with tensorflow

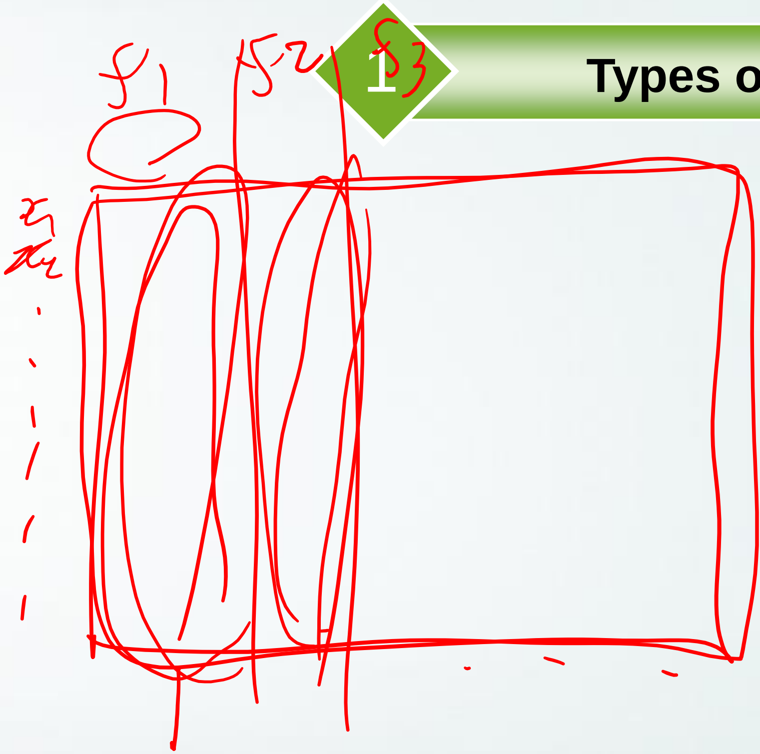
구현을 위한 딥러닝

- 고려대학교 물리학과 한승희
- mod96@naver.com

Contents

Contents

1 **Types of Problems**



scalar

Types of Problems - 1

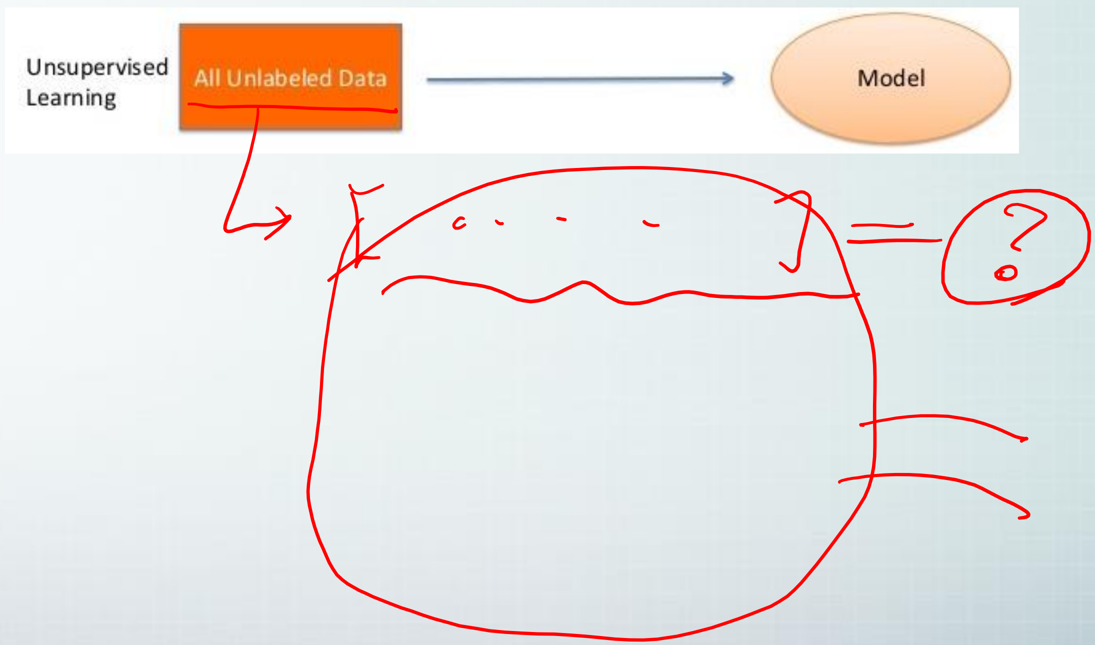
- Supervised Learning
- Unsupervised Learning
- Semi-Supervised Learning

Supervised Learning

- Majority of algorithms. Machine is trained using well-labeled data. (inputs and outputs are matched)
- Ex> Classification, Regression

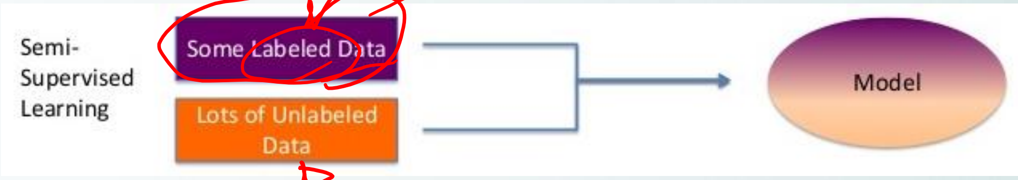
Unsupervised Learning

- Learning happens without supervision. Only inputs are used to create a model.
- Ex> Clustering



Semi - Supervised Learning

- Some data is labeled, some not. Since clean, perfectly labeled datasets aren't easy to come by, good for real world data.



Types of Problems - 2

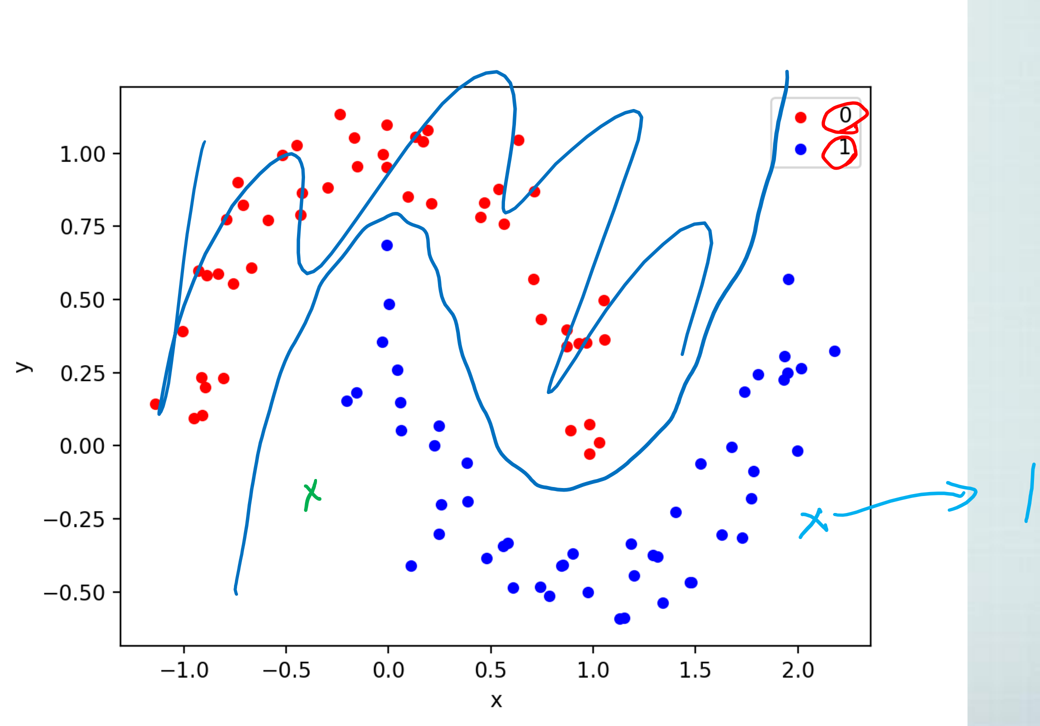
- Classification
- Regression
- Clustering
- Etc.

Sup

Un-sup.

Classification

❖ Predicts discrete number of values. The data is categorized under different labels



Classification

0 ~ 1 / standardization

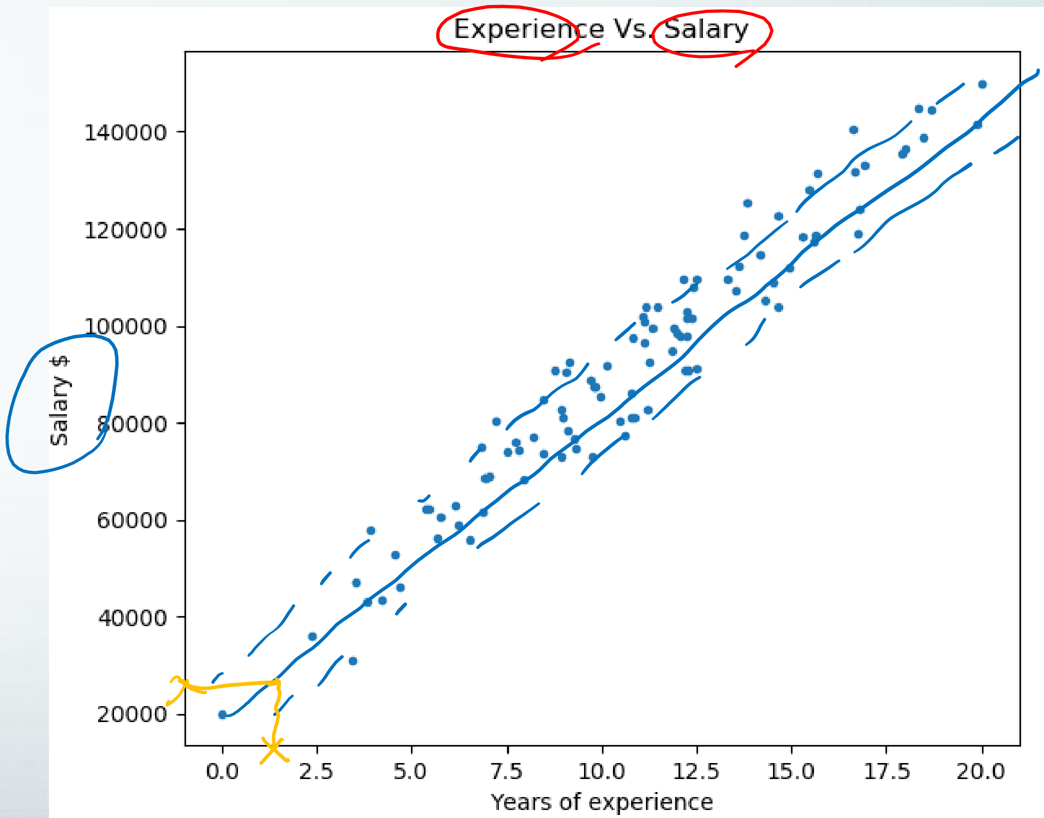
가중치

W

Sepal length	Sepal width	Petal length	Petal width	Species
5.2	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*
4.7	3.2	1.3	0.2	*I. setosa*
4.6	3.1	1.5	0.2	*I. setosa*
5.0	3.6	1.4	0.3	*I. setosa*
5.4	3.9	1.7	0.4	*I. setosa*
4.6	3.4	1.4	0.3	*I. setosa*
7.0	3.2	4.7	1.4	*I. versicolor*
6.4	3.2	4.5	1.5	*I. versicolor*
6.9	3.1	4.9	1.5	*I. versicolor*
5.5	2.3	4.0	1.3	*I. versicolor*
6.5	2.8	4.6	1.5	*I. versicolor*
5.7	2.8	4.5	1.3	*I. versicolor*
6.3	3.3	4.7	1.6	*I. versicolor*
4.9	2.4	3.3	1.0	*I. versicolor*
6.6	2.9	4.6	1.3	*I. versicolor*
6.3	3.3	6.0	2.5	*I. virginica*
5.8	2.7	5.1	1.9	*I. virginica*
7.1	3.0	5.9	2.1	*I. virginica*
6.3	2.9	5.6	1.8	*I. virginica*
6.5	3.0	5.8	2.2	*I. virginica*
7.6	3.0	6.6	2.1	*I. virginica*
4.9	2.5	4.5	1.7	*I. virginica*

Regression

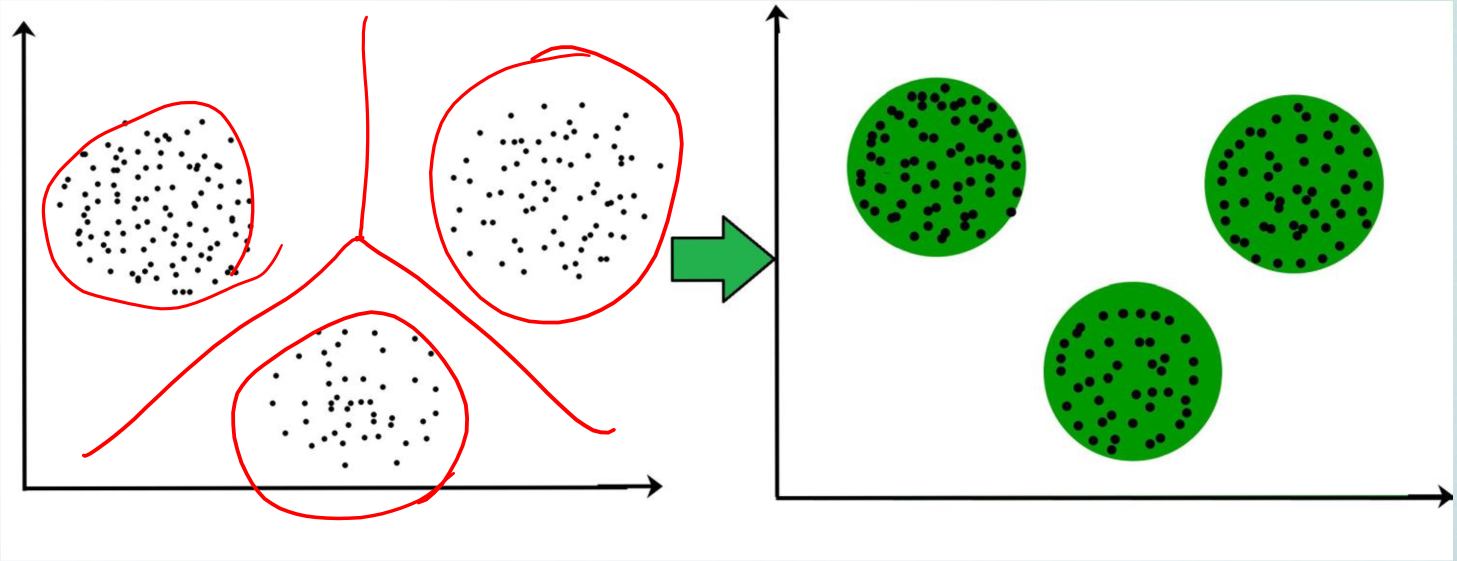
- Predicts continuous values output. Mostly analysis using statistical model which is used to predict the numeric data instead of labels.



이론
↓
실험
↓
데이터

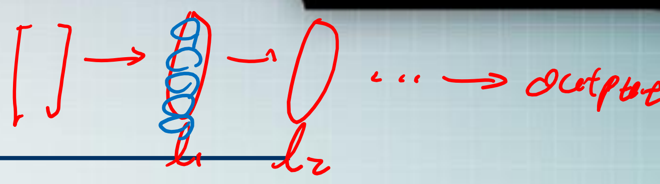
Clustering

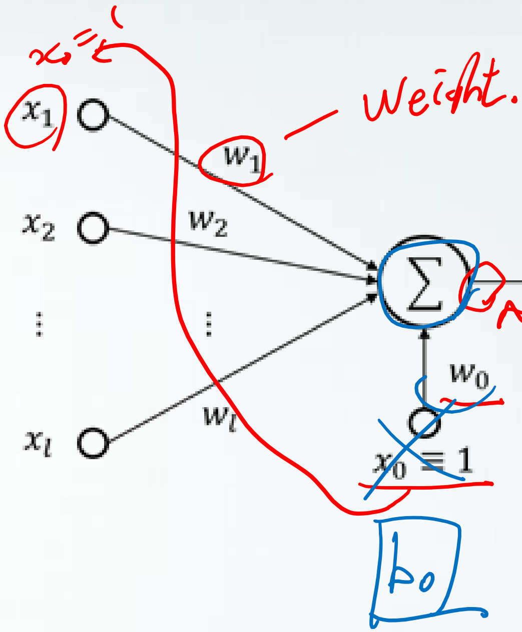
- Task of partitioning the dataset into groups, called clusters. This splits the data in such a way that points within single cluster are very similar and points in different are different.



Contents

Perceptron (Neuron)





weight.

$W = [w_0 \; w_1 \cdots w_l]$

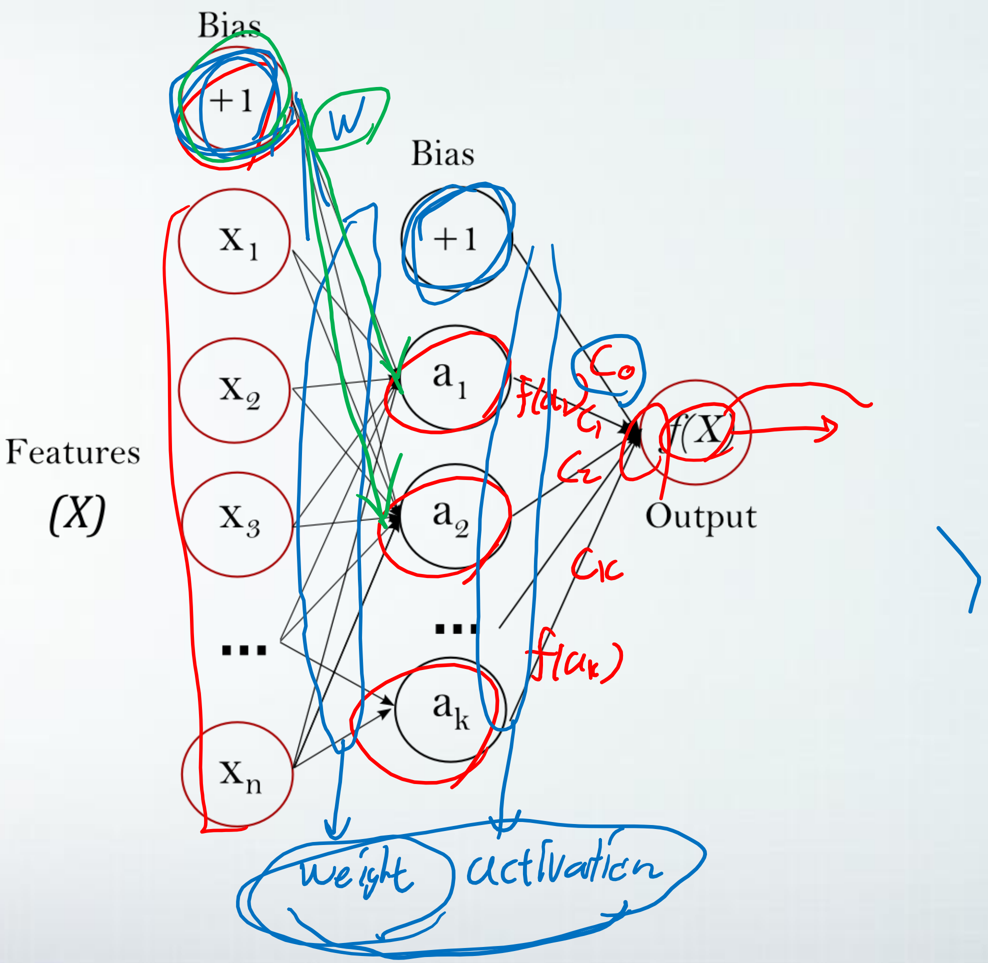
$Wx = \sum_0^l W_i x_i$

$f(\sum \sim)$

비선형 함수

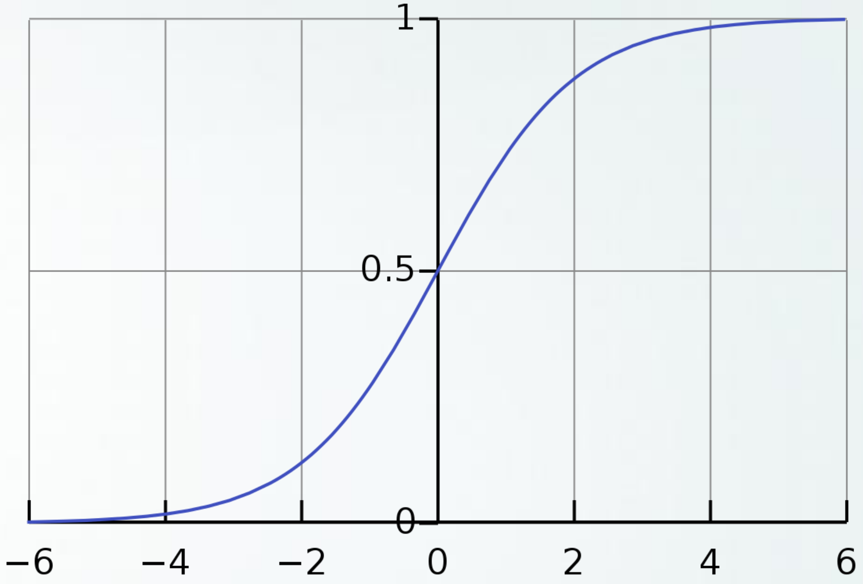
activation function

Multi Layer Perceptron



layer (tf. keras. layers. Dense)

Activation ftn (Sigmoid)



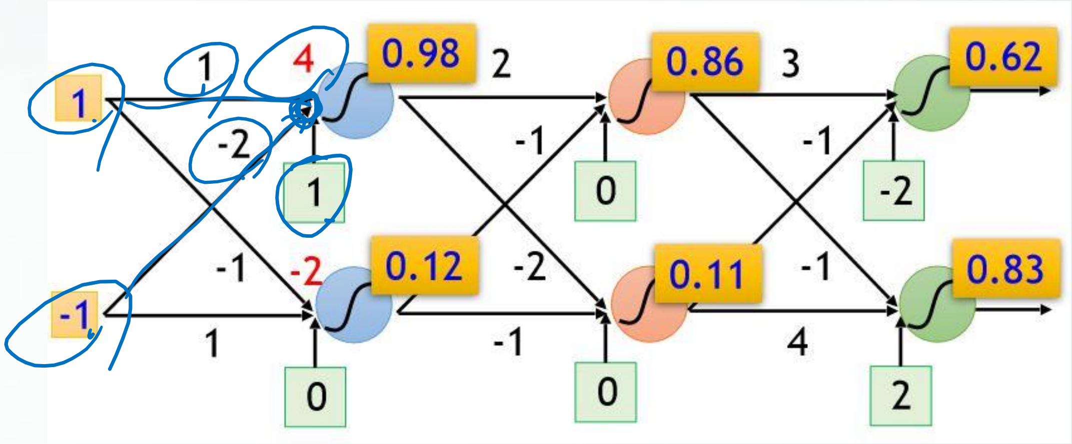
$$a(x) = \underbrace{\frac{1}{1+e^{-x}}} \quad : \ 0 \sim 1$$

$$a'(x) = - \frac{-e^{-x}}{(1+e^{-x})^2}$$

$$= \underbrace{\frac{1}{1+e^{-x}}} \times \underbrace{\frac{e^{-x}}{1+e^{-x}}}$$

$$= \underline{a(x)} \cdot \underline{(1-a(x))} \quad : 0 \sim 1$$

Example



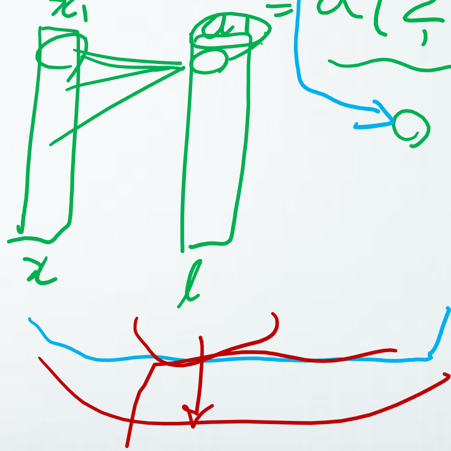
But…Why?

Universal Approximation thm

- The sum of the following form can approximate any continuous function F on $[0,1]^n$ to any degree of accuracy:

- $F(x) \approx \boxed{\sum_k c_k \sigma\left(\sum_{i=1}^{n} w_{ki} x_i + b_k\right)}$

$= a\left(\sum_i W_i x_i + b\right)$



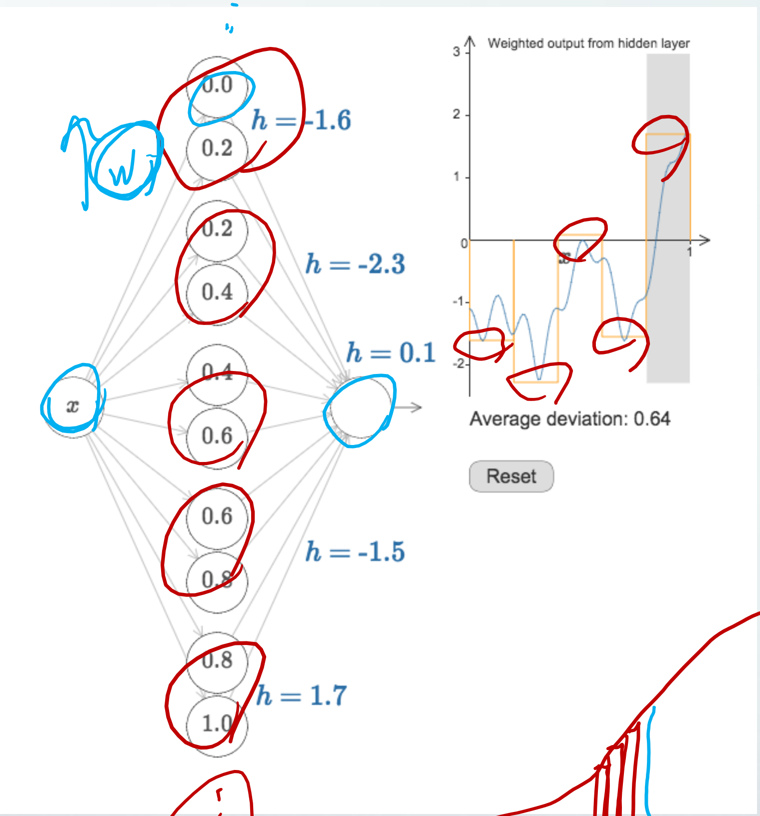
Universal Approximation thm

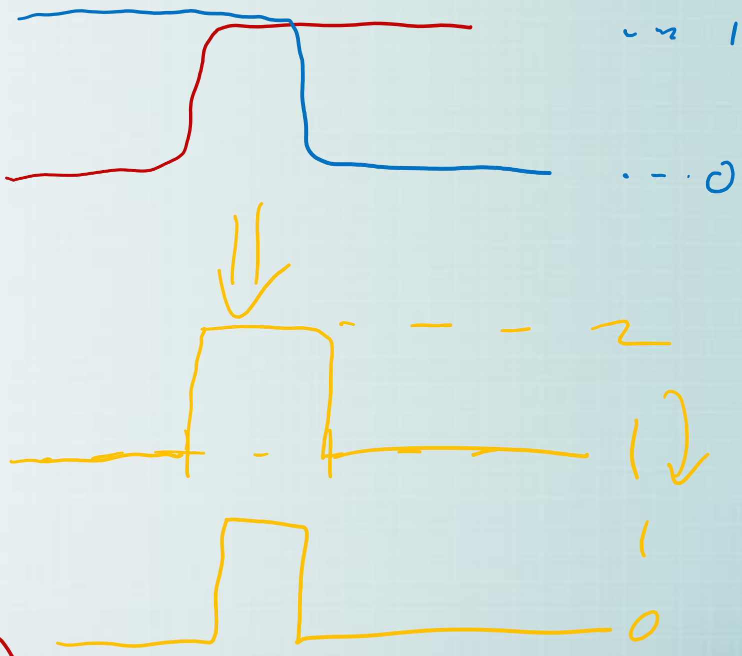
10×10×10

l_1 l_2 l_3

30 30

- See what happens with perceptron





Training – Loss ftn?

❖ **Loss ftns**

- Mean Squared Error
- Mean Absolute Error
- KL-Divergence
- Categorical Cross Entropy
- etc.

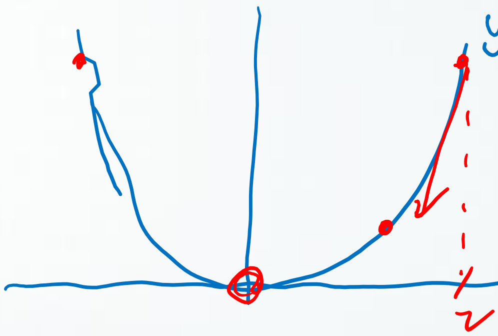
Training - Optimizer

- Gradient Descent & Backpropagation

$$\theta_{n+1} = \theta_n - \eta \cdot \nabla_\theta L(\theta)\Big|_{\theta_n}$$

학습률

$L(x) = y = x^2$



$x_0 = 2$

$x_1 = x_0 - \eta \cdot \nabla_x x^2 \Big|_{x=2}$

$= 2 - \eta \cdot \underbrace{2x}_{=4}$

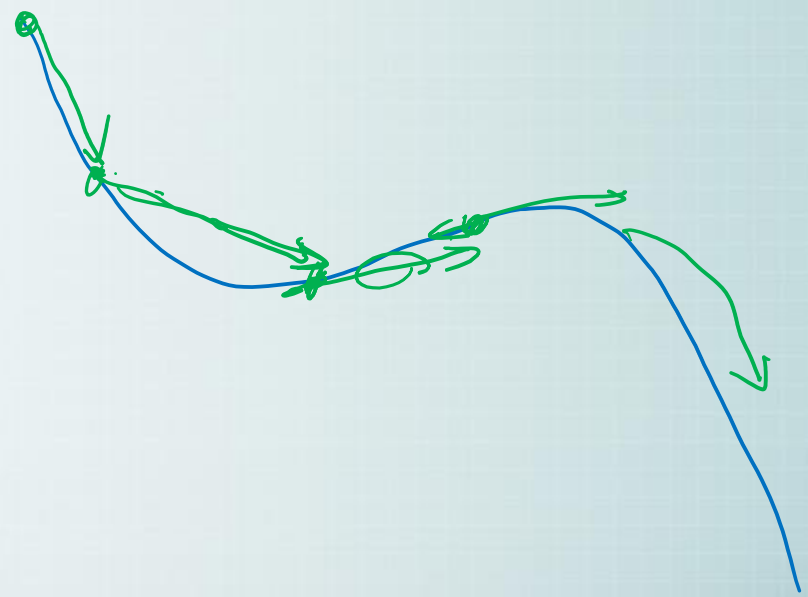
$\eta = 0.5 \Rightarrow 0$

$\eta = \frac{1}{4} \Rightarrow x_1 = 1$

Training - Optimizers

- **Momentum**

$$\Theta_{n+1} = m\,\Theta_n - \eta \cdot \nabla_\Theta L(\Theta)\Big|_{\Theta_n}$$



Training - Optimizers

- **Adagrad (adaptive gradient)**

$$\Theta_{n+1} = \Theta_n - \frac{\eta}{\sqrt{G_n + \epsilon}} \nabla_\Theta L(\Theta)\Big|_{\Theta_n}$$

$$G_n = \sum_{i=1}^{n} \left[\nabla_\Theta L(\Theta)\Big|_{\Theta_i} \right]^2$$

Training - Optimizers

- **RMSProp (root mean square propagation)**

$$\Theta_{n+1} = \Theta_n - \frac{\eta}{\sqrt{G_n + \epsilon}} \nabla_\Theta L(\Theta)\Big|_{\Theta_n}$$

$$G_n = \gamma \cdot G_{n-1} + (1 - \gamma) \left[\nabla_\Theta L(\Theta)\Big|_{\Theta_n} \right]^2$$

Training - Optimizers

빨리 수렴, overfit↓. / AdamW / SWE

❖ **$\boxed{\text{Adam}}$ (adaptive moment estimation)**

$$\hat{m}_\Theta = \frac{m_\Theta^{n+1}}{1-(\beta_1)^{n+1}} \quad \text{where} \quad m_\Theta^{n+1} = \beta_1 m_\Theta^n + (1-\beta)\nabla_\Theta L(\Theta)\Big|_{\Theta_n}$$

$$\hat{G}_\Theta = \frac{G_\Theta^{n+1}}{1-(\beta_2)^{n+1}} \quad \text{where} \quad G_\Theta^{n+1} = \beta_2 G_\Theta^n + (1-\beta_2)\left[\nabla_\Theta L(\Theta)\Big|_{\Theta_i}\right]^2$$

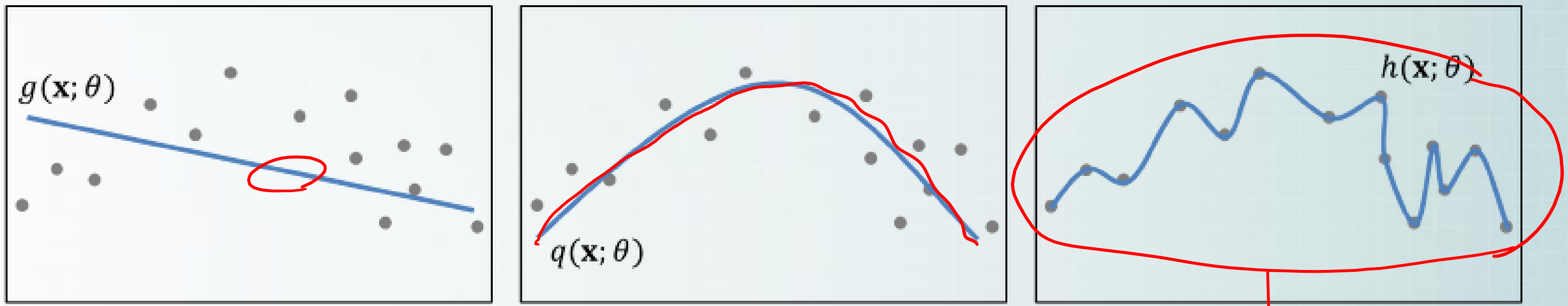
$$\Theta_{n+1} = \Theta_n - \eta \frac{\hat{m}_\Theta}{\sqrt{\hat{G}_\Theta + \epsilon}}$$

❖ **About learning rate…**

- Google exercise

Contents

Bias vs Variance

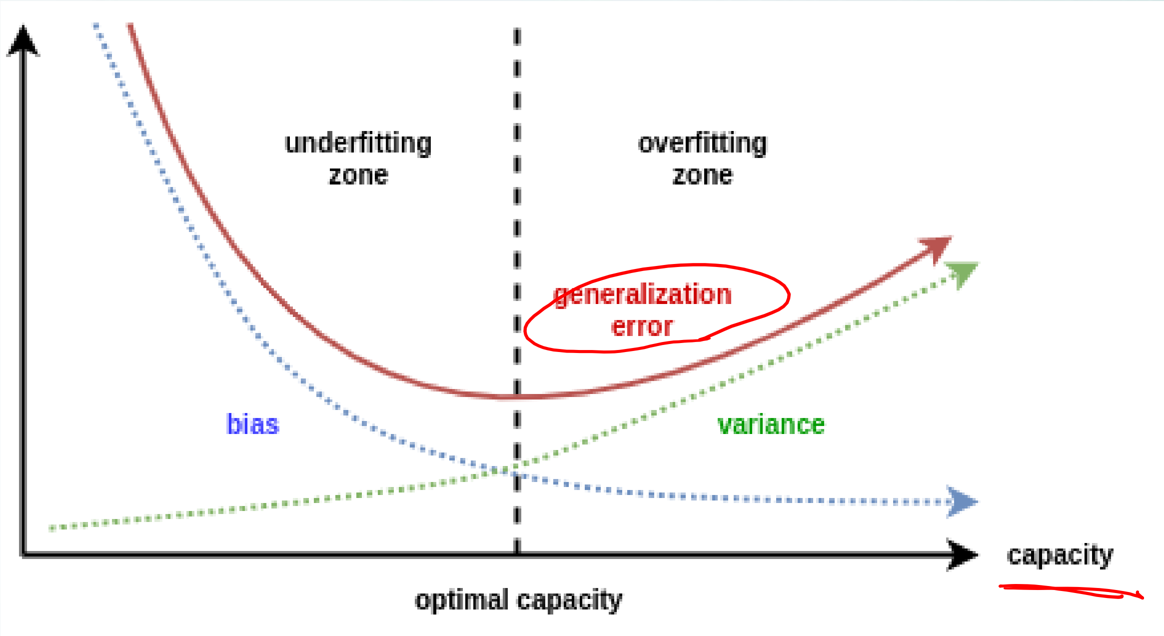


(a) Underfit (b) Ideal fit (c) Overfit

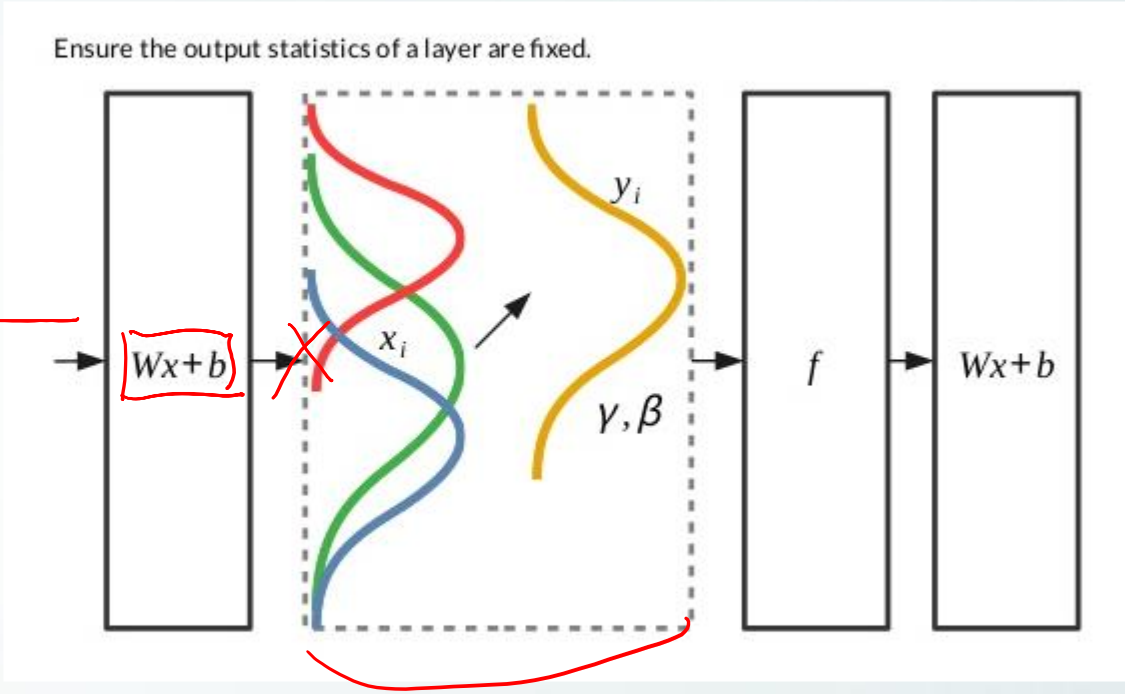
linear

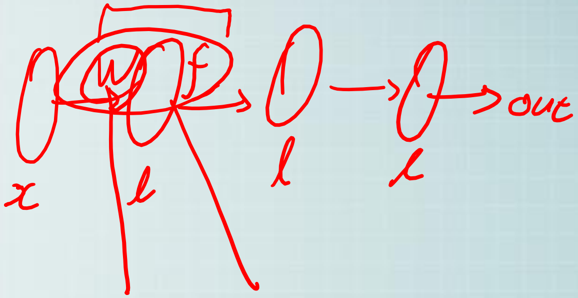
다항식 dim = data 개수

Bias vs Variance



Bias vs Variance : BN





Dense (~~activation = "sigmoid"~~)

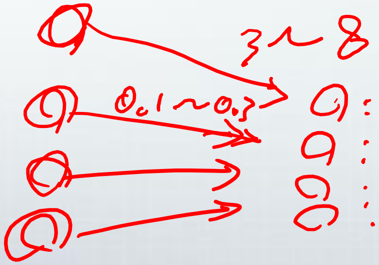
Dense

↓

BN

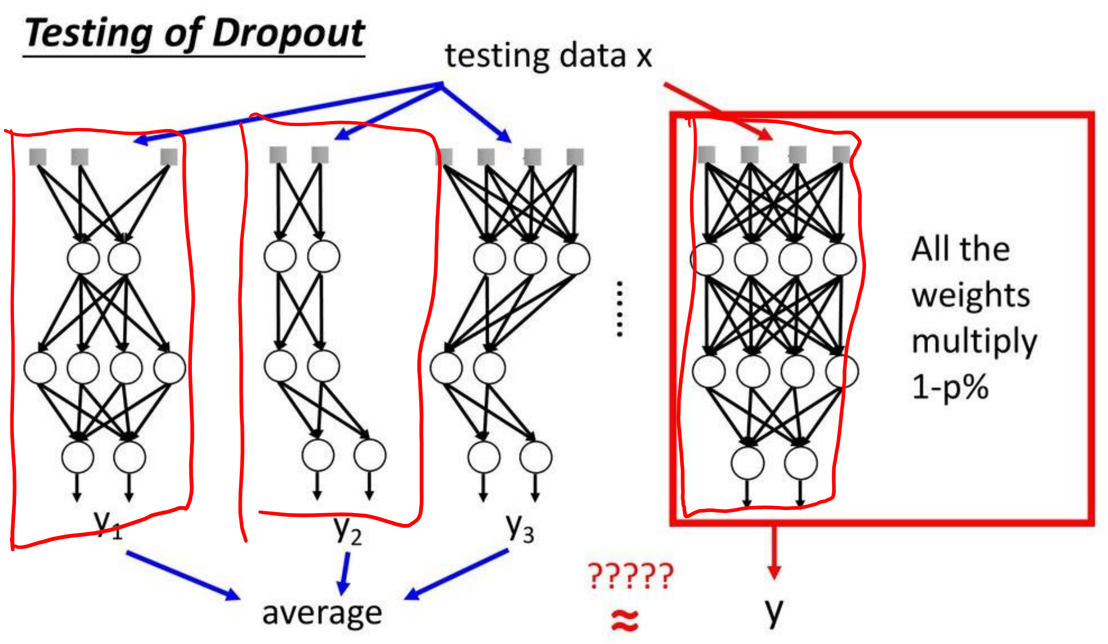
↓

Activation.



Bias vs Variance : dropout

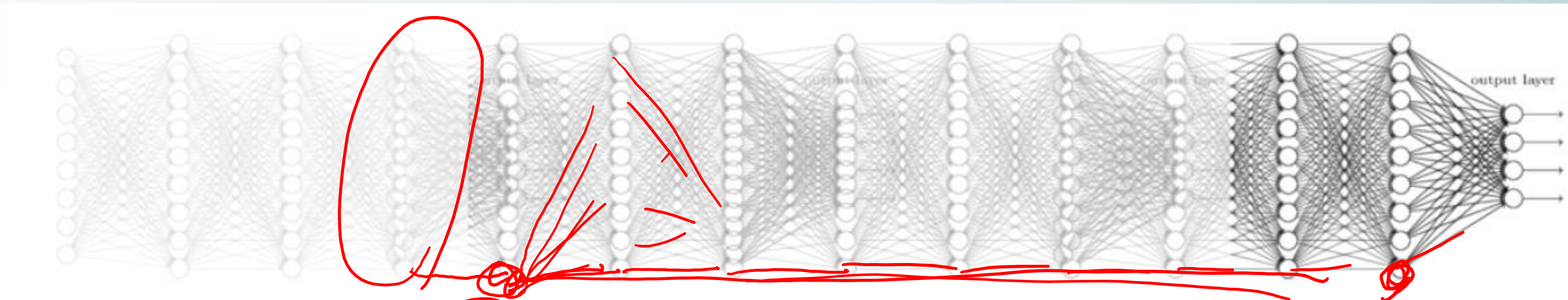
Dropout is a kind of ensemble.



Dense(uc—)

dropout(ratio=0.2)

Bias vs Variance : vanishing gradient



W

$w_{n+1} = w_n - \eta \nabla_w L$

f

$\boxed{\nabla_w L(w)}$

$a : 0 \sim 1$

$a' : 0 \sim 1$

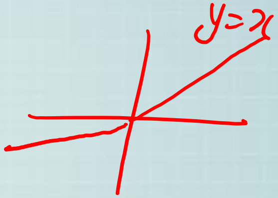
$\nabla_w L(w) \big|_{w = w_n}$

Vanishing Gradient : relu

|||

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^z)$	Multi-layer Neural Networks	

selu
elu
leaky-relu



$x > 0$

$\Rightarrow 1$

Contents

Normalization vs Standardization

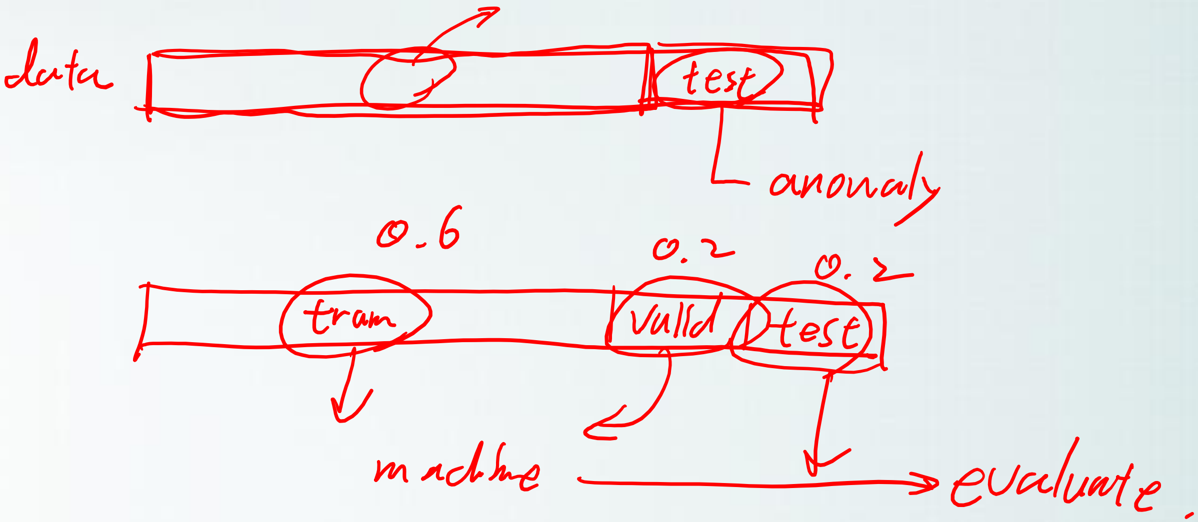
- **Normalization :**

$$\boxed{X' = \frac{X - X_{min}}{X_{max} - X_{min}}}$$

- **Standardization :**

$$\boxed{X' = \frac{X - \mu}{\sigma}}$$

Train-Valid-Test split



Contents

5 Implementation

Pima-Indian Diabetes dataset

- https://www.kaggle.com/kumargh/pimaindiansdiabetescsv

Credit Card Fraud Detection

- https://www.kaggle.com/mlg-ulb/creditcardfraud/