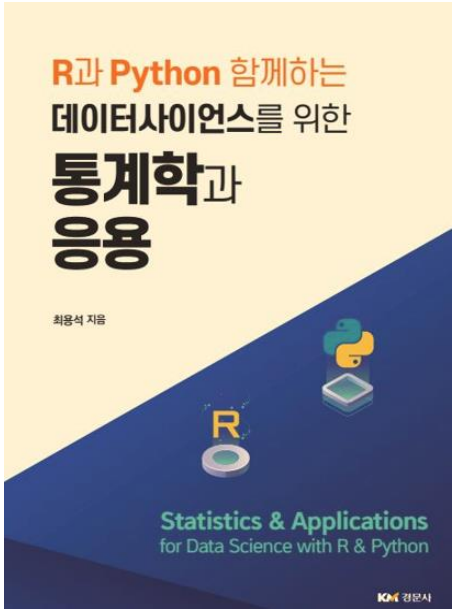
Techniques and Applications of Multivariate Analysis

A. Machine Learnings (ML)

Contents



2025, Chapter 9 Machine Learnings

A.1 Comprehension of ML

A.2 SVM(support vector machine)

A.3 ANN(Artificial Neural Network)

A.4 R for ML : Practice Time

Application:

Hierarchical CA, MDS, SVM and DNN in Text Mining

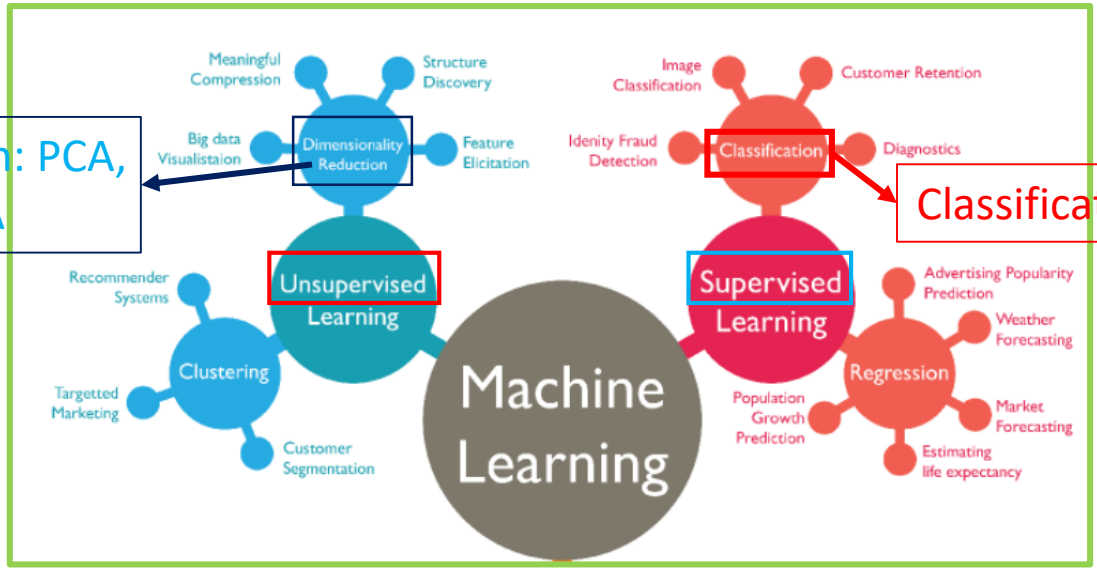
A.1 Comprehension of ML

- **Definition of ML**(machine learning) **:**

ML is **the study of computer algorithms that can improve automatically through experience and by the use of data.**

- Recently, ML became very important and useful approaches for classification and clustering of Big data which are structured and unstructured data.

- It is seen as a part of AI(artificial intelligence).



Dimension Reduction: PCA, CA, Biplot, MDS, CRA

Classification + DA

https://velog.io/@kjune1236/Machine-Learning

A.1 Comprehension of ML

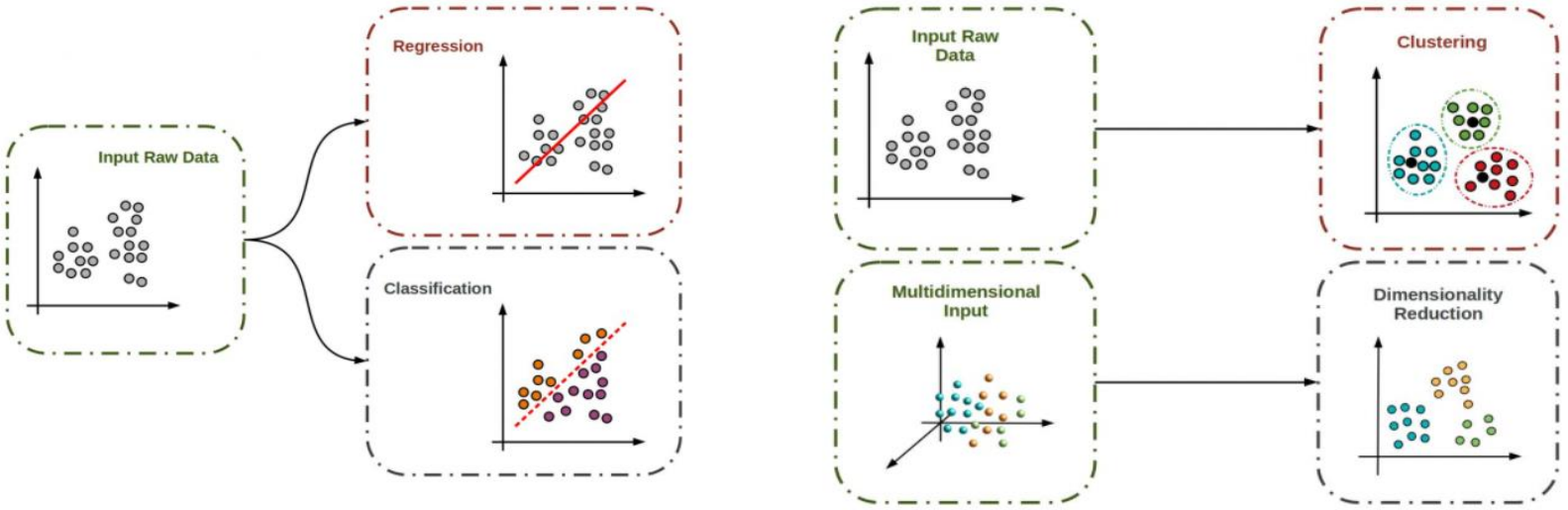
- **Definition of AI(artificial intelligence) :**

AI is what allows to imitate **human intelligence activities.**

AI refers to the field of **computer engineering** and **information technology** that studies the **thinking, learning, imitation** and **self-development** that human intelligence can do.

A.1 Comprehension of ML

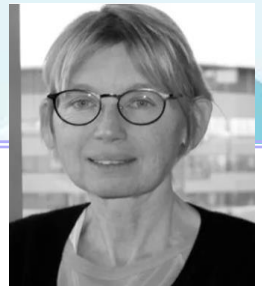
- **Learnings : Supervised vs. Unsupervised (James et al., 2013)**
- **Supervised : the goal is to use the input to predict the values of the outputs**
  - **SVM(support vector machine), L(Q)DA, Classification Trees, Random Forest,…**
- **Unsupervised : the goal is to directly discover interesting thing about inputs for each observation without a supervisor or teacher (often part of EDA)**
  - **PCA, Cluster Analysis(K-means), Biplot, MDS, CRA…**



Supervised Learning *Unsupervised Learning*

https://starship-knowledge.com/supervised-vs-unsupervised-vs-reinforcement

A.2 SVM(support vector machine)

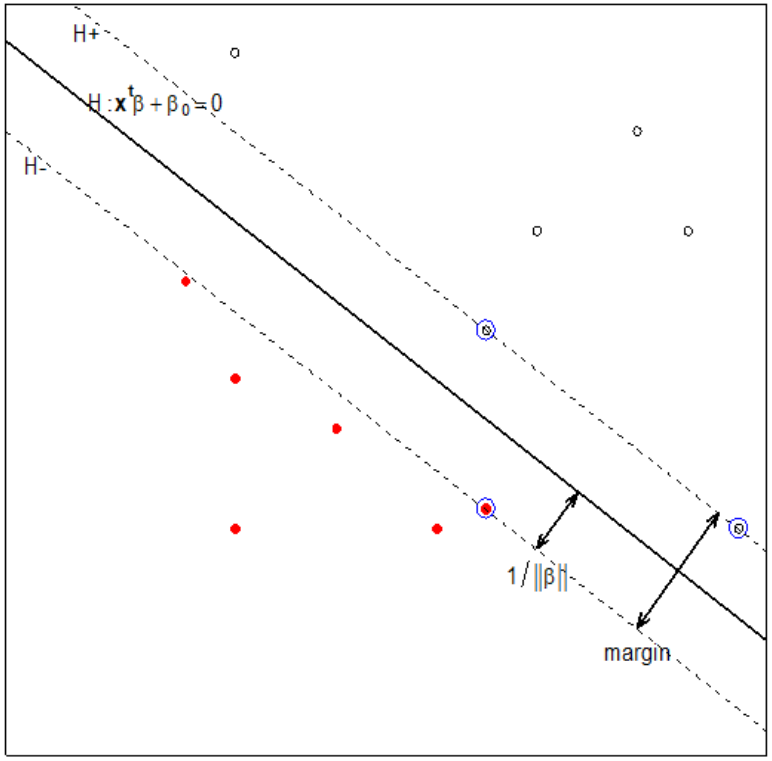


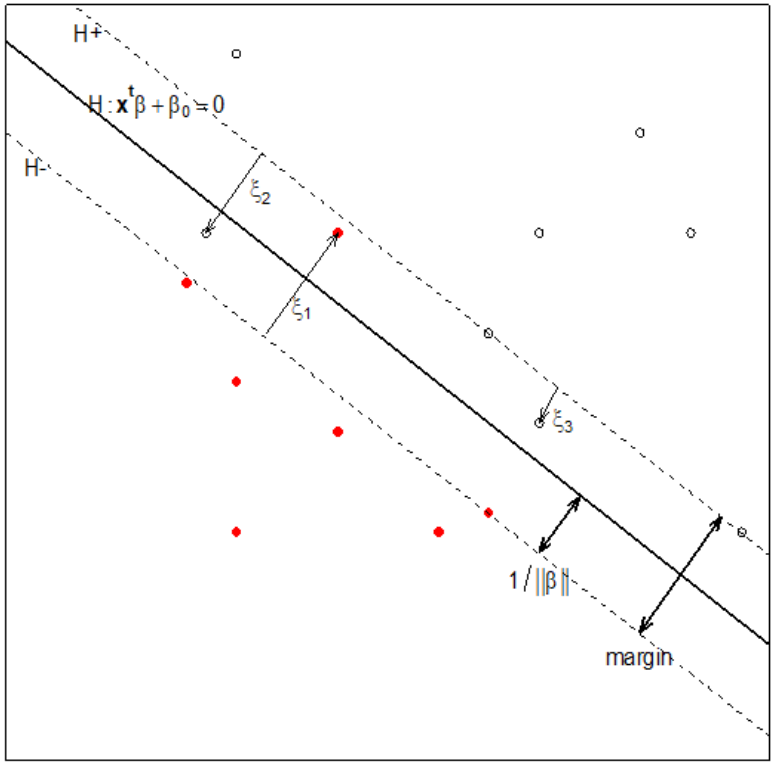
Corinna Cortes(1961~, Denmark)

Vladimir Vapnik(1936 ~, Russia)

- **Definition:**
- SVM was developed by Cortes and Vapnik(1995) for binary classification : Binary response variable
- SVM is a powerful technique for general(nonlinear) data classification that separates data using separating hyperplanes. This is based on the algorithms that analyze data for classification and regression analysis.
- Moreover it is especially useful for data whose distribution is unknown (also known as non-regularity in data).
- **Note: Usually, traditional DA(LDA and QDA) need the assumption of MVN and homogeneity of covariance matrices in clusters or groups.**

A.2 SVM(support vector machine)





Separable case: points on the dot line are support vectors	Non-separable case: slack variables for each observation : ξ_i

Linear SVM : Linearly Separable SVM vs. Linearly Non-separable SVM

A.2 SVM(support vector machine)

- 여기에 수식을 입력하십시오.

Algorithms for Linear SVM

A learning set of data : $\mathcal{L} = \{(\boldsymbol{x}_i, y_i) : i = 1, \ldots, n\}$, s.t $\boldsymbol{x}_i \in \mathbb{R}^p$, $y_i \in \{-1, 1\}$

Separating function

Maximum Margin Hyper-plane : $H = \{\boldsymbol{x} : f(\boldsymbol{x}) = \beta_0 + \boldsymbol{x}^T\boldsymbol{\beta}\}$

where $\boldsymbol{x} = (x_1, \ldots, x_p)^T$, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T$, β_0: bias

$\Rightarrow$ $H_+ : \{\boldsymbol{x} : \beta_0 + \boldsymbol{x}^T\boldsymbol{\beta} = +1\}$, $H_- : \{\boldsymbol{x} : \beta_0 + \boldsymbol{x}^T\boldsymbol{\beta} = -1\}$

(1) Optimization Problem for Linearly Separable SVM

Optimization : $Max_H \frac{2}{\|\boldsymbol{\beta}\|} \Leftrightarrow \min_{\boldsymbol{\beta}, \beta_0} \frac{1}{2}\|\boldsymbol{\beta}\|^2$, s.t $y_i(\boldsymbol{x}^T\boldsymbol{\beta} + \beta_0) \geq 1$, $i = 1, \ldots, n$

To find optimal separating hyper plane H $\leftrightarrow$ *To estimate* β_0 and β

A.2 SVM(support vector machine)

Optimization Problems: Primal and Dual Problems

*Lagrag*ian Multipler Method :

[Step 1] $L_p(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 - \sum_{i=1}^{n}\alpha_i y_i(\beta_0 + \boldsymbol{x}_i{}^T\boldsymbol{\beta}) + \sum_{i=1}^{n}\alpha_i$ s.t $\alpha_i \geq 0,\ i = 1, \ldots, n$

$Min_{\beta_0, \boldsymbol{\beta}} L_p \Rightarrow L_p(\hat{\beta}_0, \hat{\boldsymbol{\beta}}, \boldsymbol{\alpha}) = L_d(\boldsymbol{\alpha})$ with $\sum_{i=1}^{n}\alpha_i y_i = 0$ and $\hat{\boldsymbol{\beta}} = \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i$

[Step 2] $L_d(\alpha) = 1_n^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^T H\boldsymbol{\alpha}$ $s.t$ $\boldsymbol{\alpha} \geq 0, \boldsymbol{\alpha}^T\boldsymbol{y} = 0$

$\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n)^T$, $\boldsymbol{y} = (y_1, \cdots, y_n)^T$, $H = (h_{ij})$ with $h_{ij} = y_i y_j(\boldsymbol{x}_i{}^T\boldsymbol{x}_j)$: $n \times n$

$Max_{\boldsymbol{\alpha}} L_d \Rightarrow \hat{\boldsymbol{\alpha}} = (\hat{\alpha}_1, \cdots, \hat{\alpha}_n)^T$ and $\hat{\boldsymbol{\beta}} = \sum_{i=1}^{n}\hat{\alpha}_i y_i \boldsymbol{x}_i$

[Step 3] $\hat{f}(\boldsymbol{x}) = \hat{\beta}_0 + \boldsymbol{x}^T\hat{\boldsymbol{\beta}} = \hat{\beta}_0 + \sum_{i=1}^{n}\hat{\alpha}_i y_i \boldsymbol{x}^T\boldsymbol{x}_i$: Linear SVM classifier

with $\hat{\beta}_0 = \frac{1}{n_{sv}}\sum_{k=1}^{n_{sv}}\left(\frac{1 - y_i\boldsymbol{x}_i^T\hat{\boldsymbol{\beta}}}{y_i}\right)$

(1) Optimization Problems for Linear Separable Case of SVM :

$$\min_{\boldsymbol{\beta},\beta_0} \frac{1}{2}\|\boldsymbol{\beta}\|^2 \;\; s.t \;\; y_i(\beta_0 + \boldsymbol{x}_i{}^T\boldsymbol{\beta}) \geq 1, \;\; i = 1, \ldots, n$$

Lagrangian Multiplier Method :

[Step 1] $L_p(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 - \sum_{i=1}^{n}\alpha_i y_i(\beta_0 + \boldsymbol{x}_i{}^T\boldsymbol{\beta}) + \sum_{i=1}^{n}\alpha_i$

- $\frac{\partial L_p}{\partial \beta_0} = 0 \Rightarrow -\sum_{i=1}^{n}\alpha_i y_i = 0 \Rightarrow \sum_{i=1}^{n}\alpha_i y_i = 0$
- $\frac{\partial L_p}{\partial \boldsymbol{\beta}} = 0 \Rightarrow \boldsymbol{\beta} - \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i = 0 \Rightarrow \hat{\boldsymbol{\beta}} = \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i$

$\Rightarrow L_p(\hat{\beta}_0, \hat{\boldsymbol{\beta}}, \boldsymbol{\alpha}) = \frac{1}{2}\|\hat{\boldsymbol{\beta}}\|^2 - \sum_{i=1}^{n}\alpha_i y_i(\hat{\beta}_0 + \boldsymbol{x}_i{}^T\hat{\boldsymbol{\beta}}) + \sum_{i=1}^{n}\alpha_i$

- $\frac{1}{2}\|\hat{\boldsymbol{\beta}}\|^2 = \frac{1}{2}\sum_{i}^{n}\sum_{j}^{n}\alpha_i\alpha_j y_i y_j(\boldsymbol{x}_i{}^T\boldsymbol{x}_j)$
- $-\sum_{i=1}^{n}\alpha_i y_i(\hat{\beta}_0 + \boldsymbol{x}_i{}^T\hat{\boldsymbol{\beta}}) = -\hat{\beta}_0\sum_{i=1}^{n}\alpha_i y_i - \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i{}^T\hat{\boldsymbol{\beta}}$

$$= -\sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i{}^T\sum_{j=1}^{n}\alpha_j y_j \boldsymbol{x}_j \left(\because \sum_{i=1}^{n}\alpha_i y_i = 0, \; \hat{\boldsymbol{\beta}} = \sum_{j=1}^{n}\alpha_j y_j \boldsymbol{x}_j\right)$$

$$= -\sum_{i}^{n}\sum_{j}^{n}\alpha_i\alpha_j y_i y_j(\boldsymbol{x}_i{}^T\boldsymbol{x}_j)$$

(1) Optimization Problems for Linear Separable Case of SVM :

[Step 2] $L_d(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i}^{n} \sum_{j}^{n} \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i{}^T \boldsymbol{x}_j) = 1_n^T \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}^T H \boldsymbol{\alpha}$

where $n \times n$ $H = (h_{ij})$ with $h_{ij} = y_i y_j (\boldsymbol{x}_i{}^T \boldsymbol{x}_j)$

Now $Max_{\boldsymbol{\alpha}} L_d$ *s.t* $\boldsymbol{\alpha} \geq 0, \boldsymbol{\alpha}^T \boldsymbol{y} = 0$

- $\frac{\partial L_d}{\partial \boldsymbol{\alpha}} = 0 \Rightarrow \hat{\boldsymbol{\alpha}} = (\hat{\alpha}_1, \ldots, \hat{\alpha}_n)^T \Rightarrow \hat{\boldsymbol{\beta}} = \sum_{i=1}^{n} \hat{\alpha}_i y_i \boldsymbol{x}_i$

[Step 3] $\hat{f}(\boldsymbol{x}) = \hat{\beta}_0 + \boldsymbol{x}^T \hat{\boldsymbol{\beta}} = \hat{\beta}_0 + \sum_{i=1}^{n} \hat{\alpha}_i y_i \boldsymbol{x}^T \boldsymbol{x}_i$ with $\hat{\beta}_0 = \frac{1}{n_{sv}} \sum_{i=1}^{n_{sv}} \left(\frac{1 - y_i \boldsymbol{x}_i^T \hat{\boldsymbol{\beta}}}{y_i} \right)$

A.2 SVM(support vector machine)

(2) Optimization Problem for Linearly Non-separable SVM

$$\min_{\boldsymbol{\beta},\beta_0} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + c\sum_{i=1}^{n}\xi_i \ , \ \text{s.t} \ \ \xi_i \geq 0, \ \ y_i(\boldsymbol{x}^T\boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i, \ i = 1, \ \ldots \ , n$$

where c: tuning(cost) parameter controling the size of the slack variables ξ_i for each observation $(\boldsymbol{x}_i, y_i)$ in $\mathcal{L}$, $i = 1, \ \ldots \ , n$.

SVM classifiers : Linear case and Non-linear case

1) Linear SVM classifier :

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p, \text{ inner product: } <\boldsymbol{x}, \boldsymbol{y}> \ = \boldsymbol{x}^T\boldsymbol{y} = \sum_{j=1}^{p} x_j y_j$$

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{n}\alpha_i y_i <\boldsymbol{x}, \boldsymbol{x}_i> \quad \Longrightarrow \quad \widehat{f}(\boldsymbol{x}) = \widehat{\beta}_0 + \sum_{i=1}^{n}\widehat{\alpha}_i y_i \boldsymbol{x}^T\boldsymbol{x}_i$$

2) Non-linear SVM classifier :

$\Phi : \mathbb{R}^p \mapsto \mathbb{H}$:
a nonlinear function into a very high-dimensional Hilbert Space $\mathbb{H}$

Non-linear transformation with Kernel trick

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{n}\alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i) \Longrightarrow \quad \widehat{f}(\boldsymbol{x}) = \widehat{\beta}_0 + \sum_{i=1}^{n}\widehat{\alpha}_i y_i \Phi(\boldsymbol{x})^T\Phi(\boldsymbol{x}_i)$$

where $K(\boldsymbol{x}, \boldsymbol{y}) = <\Phi(\boldsymbol{x}), \Phi(\boldsymbol{y})> = \Phi(\boldsymbol{x})^T\Phi(\boldsymbol{y}), \ \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p$: Inner product

(2) Optimization Problems for Linear Non-separable Case of SVM :

$$\min_{\boldsymbol{\beta},\beta_0} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + c\sum_{i=1}^{n}\xi_i \;\; s.t\; \xi_i \geq 0, y_i(\beta_0 + \boldsymbol{x}_i^T\boldsymbol{\beta}) \geq 1-\xi_i, i = 1, \ldots, n$$

[Step 1] $L_p(\beta_0, \boldsymbol{\beta}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\eta}) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + c\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left(y_i(\beta_0 + \boldsymbol{x}_i^T\boldsymbol{\beta} - (1-\xi_i)\right) - \sum_{i=1}^{n}\eta_i\xi_i$

- $\frac{\partial L_p}{\partial \beta_0} = 0 \Rightarrow -\sum_{i=1}^{n}\alpha_i y_i = 0 \Rightarrow \sum_{i=1}^{n}\alpha_i y_i = 0$
- $\frac{\partial L_p}{\partial \boldsymbol{\beta}} = 0 \Rightarrow \boldsymbol{\beta} - \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i = 0 \Rightarrow \hat{\boldsymbol{\beta}} = \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i$
- $\frac{\partial L_p}{\partial \eta_i} = 0 \Rightarrow c - \alpha_i - \eta_i = 0 \Rightarrow \alpha_i = c - \eta_i \Rightarrow 0 \leq \alpha_i < c$

[Step 2] $L_d(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i}^{n}\sum_{j}^{n}\alpha_i\alpha_j y_i y_j(\boldsymbol{x}_i^T\boldsymbol{x}_j) = 1_n^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^T H\boldsymbol{\alpha}$

where $n \times n$ $H = (h_{ij})$ with $h_{ij} = y_i y_j(\boldsymbol{x}_i^T\boldsymbol{x}_j)$

- $\frac{\partial L_d}{\partial \boldsymbol{\alpha}} = 0 \Rightarrow \hat{\boldsymbol{\alpha}} = (\hat{\alpha_1}, \ldots, \hat{\alpha_n})^T \Rightarrow \hat{\boldsymbol{\beta}} = \sum_{i=1}^{n}\hat{\alpha_i} y_i \boldsymbol{x}_i = \sum_{i=1}^{n}(c - \eta_i) y_i \boldsymbol{x}_i$

[Step 3] $\hat{f}(\boldsymbol{x}) = \hat{\beta_0} + \boldsymbol{x}^T\hat{\boldsymbol{\beta}} = \hat{\beta_0} + \sum_{i=1}^{n}\hat{\alpha_i} y_i \boldsymbol{x}^T\boldsymbol{x}_i$ with $\hat{\beta_0} = \frac{1}{n_{sv}}\sum_{i=1}^{n_{sv}}\left(\frac{1 - y_i\boldsymbol{x}_i^T\hat{\boldsymbol{\beta}}}{y_i}\right)$

(3) Optimization Problems for Non-linear Case of SVM :

$$\mathcal{L}_H = \{(\Phi(\boldsymbol{x}_i), y_i) : i = 1, \ldots, n\} \; s.t \;\; \boldsymbol{x}_i \in \mathbb{R}^p, \quad y_i \in \{-1, 1\}$$

$$\min_{\boldsymbol{\beta}, \beta_0} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + c\sum_{i=1}^{n}\xi_i \;\; s.t \;\; \xi_i \geq 0,\; y_i(\beta_0 + \Phi(\boldsymbol{x}_i)^T\boldsymbol{\beta}) \geq 1 - \xi_i \;, i = 1, \ldots, n$$

[Step 1] $$L_p(\beta_0, \boldsymbol{\beta}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\eta}) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + c\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\big(y_i(\beta_0 + \Phi(\boldsymbol{x}_i)^T\boldsymbol{\beta} - (1 - \xi_i)\big) - \sum_{i=1}^{n}\eta_i\xi_i$$

- $\dfrac{\partial L_p}{\partial \beta_0} = 0 \;\Rightarrow\; \sum_{i=1}^{n}\alpha_i y_i = 0$
- $\dfrac{\partial L_p}{\partial \boldsymbol{\beta}} = 0 \;\Rightarrow\; \hat{\boldsymbol{\beta}} = \sum_{i=1}^{n}\alpha_i y_i \Phi(\boldsymbol{x}_i)$
- $\dfrac{\partial L_p}{\partial \eta_i} = 0 \;\Rightarrow\; \alpha_i = c - \eta_i \;\Rightarrow 0 \leq \alpha_i < c$

[Step 2] $$L_d(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i}^{n}\sum_{j}^{n}\alpha_i\alpha_j y_i y_j \Phi(\boldsymbol{x}_i)^T\Phi(\boldsymbol{x}_j) = 1_n^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^T H\boldsymbol{\alpha}$$

where $n \times n$ $H = (h_{ij})$ with $h_{ij} = y_i y_j \Phi(\boldsymbol{x}_i)^T\Phi(\boldsymbol{x}_j) = y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$

- $\dfrac{\partial L_d}{\partial \boldsymbol{\alpha}} = 0 \;\Rightarrow\; \hat{\boldsymbol{\alpha}} = (\widehat{\alpha_1}, \ldots, \widehat{\alpha_n})^T \;\Rightarrow\; \hat{\boldsymbol{\beta}} = \sum_{i=1}^{n}\widehat{\alpha_i} y_i \Phi(\boldsymbol{x}_i)$

[Step 3] $$\widehat{f}(\boldsymbol{x}) = \widehat{\beta_0} + \Phi(\boldsymbol{x})^T\hat{\boldsymbol{\beta}} = \widehat{\beta_0} + \sum_{i=1}^{n}\Phi(\boldsymbol{x})^T\Phi(\boldsymbol{x}_i) = \widehat{\beta_0} + \sum_{i=1}^{n}\widehat{\alpha_i} y_i K(\boldsymbol{x}, \boldsymbol{x}_i)$$

with $$\widehat{\beta_0} = \frac{1}{n_{sv}}\sum_{i=1}^{n_{sv}}\left(\frac{1 - y_i\Phi(\boldsymbol{x}_i)^T\hat{\boldsymbol{\beta}}}{y_i}\right)$$

A.2 SVM(support vector machine)

Kernel functions : ➡ Best choice : polynomial and RBF

$K(\boldsymbol{x}, \boldsymbol{y})$, $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p$, scale parameter $\gamma > 0$, integer d and $c \geq 0$.

kernel	formular $K(\boldsymbol{x}, \boldsymbol{y})$	parameters
linear	$\boldsymbol{x}^T\boldsymbol{y}$	none
polynomial	$(\gamma \boldsymbol{x}^T\boldsymbol{y} + c)^d$	γ, d, c
Gaussian radial basis function(RBF)	$\exp(-\gamma\|\boldsymbol{x} - \boldsymbol{y}\|^2)$	γ
sigmoid $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$\tanh(\gamma \boldsymbol{x}^T\boldsymbol{y} + c)$	γ, c

Note: Text classification: linear kernel is better than RBF kernel.

$\gamma \in (2^{-1}, 2)$, $c \in (2^2, 2^4)$

Example: $p = 2$, $d = 2 \Rightarrow \boldsymbol{x} = (x_1, x_2)^T$ and $\boldsymbol{y} = (y_1, y_2)^T$

$$\Rightarrow K(\boldsymbol{x}, \boldsymbol{y}) = (\gamma \boldsymbol{x}^T\boldsymbol{y} + c)^d = (\gamma(x_1 y_1 + x_2 y_2) + c)^2$$

$$K(\boldsymbol{x}, \boldsymbol{y}) = <\Phi(\boldsymbol{x}), \Phi(\boldsymbol{y})> = \Phi(\boldsymbol{x})^T\Phi(\boldsymbol{y}), \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p \text{ : inner product}$$

$$\Rightarrow \Phi(\boldsymbol{x}) = \gamma(x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2c}\,x_2, c)^T \text{ and similarly for } \Phi(\boldsymbol{y})$$

A.2 SVM(support vector machine)

Example 1. Riding Mower Data : [Data 6.3.1]

```
ridingmower<-read.table("ridingmower2.txt", header=T)
attach(ridingmower)
ridingmower

## choice of gamma and cost parameters in linear kernel
tune.out=tune.svm(Pop~., data=ridingmower, kernel="linear",
gamma = 2^(-1:1), cost=2^(2:4))
summary(tune.out)

## SVM fit in linear kernel
svmfit=svm(Pop~., data=ridingmower, kernel="linear",
    gamma=0.5, cost=4, scale=F)
summary(svmfit)

## Performance in linear kernel
svmfit$index
ypred=predict(svmfit, ridingmower)
table(truth=ridingmower$Pop, predict=ypred)
mean(Pop==ypred)
```

library(e1071) for SVM

```
## choice of gamma parameter in RBF kernel
tune.out=tune.svm(Pop~., data=ridingmower, kernel="radial",
gamma = 2^(-1:1))
summary(tune.out)

## SVM fit in RBF kernel
svmfit=svm(Pop~., data=ridingmower, kernel="radial",
    gamma=0.5, scale=F)
summary(svmfit)

## Performance in RBF kernel
ypred=predict(svmfit, ridingmower)
table(truth=ridingmower$Pop, predict=ypred)
mean(Pop==ypred)
```

A.2 SVM(support vector machine)

Result 1.

linear kernel	RBF kernel
> summary(tune.out)	
- best parameters: gamma cost 0.5 4 - best performance: 0.1666667 - Detailed performance results: gamma cost error dispersion 1 0.5 4 0.1666667 0.2222222 2 1.0 4 0.1666667 0.2222222 3 2.0 4 0.1666667 0.2222222 4 0.5 8 0.2166667 0.2363561 5 1.0 8 0.2166667 0.2363561 6 2.0 8 0.2166667 0.2363561 7 0.5 16 0.2166667 0.2363561 8 1.0 16 0.2166667 0.2363561 9 2.0 16 0.2166667 0.2363561	- best parameters: gamma 0.5 - best performance: 0.2 - Detailed performance results: gamma error dispersion 1 0.5 0.20 0.269888 2 1.0 0.25 0.274986 3 2.0 0.30 0.269888
> table(truth=iris$Species, predict=ypred)	
predict truth 미소유 소유 미소유 9 3 소유 2 10	predict truth 미소유 소유 미소유 12 0 소유 0 12
> mean(Pop==ypred)	
[1] 0.7916667	[1] 1

A.2 SVM(support vector machine)

Example 2. Iris Flowers Data : [Data 1.3.4]

```
install.packages("e1071")
library(e1071)
data(iris)
attach(iris)

## choice of gamma and cost parameters
tune.out=tune.svm(Species~., data=iris, kernel="linear",
gamma = 2^(-1:1), cost=2^(2:4))
summary(tune.out)

## SVM fit
svmfit=svm(Species~., data=iris, kernel="linear",
    gamma=0.5, cost=8, scale=F)
summary(svmfit)

## Performance
svmfit$index
ypred=predict(svmfit, iris)
table(truth=iris$Species, predict=ypred)
mean(Species==ypred)
```

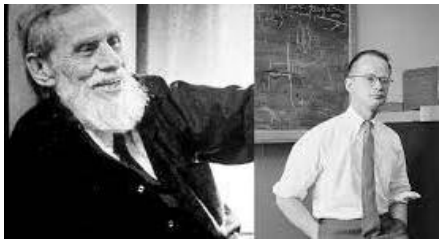
A.2 SVM(support vector machine)

Result 2.

```
> summary(tune.out)

Parameter tuning of 'svm':
- sampling method: 10-fold cross validation
- best parameters:
 gamma cost
   0.5    8

- best performance: 0.03333333
- Detailed performance results:
   gamma cost       error dispersion
1    0.5    4 0.04000000 0.04661373
2    1.0    4 0.04000000 0.04661373
3    2.0    4 0.04000000 0.04661373
4    0.5    8 0.03333333 0.04714045
5    1.0    8 0.03333333 0.04714045
6    2.0    8 0.03333333 0.04714045
7    0.5   16 0.04666667 0.05488484
8    1.0   16 0.04666667 0.05488484
9    2.0   16 0.04666667 0.05488484

> table(truth=iris$Species, predict=ypred)
            predict
truth        setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         47         3
  virginica       0          0        50

> mean(Species==ypred)
[1] 0.98
```

A.3 ANN (artificial neural network)

Warren Sturgis McCulloch (1898 – 1969) : neurophysiologist and logician Walter Pitts (1923 – 1969) : mathematician *A Logical Calculus of Ideas Immanent in Nervous Activity* (1943). They were the first to create a mathematical model of a neuron, inspired by the concept of a biological neuron.



Computing Machine Model

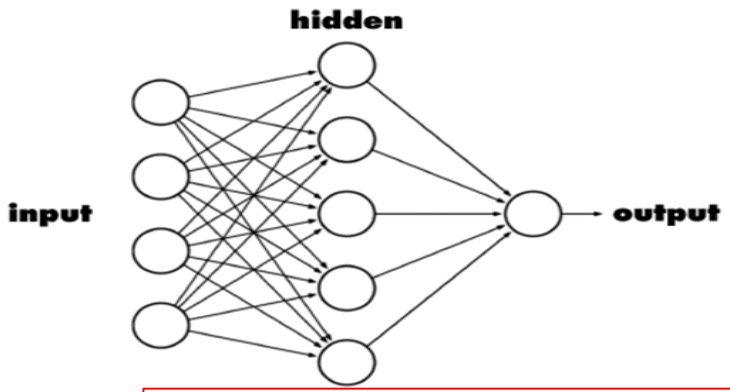
Definition

Computational model that mimics the pattern of the human mind or constructed a simplified abstraction of the process of neuron activity in the human brain.

- The development of a neural network is inspired by human brain activities.
- The neural network is a network made up of artificial neurons (or nodes).

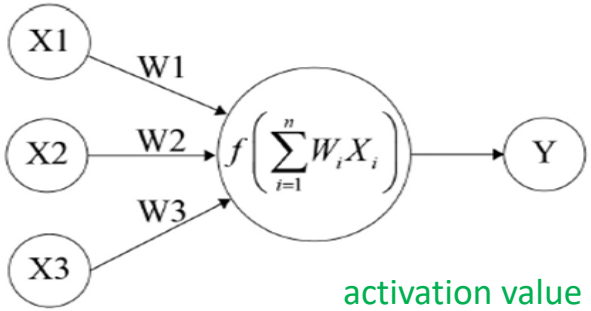
Sstructure of ANN

Three types of neurons



Connection strength between neurons

Single-layer perception with 3 inputs



activation value

Inputs: x_1, x_2, x_3 → $u = \sum w_j x_j$

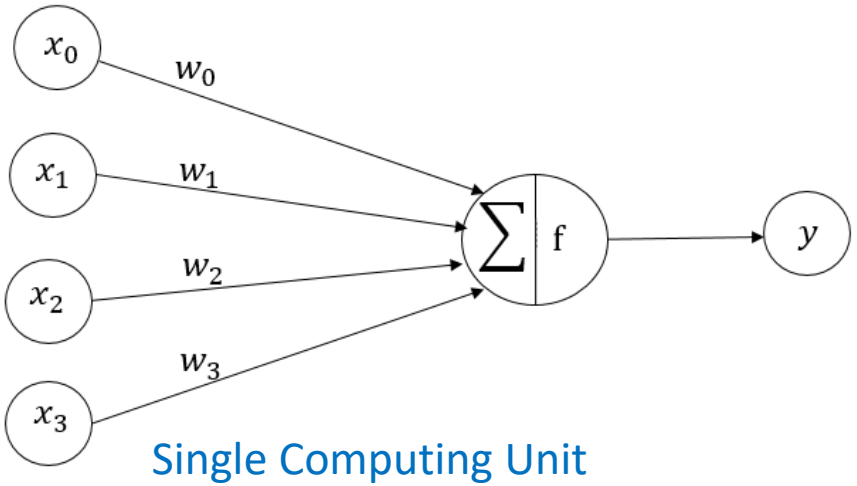
Weights: w_1, w_2, w_3

Output: $y = f(u) = \begin{cases} +1 & \text{if } \boldsymbol{x} \in C_1 \\ -1 & \text{if } \boldsymbol{x} \in C_2 \end{cases}$

Structure of Artificial Neural Network

Activation values and functions of Single-layer and Multi-layer

(1) Single-layer perception with 3 inputs and 1 output



Single Computing Unit

$$u = w_0 + \sum_{j=1}^{3} w_j x_j, \quad f(u) = \begin{cases} +1 & \text{if } \boldsymbol{x} \in C_1 \\ -1 & \text{if } \boldsymbol{x} \in C_2 \end{cases}$$

(2) Multi-layer perception with p inputs and 1 output

$$u_k = w_{0k} + \sum_{j=1}^{p} w_{jk} x_j \,, \quad f(u_k) = \begin{cases} +1 & \text{if } \boldsymbol{x} \in C_1 \\ -1 & \text{if } \boldsymbol{x} \in C_2 \end{cases}, \quad k = 1, \ldots, s$$

$$= w_{0k} + \boldsymbol{x}^T \boldsymbol{w}_k \text{ with } \boldsymbol{x} = (x_1, \ldots, x_p)^T \text{ and } \boldsymbol{w}_k = (w_{1k}, \ldots, w_{pk})^T$$

A.3 ANN(artificial neural network)

Applications : classification, clustering, and prediction

Process:

1) Input neurons receive the input information;
the higher the input values,
the greater the activation values.

the activation value is passed through the network in regard to weights and transfer functions(activation functions) in the graph.

A.3 ANN(artificial neural network)

2) The hidden neurons (or output neurons) then sum up the activation values and modify the summed values with the transfer function.

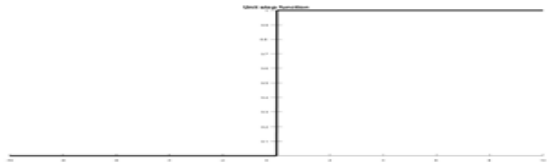
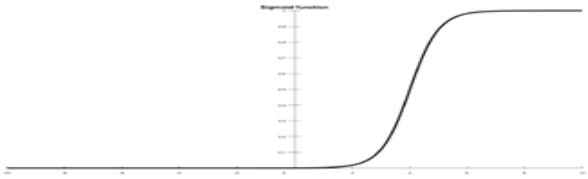
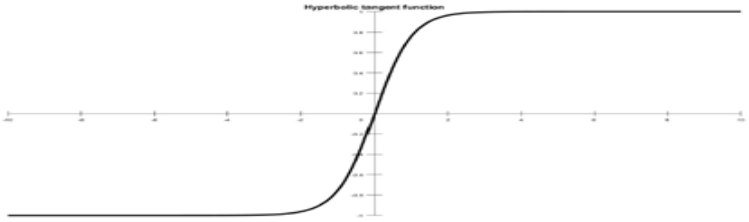
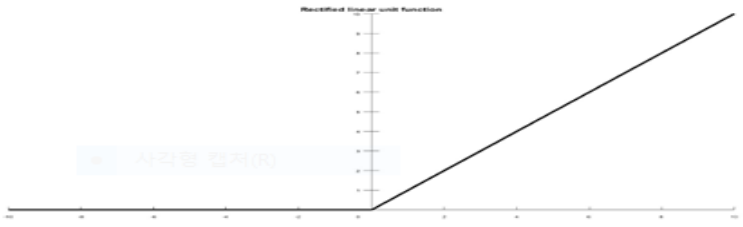
➡ The activation value then flows through hidden neurons and stops when it reaches the output nodes.

As a result, one can use the output value from the output neurons to classify the data.

A.3 ANN(artificial neural network)

Advantages of a neural network
1. it can detect a nonlinear relationship between the dependent and independent variable. 2. one can efficiently train large datasets using the parallel architecture. 3. it is a non-parametric model so that one can eliminate errors in the estimation of parameters.
Disadvantages
it often converges to the local minimum rather than the global minimum. Also, it might over-fit when the training process goes on for too long.

A.3 ANN(artificial neural network)

Activation function	$f(x)$	Shape
linear	x	
unit step	$I(x \geq 0)$	
sigmoid	$1/(1+e^{-x})$	
hyperbolic tangent	$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	
rectified liner unit	$\begin{cases} 1, & x < 0 \\ x, & x \geq 0 \end{cases}$	

Note : (1) sigmod function and $\tanh(x)$ are more robust for outliers.
(2) $\tanh(x)$ converges faster than sigmoid function in ANN.

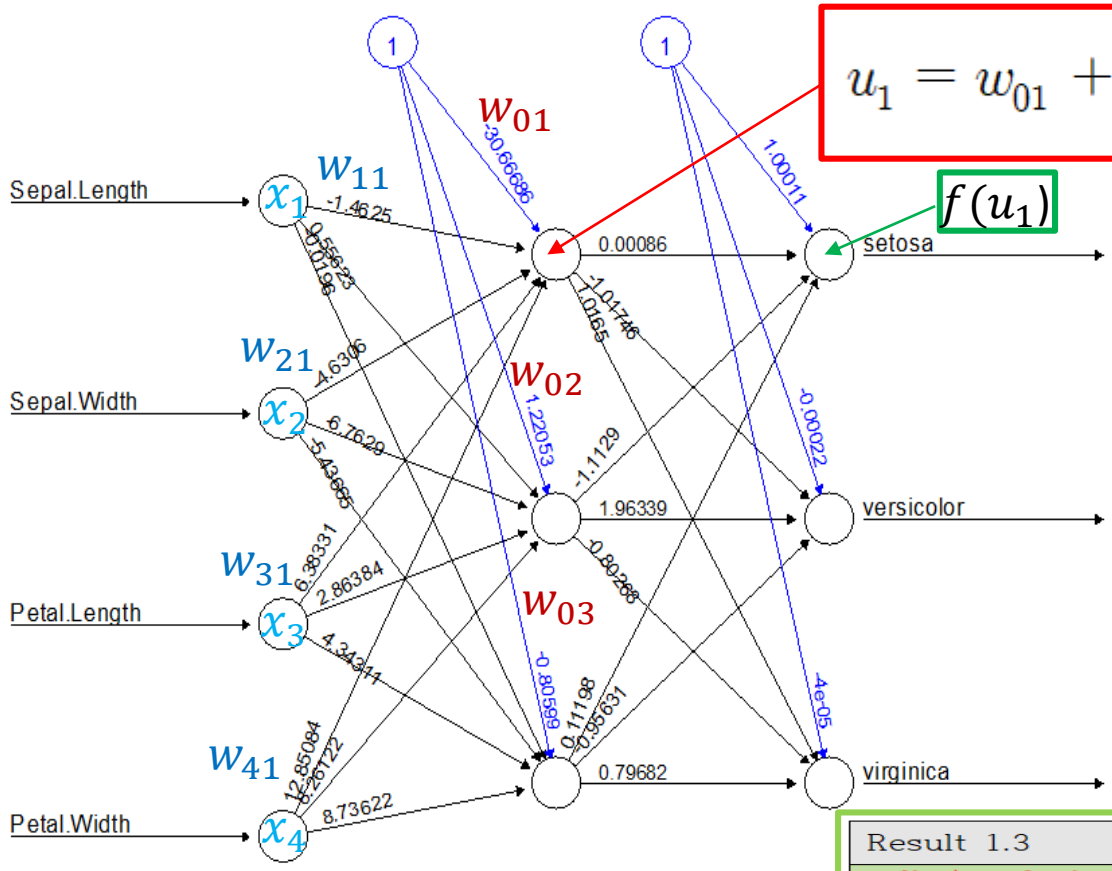
A.3 ANN(artificial neural network)

Example 1. ANN for Iris Flowers Data using library(neuralnet)

```
## Neural Network for Iris  Flowers Data Using neuralnet
install.packages("neuralnet")
library(neuralnet)

data(iris)
attach(iris)

## Traning and Testing data
ind=sample(2, nrows(iris), replace=T, prob=c(0.60, 0.40))
train=iris[ind==1,]
test=iris[ind==2,]

## ANN model using neuralnet
model_3= neuralnet(Species~., data=train, hidden=3, act.fct = "logistic"
plot(model_3);model_3

## Performance & Confusion Matrix
predict_3=compute(model_3, test)
idx <- apply(predict_3$net.result, 1, which.max)
predicted_3 <- c('setosa', 'versicolor', 'virginica')[idx]
confusion=table(test$Species, predicted_3)
EAER=(1-sum(diag(prop.table(confusion))))*100
list(confusion, EAER)
```

A.3 ANN(artificial neural network)

Result 1.1 Three-layer : Multilayer NN



Result 1.3

```
> list(confusion, EAER)
[[1]]
            predicted_3
             setosa versicolor virginica
  setosa         20          0         0
  versicolor      0         17         3
  virginica       0          0        20

[[2]]
[1] 5
```

A.3 ANN(artificial neural network)

Result 1.2

```
> model_3

$weights
$weights[[1]]
$weights[[1]][[1]]
          [,1]        [,2]       [,3]
[1,] -30.666855   1.2205297 -0.80598664
[2,]  -1.462497   0.5562277 -0.01960306
[3,]  -4.630602  -6.7628997 -5.43664654
[4,]   6.383312   2.8638401  4.34310663
[5,]  12.850844   8.2612229  8.73621594

$result.matrix
                                    [,1]
error                       1.882300e+00
reached.threshold           9.749576e-03
steps                       9.263000e+03
Intercept.to.1layhid1      -3.066686e+01
Sepal.Length.to.1layhid1   -1.462497e+00
Sepal.Width.to.1layhid1    -4.630602e+00
Petal.Length.to.1layhid1    6.383312e+00
Petal.Width.to.1layhid1     1.285084e+01
Intercept.to.1layhid2       1.220530e+00
Sepal.Length.to.1layhid2    5.562277e-01
Sepal.Width.to.1layhid2    -6.762900e+00
Petal.Length.to.1layhid2    2.863840e+00
Petal.Width.to.1layhid2     8.261223e+00
Intercept.to.1layhid3      -8.059866e-01
Sepal.Length.to.1layhid3   -1.960306e-02
Sepal.Width.to.1layhid3    -5.436647e+00
Petal.Length.to.1layhid3    4.343107e+00
Petal.Width.to.1layhid3     8.736216e+00
Intercept.to.setosa         1.000109e+00
1layhid1.to.setosa          8.611765e-04
1layhid2.to.setosa         -1.112905e+00
1layhid3.to.setosa          1.119811e-01
Intercept.to.versicolor    -2.193551e-04
1layhid1.to.versicolor     -1.017464e+00
1layhid2.to.versicolor      1.963388e+00
1layhid3.to.versicolor     -9.563123e-01
Intercept.to.virginica     -4.338606e-05
1layhid1.to.virginica       1.016501e+00
1layhid2.to.virginica      -8.026799e-01
1layhid3.to.virginica       7.968166e-01
```

w_{01} w_{02} w_{03} w_{j1} w_{j2} w_{j3}

$|w_k - w_{k+1}| \leq 10^{-2}$: *stoping rule*

Interpretations

The estimated weight ranges from −30.667 to 12.851: the intercepts of the first hidden layer are −30.0667, 1.221 and −0.806, and the two weights leading to the first hidden neuron are estimated as −1.462(Sepal.Length), −4.631(Sepal.Width), 6.3833(Petal.Length), and 12.851(Petal.Width).

Training process needed 9263 steps until all the absolute partial derivatives of the error function were lower than 0.01 (9.749576e-03 specified in the threshold). The error refers to the likelihood of calculating Akaike Information Criterion (AIC).

A.3 ANN(artificial neural network)

Example 2. Iris Flowers Data using library(nnet)

```
## Neural Network for Iris  Flowers Data Using nnet
library(nnet)
data(iris)
attach(iris)

## Traning and Testing data
ind=sample(2, length(iris), replace=T, prob=c(0.60, 0.40))
train=iris[ind==1,]
test=iris[ind==2,]

## ANN model using  nnet
model_3= nnet(Species~., data=train, size=3)
model_3
## Performance & Confusion Matrix
predict=predict(model_3, test, type="class")
confusion=table(test$Species, predict)
EAER=(1-sum(diag(prop.table(confusion))))*100
list(confusion, EAER)
```

# of hidden layers

Result 2.1

```
> model_3
a 4-3-3 network with 27 weights
inputs: Sepal.Length Sepal.Width Petal.Length Petal.Width
output(s): Species
options were - softmax modelling

> list(confusion, EAER)
[[1]]
            predict
             setosa versicolor virginica
  setosa         20          0         0
  versicolor      0         17         3
  virginica       0          0        20

[[2]]
[1] 5
```

Comparison of Traditional DA, Logistic DA, SVM and ANN

[Data 6.6.1] Riding Mower Data

Misclassification Error Rate	LDA		SVM		Logistic DA
	RSM	CVM	Linear Kernel	RBF Kernel	RSM
	12.50%	20.83%	20.83%	0.00%	16.67%

Note: Choi(2025, Chapter 9)

: Statistics& Applications for Data Science with R & Python.

Tree	Random Forest	ANN
25.00%	0.00%	16.67%

[Data 6.6.2] Iris Flowers Data

Misclassification Error Rate	QDA		SVM		ANN
	RSM	CVM	Linear Kernel	RBF Kernel	CVM
	2%	2.67%	2%	-	5%

A.4 R for ML : Practice Time

R-Code:

SVM	ANN
library(e1071) svm() tune.svm()	library(neuralnet)/library(nnet) neuralnet()/nnet()
gamma = 2^(-1:1), cost=2^(2:4) kernel=" linear/polynomial/radial"	hidden=3, act.fct = "logistic/tanh“/size=3 ## Training and Testing data ind=sample(2, nrows(iris), replace=T, prob=c(0.60, 0.40)) train=iris[ind==1,] test=iris[ind==2,] length(iris)