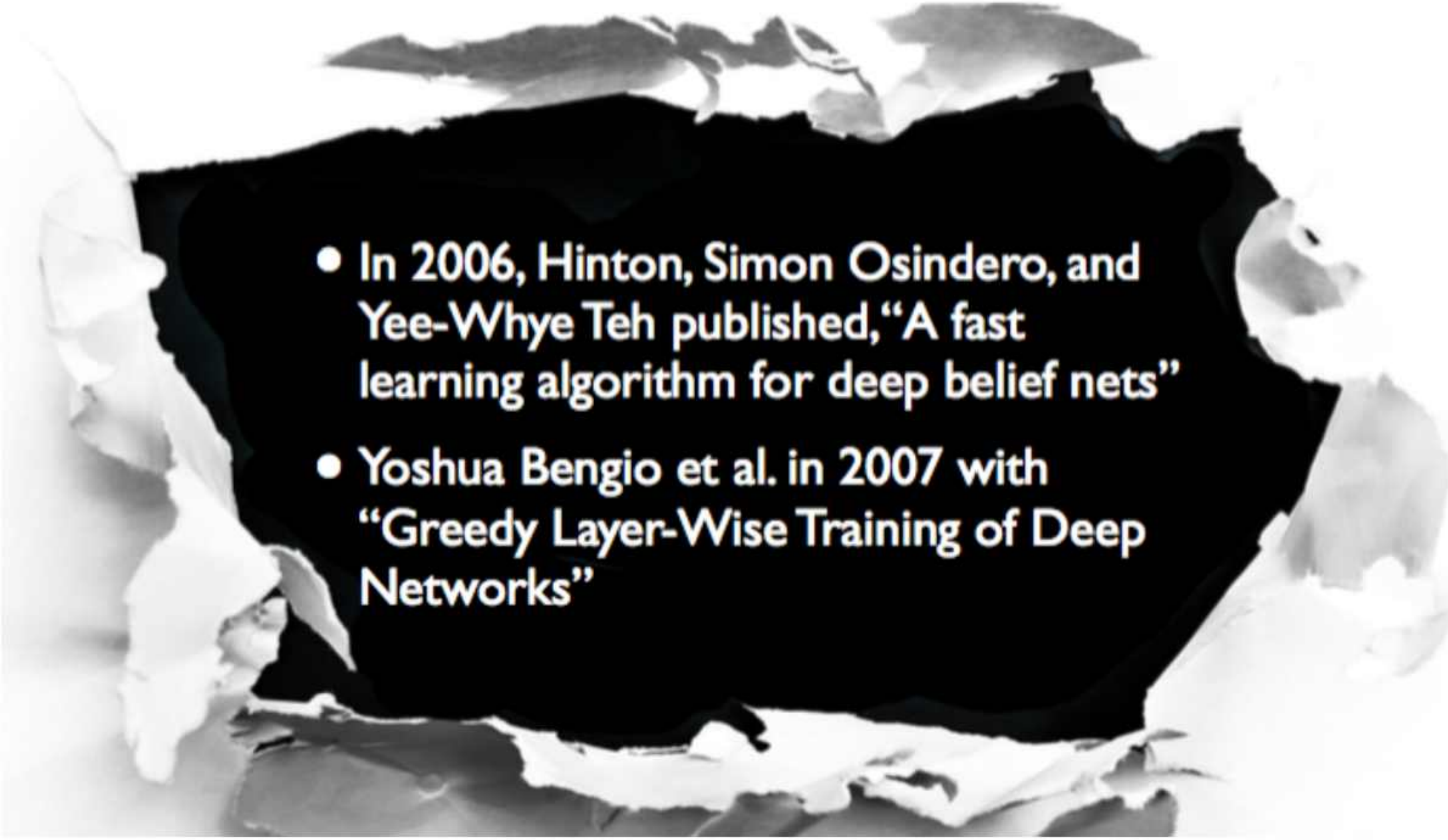
CIFAR

- Canadian Institute for Advanced Research (CIFAR)
- CIFAR encourages basic research *without direct application*, was what motivated **Hinton** to move to Canada in 1987, and funded his work afterward.

CIFAR
CANADIAN INSTITUTE
for ADVANCED RESEARCH

http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning-part-4/

BREAKTHROUGH – HINTON(2006), BENGIO(2007)



BREAKTHROUGH - HINTON(2006), BENGIO(2007)

- Neural networks with many layers really could be trained well, *if the weights are initialized in a clever way* rather than randomly.
- Deep machine learning methods are more efficient for difficult problems than shallow methods.
- Rebranding to ***Deep Nets***, ***Deep Learning***

선형회귀(Linear Regression)

1. 회귀분석 (HTTP://MATH7.TISTORY.COM/118 에서 발췌)

- 점들이 퍼져있는 형태에서 패턴을 찾아내고,
  이 패턴을 활용해서 무언가를 예측하는 분석.
- 새로운 표본을 뽑았을 때 평균으로 돌아가려는 특징이 있기 때문에 붙은 이름
- 회귀(回歸 돌 회, 돌아갈 귀)라는 용어는 일반적으로 '돌아간다'는 정도로만 사용하기 때문에 회귀로부터 '예측'이라는 단어를 떠올리기는 쉽지 않다.

2. LINEAR REGRESSION

- 2차원 좌표에 분포된 데이터를 1차원 직선 방정식을 통해 표현되지 않은 데이터를 예측하기 위한 분석 모델.
- 머신러닝 입문에서는 기본적으로 2차원이나 3차원까지만 정리한다.
- 여기서는 편의상 1차원 직선으로 정리하고 있다. xy축 좌표계에서 직선을 그렸다고 생각하면 된다.

3. HYPOTHESIS

- Linear Regression에서 사용하는 1차원 방정식을 가리키는 용어로, 우리말로는 가설이라고 한다. 수식에서는 h(x) 또는 H(x)로 표현된다.
- 최저점(minimize cost)이라는 정답을 찾기 위한 가정이기 때문에 가설이라고 부를 수 있다.
- H(x) = Wx + b
  ==> x에 대한 1차 방정식

4. COST (비용)

- 앞에서 설명한 **Hypothesis** 방정식에 대한 비용(cost)으로 방정식의 결과가 크게 나오면 좋지 않다고 얘기하고 루프를 돌 때마다 **W**와 **b**를 비용이 적게 발생하는 방향으로 수정하게 된다.
- 미분을 사용해서 스스로 최저 비용을 찾아간다.
- **Gradient Descent Algorithm**을 사용해서 최저 비용을 찾는다.

5. COST 함수

- **Hypothesis 방정식을 포함하는 계산식**
- **현재의 기울기(W)와 절편(b)에 대해 비용을 계산해 주는 함수**
- **W와 b가 변함에 따라 반드시 convex(오목)한 형태로 설계되어야 하는 것이 핵심.**
- **convex하지 않다면, 경사를 타고 내려갈 수 없기 때문에 최저점 계산이 불가능해질 수 있다.**
- **Linear Regression을 비롯한 머신러닝 전체에서 최소 비용을 검색하기 위한 역할 담당**

6. GRADIENT DESCENT ALGORITHM

- 딥러닝의 핵심 알고리즘
- 경사타고 내려가기, 경사하강법 등의 여러 용어로 번역되었다.
- 미분을 사용해서 비용이 작아지는 방향으로 진행하는 알고리즘
- 생각보다 어렵지 않고 간단한 미분 정도만 이해하면 알고리즘 자체는 너무 단순하다.
- 텐서플로우에 포함된 **Optimizer**는 대부분 **Gradient Descent Algorithm**에서 파생된 방법을 사용하고 있다.

회귀(Regression)모델 용어 정리

[1] 선형 회귀(Linear Regression) : 1차 함수, 직선의 방정식

[2] 가중치(Weight) : 입력변수가 출력에 영향을 미치는 정도를 설정,
기울기 값 , 회귀 계수

[3] 편향(Bias) : 기본 출력 값이 활성화 되는 정도를 설정, y 절편,
회귀 계수

[4] 비용함수(Cost Function) : 2차 함수, 포물선의 방정식,
(예측값 - 실제값)^2

cost(비용) = 오차 = 에러 = 손실(loss)
cost(W,b) = (H(x) - y)^2

회귀(Regression)모델 용어 정리

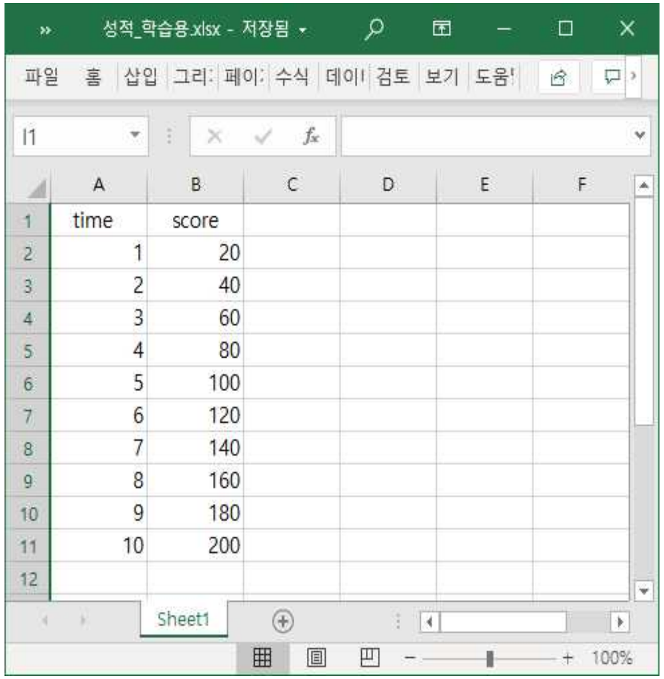
[5] 예측(가설,Hypothesis) 함수 : predict, H(x) : 예측 값,
y값: 답, 결정 값, target, label,   x값 : 입력, 피쳐(feature)

H(X) = W*X + b

[6] 경사 하강법(Gradient Descent Algorithm)
: 비용(cost, loss) 이 가장 작은 Weight(가중치) 값을 구하는 알고리즘

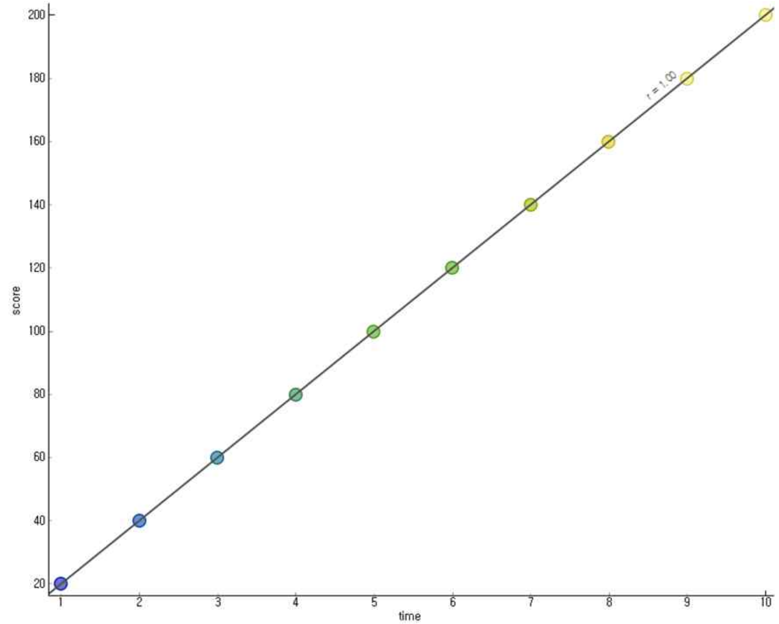
회귀용 데이터

학습시간 과 성적과의 관계?



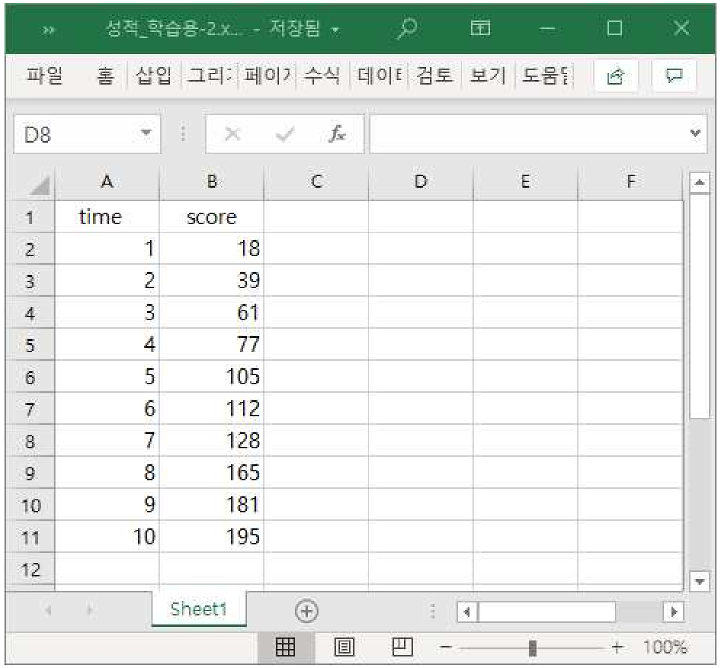
	A	B
1	time	score
2	1	20
3	2	40
4	3	60
5	4	80
6	5	100
7	6	120
8	7	140
9	8	160
10	9	180
11	10	200

time	score
1	20
2	40
3	60
4	80
5	100
6	120
7	140
8	160
9	180
10	200



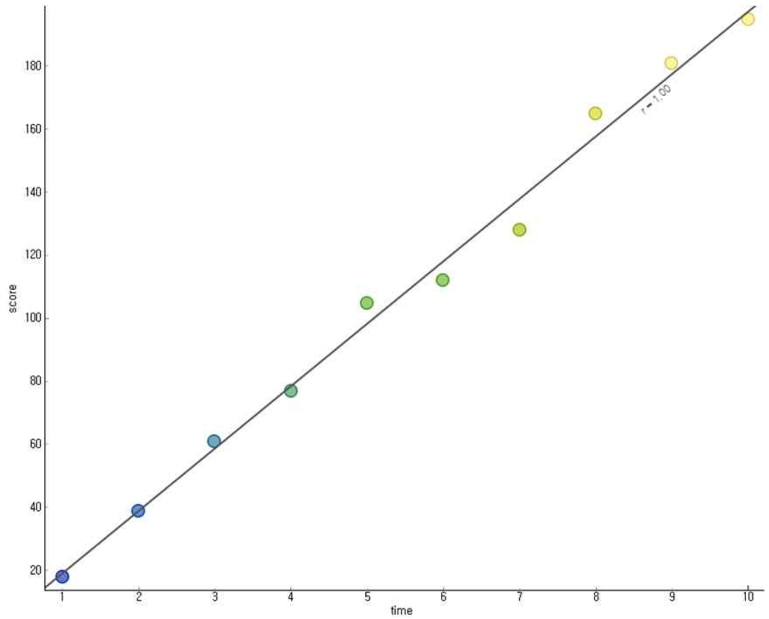
실제 오차가 있는 데이터

오차가 있는 데이터



	A	B	C	D	E	F
1	time	score				
2	1	18				
3	2	39				
4	3	61				
5	4	77				
6	5	105				
7	6	112				
8	7	128				
9	8	165				
10	9	181				
11	10	195				
12						

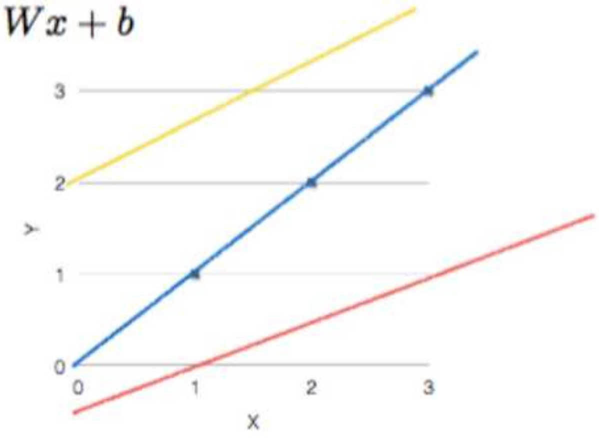
time	score
1	18
2	39
3	61
4	77
5	105
6	112
7	128
8	165
9	181
10	195



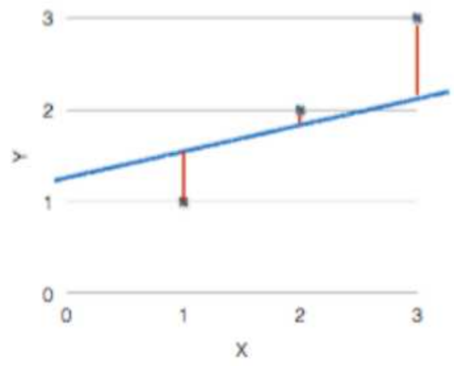
좋은 가설?

(Linear) Hypothesis

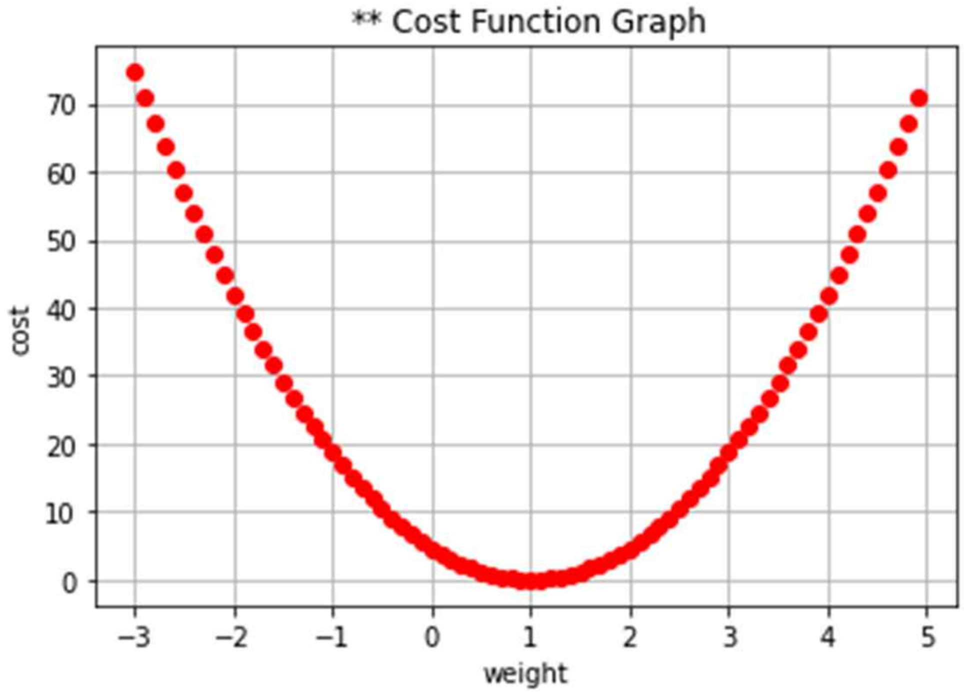
$$H(x) = Wx + b$$



Which hypothesis is better?



비용함수(Cost Function)



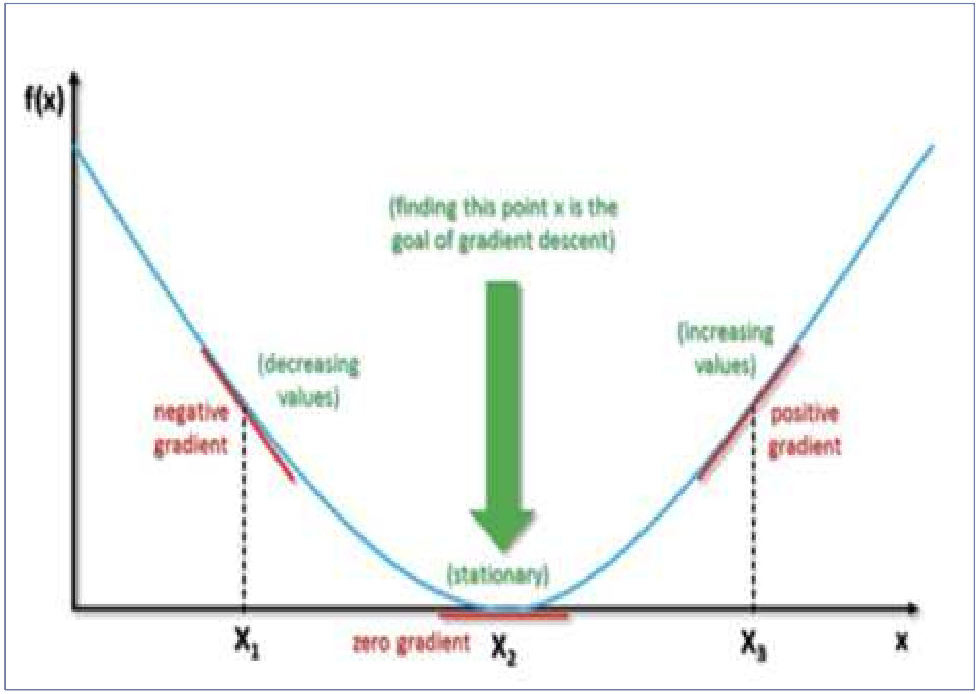
cost(W,b) = (H(x) - y)^2

미분 : 순간 변화량, 기울기, x축으로 1만큼 움직였을 때 y축으로 움직인 거리

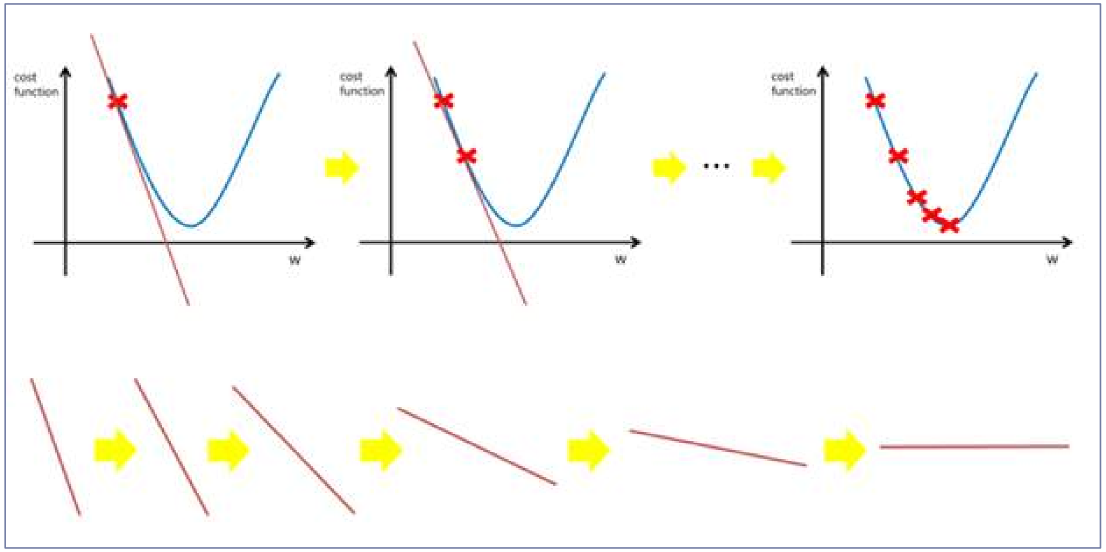
- **함수의 미분 공식 정리** : **f(x) =** $\boldsymbol{x^n}$ **=====> f'(x) = n** $* \boldsymbol{x^{(n-1)}}$

- y = 3 ===> y' = 0
- y = 2*x ===> y' = 2
- y = x^2 ===> y' = 2*x
- y = $(x+1)^2$ ===> y' = 2*(x + 1)
- y = x^2 + 2*x + 1 ===> y' = 2*x + 2
- 곱셈 공식 : (a + b)^2 = a^2 + 2*a*b + b^2

경사하강법

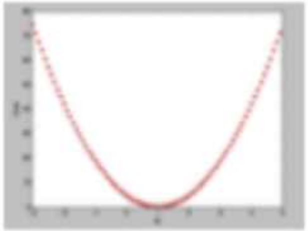


f(x)
(finding this point x is the goal of gradient descent)
(decreasing values)
negative gradient
(increasing values)
positive gradient
(stationary)
zero gradient
X_1
X_2
X_3
x



cost function
w
cost function
w
...
cost function
w

HOW IT WORKS?



- Start with initial guesses
  - Start at 0,0 (or any other value)
  - Keeping changing W and b a little bit to try and reduce cost(W, b)
- Each time you change the parameters, you select the gradient which reduces cost(W, b) the most possible
- Repeat
- Do so until you converge to a local minimum
- Has an interesting property
  - Where you start can determine which minimum you end up

비용함수의 미분 : cost(w) = $(w*x - y)^2$ 의 미분 , hx = w*x + 0

cost(w) = $w^2 * x^2$ - 2*w*x*y + y^2

cost'(w) = 2*w*x^2 - 2*x*y = 2*x*(w*x - y)

= 2*x*(hx - y)

FORMAL DEFINITION

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

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

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

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

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

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

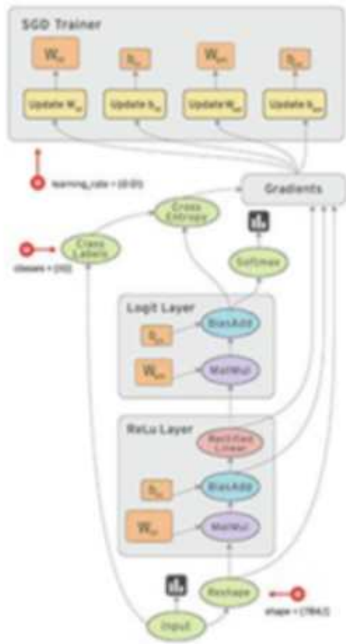
텐서플로우

- TensorFlow™ is an open source software library for numerical computation using data flow graphs.
- Python!

텐서플로우 DATA FLOW GRAPH

What is a Data Flow Graph?

- Nodes in the graph represent mathematical operations
- Edges represent the multidimensional data arrays (tensors) communicated between them.



텐서플로우 속성 : RANK

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

Rank	Math entity	Python example
0	Scalar (magnitude only)	`s = 483`
1	Vector (magnitude and direction)	`v = [1.1, 2.2, 3.3]`
2	Matrix (table of numbers)	`m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]`
3	3-Tensor (cube of numbers)	`t = [[[2], [4], [6]], [[8], [10], [12]], [[14], [16], [18]]]`
n	n-Tensor (you get the idea)	`....`

텐서플로우 속성 : SHAPE

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

Rank	Shape	Dimension number	Example
0	[]	0-D	A 0-D tensor. A scalar.
1	[D0]	1-D	A 1-D tensor with shape [5].
2	[D0, D1]	2-D	A 2-D tensor with shape [3, 4].
3	[D0, D1, D2]	3-D	A 3-D tensor with shape [1, 4, 3].
n	[D0, D1, ... Dn-1]	n-D	A tensor with shape [D0, D1, ... Dn-1].

텐서플로우 속성 : DTYPE

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

Data type	Python type	Description
DT_FLOAT	tf.float32	32 bits floating point.
DT_DOUBLE	tf.float64	64 bits floating point.
DT_INT8	tf.int8	8 bits signed integer.
DT_INT16	tf.int16	16 bits signed integer.
DT_INT32	tf.int32	32 bits signed integer.
DT_INT64	tf.int64	64 bits signed integer.

...

데이터 타입 비교

차원	R	Python	Numpy / Pandas	Tensorflow
0 차원	스칼라(scalar) : 숫자/NA/NULL/문자열/ 진리값/Factor	숫자형(number) 문자열(string)		0-D Tensor
1 차원	벡터(vector) : 한 가지 변수 타입으로 구성	리스트(list) 튜플(tuple)	ndarray / Series	1-D Tensor
2 차원	행렬(matrix) : 한 가지 변수 타입으로 구성		ndarray / x	2-D Tensor
2 차원	데이터 프레임(Data Frame) : 다양한 변수 타입으로 구성		x / DataFrame	
다차원 (2 차원이상)	배열(array) : 2 차원 이상의 행렬		ndarray / x	3-D Tensor / n-D Tensor
다차원	리스트(list) : 서로 다른 데이터 구조 포함	딕셔너리(dictionary)		

MULTI FEATURES

one-variable
one-feature

x (hours)	y (score)
10	90
9	80
3	50
2	60
11	40

multi-variable/feature

x1 (hours)	x2 (attendance)	y (score)
10	5	90
9	5	80
3	2	50
2	4	60
11	1	40

MULTI-FEATURES

Predicting exam score:

regression using three inputs (x1, x2, x3)

multi-variable/feature

x_1 (quiz 1)	x_2 (quiz 2)	x_3 (midterm 1)	Y (final)
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

HYPOTHESIS AND COST FUNCTION

Hypothesis

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

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

Cost function

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

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

MATRIX AND TRANSPOSE

Matrix multiplication

"Dot Product"

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & \\ & \end{bmatrix}$$

Transpose

$$\begin{bmatrix} 6 & 4 & 24 \\ 1 & -9 & 8 \end{bmatrix}^T = \begin{bmatrix} 6 & 1 \\ 4 & -9 \\ 24 & 8 \end{bmatrix}$$

HYPOTHESIS USING MATRIX (1)

$$w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_nx_n$$

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

$$H(X) = XW$$

HYPOTHESIS USING MATRIX (2)

$$H(x_1, x_2, x_3) = x_1w_1 + x_2w_2 + x_3w_3$$

x_1	x_2	x_3	Y
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

Test Scores for General Psychology

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

$$H(X) = XW$$

HYPOTHESIS USING MATRIX (3)

x_1	x_2	x_3	Y
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

Hypothesis using matrix

$$w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_nx_n$$

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

$$H(X) = XW$$

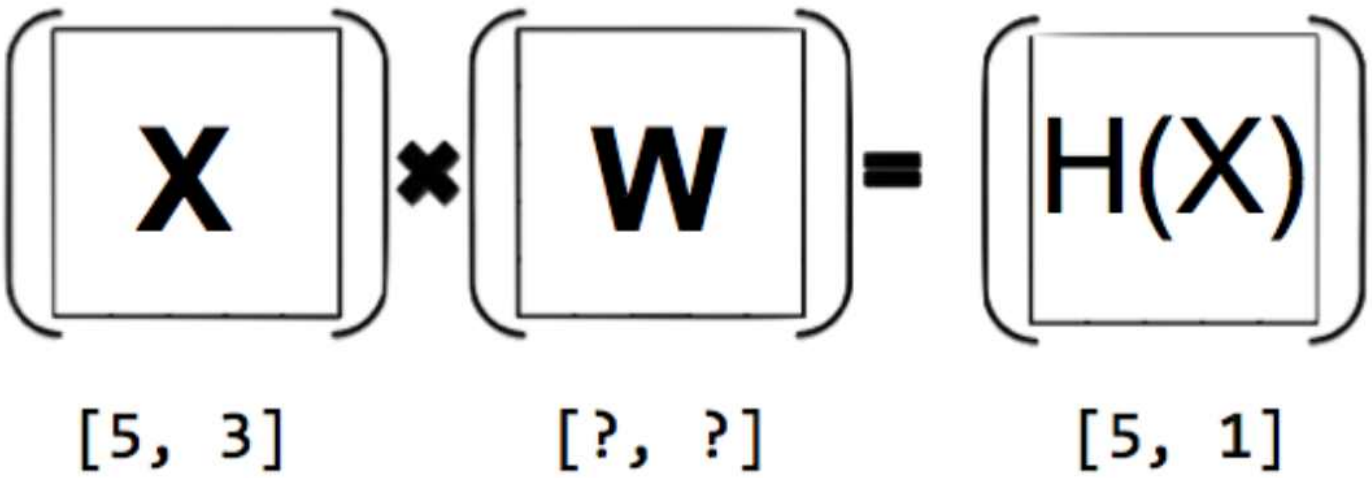
HYPOTHESIS USING MATRIX (4)

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

[5, 3] [3, 1] [5, 1]

$$H(X) = XW$$

HYPOTHESIS USING MATRIX (5)



$$H(X) = XW$$

HYPOTHESIS USING MATRIX (6)

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

[n, 3] [3, 1] [n, 1]

$$H(X) = XW$$

HYPOTHESIS USING MATRIX (7)

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

[n, 3] [?, ?] [n, 2]

$$H(X) = XW$$

HYPOTHESIS USING MATRIX (8)

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

[n, 3] **[3, 2]** [n, 2]

$$H(X) = XW$$

HYPOTHESIS USING MATRIX (9)

- Lecture (theory):

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

- Implementation (TensorFlow)

$$H(X) = XW$$

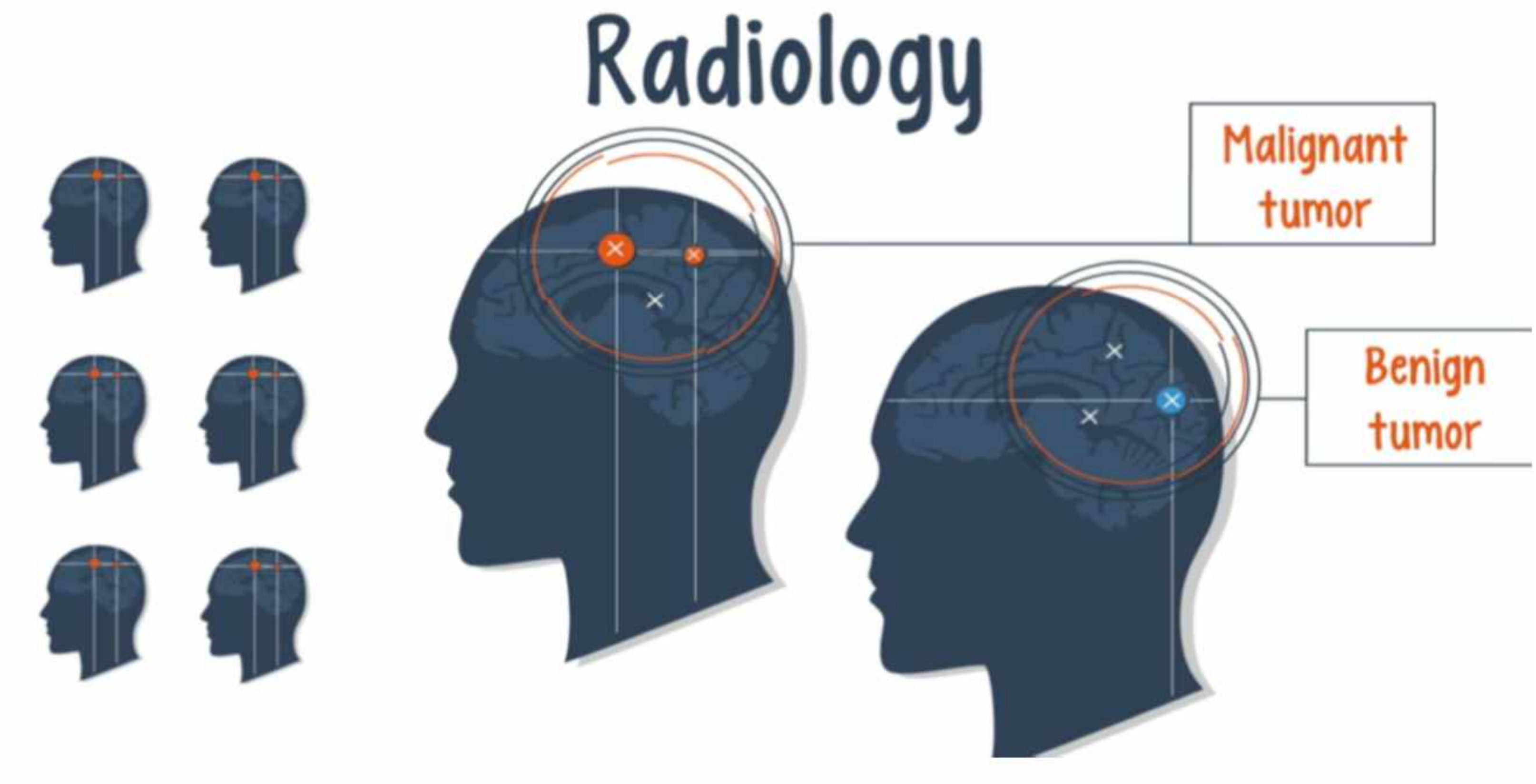
Logistic Classification

CLASSIFICATION AND ENCODING

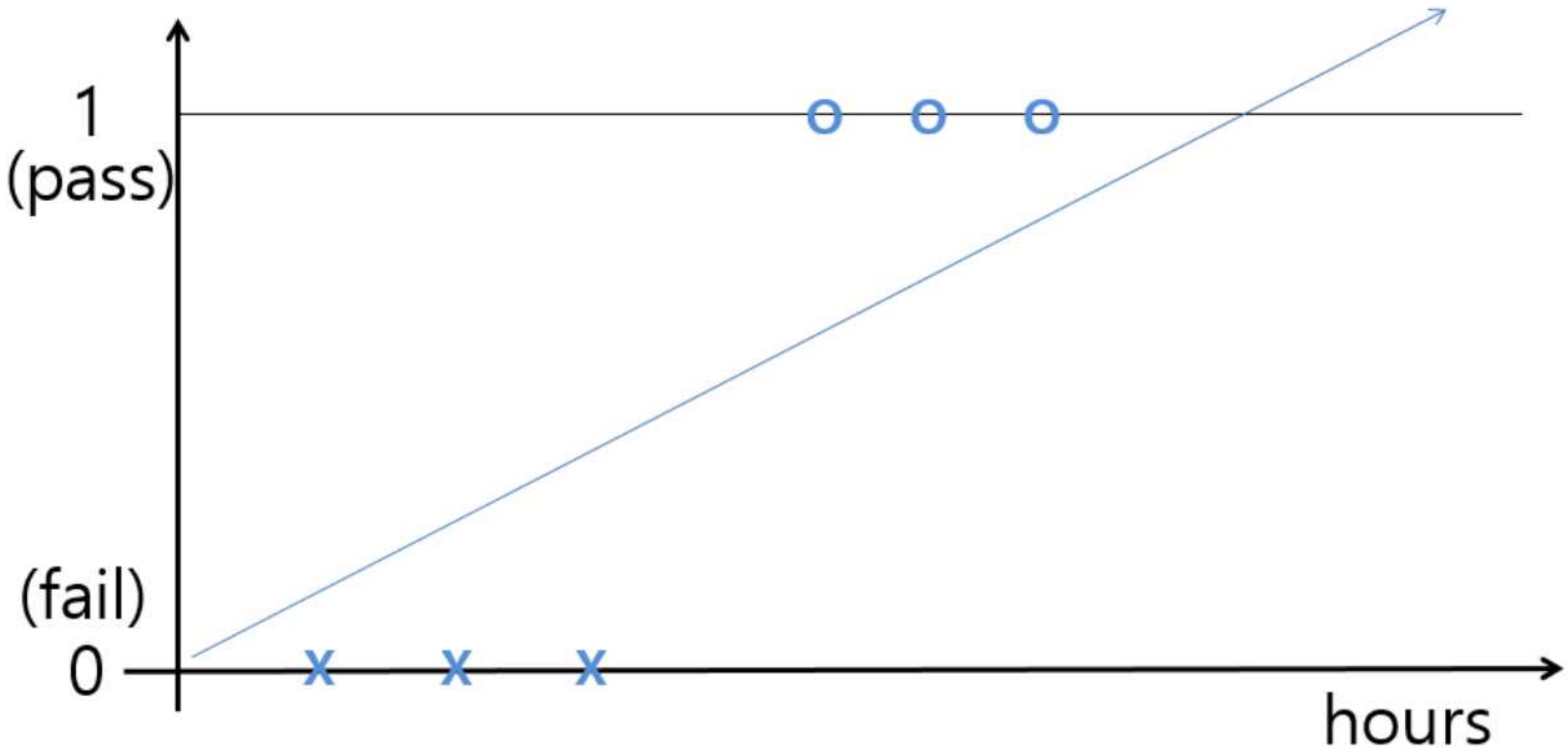
- Spam Detection: Spam or Ham
- Facebook feed: show or hide
- Credit Card Fraudulent Transaction detection: legitimate/fraud

- Spam Detection: Spam (1) or Ham (0)
- Facebook feed: show(1) or hide(0)
- Credit Card Fraudulent Transaction detection: legitimate(0) or fraud (1)

RADIOLOGY



LINEAR REGRESSION?



PROBLEMS

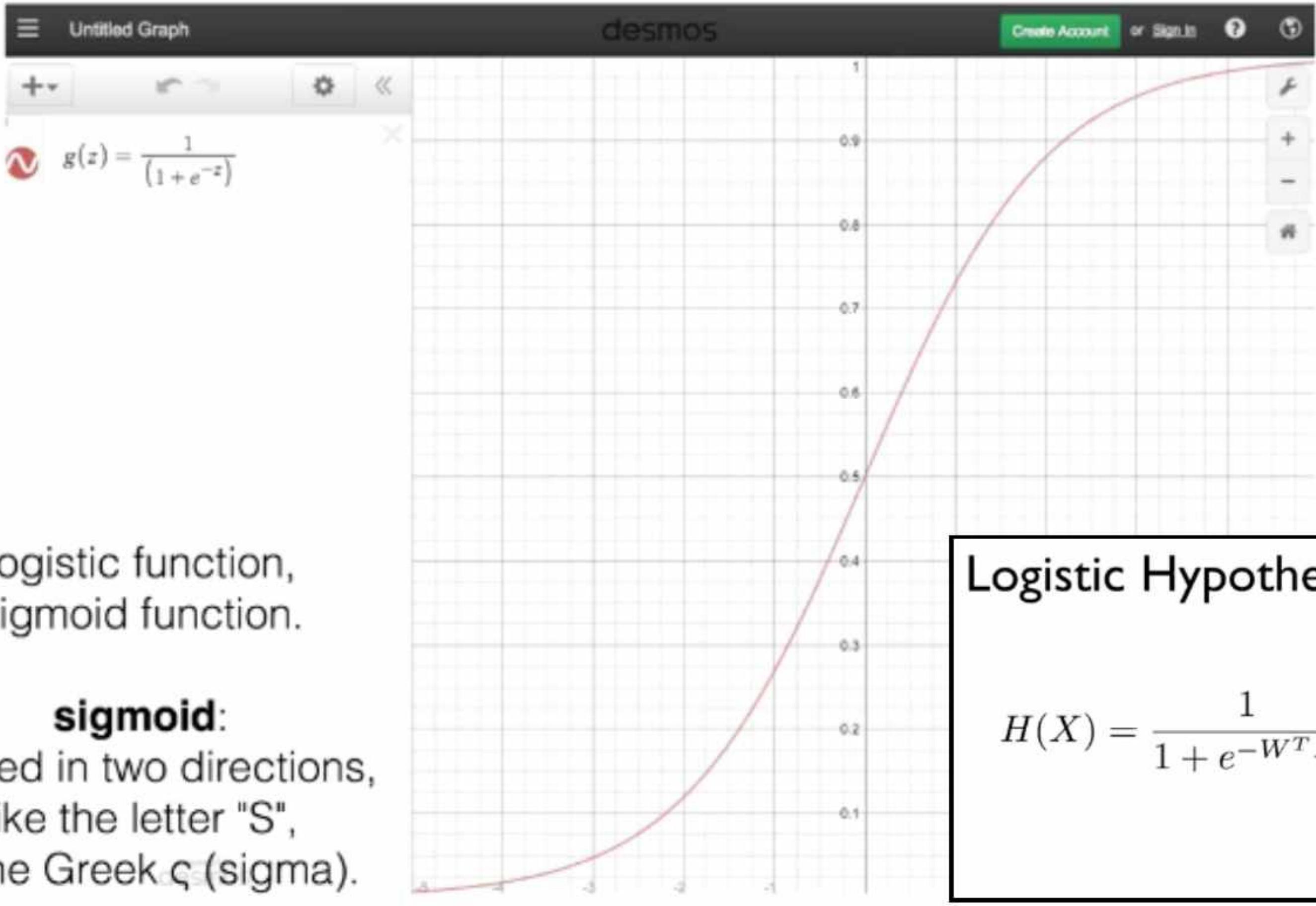
Linear regression

- We know Y is 0 or 1

  $H(x) = Wx + b$

- Hypothesis can give values large than 1 or less than 0

SIGMOID



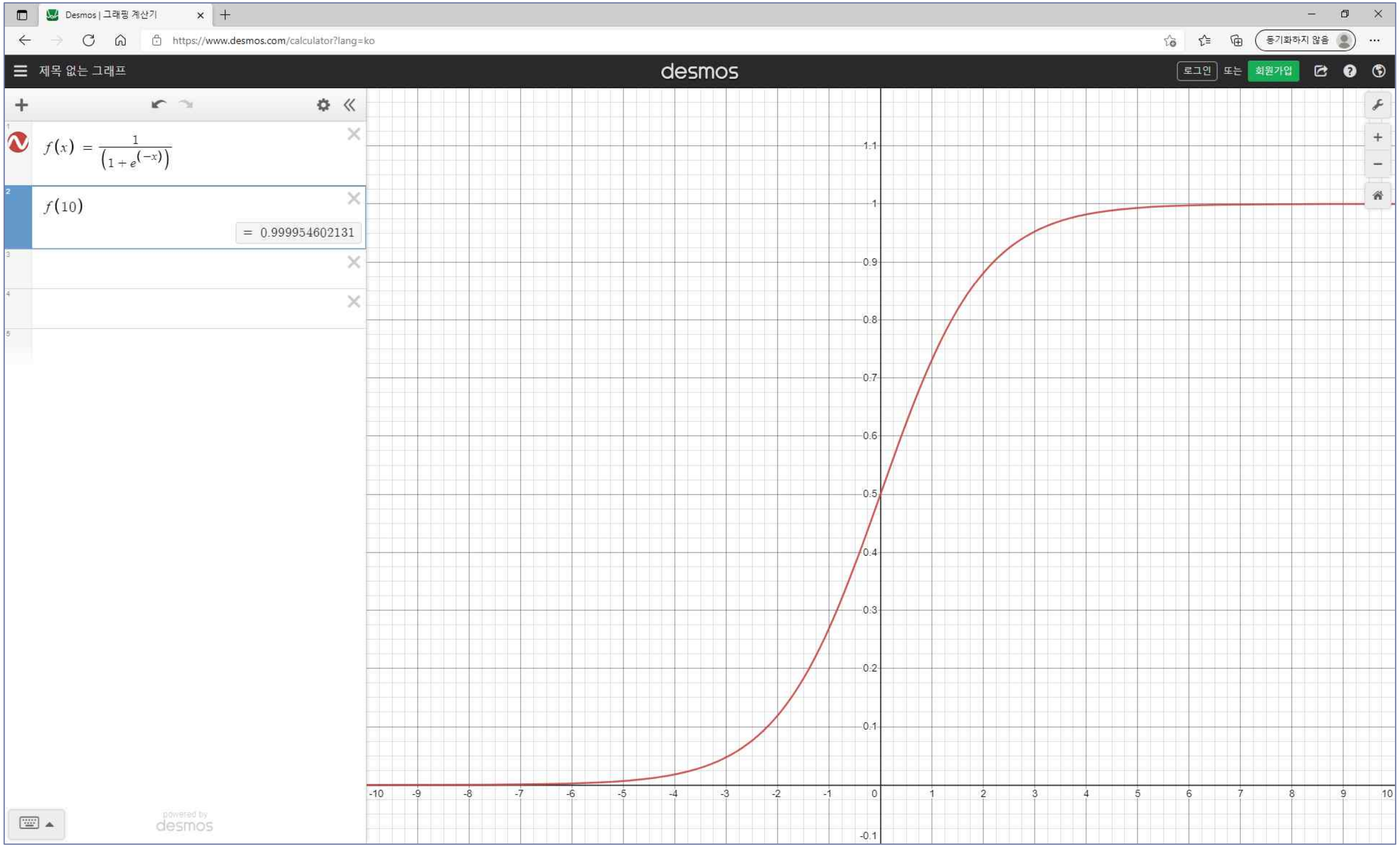
logistic function,
sigmoid function.

sigmoid:
Curved in two directions,
like the letter "S",
or the Greek ς (sigma).

Logistic Hypothesis

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

https://www.desmos.com/



예측함수 : SIGMOID 함수 사용

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

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

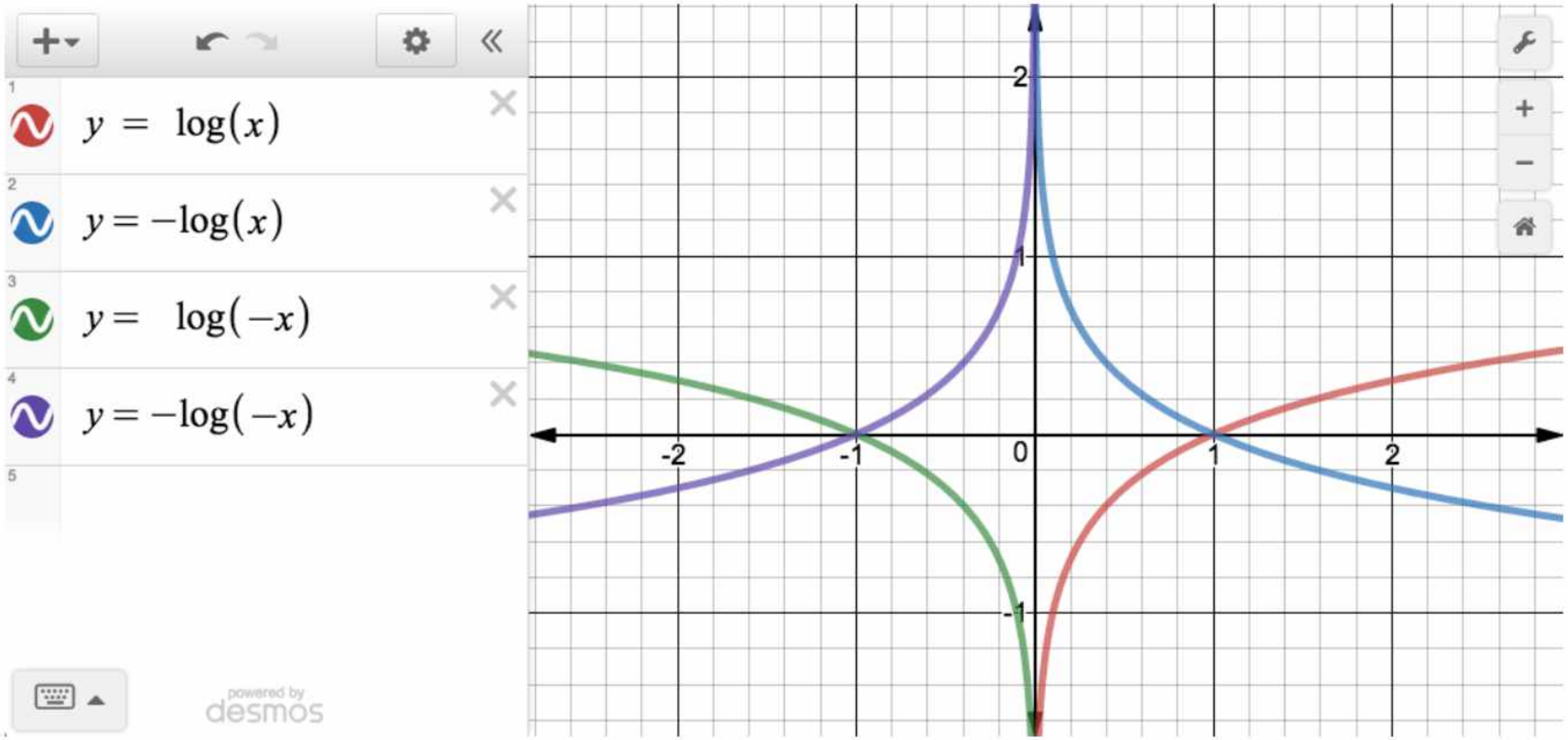
$$\boxed{H(X) = \frac{1}{1+e^{-W^TX}}}$$

NEW COST FUNCTION FOR LOGISTIC

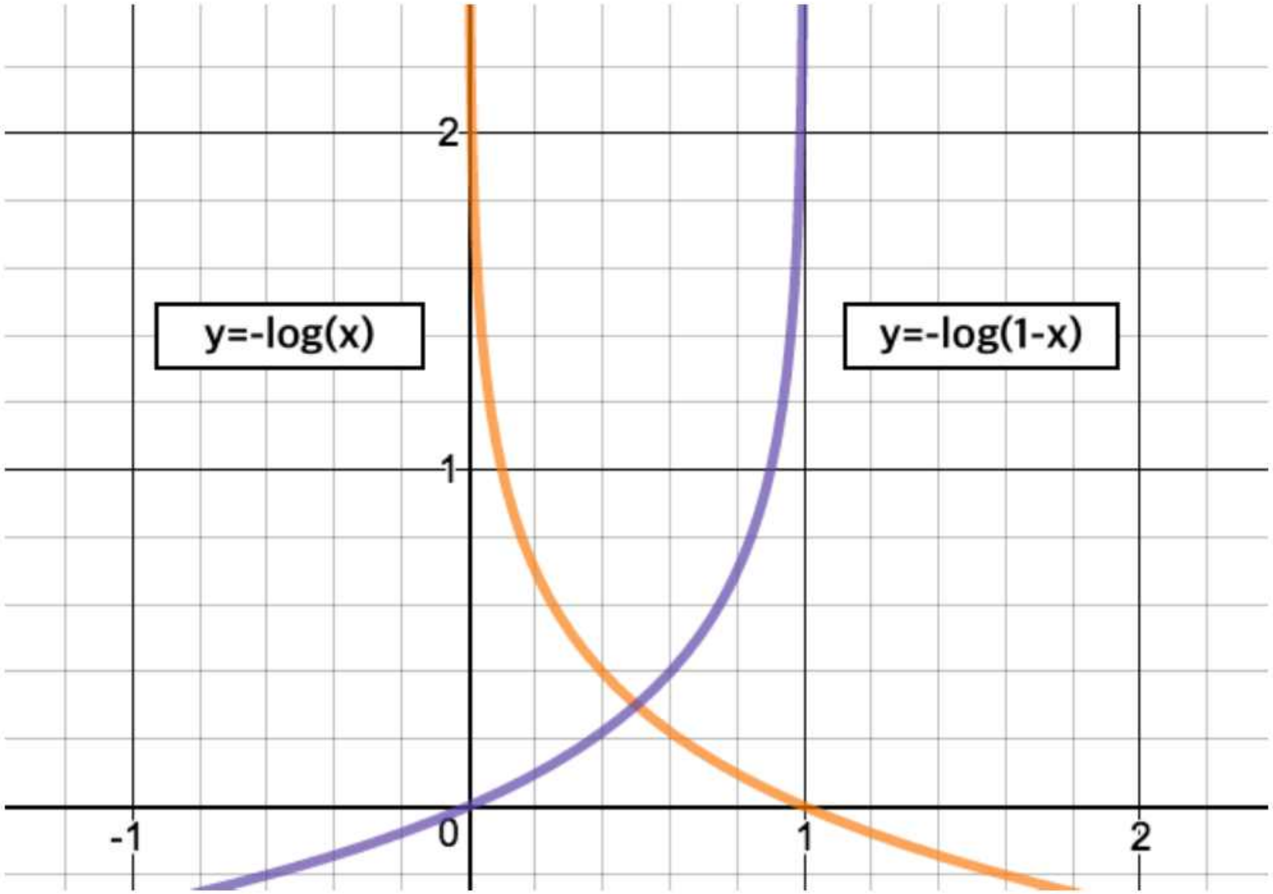
$$cost(W) = \frac{1}{m} \sum c(H(x), y)$$

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

LOG FUNCTION



LOG FUNCTIONS WE NEED TO KNOW



log(1) = 0
log(0) = -∞

-log(1) = 0
-log(0) = +∞

UNDERSTANDING COST FUNCTION

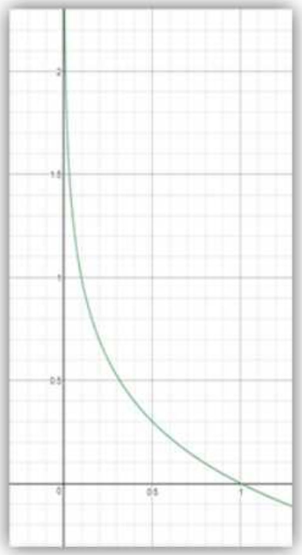
$$C(H(x),y) = \begin{cases} -\log(H(x)) & : y = 1 \\ -\log(1-H(x)) & : y = 0 \end{cases}$$

$y = 1$

$H(x) = 1,\ cost(1) = 0$

$H(x) = 0,\ cost(0) = \infty$

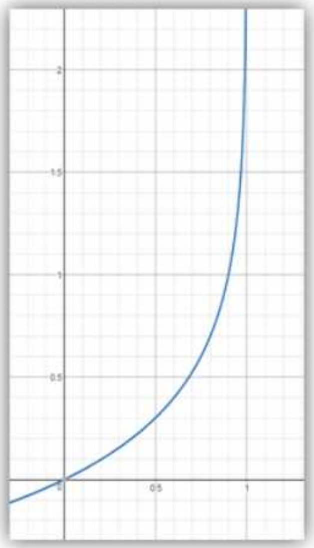
$g(z) = -\log(z)$



$y = 0$

$H(x) = 0,\ cost(0) = 0$

$H(x) = 1,\ cost(1) = \infty$



$g(z) = -\log(1-z)$

COST FUNCTION

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

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

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

GRADIENT DESCENT ALGORITHM

$$cost(W) = -\frac{1}{m}\sum ylog(H(x)) + (1-y)log(1-H(x))$$

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

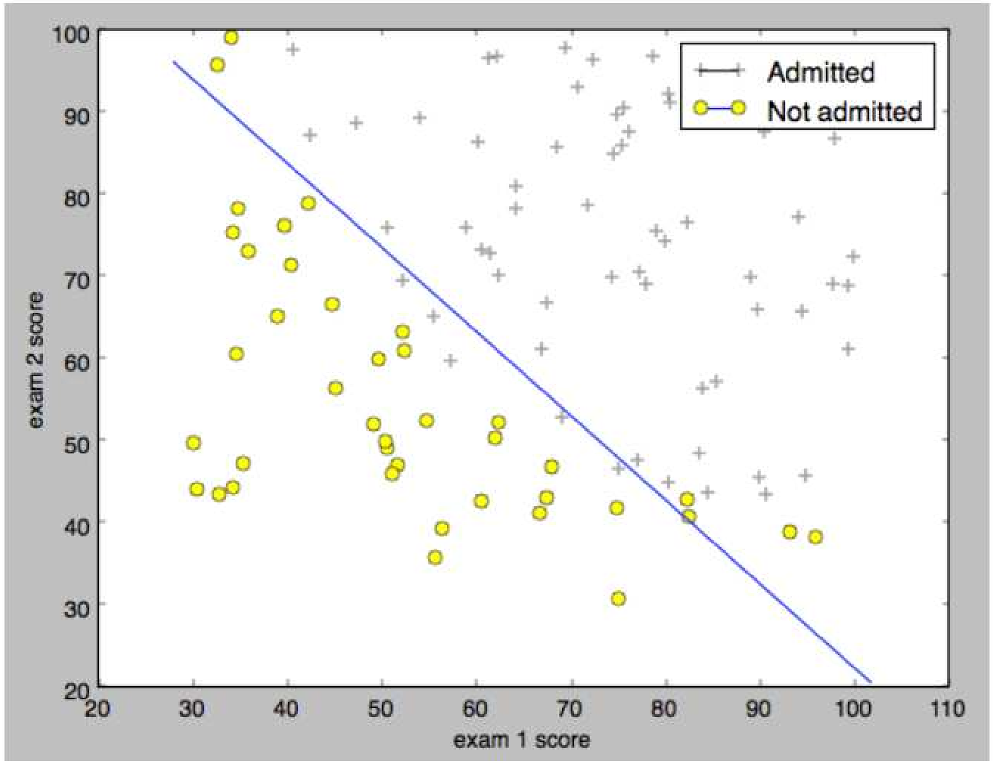
```
# cost function
cost = -tf.reduce_mean(Y*tf.log(hypothesis) + (1-Y)*tf.log(1-hypothesis))

# Minimize
a = tf.Variable(0.1) # Learning rate, alpha
optimizer = tf.train.GradientDescentOptimizer(a)
train = optimizer.minimize(cost)
```

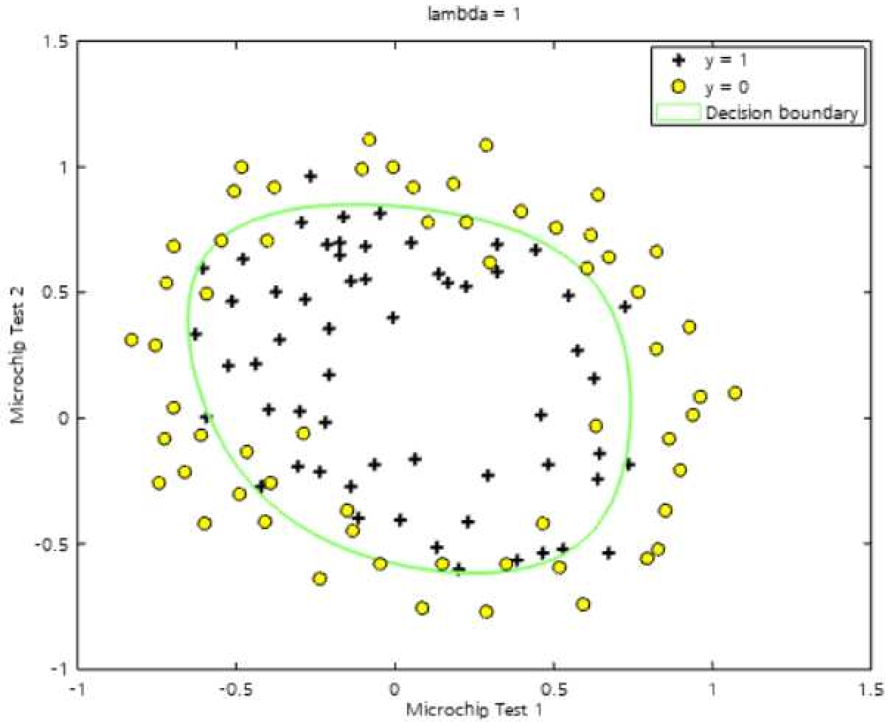
CLASSIFYING DIABETES

Classifying diabetes

-0.411765	0.165829	0.213115	0	0	-0.23696	-0.894962	-0.7	1
-0.647059	-0.21608	-0.180328	-0.353535	-0.791962	-0.0760059	-0.854825	-0.833333	0
0.176471	0.155779	0	0	0	0.052161	-0.952178	-0.733333	1
-0.764706	0.979899	0.147541	-0.0909091	0.283688	-0.0909091	-0.931682	0.0666667	0
-0.0588235	0.256281	0.57377	0	0	0	-0.868488	0.1	0
-0.529412	0.105528	0.508197	0	0	0.120715	-0.903501	-0.7	1
0.176471	0.688442	0.213115	0	0	0.132638	-0.608027	-0.566667	0
0.176471	0.396985	0.311475	0	0	-0.19225	0.163962	0.2	1

DECISION BOUNDARY

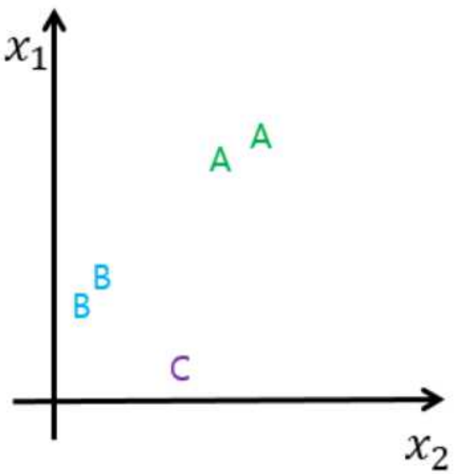


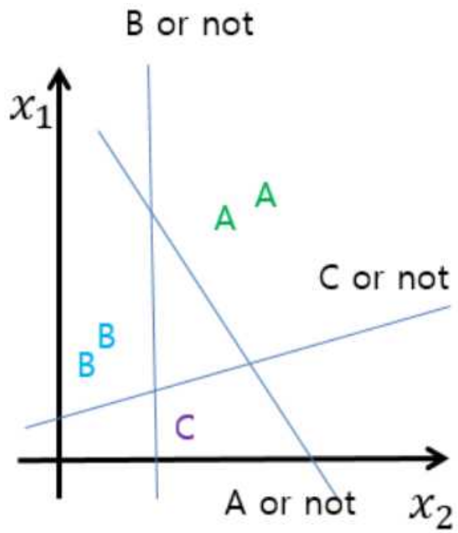


Multi Classification

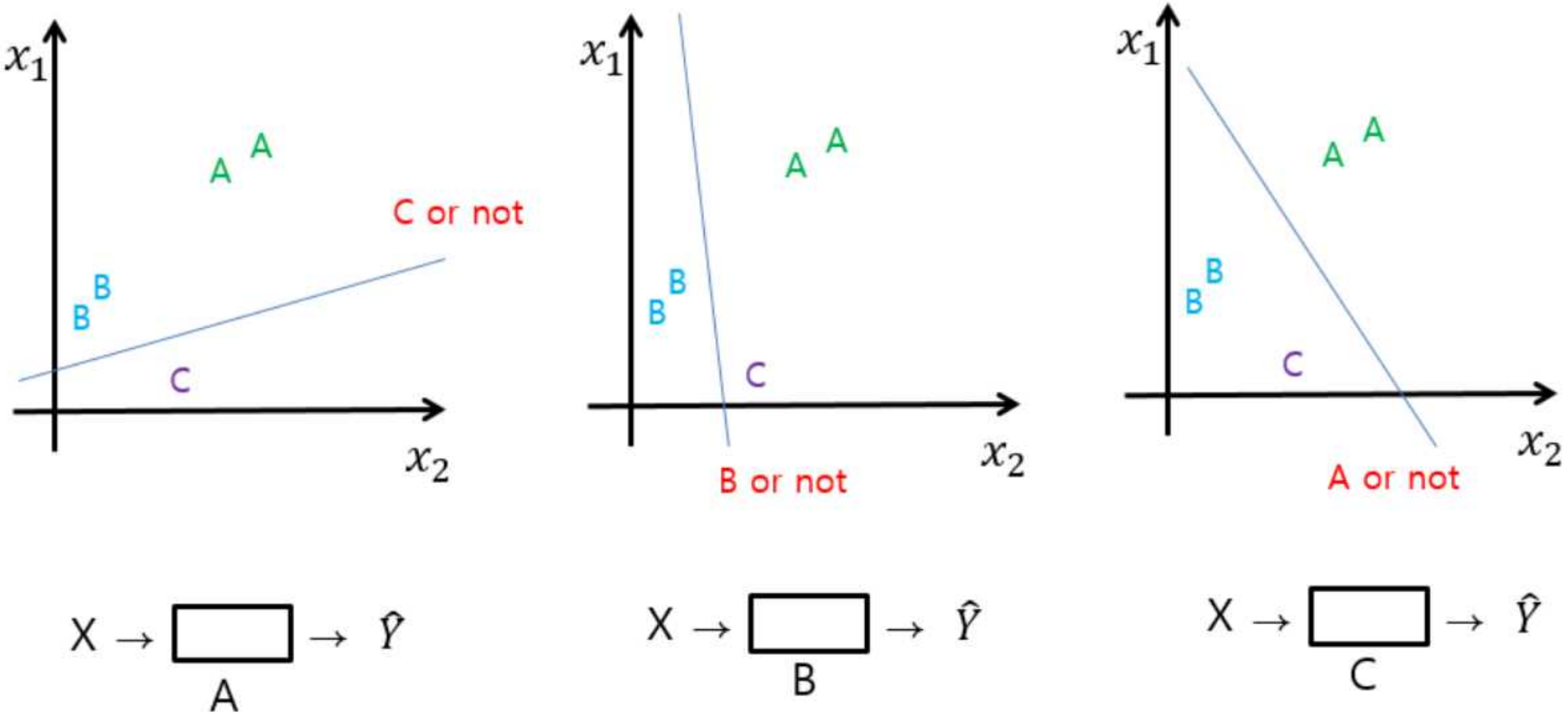
MULTINOMIAL CLASSIFICATION

x1 (hours)	x2 (attendance)	y (grade)
10	5	A
9	5	A
3	2	B
2	4	B
11	1	C





MULTINOMIAL CLASSIFICATION



MULTINOMIAL CLASSIFICATION

$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_1 x_1 + w_2 x_2 + w_3 x_3 \end{bmatrix} \qquad X \rightarrow \underset{W}{\boxed{\quad\quad}} \xrightarrow{Z} \boxed{\int} \rightarrow \hat{Y}$$

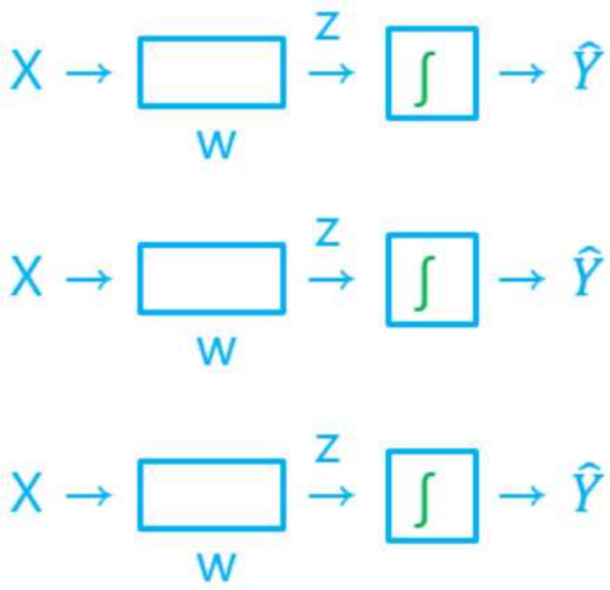
$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_1 x_1 + w_2 x_2 + w_3 x_3 \end{bmatrix} \qquad X \rightarrow \underset{W}{\boxed{\quad\quad}} \xrightarrow{Z} \boxed{\int} \rightarrow \hat{Y}$$

$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_1 x_1 + w_2 x_2 + w_3 x_3 \end{bmatrix} \qquad X \rightarrow \underset{W}{\boxed{\quad\quad}} \xrightarrow{Z} \boxed{\int} \rightarrow \hat{Y}$$

MULTINOMIAL CLASSIFICATION

$$[w_1 \ \ w_2 \ \ w_3] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [w_1 x_1 + w_2 x_2 + w_3 x_3]$$

$$\begin{bmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \ ?$$



MULTINOMIAL CLASSIFICATION

$$\begin{bmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_{A1}x_1 + w_{A2}x_2 + w_{A3}x_3 \\ w_{B1}x_1 + w_{B2}x_2 + w_{B3}x_3 \\ w_{C1}x_1 + w_{C2}x_2 + w_{C3}x_3 \end{bmatrix} = \begin{bmatrix} \hat{Y}_A \\ \hat{Y}_B \\ \hat{Y}_C \end{bmatrix}$$

$X \rightarrow \underset{w}{\boxed{\quad\quad}} \xrightarrow{z} \boxed{\int} \rightarrow \hat{Y}$ $\quad$ $X \rightarrow \underset{w}{\boxed{\quad\quad}} \xrightarrow{z} \boxed{\int} \rightarrow \hat{Y}$ $\quad$ $X \rightarrow \underset{w}{\boxed{\quad\quad}} \xrightarrow{z} \boxed{\int} \rightarrow \hat{Y}$

WHERE IS SIGMOID?

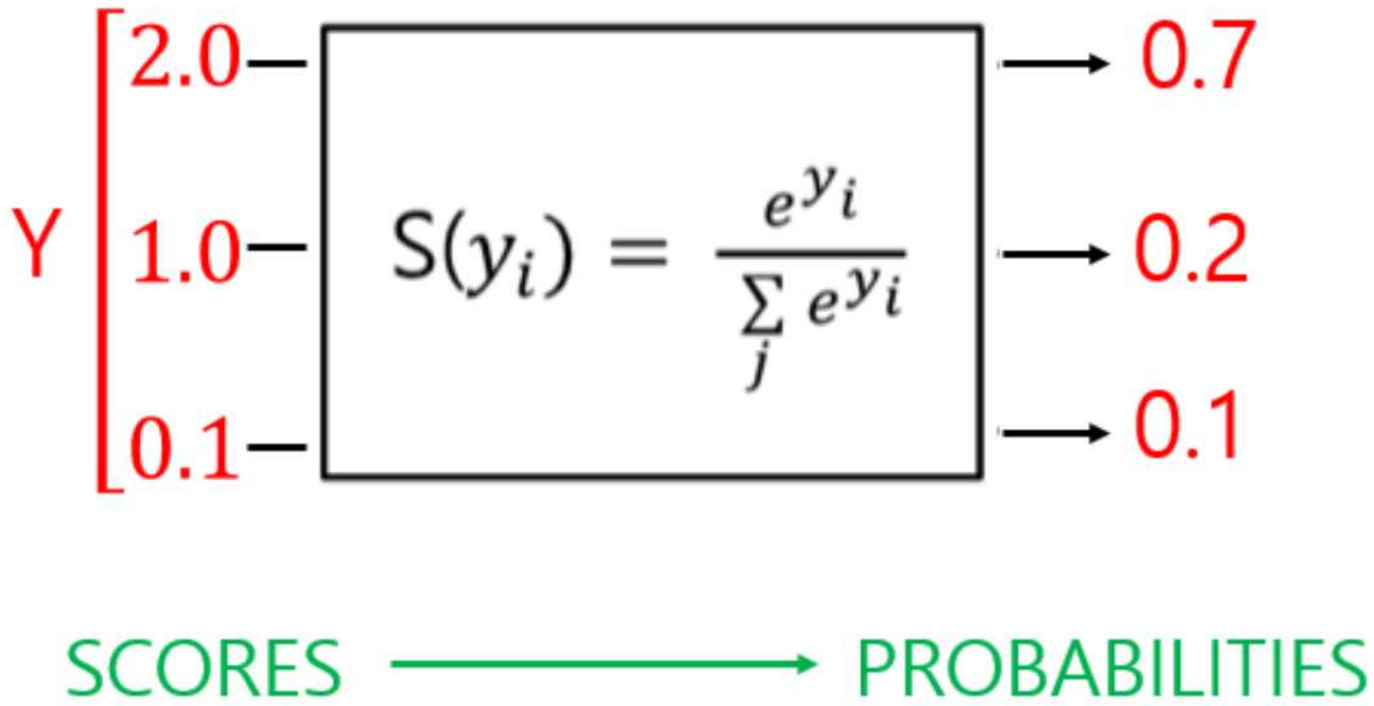
$$\begin{bmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_{A1}x_1 + w_{A2}x_2 + w_{A3}x_3 \\ w_{B1}x_1 + w_{B2}x_2 + w_{B3}x_3 \\ w_{C1}x_1 + w_{C2}x_2 + w_{C3}x_3 \end{bmatrix} = \begin{bmatrix} \hat{Y}_A \\ \hat{Y}_B \\ \hat{Y}_C \end{bmatrix} \left[\begin{matrix} \mathbf{2.0} \\ \mathbf{1.0} \\ \mathbf{0.1} \end{matrix} \right.$$

SIGMOID?

LOGISTIC CLASSIFIER

$$WX = Y \begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix} \begin{matrix} \rightarrow p = 0.7 \\ \rightarrow p = 0.2 \\ \rightarrow p = 0.1 \end{matrix}$$

SOFTMAX



SOFTMAX

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

```
tf.matmul(X,W)+b
```

$$XW = Y \begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix} \rightarrow \boxed{S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}} \rightarrow \begin{matrix} 0.7 \\ 0.2 \\ 0.1 \end{matrix}$$

SCORES $\longrightarrow$ PROBABILITIES

SOFTMAX_CROSS_ENTROPY_WITH_LOGITS

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

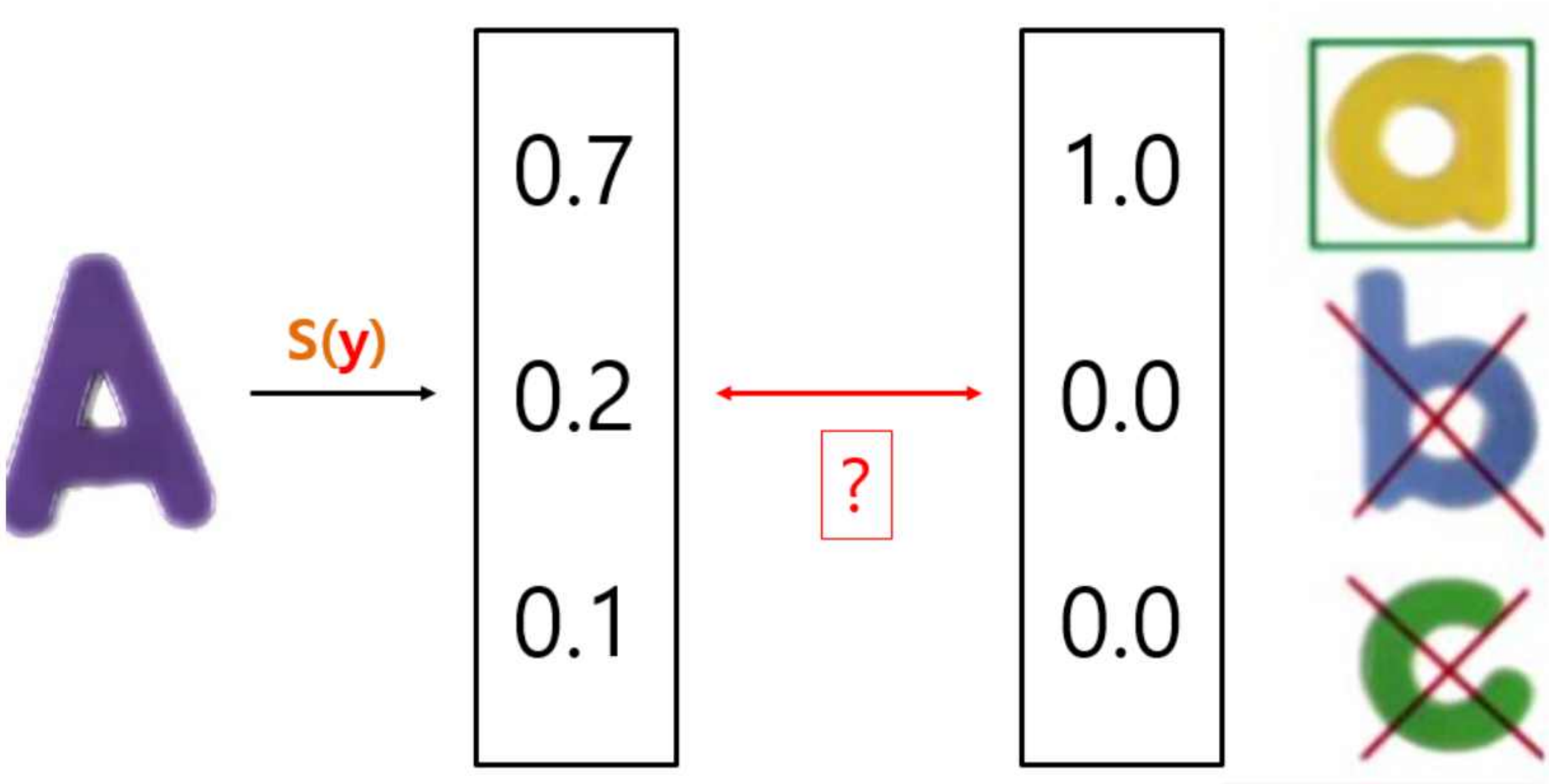
1

```
# Cross entropy cost/loss
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
```

2

```
# Cross entropy cost/loss
cost_i = tf.nn.softmax_cross_entropy_with_logits(logits=logits,
                                                 labels=Y_one_hot)
cost = tf.reduce_mean(cost_i)
```

ONE-HOT ENCODING



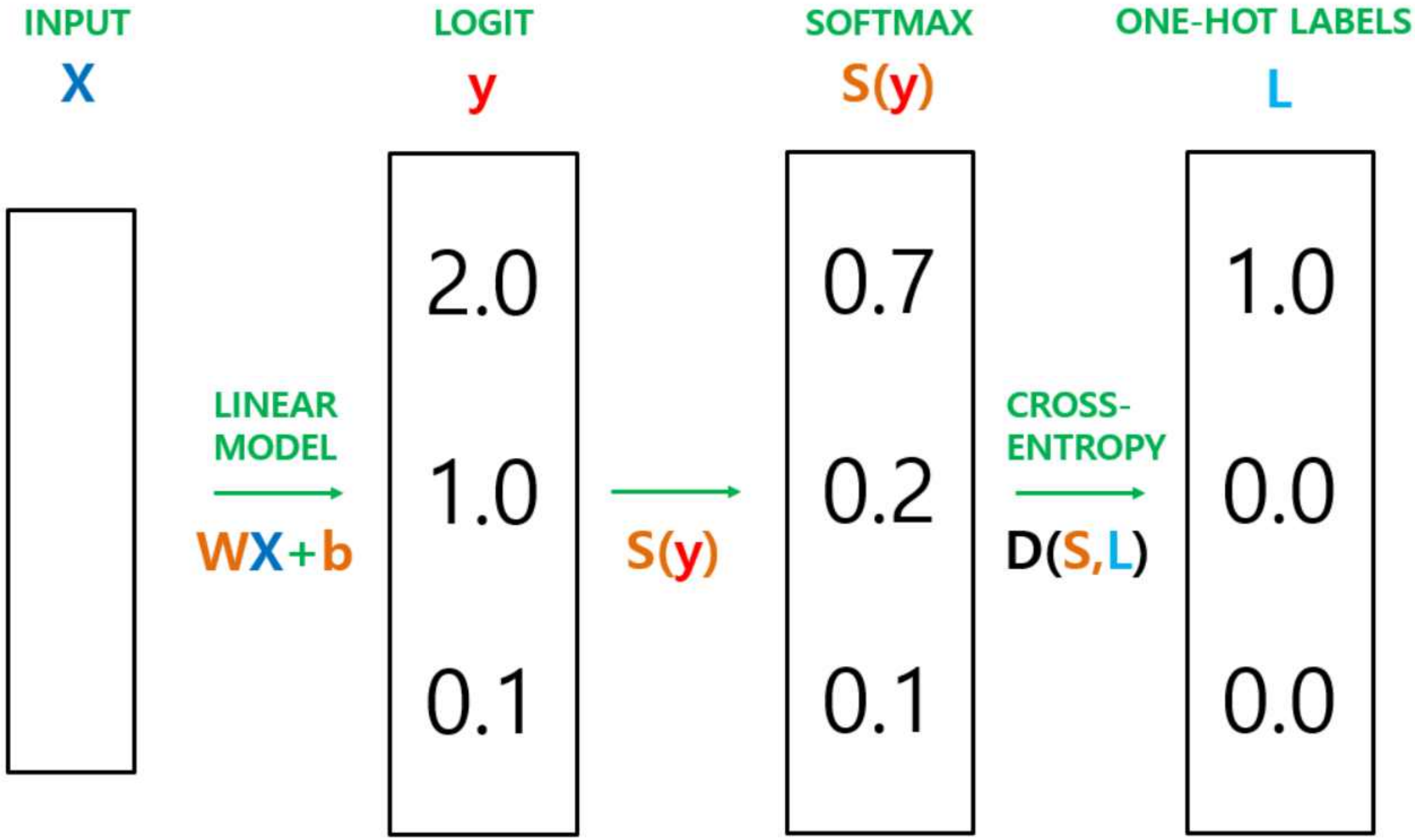
COST FUNCTION

S(y) L

$$\begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix} \qquad D(S, L) = -\sum_i L_i \log(S_i) \qquad \begin{bmatrix} 1.0 \\ 0.0 \\ 0.0 \end{bmatrix}$$

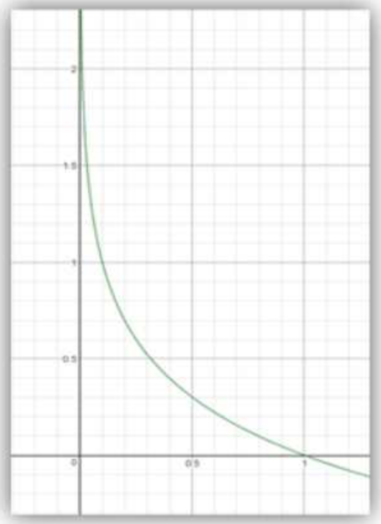
CROSS - ENTROPY

COST FUNCTION



CROSS-ENTROPY COST FUNCTION

$$-\sum_i L_i \log(S_i) \rightarrow -\sum_i L_i \log(\hat{y}_i) \rightarrow \sum_i L_i * -\log(\hat{y}_i)$$



$$L = \begin{array}{c} \mathsf{A} \\ \mathsf{B} \end{array}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \mathrm{B}$$

$$\hat{Y} = \begin{array}{c} \mathsf{A} \\ \mathsf{B} \end{array}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \mathrm{B}(0), \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} * -\log\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} * \begin{bmatrix} \infty \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0$$

$$\hat{Y} = \begin{array}{c} \mathsf{A} \\ \mathsf{B} \end{array}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \mathrm{A}(\mathrm{X}), \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} * -\log\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} * \begin{bmatrix} 0 \\ \infty \end{bmatrix} = \begin{bmatrix} 0 \\ \infty \end{bmatrix} = \infty$$

LOGISTIC COST VS. CROSS ENTROPY

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

$$D(S, L) = -\sum_{i} L_i \log(S_i)$$

COST FUNCTION : CROSS ENTROPY

$$L = \frac{1}{N}\sum_{i} D(S(WX_i + b), L_i)$$

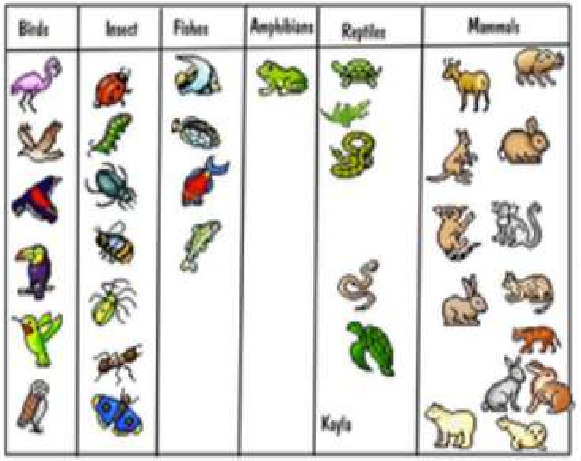
LOSS

TRAINING SET

```
# Cross entropy cost/loss
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))

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

ANIMAL CLASSIFICATION

with *softmax_cross_entropy_with_logits*



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