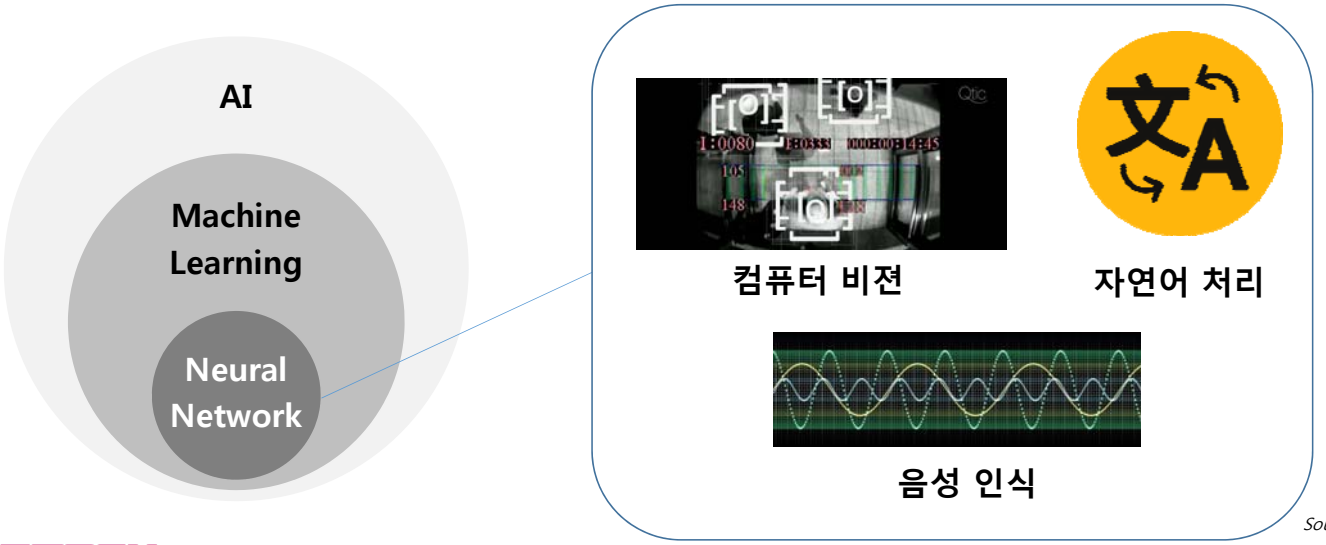
	단위별 학습내용 (Week16)
wk16-1	Neural Network
wk16-2	Convolutional Neural Network
wk16-3	웹문서 텍스트마이닝

Wk16-1 : Neural Network

1. Concepts

- 인공신경망은 기계학습(Machine Learning)의 통계적 학습 알고리즘 중 하나
- 컴퓨터 비젼, 자연어 처리, 음성 인식 등의 영역에서 활발하게 사용됨

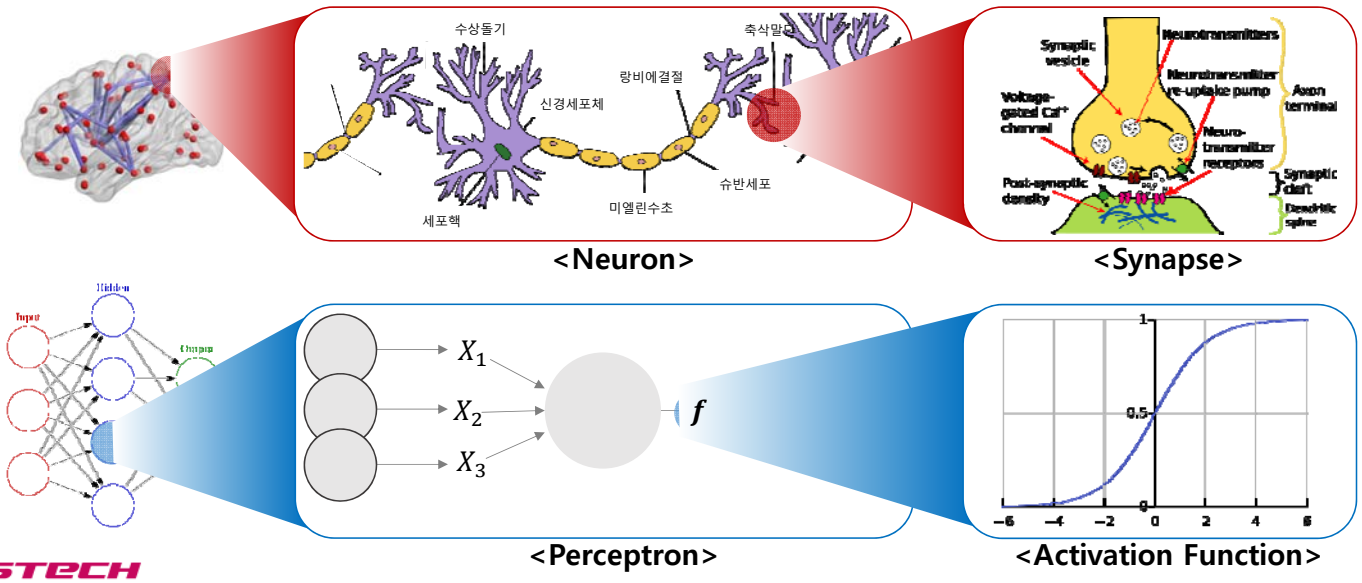


Source: Wikipedia

1. Concepts

- 신경망 모델은 (Neural Network)은 Perceptron을 한 단위로 하는 네트워크를 구축하여, 인간의 신경세포(Neuron)과 유사한 기능을 하도록 제안되었음

Source: Wikipedia

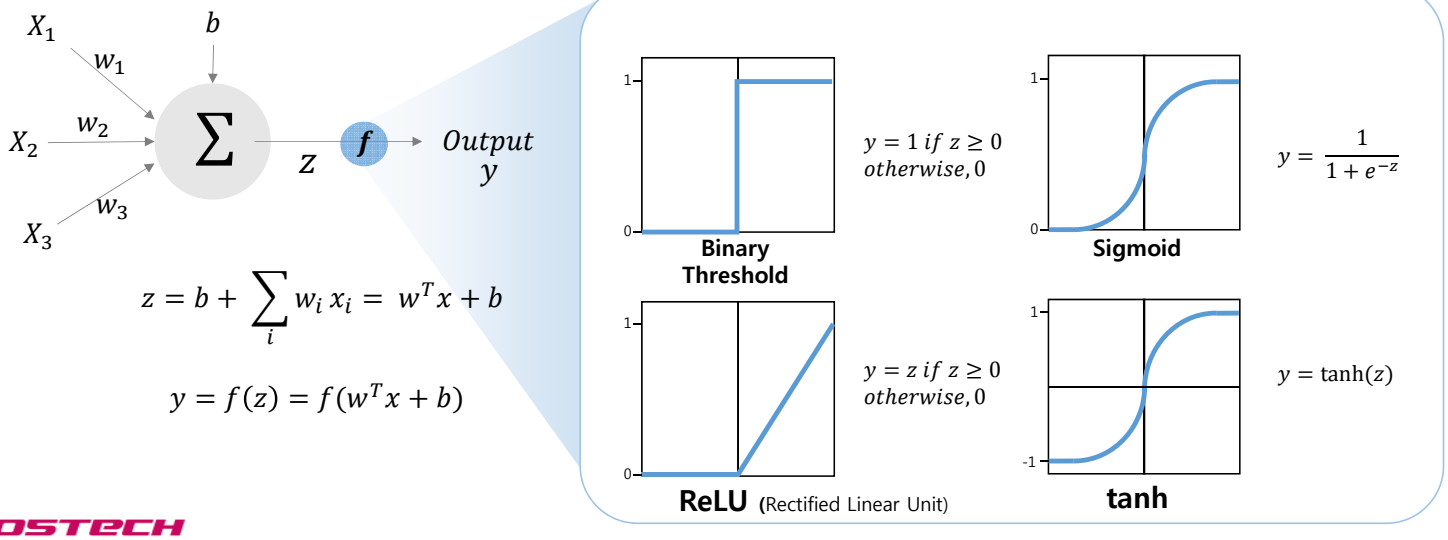


<Neuron> <Synapse>

<Perceptron> <Activation Function>

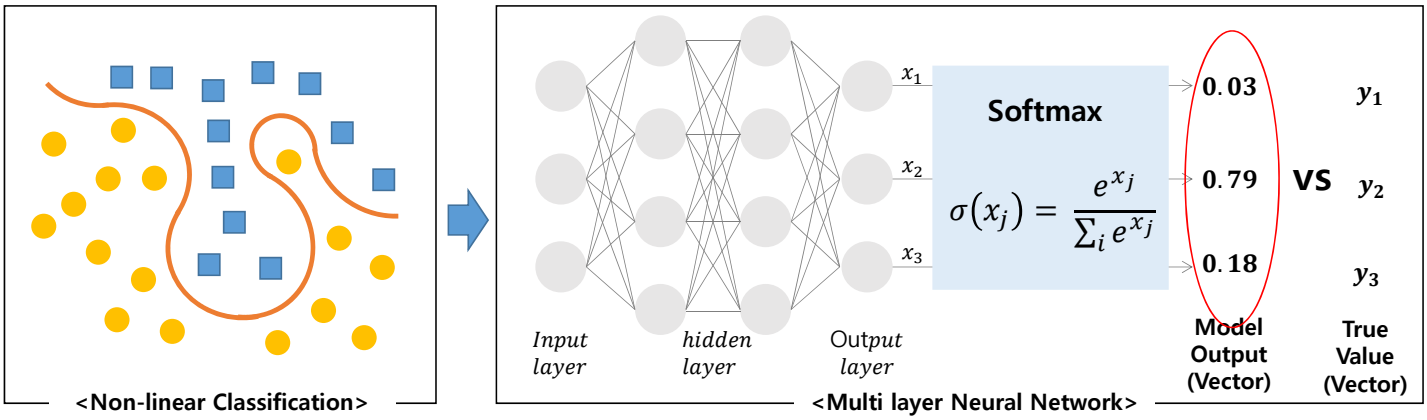
2. Perceptron – Single Layer

- 하나의 Perceptron은 단순하게 다수의 입력과 가중치의 선형 결합을 계산하는 역할을 수행
- Activation 함수에 따라 선형결합으로 생성되는 출력의 값이 결정됨



3. Multi-layer perceptron

- Perceptron으로 구성된 Single Layer들이 Multi-layer를 만듬
  - Input layer과 Output layer 사이에는 Hidden layer가 존재하여 Non-linear transformation을 수행
- Output layer에서 Softmax 함수를 통해 가장 큰 값을 손쉽게 알 수 있음.
  - Exponential 함수로 인해 항상 양수의 결과치가 도출되고 이를 통해 확률값을 도출함.



4. Neural Network 수행

- 신경망 모델 생성을 위한 패키지: mxnet

```
# lec16_1_nn.r
# Neural network

# require mxnet
install.packages("https://github.com/jeremiedb/mxnet_winbin/raw/ma
library(mxnet)

# If you have Error message "no package called XML or DiagrmmeR",
install.packages("XML")
install.packages("DiagrammeR")
#library(XML)
#library(DiagrammeR)
```

mxnet 라이브러리 설정시 아래와 같은 Error메세지가 나오면, DiagrmmeR 패키지 설치. XML도 설치필요할 수 있음

```
> library(mxnet)
Error: package or namespace load failed for 'mxnet' 
], c(lib.loc, .libPaths()), versionCheck = vI[[j]]):
 there is no package called 'DiagrammeR'
In addition: Warning message:
package 'mxnet' was built under R version 3.4.4
```


4. Neural Network 수행

- 신경망 모델 생성을 위한 패키지: mxnet
- iris 데이터를 이용

```
# lec16_1_nn.r
# Neural network

# require mxnet
install.packages("https://github.com/jeremiedb/mxnet_win
library(mxnet)

#devtools::install_github("rich-iannone/DiagrammeR")

#install.packages('devtools')
#library(devtools)

#install.packages("DiagrammeR")
#library(DiagrammeR)

# set working directory
setwd("D:/tempstore/moocr/wk16")
```

Mxnet 설치

```
# read csv file
iris<-read.csv("iris.csv")
attach(iris)
# change to numeric from character variable : Species
iris[,5] = as.numeric(iris[,5])-1
table(Species)
# check the data
head(iris, n=10)
```

```
> head(iris, n=10)
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1           5.1         3.5          1.4         0.2       0
2           4.9         3.0          1.4         0.2       0
3           4.7         3.2          1.3         0.2       0
4           4.6         3.1          1.5         0.2       0
```

Species을 Label로 활용,
각 종별로 0, 1, 2의 숫자로 변화

4. Neural Network 수행

- 학습데이터와 검증데이터

```
# split train & test dataset
# training (n=100)/ test data(n=50)
set.seed(1000)
N<-nrow(iris)
tr.idx<-sample(1:N, size=N*2/3, replace=FALSE)

# split train data and test data
train<-data.matrix(iris[tr.idx,])
test<-data.matrix(iris[-tr.idx,])

# feature & Labels
train_feature<-train[,-5]
trainLabels<-train[,5]
test_feature<-test[,-5]
testLabels <-test[,5]
```

학습데이터: 2/3 (100개)
검증데이터: 1/3 (50개)

Label은 5번째 열에 위치...
각 객체별 Feature와 Label로 분리

```
> table(trainLabels)
trainLabels
 0  1  2
31 31 38
```

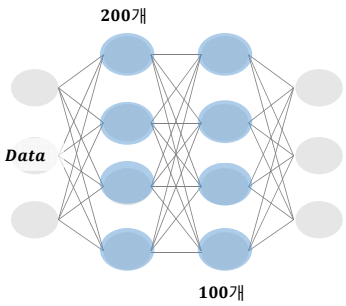

4. Neural Network 수행

- Hidden Layer 구성

```
# Build nn model
# first layers
my_input = mx.symbol.Variable('data')
fc1 = mx.symbol.FullyConnected(data=my_input, num.hidden = 200, name='fc1')
relu1 = mx.symbol.Activation(data=fc1, act.type='relu', name='relu1')
```

200개의 뉴런 형성
4개의 input 뉴런과 모두 연결되도록
(Sepal.Length, Sepal.Width, Petal.Length, Petal.Width)

Sigmoid Function 대신 ReLU Function 사용




```
# second layers
fc2 = mx.symbol.FullyConnected(data=relu1, num.hidden = 100, name='fc2')
relu2 = mx.symbol.Activation(data=fc2, act.type='relu', name='relu2')
```

100개의 뉴런 형성
첫번째 Layer의 200개의 뉴런과 모두 연결

4. Neural Network 수행

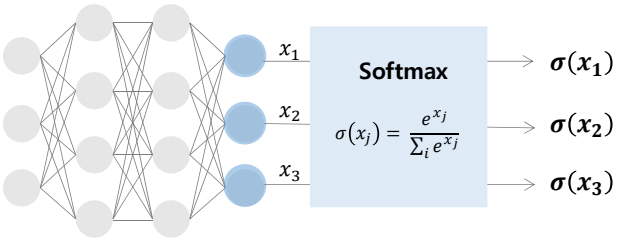
- **Output Layer 구성**

```
# third layers
fc3 = mx.symbol.FullyConnected(data=relu2, num.hidden = 3, name='fc3')

# softmax
softmax = mx.symbol.SoftmaxOutput(data=fc3, name='sm')
```

3개로 분류(0,1,2)해야하므로
3개의 Output 뉴런을 생성

Softmax의 결과를 통해 가장 큰 값을 선택



5. Neural Network 모델 학습

- **모델 학습**

앞에서 만든 Layer를 이용해서 모델을 형성 및 학습

```
# training
mx.set.seed(123)
device <- mx.cpu()
model <- mx.model.FeedForward.create(softmax,
                                     optimizer = "sgd",
                                     array.batch.size=10,
                                     num.round = 300, learning.rate=0.1,
                                     X=train_feature, y=trainLabels, ctx=device,
                                     eval.metric = mx.metric.accuracy,
                                     array.layout = "rowmajor",
                                     epoch.end.callback=mx.callback.log.train.metric(100))
graph.viz(model$symbol)
```

Stochastic Gradient Descent
batch size = 10 (총 10개 그룹)

Iteration(epoch): 300
Learning Step: 0.1

```
Start training with 1 devices
[1] Train-accuracy=0.344444444444444
[2] Train-accuracy=0.38
```

```
[298] Train-accuracy=0.89
[299] Train-accuracy=0.96
[300] Train-accuracy=0.97
```

6. Neural Network 모형- 검증데이터 결과

- 모델 테스트

```
# testing
predict_probs <- predict(model, test_feature, array.layout = "rowmajor")
predicted_labels <- max.col(t(predict_probs)) - 1
table(testLabels, predicted_labels)
sum(diag(table(testLabels, predicted_labels)))/length(predicted_labels)
```

```
> table(testLabels, predicted_labels)
          predicted_labels
testLabels  0  1  2
         0 19  0  0
         1  0 18  1
         2  0  0 12
> sum(diag(table(testLabels, predicted_labels)))/length(predicted_labels)
[1] 0.98
```