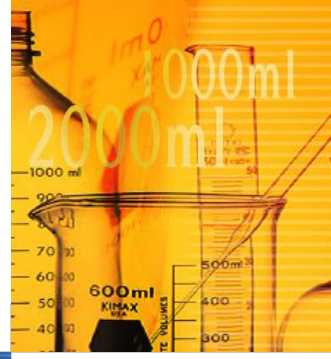
Machine learning



Chapter 6

Support Vector Machine Neural Network

Sejong Oh

Bio Information Technology Lab.

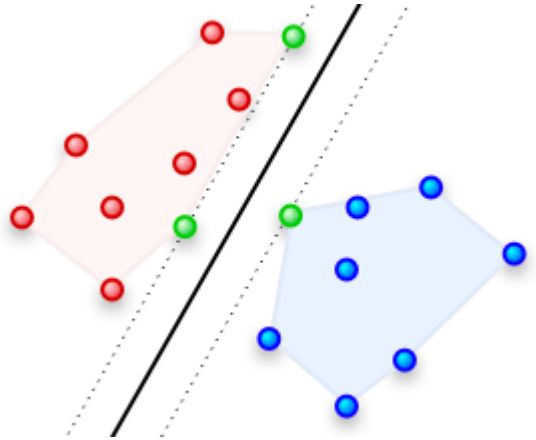
Content

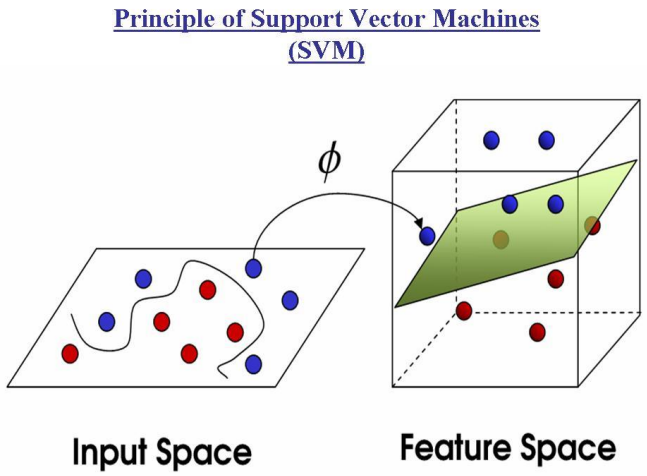
- Support Vector Machine (SVM)
- Neural Network (NN)

- SVM 과 NN 은 model 의 해석이 어렵기 때문에 black box model 이라고 불린다

Support Vector Machine

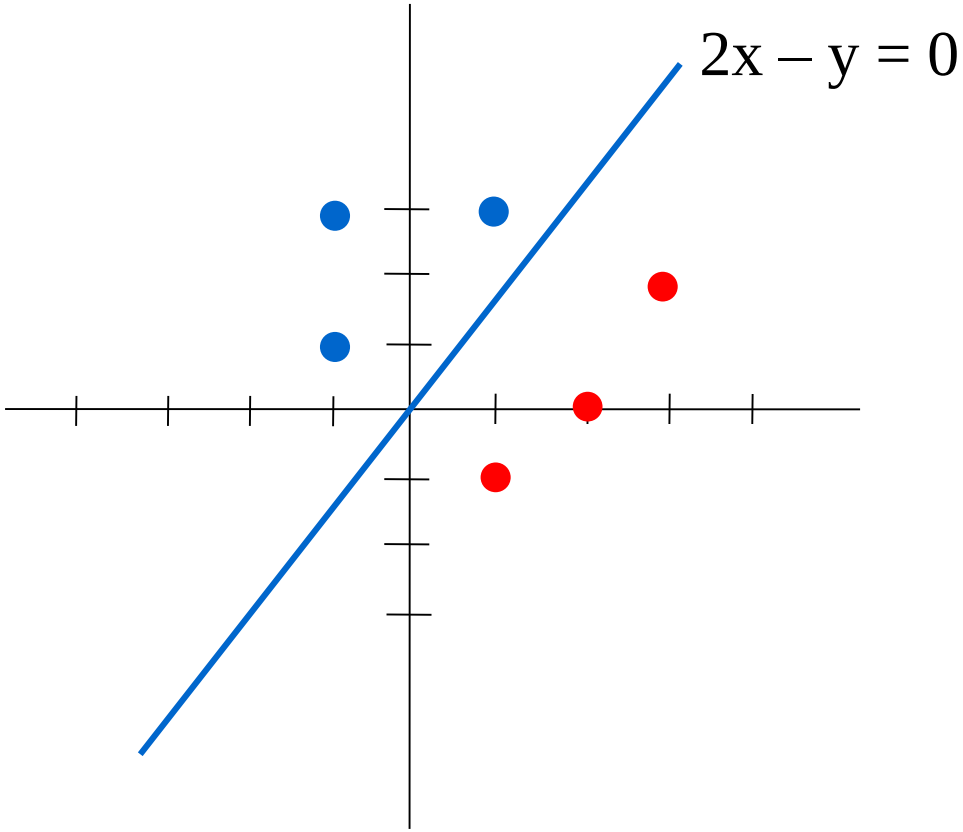
- Introduction
  - One of classification algorithm
  - Original SVM can be applied on two class problem
    - Extend to multi-class problem
  - SVM is basically linear discrimination method
  - It has two ideas
    - Finding maximum-margin hyperplane
    - Move input data onto higher dimension space (kernel method)





연두색 점 : support vector

Support Vector Machine



(-1,1) → -3
(-1,3) → -5
(1,3) → -1

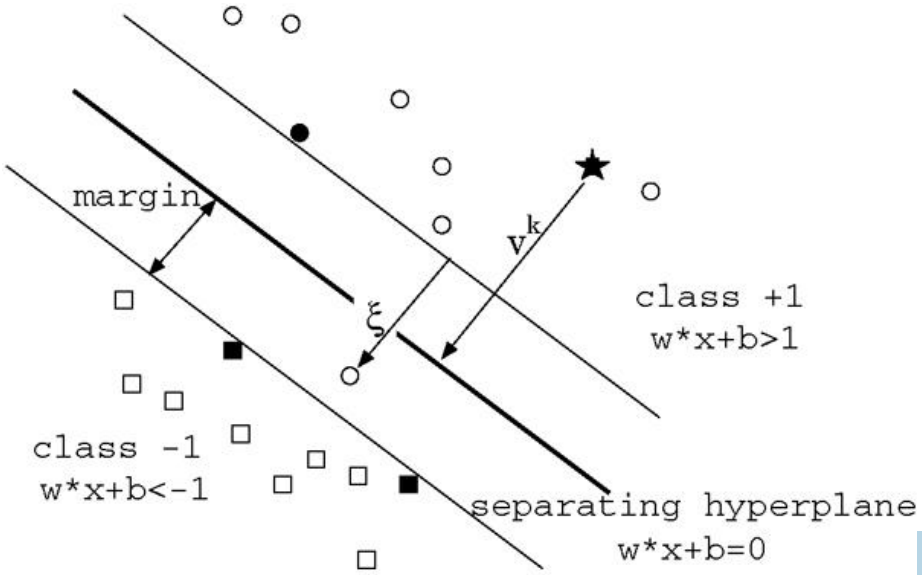
(-)

(1,-1) → 3
(2,0) → 4
(3,2) → 4

(+)

Support Vector Machine

- Support vectors
  - The hypothesis space is given by the functions f(x) = sng(*w*x + *b*), where *w* and *b* are parameters that are learned
  - Samples on the margin are the *support vectors*
  - Optimization problem: the norm of w and loss are minimized subject to right classification within the allowed error(s), ξ (soft margin)
  - Writing the classification rule in its unconstrained dual form reveals that classification is only a function of the support vectors



Support Vector Machine

- Major application area
  - 암이나 다른 유전적 질병을 인식하기 위한 microarray data의 분류
  - 주제별로 정리된 문서나 문서에 사용된 언어의 식별 같은 텍스트 분류
  - 엔진 연소문제, 보안상 결함, 지진과 같은 드물게 일어나는 주요 사건 감식

Support Vector Machine

- Advantage
  - 범주나 수치데이터 모두에 사용가능
  - 노이즈 데이터에 영향을 크게 받지 않고 overfitting 이 잘 일어나지 않음
  - 높은 정확도

- Disadvantage
  - 최적의 모델을 찾기 위해 커널과 기타 매개변수의 여러 조합을 테스트 해보아야 한다
  - 입력 데이터셋이 feature 수와 데이터 sample 수가 많으면 훈련시간이 많이 소요될수 있다
  - 모델의 해석이 불가능하지는 않지만 어렵다

Support Vector Machine

- SVM tool : LIBSVM (Java version)
  - LIBSVM -- A Library for Support Vector Machines (http://www.csie.ntu.edu.tw/~cjlin/libsvm/)
  - Dataset format

```
2 1:0.2 2:0.2 3:0.2 4:0.2
3 1:0.2 2:0.2 3:0.2 4:0.4
3 1:0.2 2:0.2 3:0.2 4:0.8
3 1:0.2 2:0.2 3:0.4 4:0.2
3 1:0.2 2:0.2 3:0.4 4:0.6
3 1:0.2 2:0.2 3:0.4 4:1.0
3 1:0.2 2:0.2 3:0.6 4:0.4
3 1:0.2 2:0.2 3:0.6 4:0.8
3 1:0.2 2:0.2 3:0.8 4:0.2
3 1:0.2 2:0.2 3:0.8 4:0.6
3 1:0.2 2:0.
3 1:0.2 2:0.
3 1:0.2 2:0.
1 1:0.2 2:0.
```

```
-t kernel_type : set type of kernel function (default 2)
        0 -- linear: u'*v
        1 -- polynomial: (gamma*u'*v + coef0)^degree
        2 -- radial basis function: exp(-gamma*|u-v|^2)
        3 -- sigmoid: tanh(gamma*u'*v + coef0)
```

  - Training

```
java svm_train -t 0 yeast_train.dat
```

  - Testing

```
java svm_predict yeast_test.dat yeast_train.dat.model result_yeast.txt
```

SVM in R

- svm 함수 (in “e1071” package)

```
svm(tr,cl, scale=TRUE, kernel ="radial",
    degree = 3, gamma
```

  - tr : training data (matrix)
  - cl : class label of tr (factor)
  - scale : indicating the variables to be scaled
  - kernal : linear, polynomial, radial, sigmoid
  - degree: parameter needed for kernel of type polynomial
  - gamma : parameter needed for all kernels except linear (default: 1/(ncol(tr))

** Kernlab package 에서 제공하는 ksvm 함수도 있다

SVM in R

- Training

```
model = svm(tr, clTR)
```

  - tr : training data (matrix or vector)
  - clTR : class label of training data (factor type)

- Test

```
pred = predict(model, ts)      # ts : test data
```

  - pred : test data의 class 예측 값이 저장되어 있음

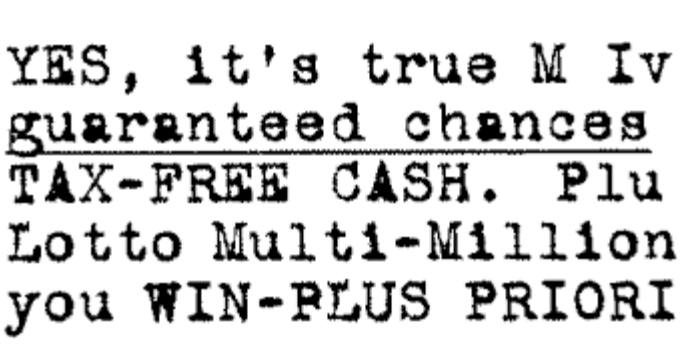
SVM in R

```
require(e1071)
data(iris)

x <- iris[,-5]      #data
cl <- iris[,5]      #class

# training with train data
model <- svm(x, cl)

# test with train data
pred <- predict(model, x)

# calculate accuracy
acc = mean(pred == cl)
# check accuracy
table(pred, cl)
```

Practice : OCR classification

- dataset : letterdata.csv

	A	B	C	D	E	F	G	H	I	J	
1	letter	xbox	ybox	width	height	onpix	xbar	ybar	x2bar	y2bar	xyl
2	T	2	8	3	5	1	8	13	0	6	
3	I	5	12	3	7	2	10	5	5	4	
4	D	4	11	6	8	6	10	6	2	6	
5	N	7	11	6	6	3	5	9	4	6	
6	G	2	1	3	1	1	8	6	6	6	
7	S	4	11	5	8	3	8	8	6	9	
8	B	4	2	5	4	4	8	7	6	6	
9	A	1	1	3	2	1	8	2	2	2	
10	J	2	2	4	4	2	10	6	2	6	
11	M	11	15	13	9	7	13	2	6	2	
12	X	3	9	5	7	4	8	7	3	8	



Practice : OCR classification

```
library(e1071)    # for svm
library(gmodels) # for CrossTable

# Read dataset
letters <- read.csv("letterdata.csv")
str(letters)

# Divide train/test data
letters_train <- letters[1:16000, ]
letters_test  <- letters[16001:20000, ]

# Training
model = svm(letters_train[,-1], letters_train[,1])

# test
result = predict(model,letters_test[,-1])
mean(result == letters_test[,1]) # print accuracy
```

Practice : OCR classification

```
# Analysis of prediction result
CrossTable(letters_test[,1], result,
    prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
    dnn = c('actual default', 'predicted default'))
```

	predicted default						
actual default	A	B	C	D	E	F	
A	151	0	0	1	0	0	
	0.038	0.000	0.000	0.000	0.000	0.000	0
B	0	129	0	1	0	0	
	0.000	0.032	0.000	0.000	0.000	0.000	0
C	0	0	133	0	2	0	
	0.000	0.000	0.033	0.000	0.000	0.000	0
D	0	2	0	162	0	0	
	0.000	0.000	0.000	0.040	0.000	0.000	0
E	0	0	3	0	139	0	
	0.000	0.000	0.001	0.000	0.035	0.000	0
F	0	1	0	0	2	148	
	0.000	0.000	0.000	0.000	0.000	0.037	0
G	0	0	1	2	0	0	

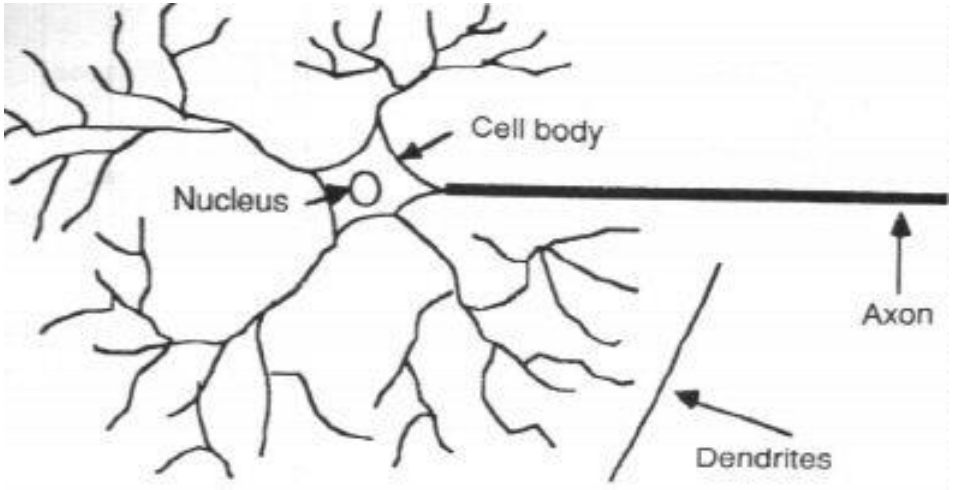
[과제 1] SVM

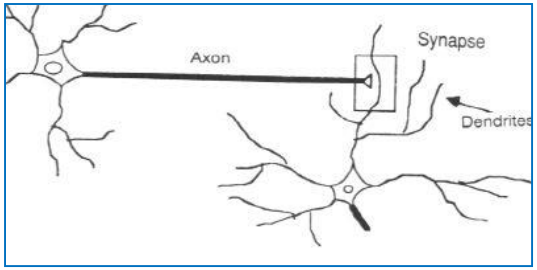
- liver.csv data 데이터를 이용하여 svm 을 테스트하시오.
  - 네개의 kernel 을 각각 적용했을 때 결과가 어떻게 나오는지 확인하시오

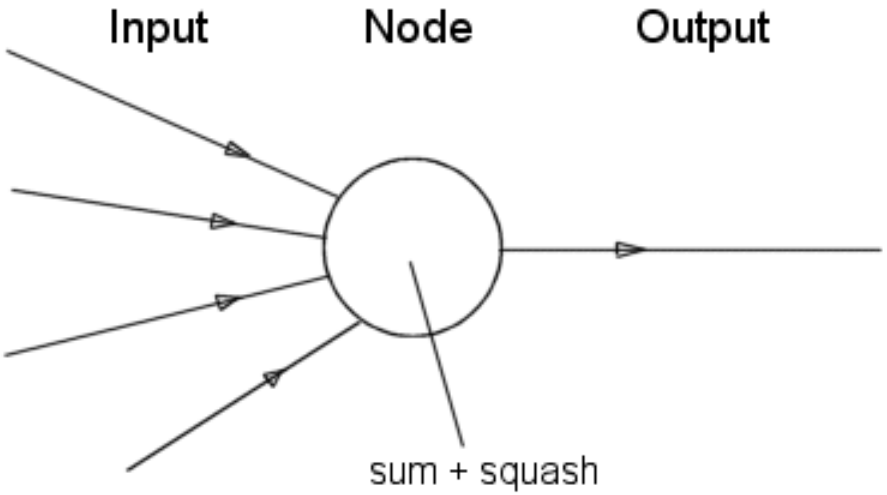
1	category	mcv	alkphos	sgpt	sgot	gammagt	drinks
2	0	85	64	59	32	23	0
3	0	86	54	33	16	54	0
4	0	91	78	34	24	36	0
5	0	87	70	12	28	10	0
6	0	98	55	13	17	17	0
7	0	91	72	155	68	82	0.5
8	0	85	54	47	33	22	0.5
9	0	79	39	14	19	9	0.5
10	0	85	85	25	26	30	0.5
11	0	89	63	24	20	38	0.5
12	0	84	92	68	37	44	0.5
13	0	89	68	26	39	42	0.5
14	0	89	101	18	25	13	0.5

Neural Network

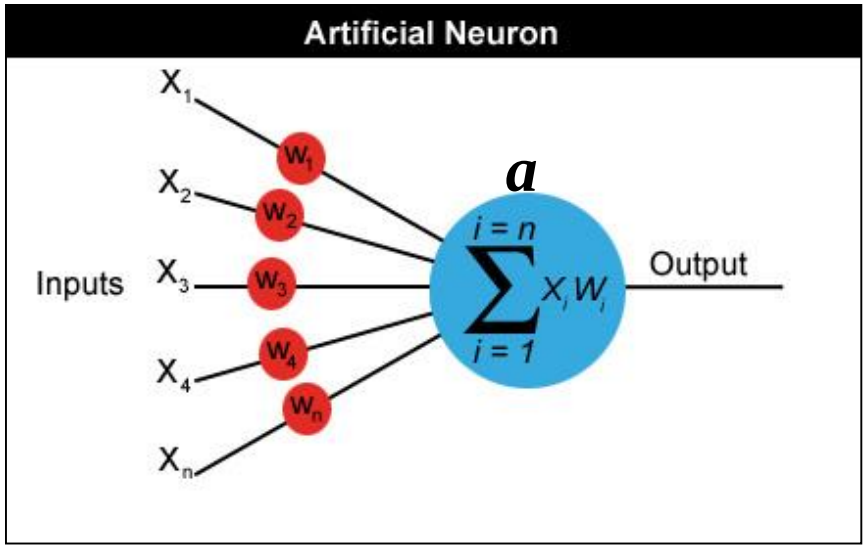
- Neurone vs. Node







Neural Network



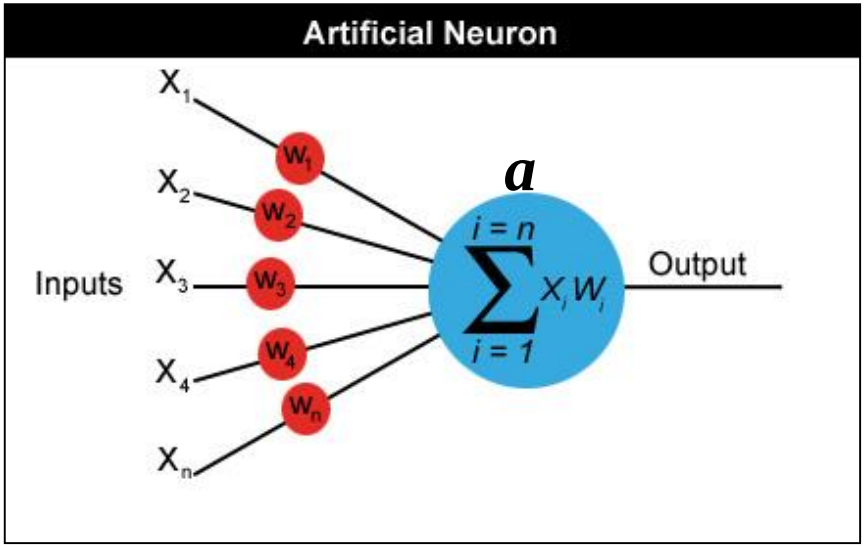
$$a = x_1w_1 + x_2w_2 + x_3w_3 \ldots + x_nw_n$$

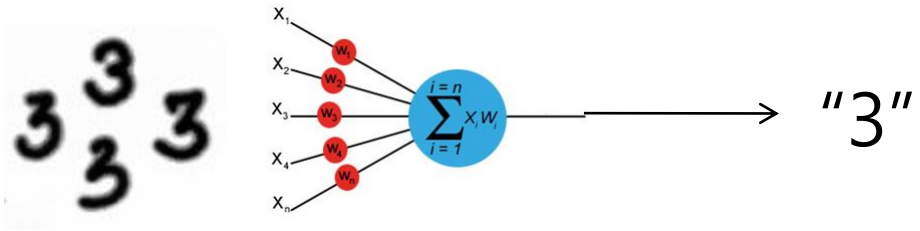
$$a = \sum_{i=0}^{i=n} w_i x_i$$

$$\text{output} = \begin{cases} 0 \text{ if } a < \text{threshold} \\ 1 \text{ if } a \geq \text{threshold} \end{cases}$$

Neural Network

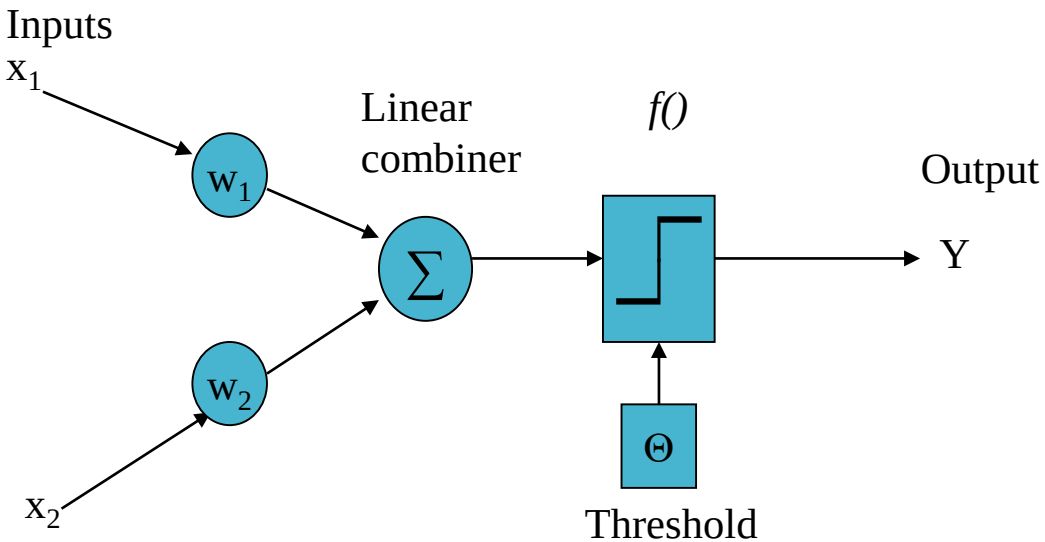
- Idea
  - If we adjust W_i value, we can get output which we want

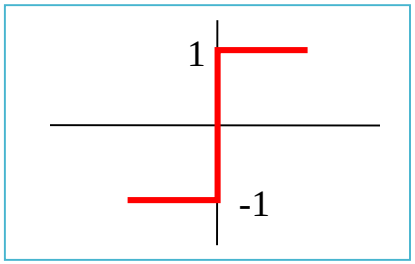




Neural Network

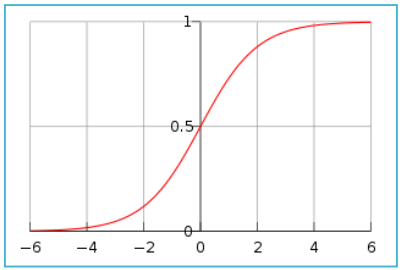
- Perceptron





$$P(t) = \begin{cases} 1 \text{ if } t > 0 \\ 0 \text{ if } t = 0 \\ -1 \text{ if } t < 0 \end{cases}$$

Hard limiter function

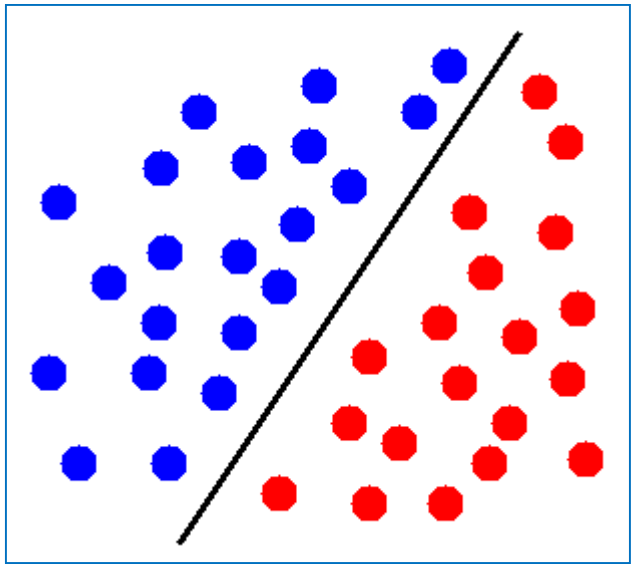


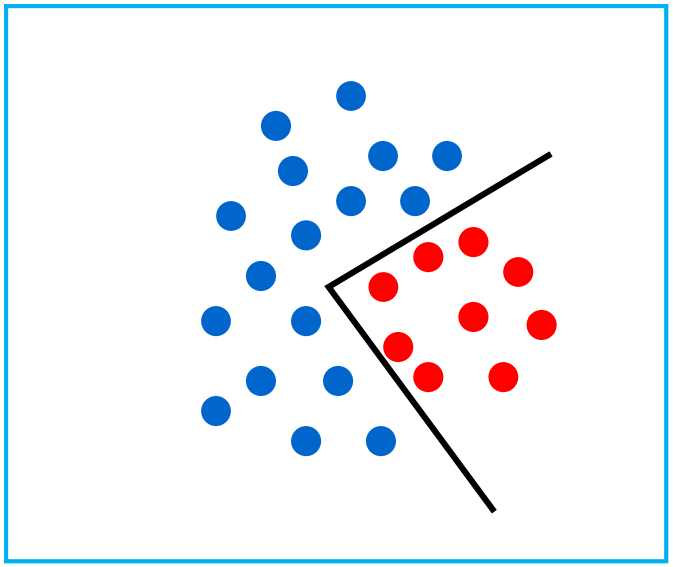
$$P(t) = \frac{1}{1 + e^{-t}}.$$

Sigmoid function

Neural Network

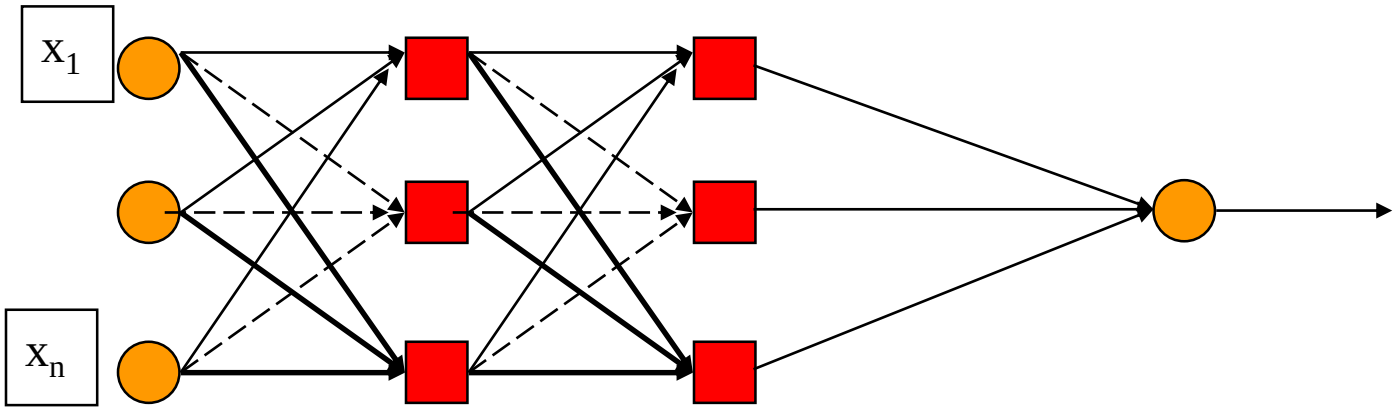
- Perceptron
  - Limitation : perceptron is only linear separable





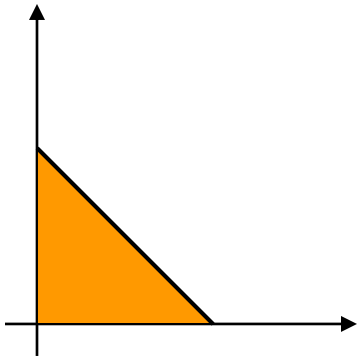
Neural Network

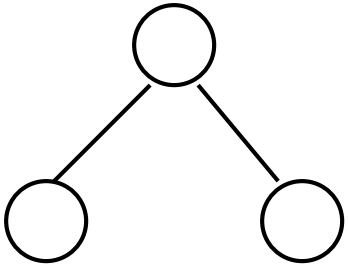
- Multilayer perceptron (back propagation model)
  - Overcomes simple perceptron's limitation



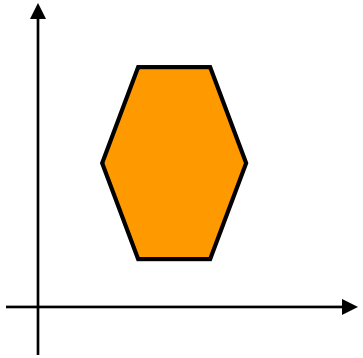
Neural Network

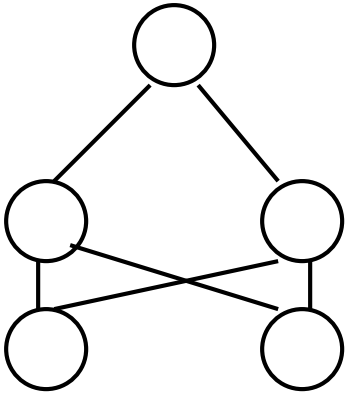
- What do each of the layers do?



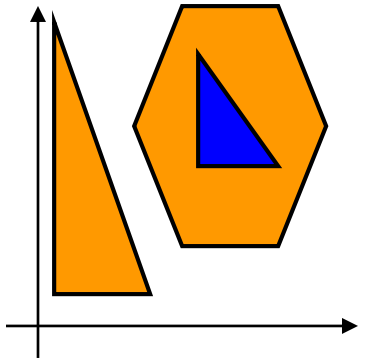


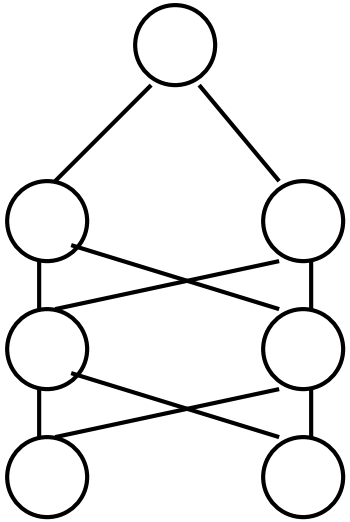
1st layer draws linear boundaries





2nd layer combines the boundaries



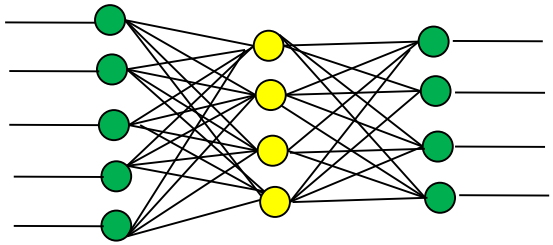


3rd layer can generate arbitrarily complex boundaries

Neural Network

- Using ANN for classification
  - (1) Design neural network for dataset

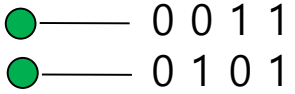
[dataset]
5 features
4 classes

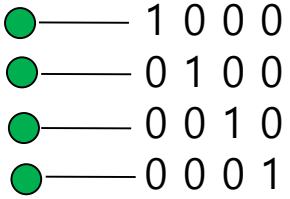


- No of input = No of feature
- No of output = No of class
- No of hidden node : ??

- Two types expression of output

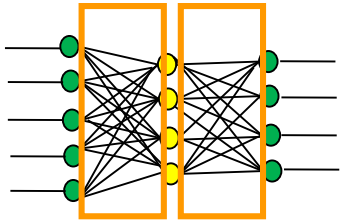
4 classes





Neural Network

- Using ANN for classification
  - (2) Prepare training/Testing data
  - (3) Training neural network
    - Result of training : weight value matrix
    - Requires long times
  - (4) Testing neural network
    - using weight value matrix



Neural Network

- Advantage
  - 분류나 수치 예측 문제에 적용 가능
  - 가장 정확한 모델링 방법중의 하나

- Disadvantage
  - Network 이 복잡해질 수 있고 훈련시간이 많이 걸린다
  - 훈련데이터에 overfitting 되거나 underfitting 되기 쉽다
  - 모델의 해석이 불가능하지는 않지만 어렵다

Neural network in R

- nnet 함수 (in “nnet” package)
- Training

```
model = nnet(tr, target, size, rang, decay, maxit)
```

  - tr : training data (matrix or vector)
  - target : output of NN (class 정보)
  - size : hidden layer 수
  - rang : Initial random weights on [-rang, rang].
  - decay : parameter for weight decay (default: 0)
  - maxit : maximum number of iterations (default: 100)
- Test

```
pred = predict(model, ts)      # ts : test data
```

Practice

```
require(nnet)
data(iris)

train = iris[,-5]                                    #data
test = train
targets = class.ind(iris[,5])                        #class

model = nnet(train, targets, size = 2,
    rang = 0.1, decay = 5e-4, maxit = 200)

# test with train data
pred <- predict(model, train)
```

```
> pred
             setosa   versicolor    virginica
  [1,] 0.989901184 0.0099054336 9.622787e-05
  [2,] 0.989423540 0.0105653736 9.621525e-05
  [3,] 0.989727770 0.0101436494 9.622322e-05
  [4,] 0.989156013 0.0109401681 9.620843e-05
  [5,] 0.989928503 0.0098680504 9.622860e-05
  [6,] 0.989830220 0.0100027224 9.622595e-05
  [7,] 0.989659249 0.0102382112 9.622140e-05
```

Practice

```
# calculate accuracy
predClass = max.col(pred)
testClass = max.col(targets)
acc = mean(predClass == testClass)

# check accuracy
table(predClass, testClass)
```

[과제 2] NN

- liver.csv data 를 이용하여 NN 을 테스트하시오
  - Training data 와 test 데이터를 반반씩 나누어서 테스트한다

1	category	mcv	alkphos	sgpt	sgot	gammagt	drinks
2	0	85	64	59	32	23	0
3	0	86	54	33	16	54	0
4	0	91	78	34	24	36	0
5	0	87	70	12	28	10	0
6	0	98	55	13	17	17	0
7	0	91	72	155	68	82	0.5
8	0	85	54	47	33	22	0.5
9	0	79	39	14	19	9	0.5
10	0	85	85	25	26	30	0.5
11	0	89	63	24	20	38	0.5
12	0	84	92	68	37	44	0.5
13	0	89	68	26	39	42	0.5
14	0	89	101	18	25	13	0.5