AI 스터디

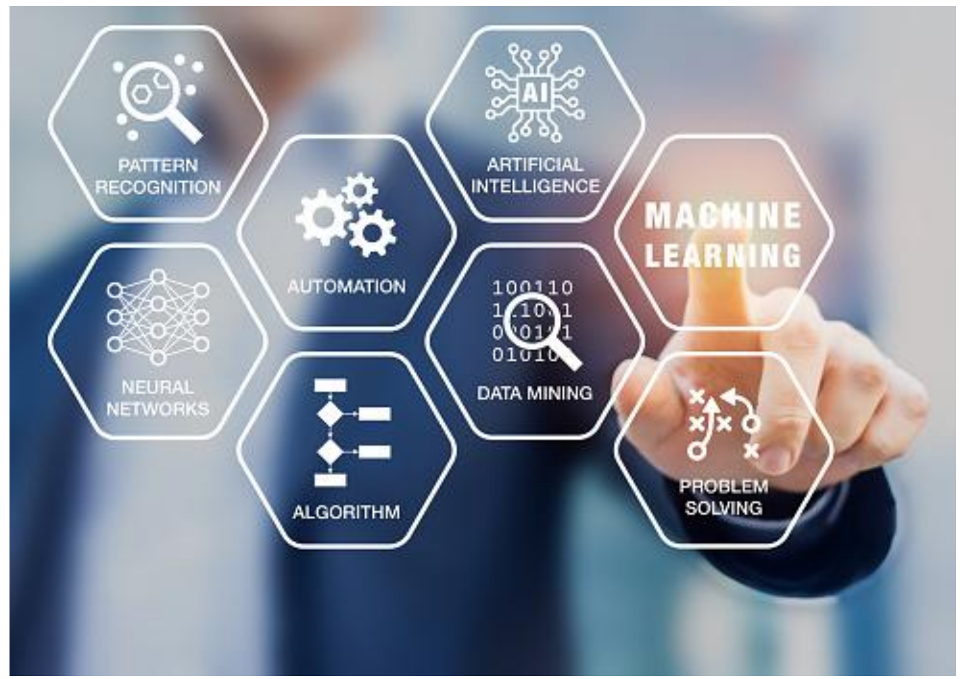
조 상 구

2025.12.24

과목 명 : AI 빅데이터

목 표 |

Python scikit-learn으로 기계학습 분야의 데이터 전처리, 다양한 알고리즘 적용을 위한 개념, 도구 및 기술 교육



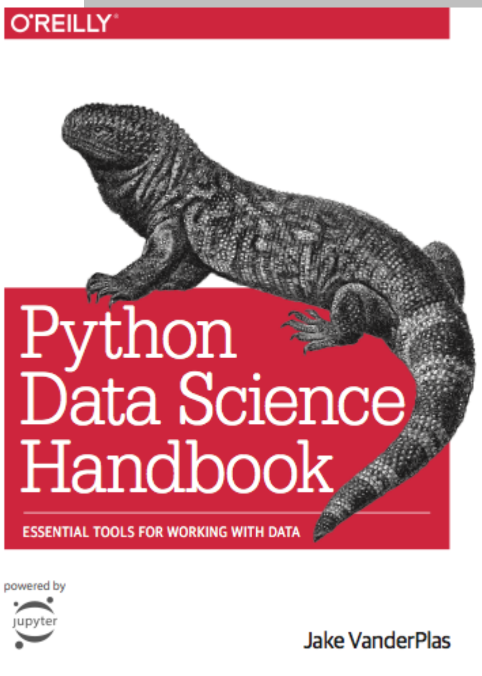
The all images are quated from https://unsplash.com/

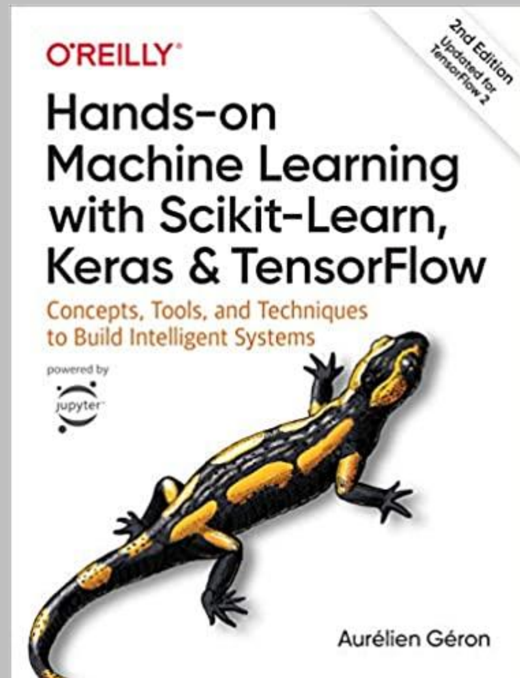
강의 계획

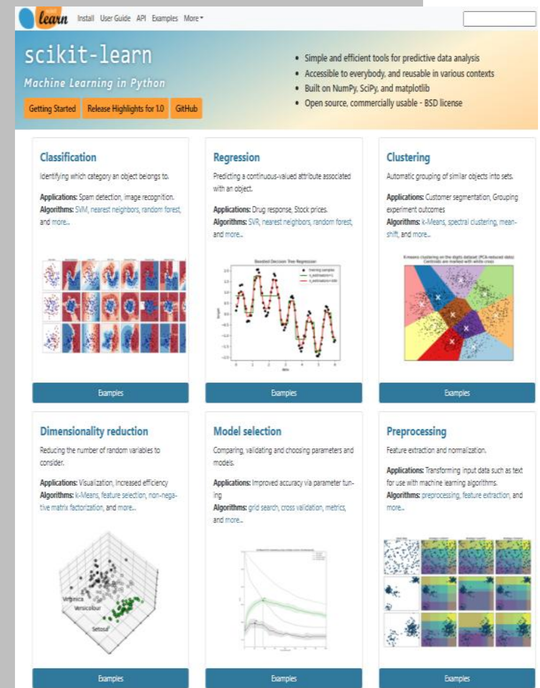
일정	강의 내용
1. Machine learining(4.2, Sat)	- Pattern recognition, data, algorithm & model - Schikit learn API, Scikit-learn getting started - Model selection and evaluation with performance index
2. Feature selection & extraction(4.2 Sat)	- Principal Component Analysis - K-Nearest Neighbor - Logistic Regression & regression - SGD(Stochastic gradient descent)
3. Supervised learning(4.8 Fri)	- Deep learning alogorithm with Keras - Gaussian Naïve Bayes - Support vector machine
4. Supervised learning(4.9 Sat)	- Decision tree, - Ensemble(Voting, Boosting, Stacking) model - Variance and Bias, Bootstrapping
5. Unsupervised model(4.9 Sat)	- K-means clustering, Logistic regression with k-Means - 1D kernel density estimation - Gaussian mixture model
6. Imbalanced data(4.15, Fri)	- Imbalanced data handling problem, Anomaly detection algorithm - Introduction to XAI (expalinable AI)

강의 교재

- 강의에 필요한 Jupyter code는 깃허브(https://github.com/조상9/aSSIST) 참고







https://github.com/jakevdp/PythonDataScienceHandbook

https://github.com/ageron/handson-ml2

https://scikit-learn.org/stable/

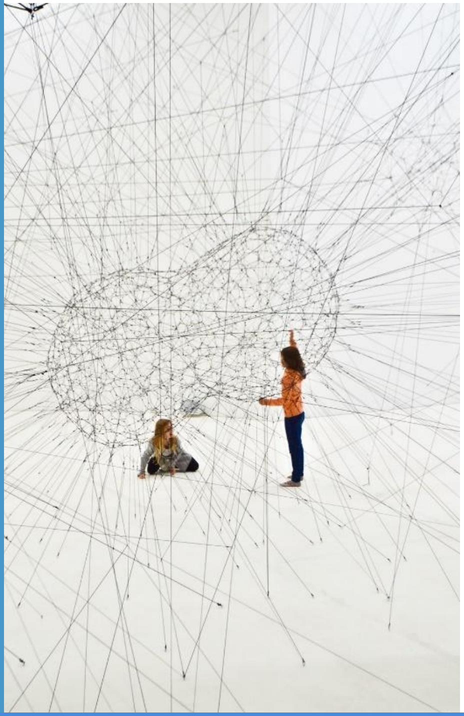
강의 목표

- 연구목적 정의, 예측모형에 투입될 데이터 전처리, 머신러닝 알고리즘과 모델의 평가와 선택 과정의 이론과 실습 능력을 갖추어 직접 실무 데이터에 적용 분석할 수 있는 인공지능 전문역량 습득

- 인공지능 연구 방법 절차를 이해하고 각 절차 단계에 필요한 python 개발 역량 습득

3. Supervised Learning

- K-nearest neighbors
- Regression, Logistic regression
- Stochastic Gradient Descent
- Deep learning algorithm

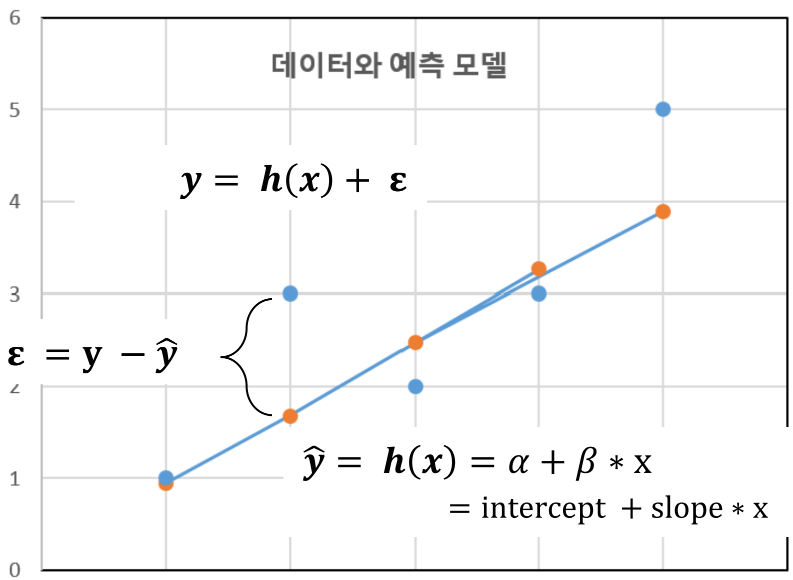


Futuristic Innovator 경복대학교 KYUNGBOK UNIVERSITY

GD(Gradient Descent) 미분 최적화

Minimize $E^2 = (Y - \widehat{Y})^2$ ⟷ Minimize Sum of squared residuals = (Observed instances – Predictions)^2

$\boxed{\text{Predictions = intercept + slope * x}}$



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

변수 2개, 인스턴스 5개
- 2개의 변수별로 5개의 미분방정식별
- Epoch이 4회이면 2*5*4 = 40번 계산

$\frac{\partial}{\partial\, intercept}$ Sum of squared residuals =

-2(1-(α + β * 1)) + -2(3-(α + β * 2))

+ -2(3-(α + β * 4)) + -2(2-(α + β * 3))

+ $\underbrace{\text{-2(5-(}\alpha + \beta \text{ * 5))}}_{\mathbf{y} - \hat{\mathbf{y}}}$ = -28

초기 값을
$\alpha = \beta = 0$

$\frac{\partial}{\partial\, slope}$ Sum of squared residuals =

-2*1(1-(α + β * 1)) + -2*2(3-(α + β * 2))

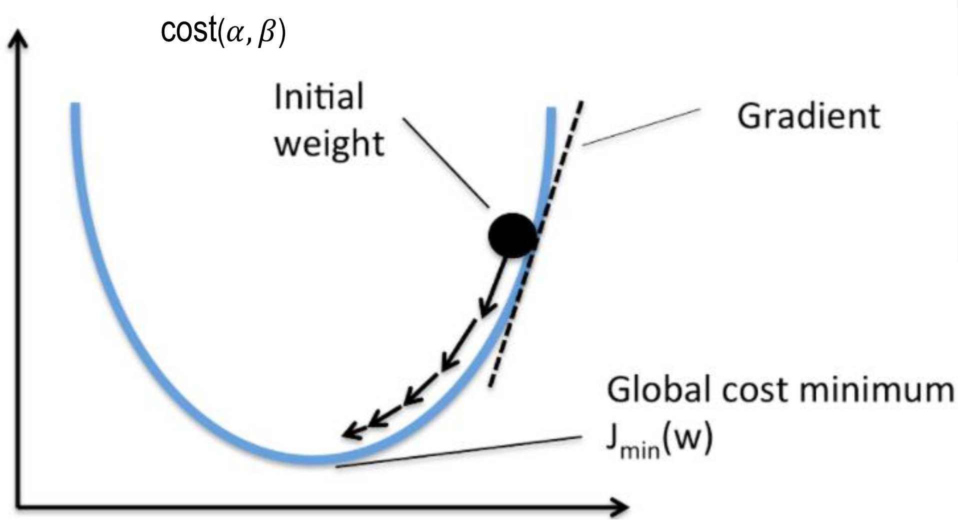
+ -2*4(3-(α + β * 4)) + -2*3(2-(α + β * 3))

+ $\underbrace{\text{-2*5(5-(}\alpha + \beta \text{ * 5))}}_{\mathbf{x}(\mathbf{y} - \hat{\mathbf{y}})}$ =-100

GD(Gradient Descent) 알고리즘 이해

- Start with initial guesses
  - Start at 0, 0 (or any other values), $\alpha = \beta = 0$ with initial parameters
  - Keeping changing α, β a little bit with learning rate to try and reduce cost(α, β), error
- Each time you change the parameters (α, β), select the gradient which reduces cost(α, β) the most possible
- Repeat, Do so until you converge to a local minimum

$$E^2 = (Y - \hat{Y})^2$$



Start at:
$\alpha = \beta = 0$

$$\begin{cases} \dfrac{\partial\ sum(sqrt(error)}{\partial\ intercept} = -28 \\[2ex] \dfrac{\partial\ sum(sqrt(error)}{\partial\ intercept} = -100 \end{cases}$$

learning rate (0.01)

$$New\ \alpha = -28 * 0.01$$
$$New\ \beta = -100 * 0.01$$

$For\ loop:$

$$\alpha = \alpha - learning\ rate * \alpha$$
$$\beta = \beta - learning\ rate * \beta$$

$Until\ to\ minimum$ cost(α, β)

GD(Gradient Descent)
연산의 복잡도 기하급수 증가

데이터의 크기가 1,000,000개이고 유전자(genes) 유형이 20,000개인 변수로 이루어진 다중 회귀 모델을 gredient descent 알고리즘을 적용할 경우 계산량이 너무 많아짐

$$\frac{\partial}{\partial\ gene1} loss\ function()$$

1
2
3
.....
....
1,000,000 개 식(terms)

$$\frac{\partial}{\partial\ gene2} loss\ function()$$

1
2
3
.....
....
1,000,000 개 식(terms)

$$\frac{\partial}{\partial\ gene20{,}000} loss\ function()$$

1
2
3
.....
....
1,000,000 개 식(terms)

gredient descent

20,000개의 미분방정식별로 1,000,000개의 식을 1,000번의 step(epochs = 1,000)으로 gradient descent를 수행하면,
총 2,300,000,000,000 개 식을 계산
(비효율적)

stochastic gredient descent

임의의 한 점을 선택하여 instance별로 gradient을 적용하면 epoch의 회수 * 1,000,000개 식을 계산
(효율적)

SGD(Stochastic Gradient Descent)
엑셀로 익히기

learning rate	Iterations	Data x	Data y	model 절편 (α)	model 기울기 (β)	model y_hat	y_hat − y error
0.01							
1 epoch	0	1	1	0.0000	0.0000	0.0000	−1.0000
	1	2	3	0.0100	0.0100	0.0300	−2.9700
	2	4	3	0.0397	0.0694	0.3173	−2.6827
	3	3	2	0.0665	0.1767	0.5967	−1.4033
	4	5	5	0.0806	0.2188	1.1746	−3.8254
2 epoch	5	1	1	0.1188	0.4101	0.5289	−0.4711
	6	2	3	0.1235	0.4148	0.9531	−2.0469
	7	4	3	0.1440	0.4557	1.9669	−1.0331
	8	3	2	0.1543	0.4971	1.6455	−0.3545
	9	5	5	0.1579	0.5077	2.6963	−2.3037
3 epoch	10	1	1	0.1809	0.6229	0.8038	−0.1962
	11	2	3	0.1829	0.6248	1.4325	−1.5675
	12	4	3	0.1985	0.6562	2.8233	−0.1767
	13	3	2	0.2003	0.6633	2.1901	0.1901
	14	5	5	0.1984	0.6575	3.4862	−1.5138
4 epoch	15	1	1	0.2135	0.7332	0.9468	−0.0532
	16	2	3	0.2141	0.7338	1.6816	−1.3184
	17	4	3	0.2273	0.7601	3.2678	0.2678
	18	3	2	0.2246	0.7494	2.4729	0.4729
	19	5	5	0.2199	0.7352	3.8961	−1.1039

Initialized settings

Let α=β=0,

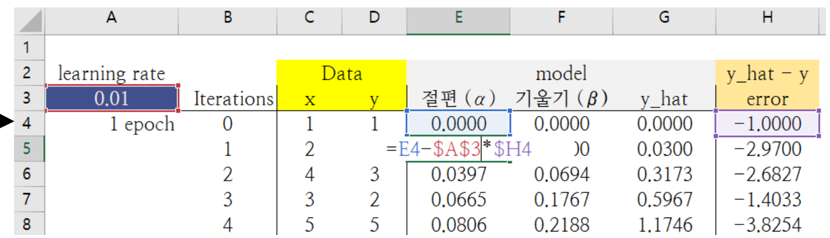
y_hat = α + β* X(instance)

error = y_hat - y

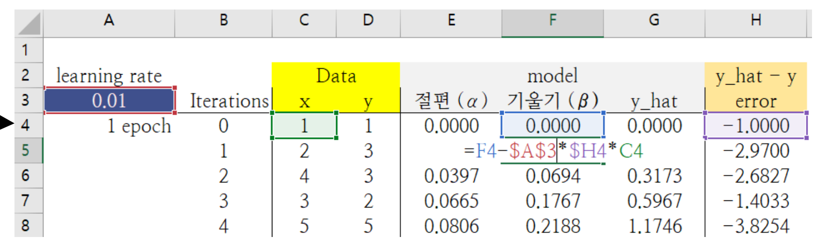
Recursive iterations

$$\alpha_{j+1} = \alpha_j - learning\ rate * error_j$$

$$\beta_{j+1} = \beta_j - learning\ rate * error_j * x_j$$

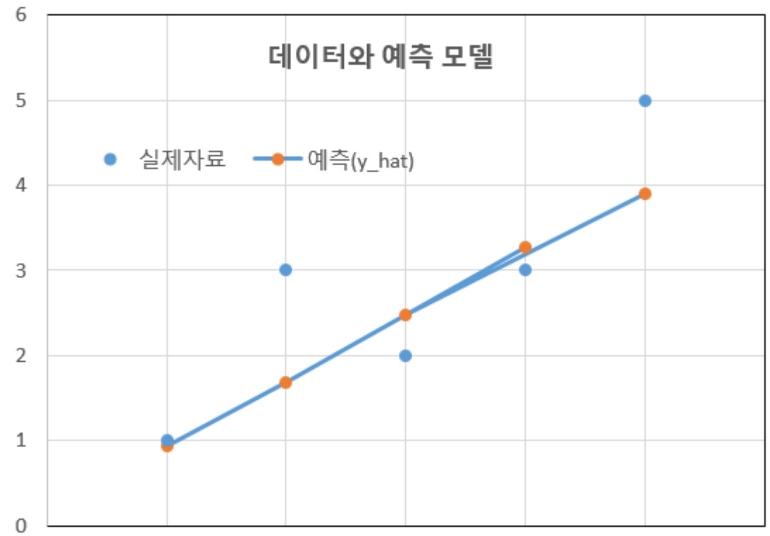


	A	B	C	D	E	F	G	H
1								
2	learning rate		Data		model			y_hat − y
3	0.01	Iterations	x	y	절편 (α)	기울기 (β)	y_hat	error
4	1 epoch	0	1	1	0.0000	0.0000	0.0000	−1.0000
5		1	2	=E4−A3*$H4		)0	0.0300	−2.9700
6		2	4	3	0.0397	0.0694	0.3173	−2.6827
7		3	3	2	0.0665	0.1767	0.5967	−1.4033
8		4	5	5	0.0806	0.2188	1.1746	−3.8254



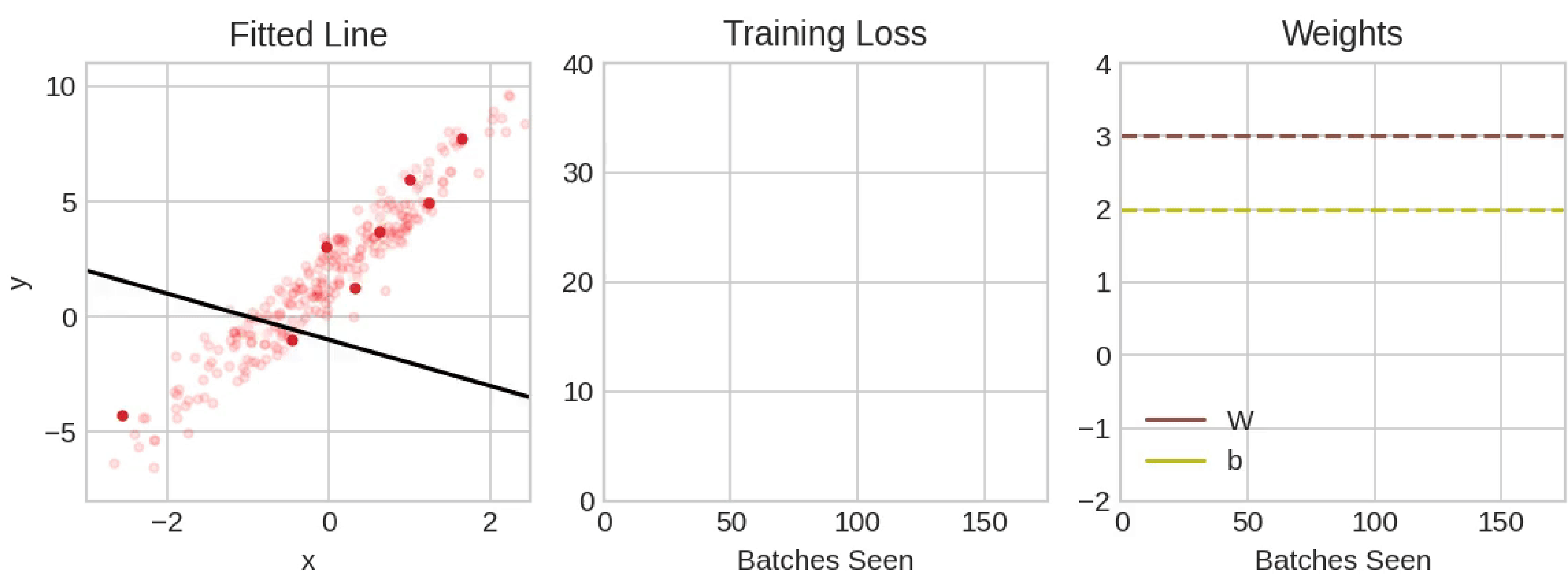
	A	B	C	D	E	F	G	H
1								
2	learning rate		Data		model			y_hat − y
3	0.01	Iterations	x	y	절편 (α)	기울기 (β)	y_hat	error
4	1 epoch	0	1	1	0.0000	0.0000	0.0000	−1.0000
5		1	2	3	=F4−A3*$H4*C4			−2.9700
6		2	4	3	0.0397	0.0694	0.3173	−2.6827
7		3	3	2	0.0665	0.1767	0.5967	−1.4033
8		4	5	5	0.0806	0.2188	1.1746	−3.8254

SGD(Stochastic Gradient Descent)
엑셀 그래프로 익히기



SGD(Stochastic Gradient Descent)
how to train a neural network

- What problem to solve : the **loss function** measures the disparity between the the target's true value and the value the model predicts.
- How to solve it : the **optimizer** is an algorithm that adjusts the weights to minize the loss function.

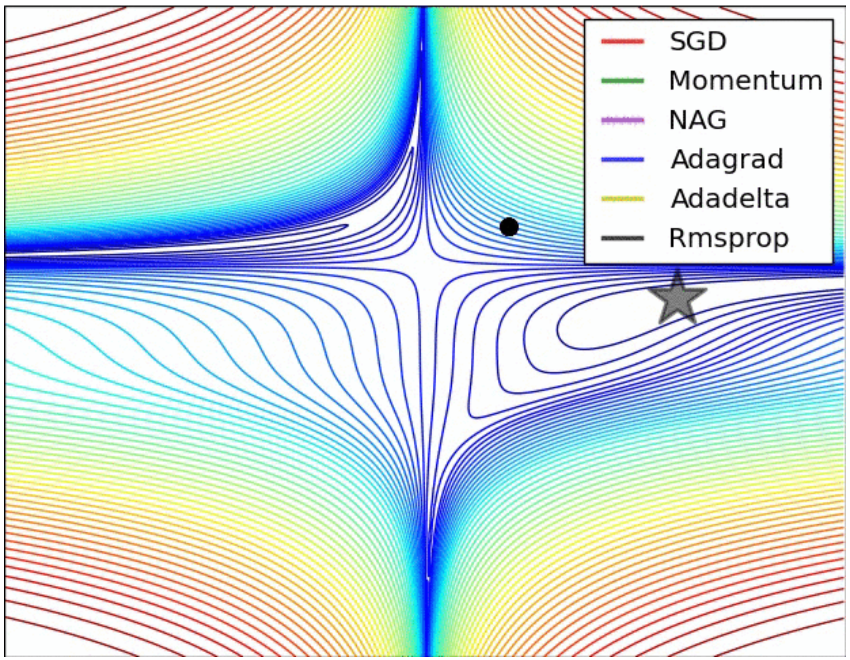


Minimize mean absolute error or MAE
With SGD algorithm

https://www.kaggle.com/ryanholbrook/stochastic-gradient-descent

SGD(Stochastic Gradient Descent)
Newton's Optimization method(2차 미분)

Convex형태의 함수를 갖는 손실함수에서 최적화(최저점으로 활강)를 위한 알고리즘은 Quasi-newton 알고리즘, L-BFG, BGF, Adam의 다양한 알고리즘 사용



What's In a Name?

The **gradient** is a vector that tells us in what direction the weights need to go. More precisely, it tells us how to change the weights to make the loss change *fastest*. We call our process gradient **descent** because it uses the gradient to *descend* the loss curve towards a minimum.

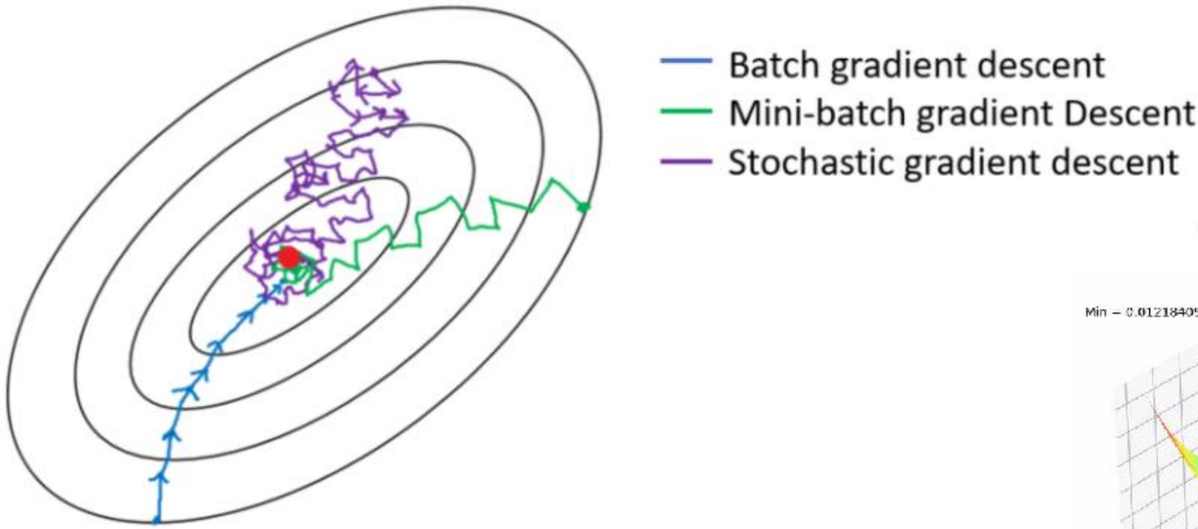
Stochastic means "determined by chance."
Our training is *stochastic* because the minibatches are *random samples* from the dataset. And that's why it's called SGD!

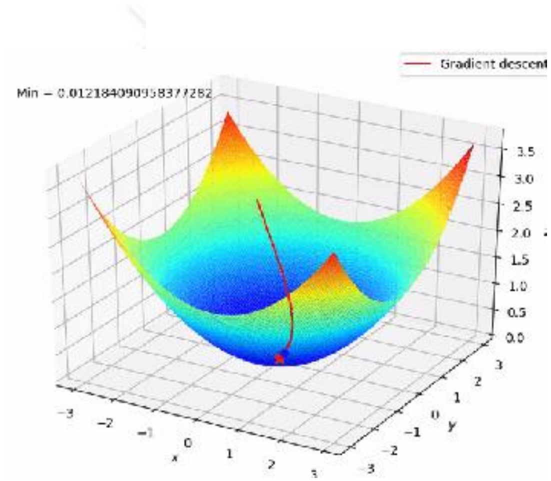
SGD(Stochastic Gradient Descent) 알고리즘의 종류

- Repeat, Do so until you converge to a local minimum

$J(\theta)$ = cost(α, β), $\theta = (\alpha, \beta)$

$$\longrightarrow \quad \theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$



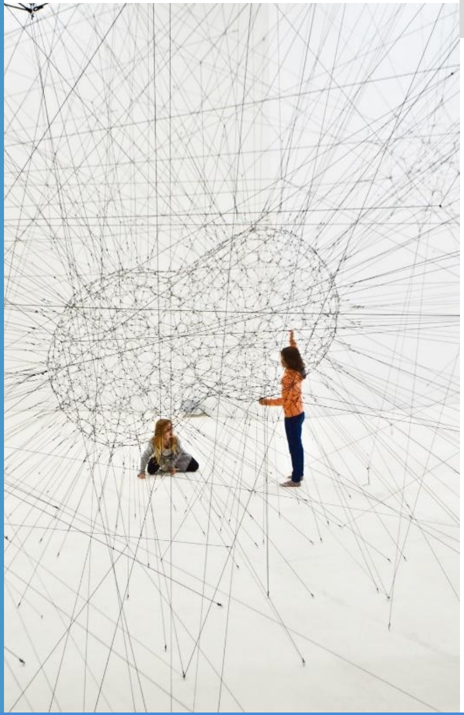


2022.4.8(Fri)

일정	강의 내용
1. Machine learining(4.2, Sat)	- Pattern recognition, data, algorithm & model - Schikit learn API, Scikit-learn getting started - Model selection and evaluation with performance index
2. Feature selection& extraction(4.2 Sat)	- Principal Component Analysis - K-Nearest Neighbor - Logistic Regression & regression - SGD(Stochastic gradient descent)
3. Supervised learning(4.8 Fri)	- Deep learning alogorithm with Keras - Gaussian Naïve Bayes - Support vector machine
4. Supervised learning(4.9 Sat)	- Decision tree, - Ensemble(Voting, Boosting, Stacking) model - Variance and Bias, Bootstrapping
5. Unsupervised model(4.9 Sat)	- K-means clustering, Logistic regression with k-Means - 1D kernel density estimation - Gaussian mixture model
6. Imbalanced data(4.15, Fri)	- Imbalanced data handling problem, Anomaly detection algorithm - Introduction to XAI (expalinable AI)

3. Supervised Learning

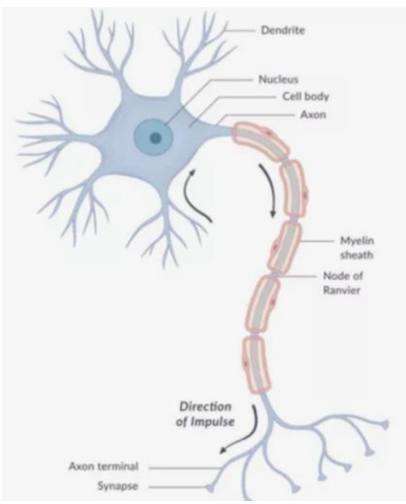
- K-nearest neighbors
- Regression, Logistic regression
- Stochastic Gradient Descent
- Deep learning algorithm

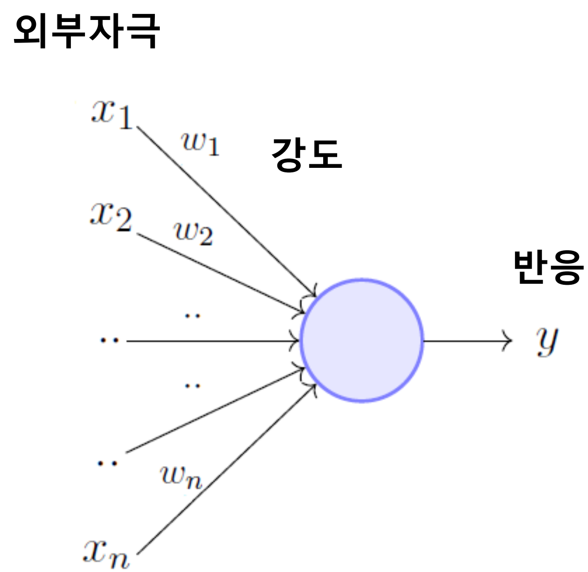


Neural network
뉴론

- 인간의 뉴런과 축삭돌기를 모방한 인공신경망 구조
- 퍼셉트론(perceptron) : 외부자극과 강도의 총합이 특정한 역치(threshold, θ)를 넘으면 반응(y)을 하는 단일 뉴런

[이진분류 뉴런]





$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i \geq \theta$$
$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i < \theta$$

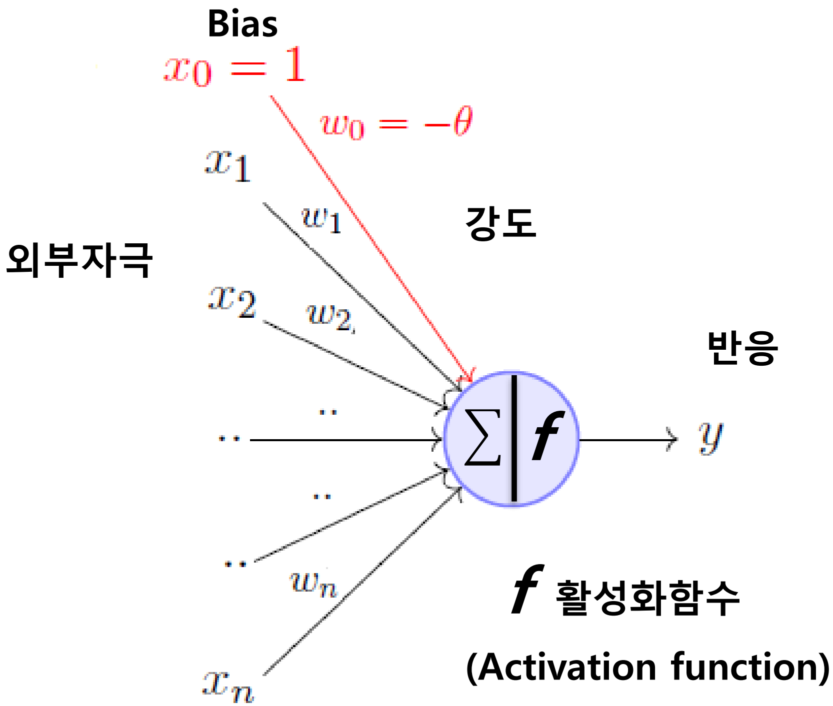
Rewriting the above,

$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i - \theta \geq 0$$
$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i - \theta < 0$$

Neural network
뉴론

- 인간의 뉴런과 축삭돌기를 모방한 인공신경망 구조

[이진분류 뉴런]



A more accepted convention,

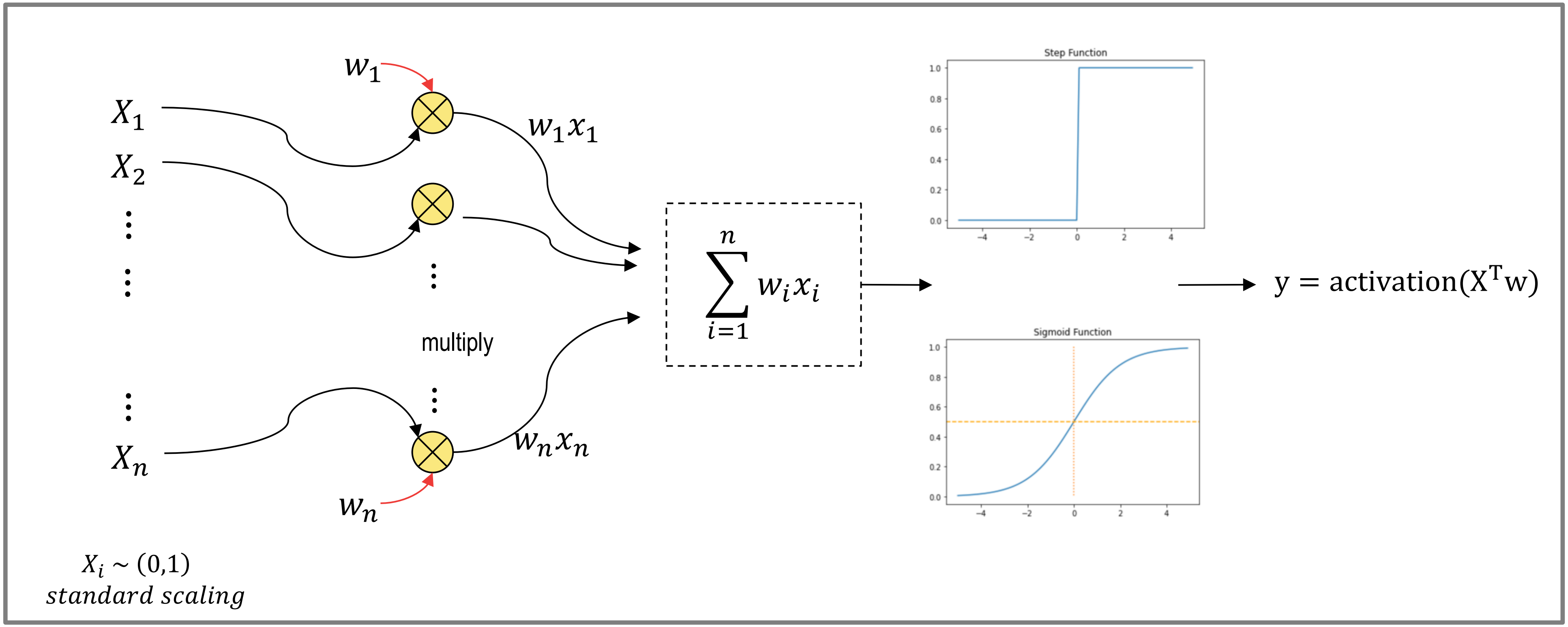
$$y = 1 \quad if \sum_{i=0}^{n} w_i * x_i \geq 0$$

$$= 0 \quad if \sum_{i=0}^{n} w_i * x_i < 0$$

$$where, \quad x_0 = 1 \quad and \quad w_0 = -\theta$$

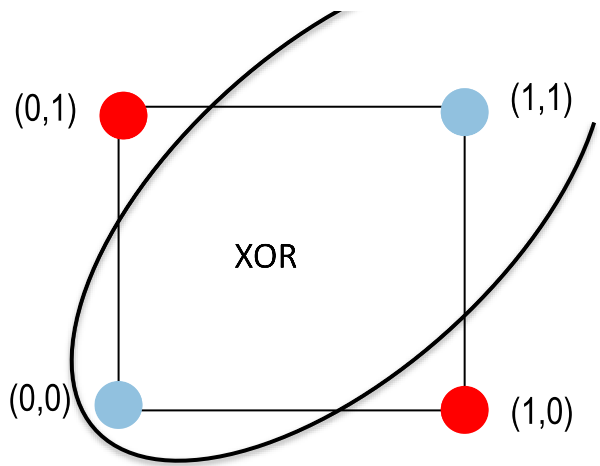
Neural network
단일 뉴런의 수학적 구조

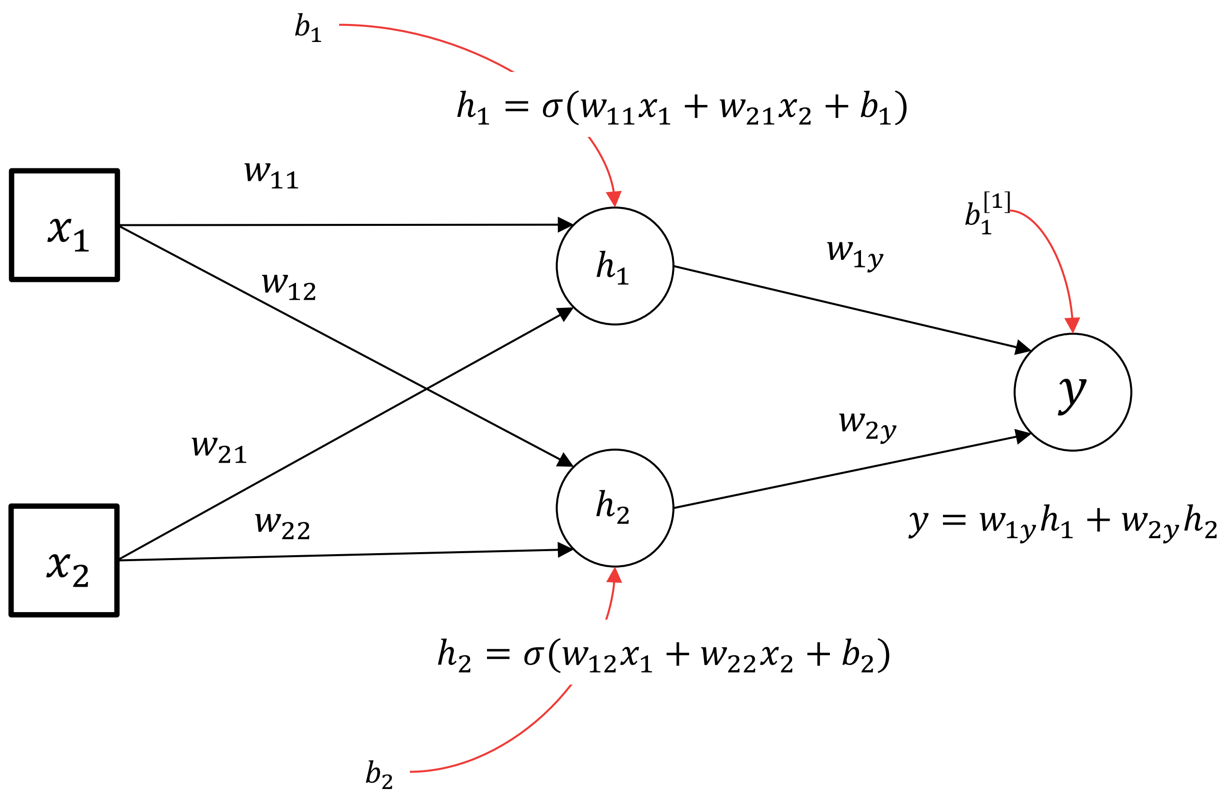
Input | **Weights** | **Weighted sum:** $z = X^T\mathbf{w}$ | **Activation function** | **Output:** $\mathbf{y} = \mathbf{h_w}(\mathbf{x})$



Neural network
XOR problem solving

$h(x)$, *Non linear function*

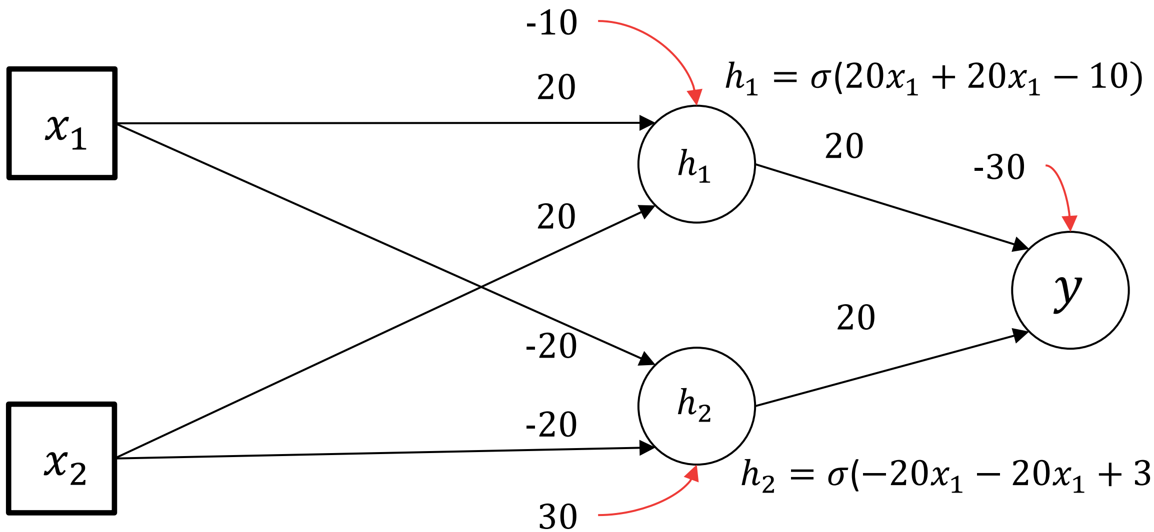




Neural network

XOR problem solving

Rule based problem solving



$h_1 = \sigma(20x_1 + 20x_1 - 10)$

$h_2 = \sigma(-20x_1 - 20x_1 + 30)$

$y = \sigma(20h_1 + 20h_1 - 30)$

$x_1 = 0, \quad x_2 = 0$
$x_1 = 1, \quad x_2 = 1$
$x_1 = 0, \quad x_2 = 1$
$x_1 = 1, \quad x_2 = 0$

$\sigma(20x_1 + 20x_2 - 10) \approx 0$
$\sigma(20x_1 + 20x_2 - 10) \approx 1$
$\sigma(20x_1 + 20x_2 - 10) \approx 1$
$\sigma(20x_1 + 20x_2 - 10) \approx 1$

$\sigma(\text{-}20x_1 - 20x_2 + 30) \approx 1$
$\sigma(\text{-}20x_1 - 20x_2 + 30) \approx 0$
$\sigma(\text{-}20x_1 - 20x_2 + 30) \approx 1$
$\sigma(\text{-}20x_1 - 20x_2 + 30) \approx 1$

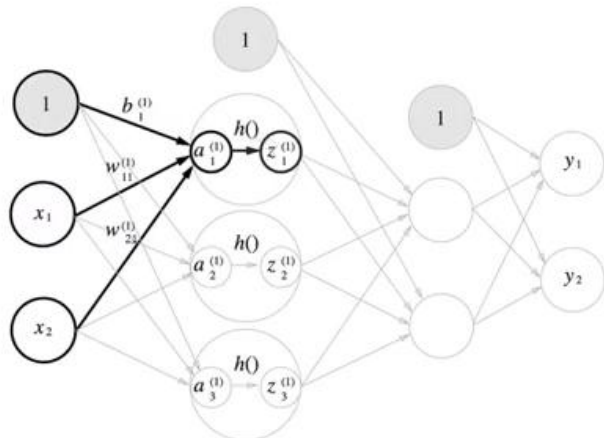
$\sigma(20h_1 + 20h_2 - 30) = 0$
$\sigma(20h_1 + 20h_2 - 30) = 0$
$\sigma(20h_1 + 20h_2 - 30) = 1$
$\sigma(20h_1 + 20h_2 - 30) = 1$

$h_1 = 0, h_2 = 1$
$h_1 = 1\ h_2 = 0$
$h_1 = 1, h_2 = 1$
$h_1 = 1, h_2 = 1$

Neural network
실습 코드 (numpy)

- 출력층의 뉴런(노드)가 한 개의 회귀(regression)모델로 추정해야할 파라미터는 각 층의 전 단계 뉴런의 개수와 bias를 더한 값

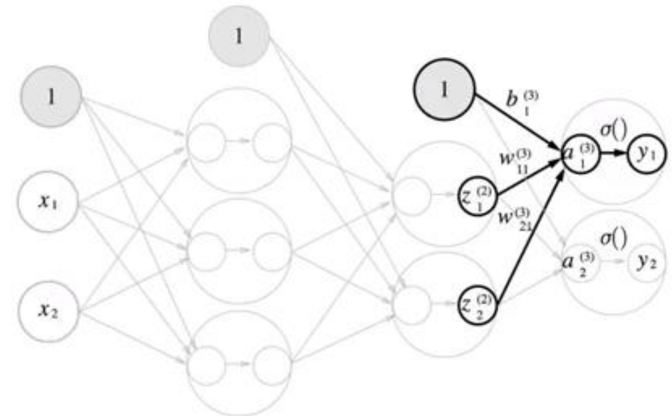
[입력층에서 1층 layer]



$$(a_1^{(1)}\ \ a_2^{(1)}\ \ a_3^{(1)}) = (x_1\ \ x_2)\begin{pmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \end{pmatrix} + (b_1^{(1)}\ \ b_2^{(1)}\ \ b_3^{(1)})$$

- 선형변환함수와 활성화함수로 중첩 네트워크 구성

[1층 layer에서 출력층]

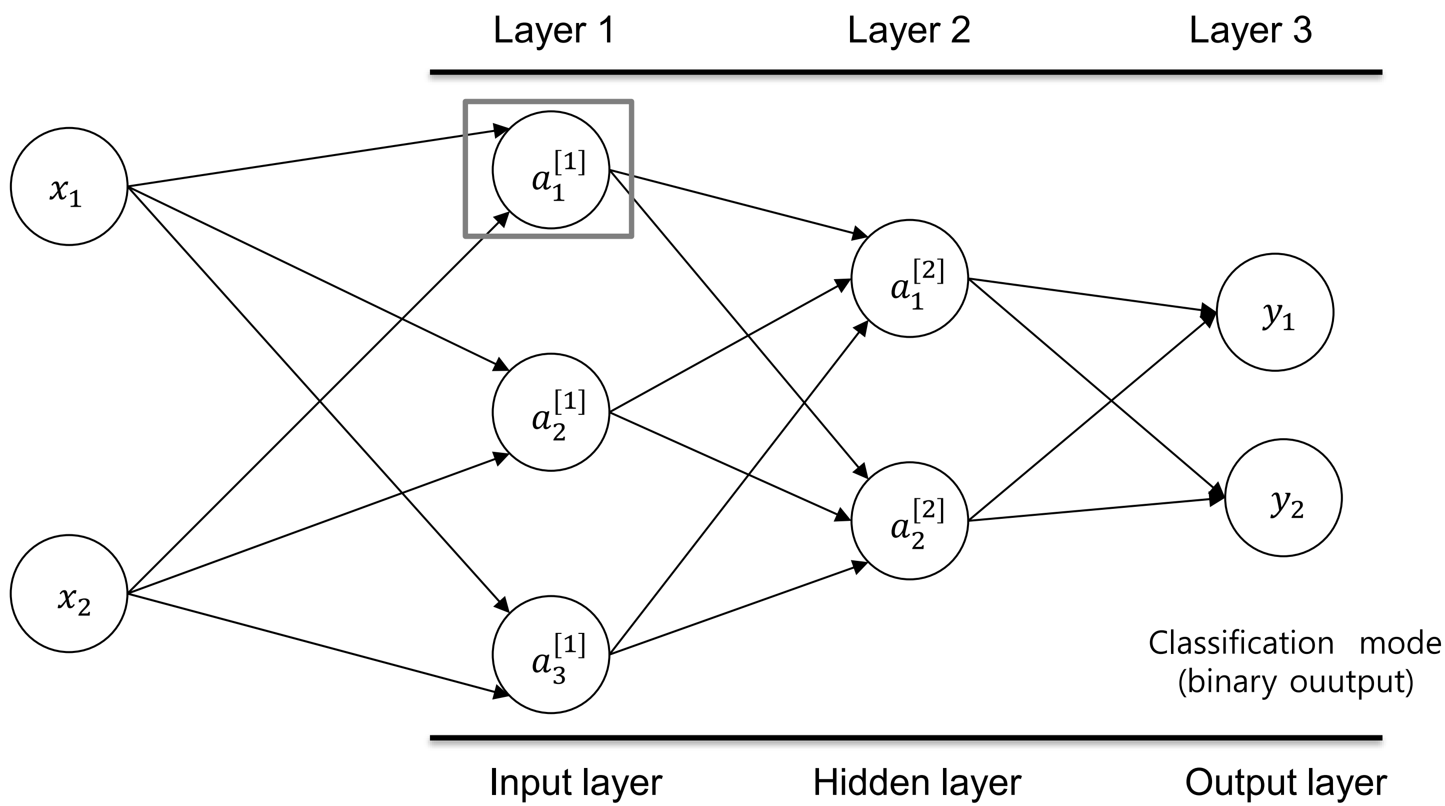


$$(a_1^{(3)}\ \ a_2^{(3)}) = (z_1^{(2)}\ \ z_2^{(2)})\begin{pmatrix} w_{21}^{(3)} & w_{22}^{(3)} \\ w_{31}^{(3)} & w_{32}^{(3)} \end{pmatrix} + (b_1^{(3)}\ \ b_2^{(3)})$$

- 소프트맥스(softmax)함수
- 확률벡터로 변환하는 함수

Neural network
Hidden layers

- 출력층의 뉴런(노드)가 한 개의 회귀(regression)모델로 추정해야할 파라미터는 각 층의 전단계 뉴런의 개수와 bias를 더한 값



$a_i^{[j]}$:

j^{th} *Layer*
i^{th} *Neuron*

추정 parameters : $2*3+3$ $3*2+2$ $2*2+1$

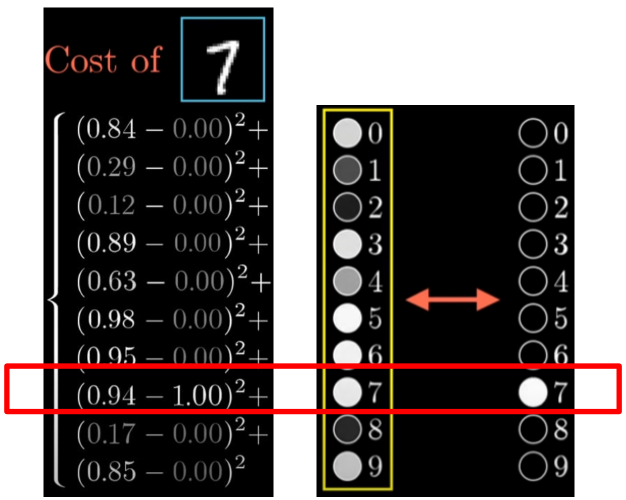
Neural network
역전파(Backpropagation)

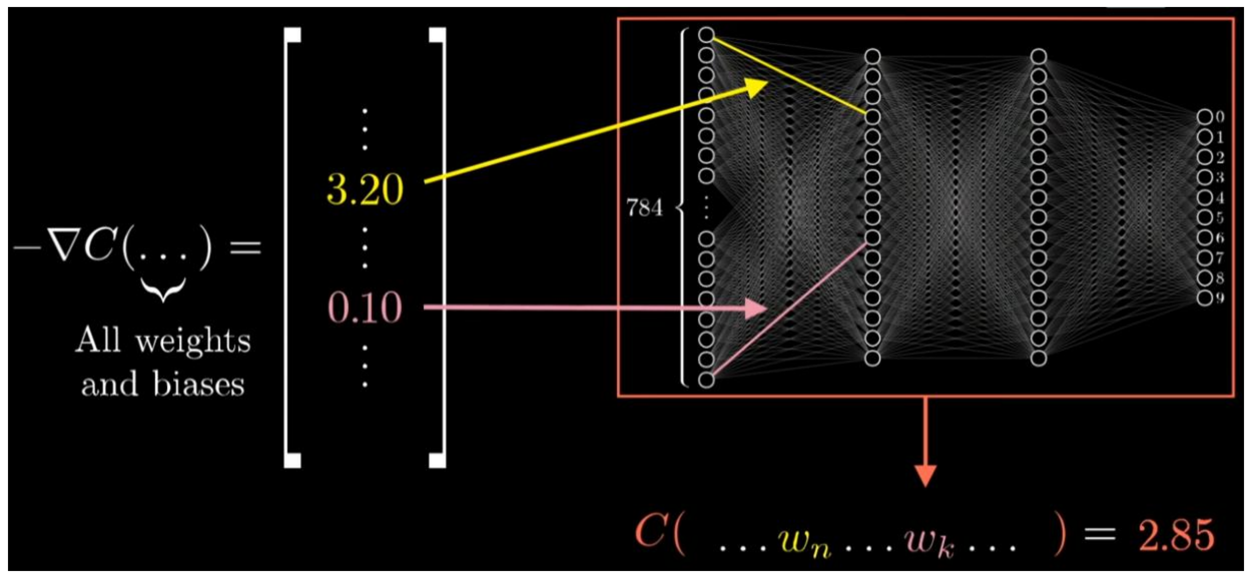
- 전체 인공신경망에서 손실함수(L, loss function)을 최소화하는 W(w_i, bias 포함) 를 찾는 방법

✓ 인공신경망이 '7'번을 예측한 확률벡터에 대한 교차엔트로피 손실 '2.85'

✓ 교차엔트로피 손실에 대한 입력층과 첫번째 hidden layer의 weight와 bias의 전미분값(민감도)은 각각 ' 3.2 ' 와 ' 0.1'

✓ W를 아주 조금씩 변동시키면서 손실함수가 최소가 되는 W를 gradient descent방법으로 최적의 W를 도출





https://www.youtube.com/watch?v=Ilg3gGewQ5U&t=566s

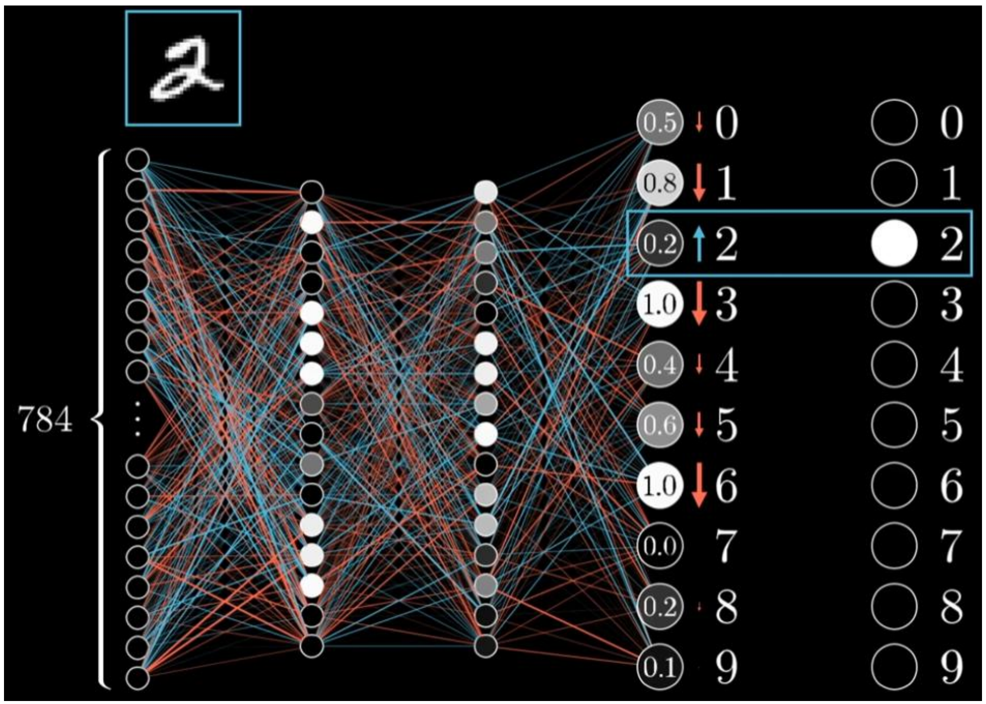
Neural network

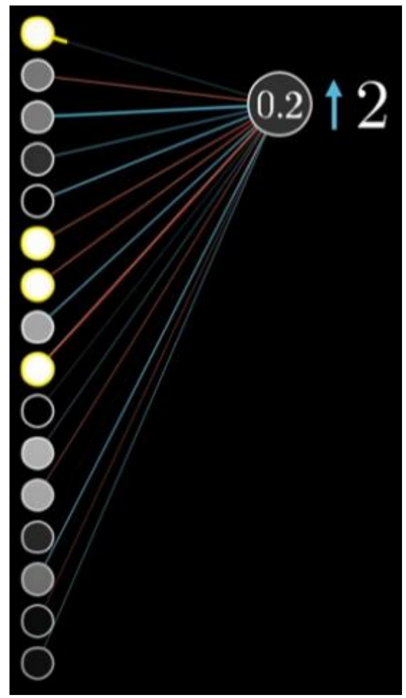
역전파(Backpropagation) gradient descent

- 실제 '2 ' 번을 예측하지 못한 손실함수(L, loss function)를 줄이는 방법은 backward방향으로 파라미터와 활성화함수를 gradient descent하게 변화(“Neurons that fire together, wire together.” - Donald Hebb)

$$0.2 = \sigma(w_0a_0 + w_1a_1 + \cdots + w_{n-1}a_{n-1} + b)$$

Backward pass





Increase b

Increase w_i
in proportion to a_i

Change a_i

- ❖ Backward pass
- ❖ Chain rule deravatives
- ❖ Gradient descent

https://www.youtube.com/watch?v=Ilg3gGewQ5U&t=566s

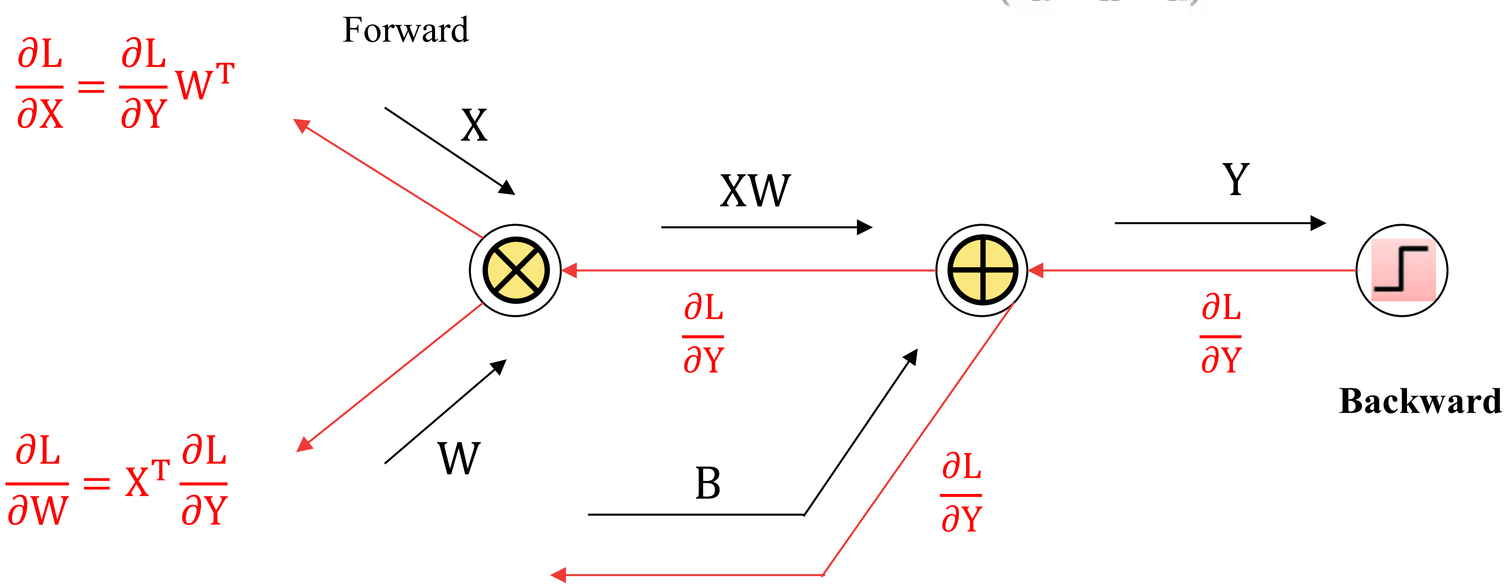
Neural network
역전파(Backpropagation)

- 전체 인공신경망에서 손실함수(L, loss function)을 최소화하는 W(w_i, bias 포함) 를 찾는 방법

1. **Chain Rule**
2. **Gredient Descent**

$$Y = XW + B$$

$$\begin{pmatrix} y_1 & y_2 & y_3 \end{pmatrix} = \begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix} + \begin{pmatrix} b_1 & b_2 & b_3 \end{pmatrix}$$

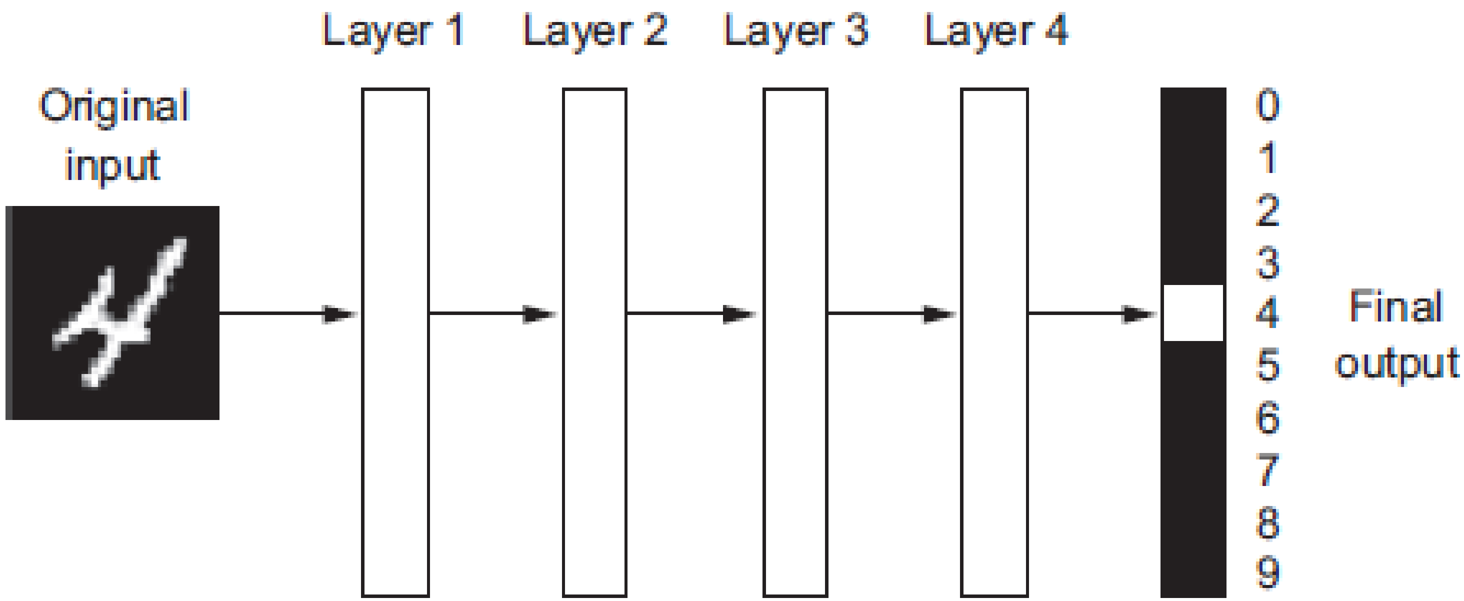


https://ml-cheatsheet.readthedocs.io/en/latest/backpropagation.html

Neural network
Deep representations

- MNIST 수기체 '4'의 이미지가 여러 층(layers)에 따라 이미지는 변환(representation)하는 방법

[A deep neural network for digit classification]

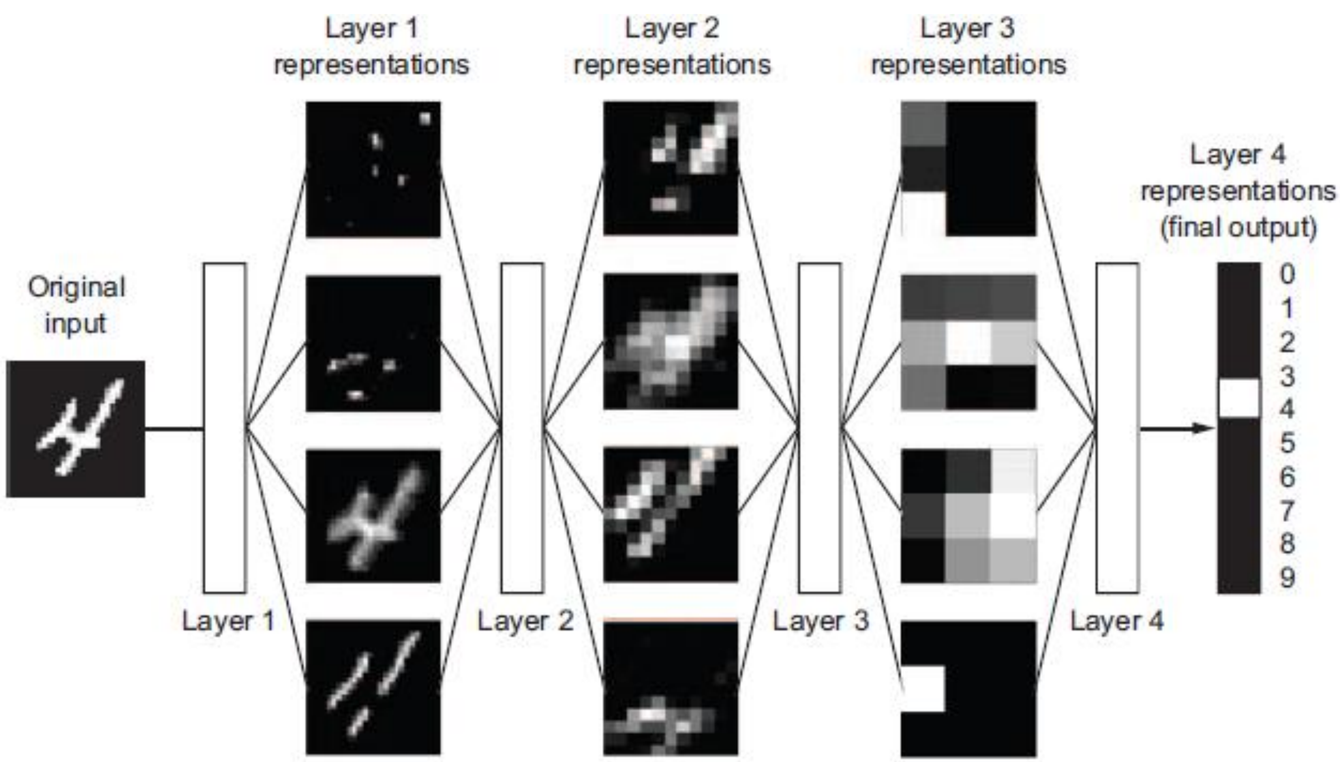


Deep Learning with Python, Francois Chollet, O'RELLY

Neural network
Deep representations

- 딥러닝 네트워크는 여러 층(layers)에 따라 숫자 이미지를 표현(representation)으로 변환
- 다단계로 확인하면 원본이미지가 연속적인 필터를 거쳐 증류 작업처럼 점점 더 정제(purification)

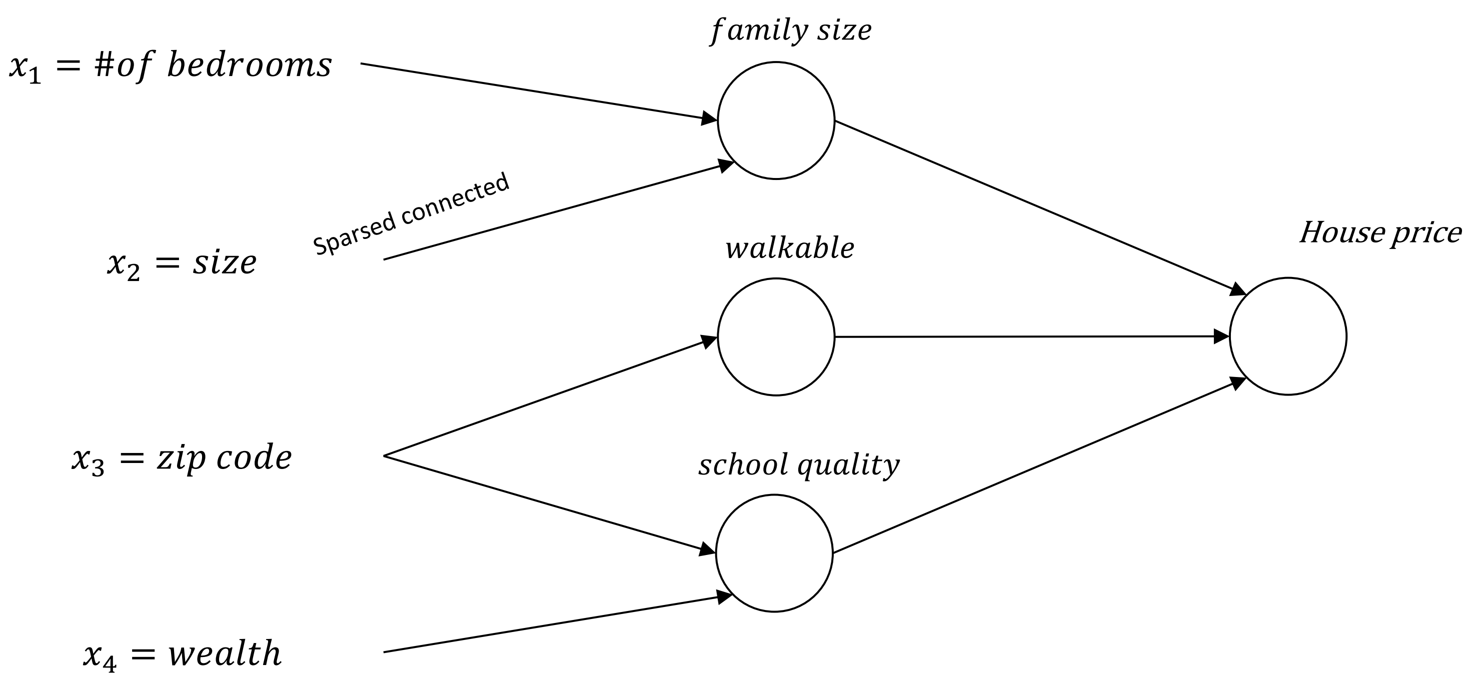
[A deep representations learned by a digit-calassification model]



Deep Learning with Python, Francois Chollet, O'RELLY

Neural network
사회과학적 해석

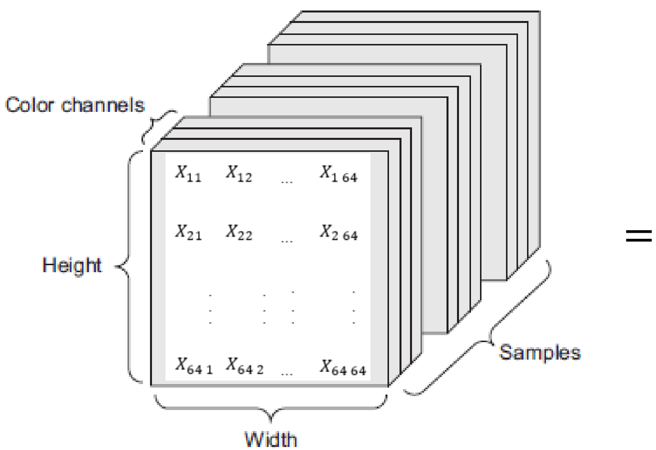
- 출력층의 뉴런(노드)가 회귀(regression)모델

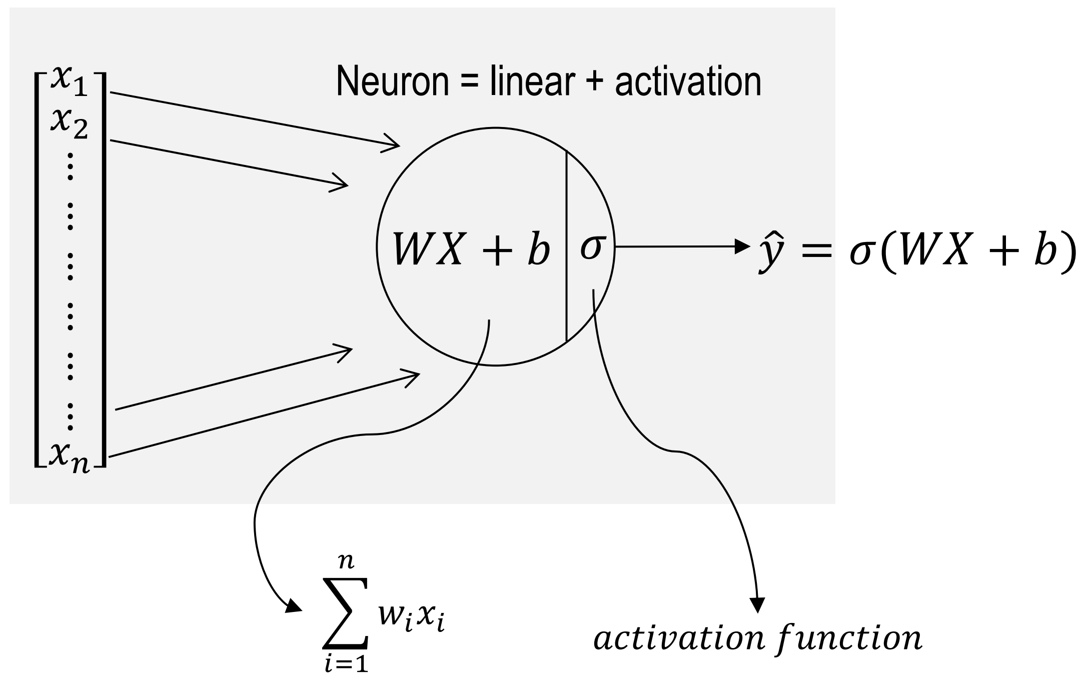


Deep learning & logistic regression
Goal 1 : 사진에서(이미지) 고양이와 개를 구분하는 이진분류

- Step 1 : initialize parameters, w(weight), b(bias)
- Step 2 : find the optimal w, b

  Min Loss function = $-[y\log\hat{y} + (1-y)\log(1-\hat{y})]$
- Step 3 : predict y to use $\hat{y} = \sigma(WX + b)$





4D tensor, 12,288 = pixel * RGB = (64 x 64)*3

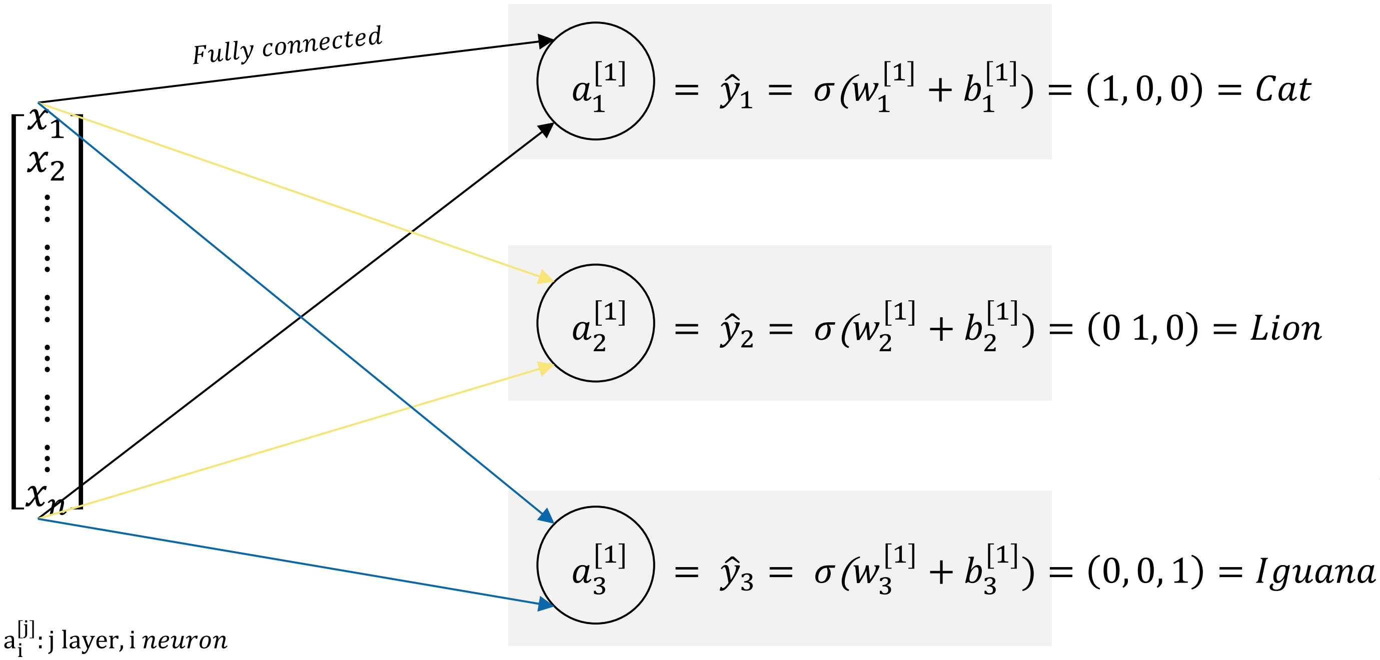
parameters :$(12{,}288 + 1)$

Deep learning & logistic regression
Goal 2 : 사진에서(이미지) 고양이, 사자, 이구아나를 분류하는 multi label classification

- Step 1 : initialize parameters, w(weight), b(bias)
- Step 2 : find the optimal w, b

  Min Loss function = $-\sum_{k=1}^{3}[y_k \log \hat{y}_k + (1-y_i)\log(1-\hat{y}_k)]$, MLE funtion
- Step 3 : predict y to use $\hat{y} = \sigma(WX+b)$

(1,1,0)출력 가능 즉 고양이와 사자가 같이 있는 이미지



$a_i^{[j]}$: j layer, i $neuron$
Neurons are independent.

Three logistic models

parameters :$(12{,}288+1)*3$

Deep learning & logistic regression

Goal 3 : 사진에서(이미지) 고양이, 사자, 이구아나를 분류하는 multi class classification

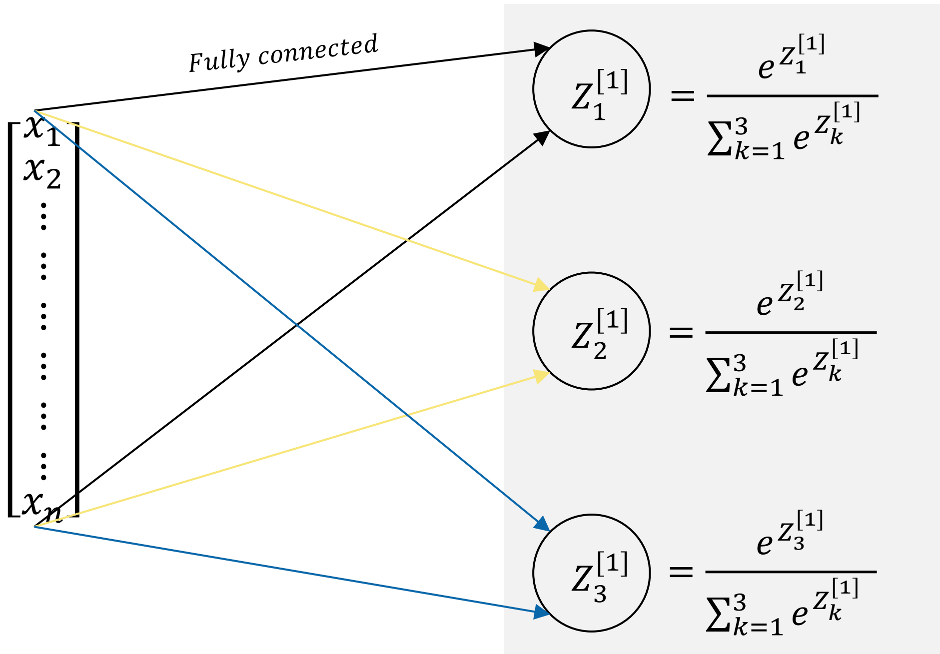
- Step 1 : initialize parameters, w(weight), b(bias)
- Step 2 : find the optimal w, b

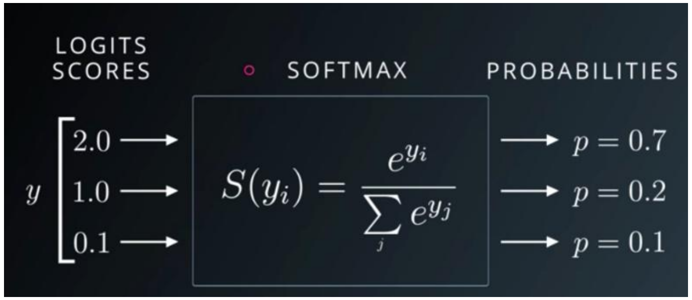
  Let $Z = WX + b,\quad z_i^{[j]} = w_i^{[j]} + b_i^{[j]}, layer\ j, neuron\ i$

  Min Loss function = $-\sum_{k=1}^{3} y_k \log \hat{y}_k$, cross entropy loss function

- Step 3 : predict y to use $\hat{y} = \sigma(Z)$

Softmax activation function





$a_i^{[j]}$: j layer, i $neuron$

Neurons are not independent.

Deep learning & logistic regression
Softmax function

The standard (unit) softmax function $\sigma : \mathbb{R}^K \to (0,1)^K$ is defined when K is greater than one by the formula

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{for } i = 1, \dots, K \text{ and } \mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K.$$

```
In [46]:
import numpy as np
a = [1.0, 2.0, 3.0, 4.0, 1.0, 2.0, 3.0]
np.exp(a) / np.sum(np.exp(a))

Out[46]: array([0.02364054, 0.06426166, 0.1746813 , 0.474833  , 0.02364054,
       0.06426166, 0.1746813 ])
```

```
In [47]:
print(np.sum(np.exp(a) / np.sum(np.exp(a))))

0.9999999999999999
```

Overflow

```
In [43]:
a = np.array([1.0, 2.0, 3.0, 4.0, 1.0, 2.0, 3.0])
a = a+10000
print(a)
np.exp(a) / np.sum(np.exp(a))

[10001. 10002. 10003. 10004. 10001. 10002. 10003.]
<ipython-input-43-bd6e323e1c75>:4: RuntimeWarning: overflow encountered in exp
  np.exp(a) / np.sum(np.exp(a))
<ipython-input-43-bd6e323e1c75>:4: RuntimeWarning: invalid value encountered in true_divide
  np.exp(a) / np.sum(np.exp(a))
Out[43]: array([nan, nan, nan, nan, nan, nan, nan])
```

```
In [48]:
np.exp(1000)

<ipython-input-48-47a6eab891c2>:1: RuntimeWarning: overflow encountered in exp
  np.exp(1000)
Out[48]: inf
```

Overflow 방지

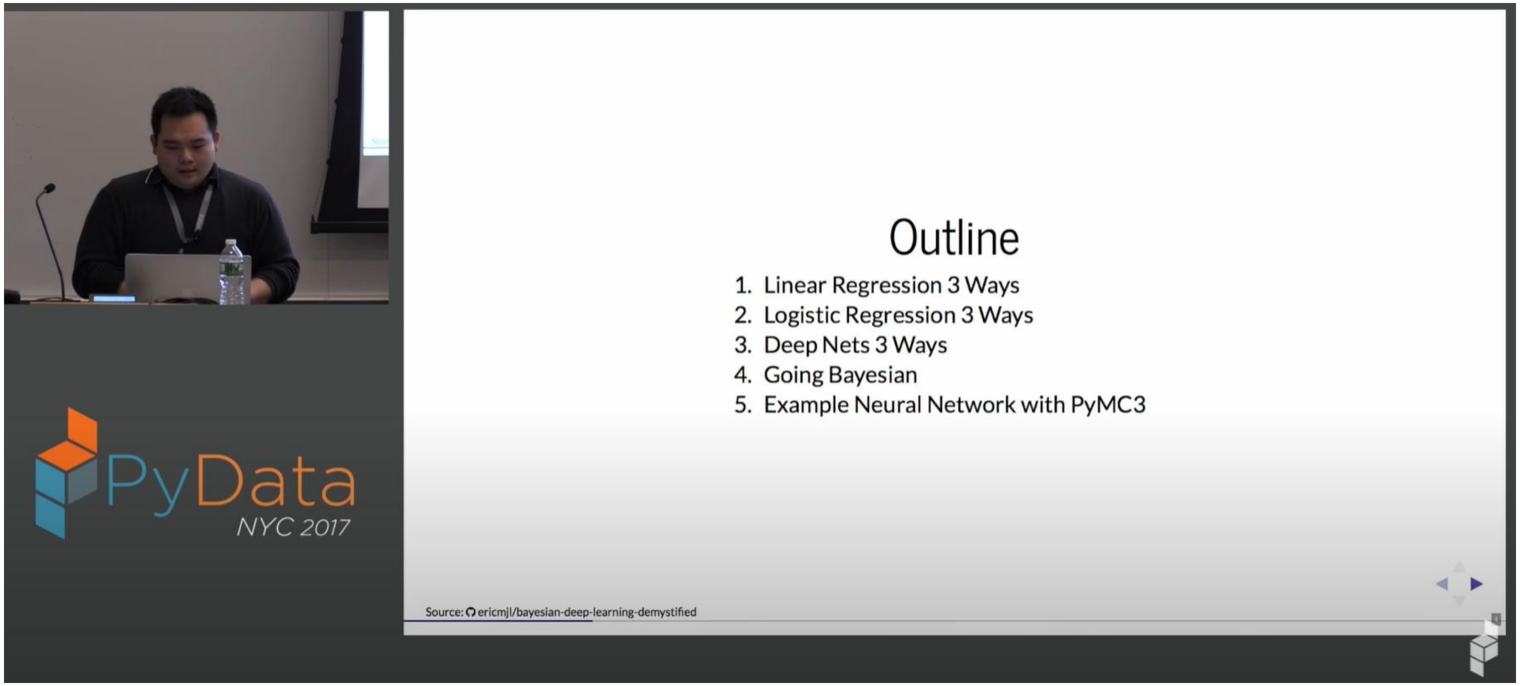
```
a = a -np.max(a)
np.exp(a) / np.sum(np.exp(a))

array([0.02364054, 0.06426166, 0.1746813 , 0.474833  , 0.02364054,
       0.06426166, 0.1746813 ])
```

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

Eric J. Ma - An Attempt At Demystifying Bayesian Deep Learning

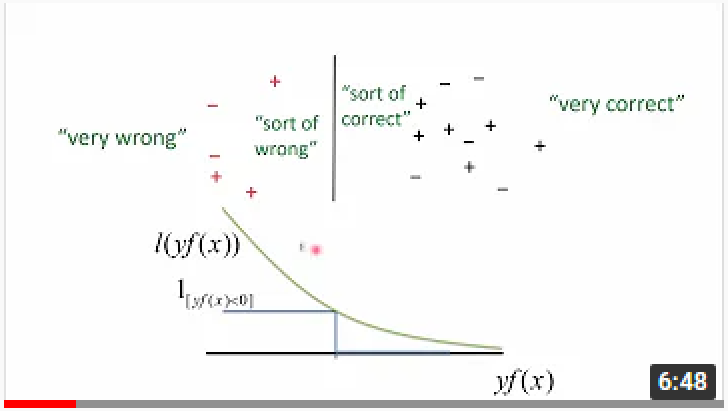
딥 러닝은 행렬에 대한 함수의 구성

Deep learning is nothing more than **compositions of functions on matrices.**



https://www.youtube.com/watch?v=s0S6HFdPtlA

Principles of Machine Learning | Loss Function for Classification

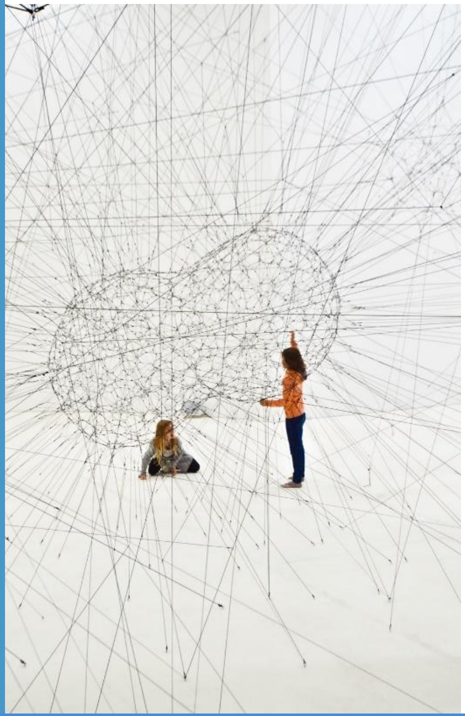


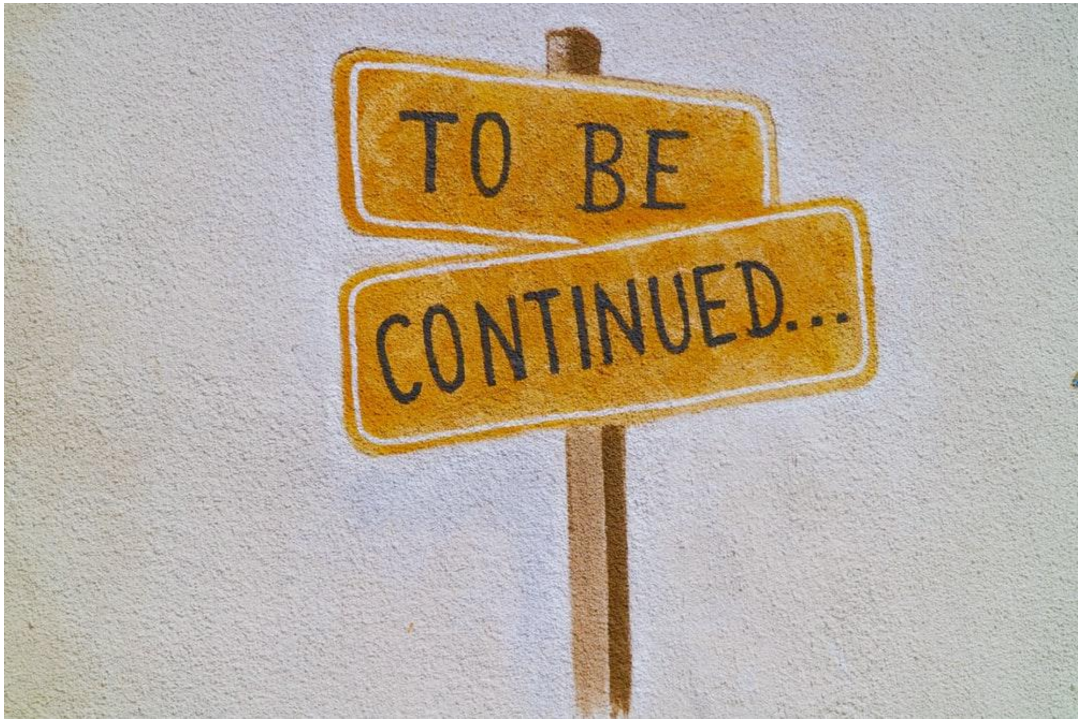
Principles of Machine Learning | Loss Function for Classification

13K views • 3 years ago

Damn! I Love Data

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




TO BE
CONTINUED...