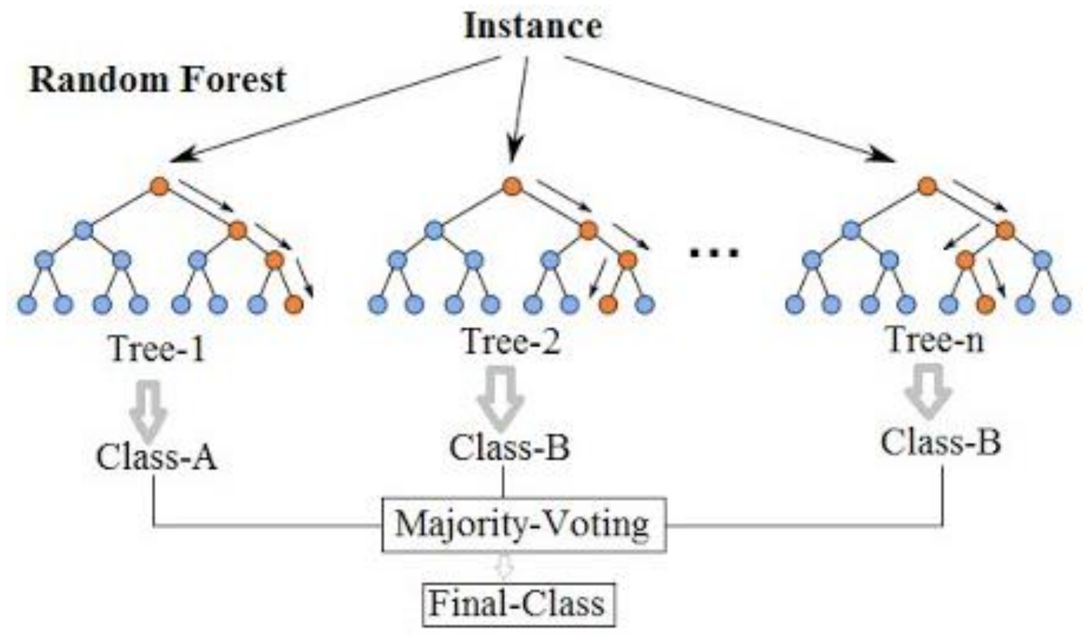
분류 및 예측 (3) : Random Forest

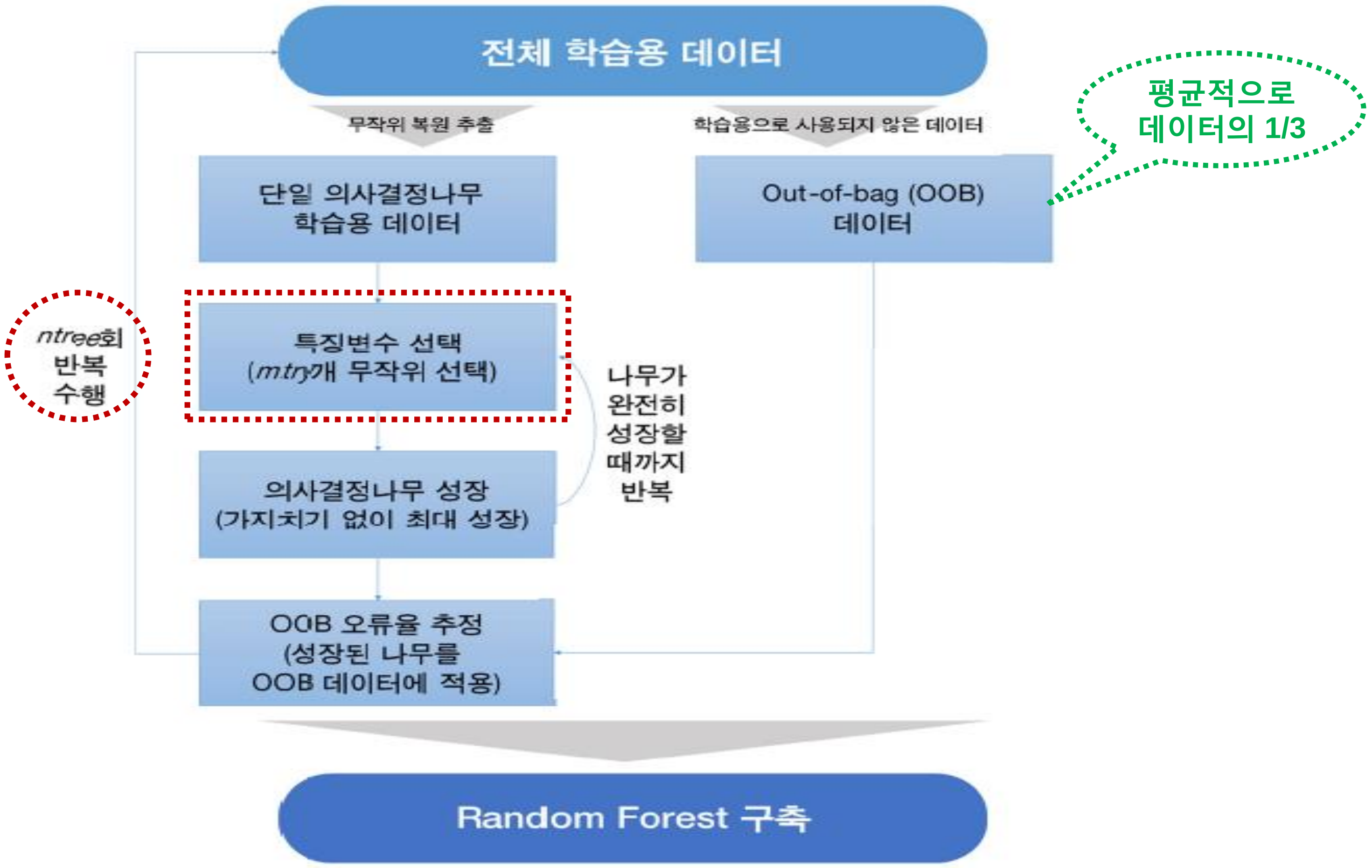
Random Forest – concept



- Breiman이 2001년에 개발
- 하나가 아닌 여러 개의 나무로 확장시킨 의사결정나무 기반의 Ensemble 기법

Randomization
- Bootstrap samples **(bagging)**
- Random selection of $K \leqslant p$ split variables **(random input selection)**

Random Forest – algorithm

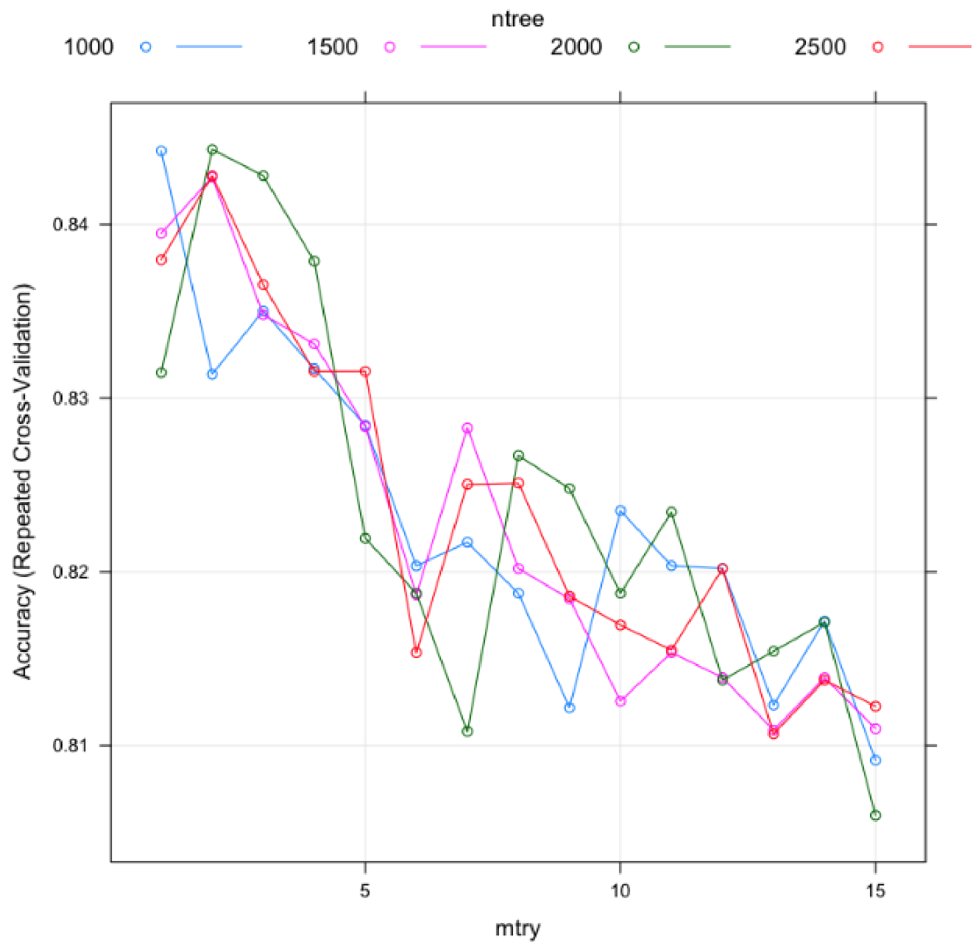

전체 학습용 데이터
무작위 복원 추출
학습용으로 사용되지 않은 데이터
평균적으로 데이터의 1/3
단일 의사결정나무 학습용 데이터
Out-of-bag (OOB) 데이터
*ntree*회 반복 수행
특징변수 선택 (*mtry*개 무작위 선택)
나무가 완전히 성장할 때까지 반복
의사결정나무 성장 (가지치기 없이 최대 성장)
OOB 오류율 추정 (성장된 나무를 OOB 데이터에 적용)
Random Forest 구축

Source: 김성진 & 안현철, "랜덤 포레스트를 활용한 기업채권등급평가 모형", 2014.

Random Forest – pros & cons

- 숲의 크기(나무의 수)가 커질수록 일반화 오류가 특정 값으로 수렴하게 되어 over-fitting을 피할 수 있음
- 전체 학습용 데이터에서 무작위로 복원 추출된 데이터를 사용함으로써 잡음이나 outlier로부터 크게 영향을 받지 않음
- 분석가가 입력변수 선정으로부터 자유로울 수 있음
- Class의 빈도가 불균형일 경우 타 기법에 비해 우수한 예측력을 보임
- 최종 결과에 대한 해석이 어려움

Random Forest – considered heuristics



Custom Tuning of Random Forest parameters in R

mtry: Number of variables randomly sampled as candidates at each split.

ntree: Number of trees to grow.

Source: http://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/

Related Packages

- Using "randomForest" packages
- Related functions
  - randomForest()
  - importance()
  - varImpPlot()
  - plot()
  - predict()

Ensemble Learning using randomForest

```
install.packages("randomForest")
library(randomForest); library(caret); library(ROCR)

cb <- read.delim("Hshopping.txt", stringsAsFactors=FALSE)
cb$반품여부 <- factor(cb$반품여부)

set.seed(1)
inTrain <- createDataPartition(y=cb$반품여부, p=0.6, list=FALSE)
cb.train <- cb[inTrain,]
cb.test <- cb[-inTrain,]

set.seed(123)
rf_model <- randomForest(반품여부 ~ .-ID, data=cb.train, ntree=50, mtry=2)

rf_model
```

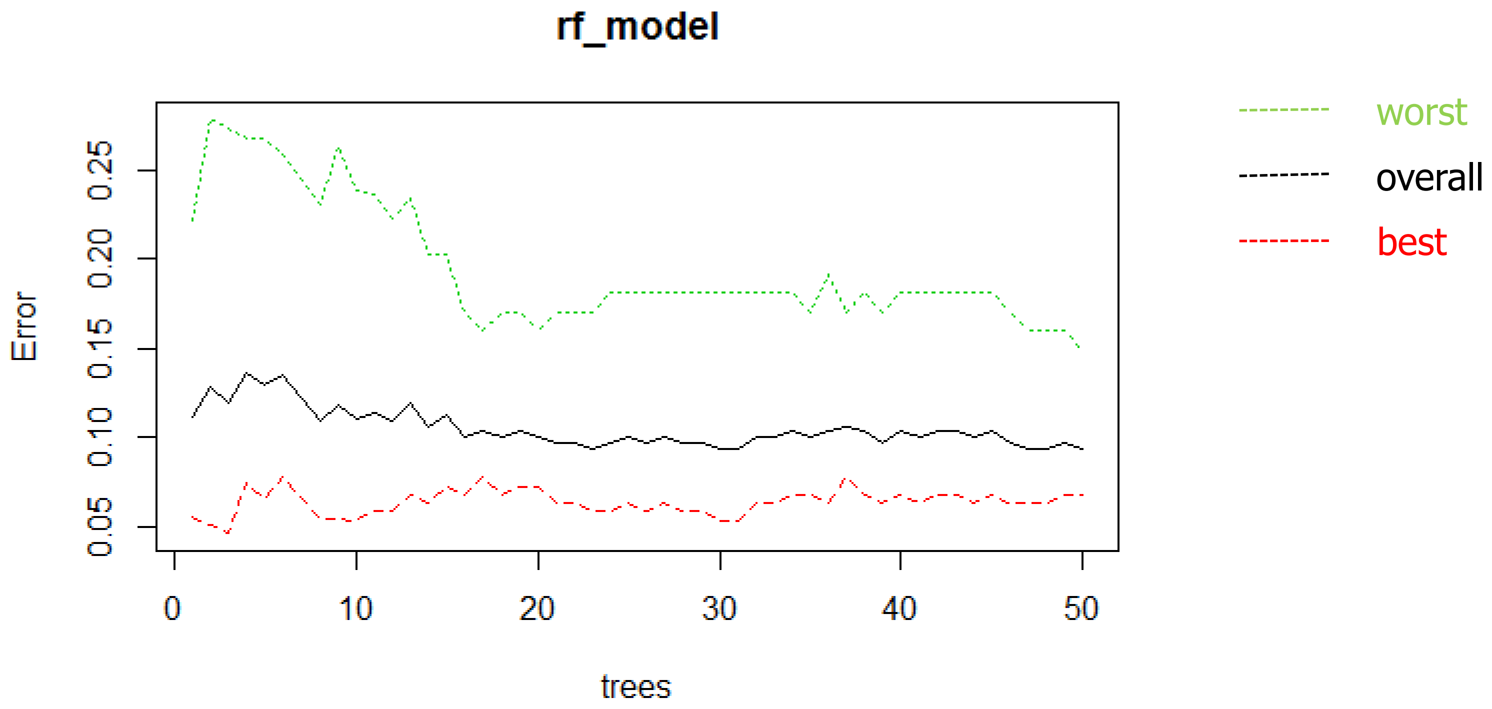
default value is sqrt(p) where p is number of variables in x

```
Call:
 randomForest(formula = 반품여부 ~ . - ID, data = cb.train, ntree = 50,      mtry = 2)
               Type of random forest: classification
                     Number of trees: 50
No. of variables tried at each split: 2

        OOB estimate of  error rate: 9.3%
Confusion matrix:
    0  1 class.error
0 193 14  0.06763285
1  14 80  0.14893617
```

out-of-bag 샘플을 사용하여 검증

Ensemble Learning using randomForest

- `plot(rf_model)`

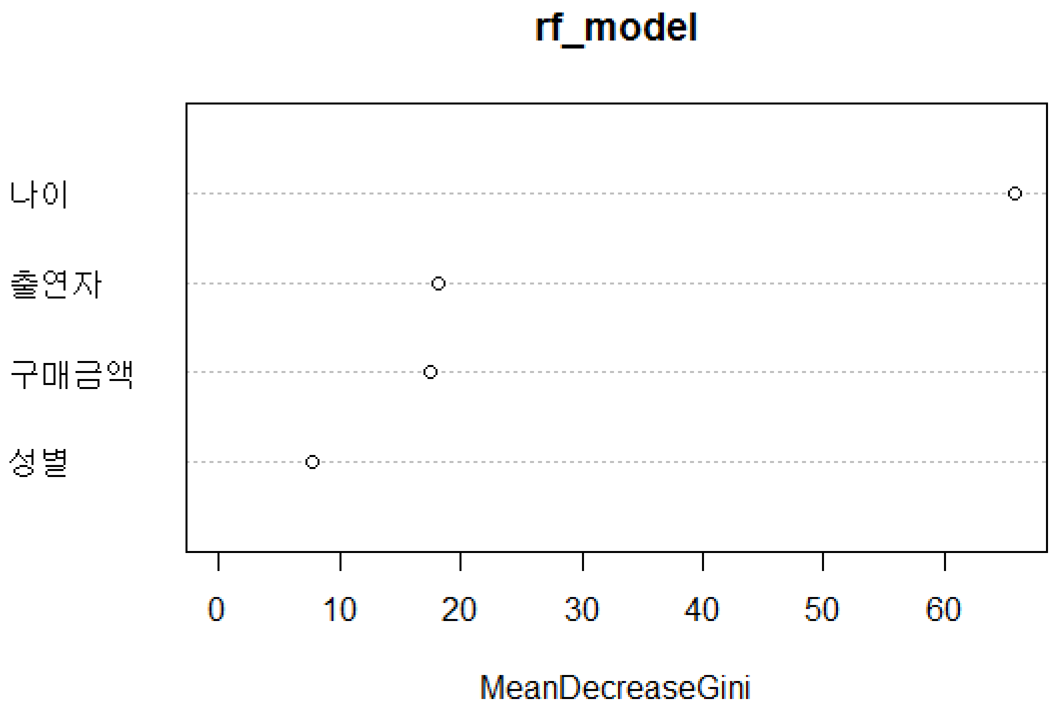


Ensemble Learning using randomForest

- `importance(rf_model)`

```
                MeanDecreaseGini
성별                    7.648143
나이                   65.864872
구매금액               17.421254
출연자                 18.225289
```

- `varImpPlot(rf_model)`



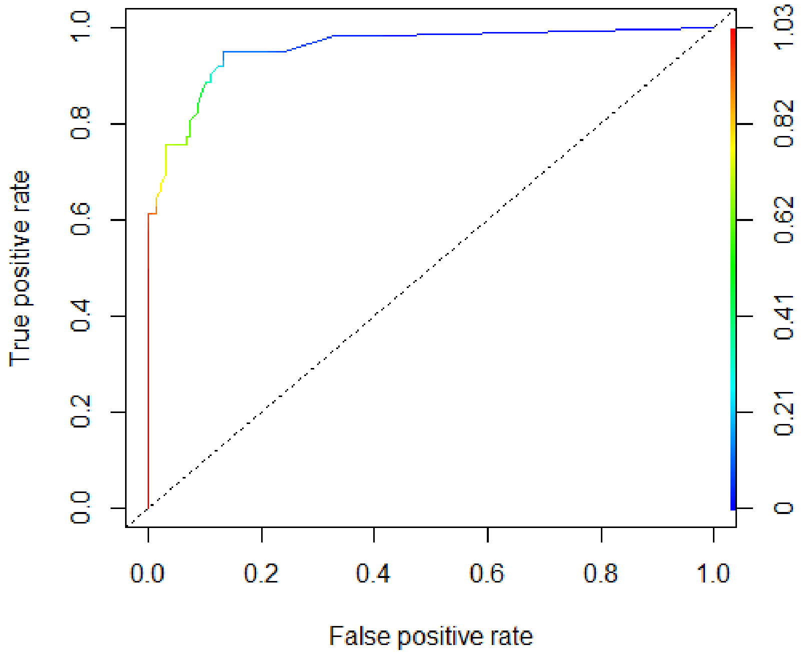
Ensemble Learning using randomForest

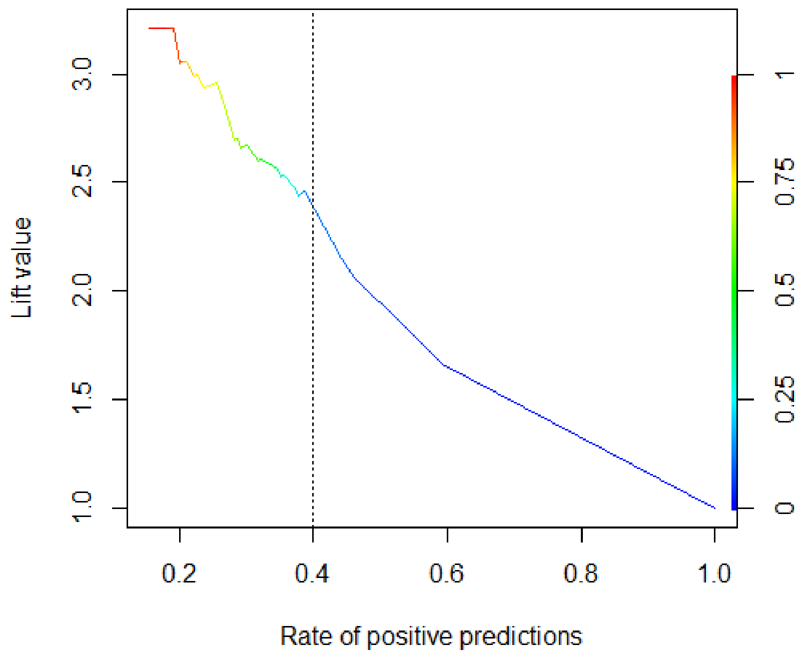
- cb.test$rf_pred <- predict(rf_model, cb.test, type="response")
- confusionMatrix(cb.test$rf_pred, cb.test$반품여부)

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 125  11
         1  12  51

               Accuracy : 0.8844
                 95% CI : (0.8316, 0.9253)
    No Information Rate : 0.6884
    P-Value [Acc > NIR] : 7.026e-11

                  Kappa : 0.7318
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9124
            Specificity : 0.8226
         Pos Pred Value : 0.9191
         Neg Pred Value : 0.8095
             Prevalence : 0.6884
         Detection Rate : 0.6281
   Detection Prevalence : 0.6834
      Balanced Accuracy : 0.8675

       'Positive' Class : 0
```

Ensemble Learning using randomForest

```
cb.test$rf_pred_prob <- predict(rf_model, cb.test, type="prob")
rf_pred <- prediction(cb.test$rf_pred_prob[,2],cb.test$반품여부)
rf_model.perf1 <- performance(rf_pred, "tpr", "fpr") # ROC-chart
rf_model.perf2 <- performance(rf_pred, "lift", "rpp")
plot(rf_model.perf1, colorize=TRUE); abline(a=0, b=1, lty=3)
plot(rf_model.perf2, colorize=TRUE); abline(v=0.4, lty=3)
```





```
performance(rf_pred, "auc")@y.values[[1]]
[1] 0.9574405
```

분류 및 예측 (4) : SVM(Support Vector Machine)

SVM – intro.

SVM(Support Vector Machine)은 지도학습모형(supervised learning)중 하나로 분류 (classification)와 회귀 (regression)에 응용하여 데이터를 분류한다.

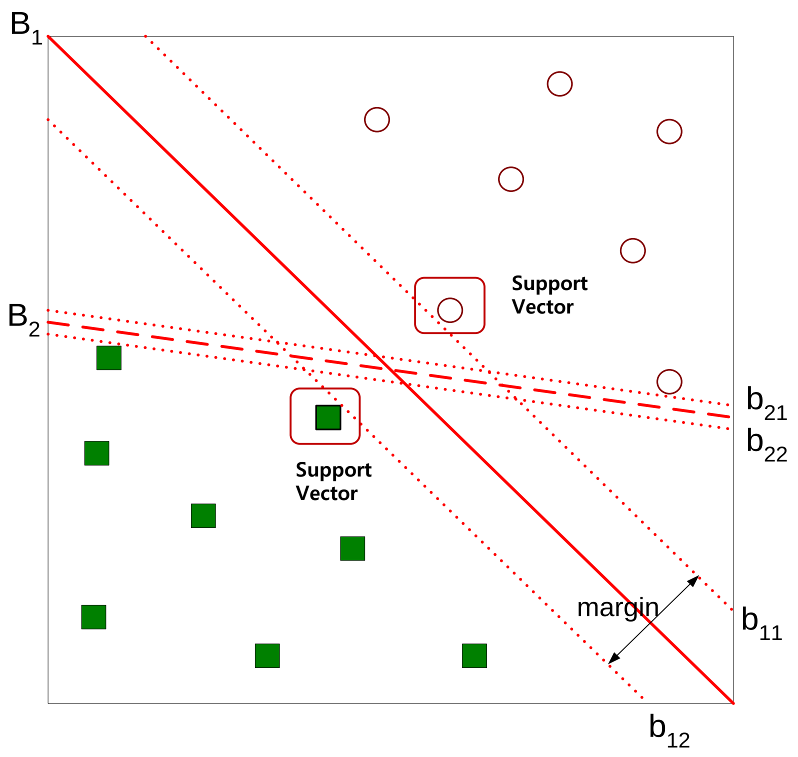
개요

- ✓ 두 카테고리 중 어느 하나에 속한 데이터의 집합이 주어졌을 때, SVM 알고리즘은 주어진 데이터 집합을 바탕으로 하여 새로운 데이터가 어느 카테고리에 속할지 판단하는 비확률적 이진 선형 분류 모델을 만든다. 만들어진 분류 모델은 데이터가 사상된 공간에서 경계로 표현되는데 SVM 알고리즘은 그 중 가장 큰 폭을 가진 경계를 찾는 알고리즘이다. SVM은 선형 분류와 더불어 비선형 분류에서도 사용될 수 있다. 비선형 분류를 하기 위해서 주어진 데이터를 고차원 특징 공간으로 사상하는 작업이 필요한데, 이를 효율적으로 하기 위해 '커널 트릭'을 사용하기도 한다.

특징

- ✓ 기존의 지도학습모형과 같이 예측을 부분에서 활용될 수 있으며 기계학습 부분에서 다른 모델에 비해서 예측률이 높다고 알려져 있다.
- ✓ 넓은 형태의 데이터 셋(많은 예측 변수를 가지고 있는)에 적합하다.
- ✓ 모델을 생성할 때는 기본적인 설정사항을 이용해 비교적 빨리 모형을 생성할 수 있다.
- ✓ 실제 응용에 있어서 인공신경망 보다 높은 성과를 내고 명백한 이론적 근거에 기반하므로 결과 해석이 상대적으로 용이하다.

SVM – margin

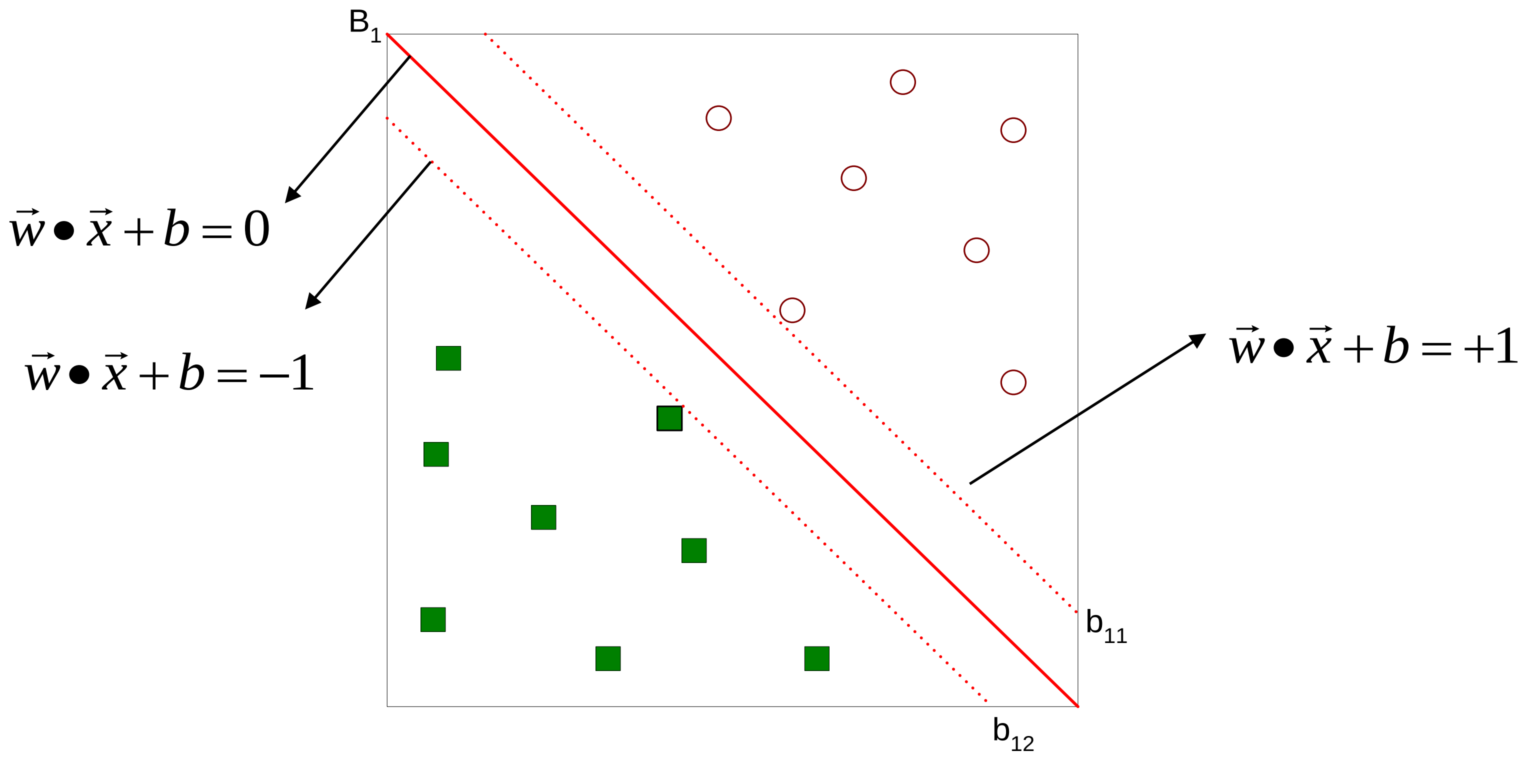


✓ 의사결정 경계: B1 B2

✓ 마진(margin)
- 의사결정경계에서 가장 가까운 Support Vector 의 근접 경계들 사이 거리 〔b11~ b12, b21~b22〕

✓ 큰 마진을 가진 의사결정 경계는 작은 마진의 경우보다 일반화 오류가 적다.
마진이 작으면 경계의 작은 움직임도 분류에 상당한 영향을 미칠 수가 있기 때문에 일반화하기가 어려운 경향이 있다.

SVM – hyperplane



B$_1$
$\vec{w}\bullet\vec{x}+b=0$
$\vec{w}\bullet\vec{x}+b=-1$
$\vec{w}\bullet\vec{x}+b=+1$
b$_{11}$
b$_{12}$

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w}\bullet\vec{x}+b \geq 1 \\ -1 & \text{if } \vec{w}\bullet\vec{x}+b \leq -1 \end{cases}$$

SVM – maximum margin hyperplane

$$w \cdot (x_1 - x_2) = 2$$

$$\|\vec{w}\| \times d = 2$$

$$d = \frac{2}{\|\vec{w}\|}$$

$$\Downarrow$$

$$\text{Margin} = \frac{2}{\|\vec{w}\|}$$

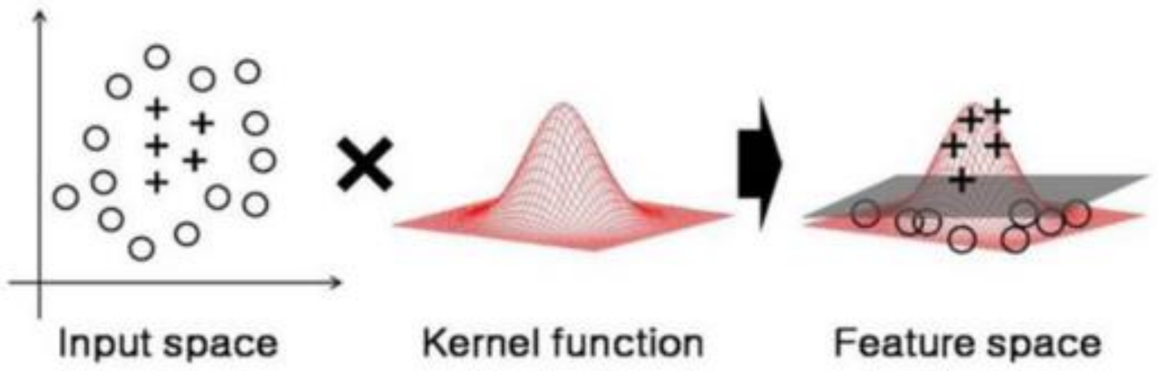
마진의 최대화

$$\min_{\mathrm{w}} \frac{\|\vec{w}\|^2}{2}$$

where

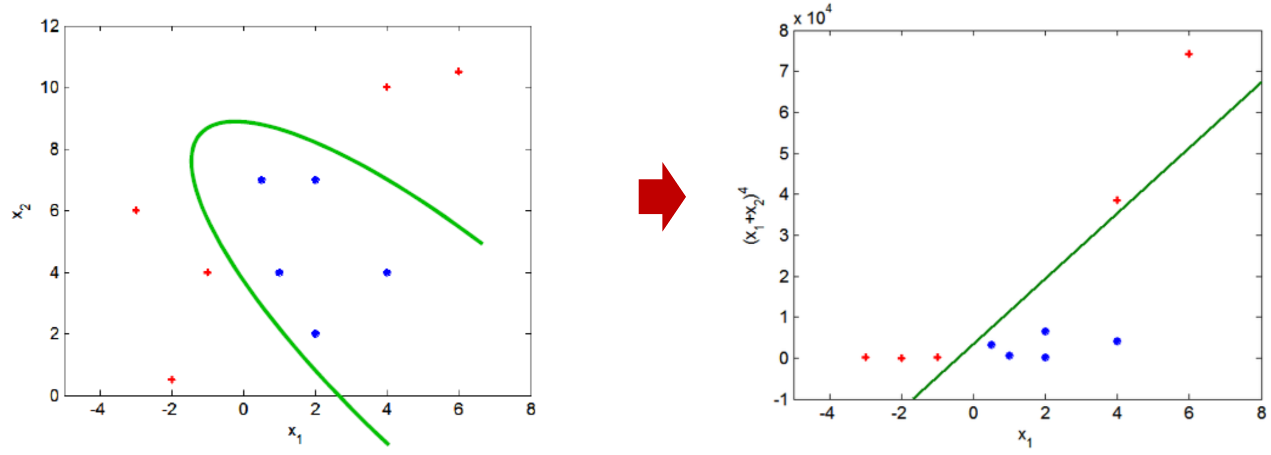
$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}} + \mathrm{b} \geq 1 \\ -1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}} + \mathrm{b} \leq -1 \end{cases}$$

SVM – kernel trick



(complex in low dimensions)

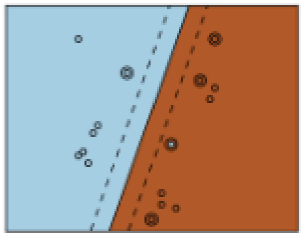
(simple in higher dimensions)

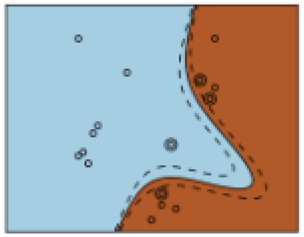


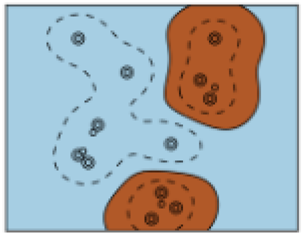
SVM – types of kernels

Kernel	Formula	Parameters	R name
Linear	$\mathbf{u}^\mathbf{T}\mathbf{v}$	none	
Polynomial	$\gamma(\mathbf{u}^\mathbf{T}\mathbf{v} + c_0)^d$	γ, d, c_0	gamma=γ coef0=c_0 degree=d
default kernel Gaussian Radial basis fct.	$exp[-\gamma\|\mathbf{u} - \mathbf{v}\|^2]$	γ	gamma=γ
Sigmoid	$\tanh[\ \gamma[\mathbf{u}^\mathbf{T}\mathbf{v} + c_0]]$	γ, c_0	gamma=γ coef0=c_0

See e1071.pdf for more info.


Linear

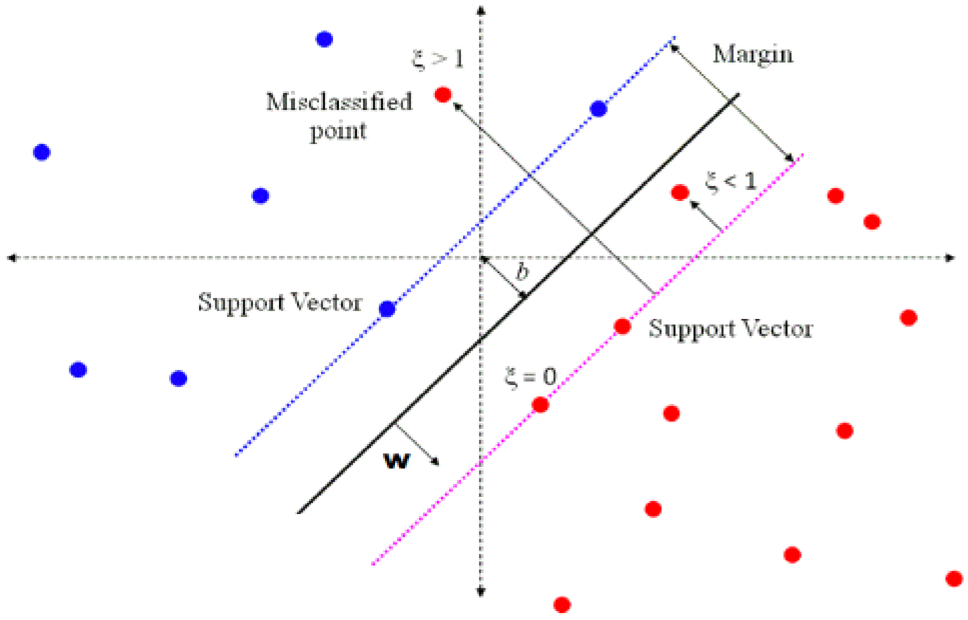

Polynomial


RBF

Source: *http://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html*

SVM – hard vs. soft margin

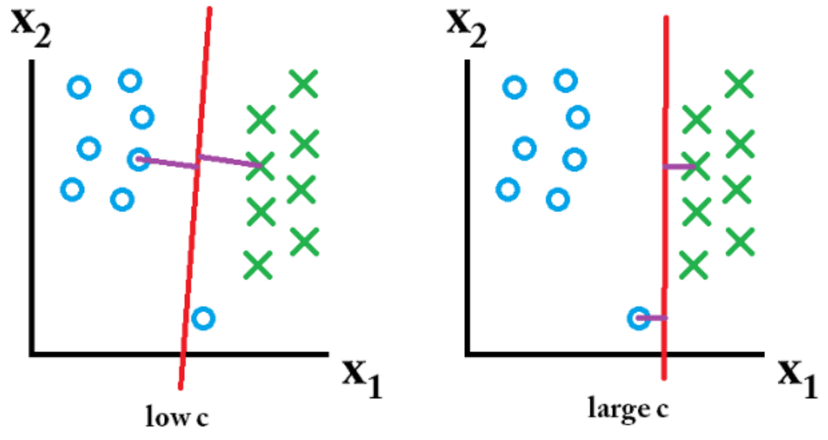
- Hard-margin vs. Soft-margin
  - SVM searches for two things: a hyperplane with the largest margin, and a hyperplane that correctly separates as many instances as possible
  - If all training data points are properly classified, the margin is usually relatively thin so it may be called a hard-margin.
  - But we often deal with a set which is not linearly separable or have some outliers. Then, it's better to soften the margin and allow for some misclassification in exchange to the grouping generalization.



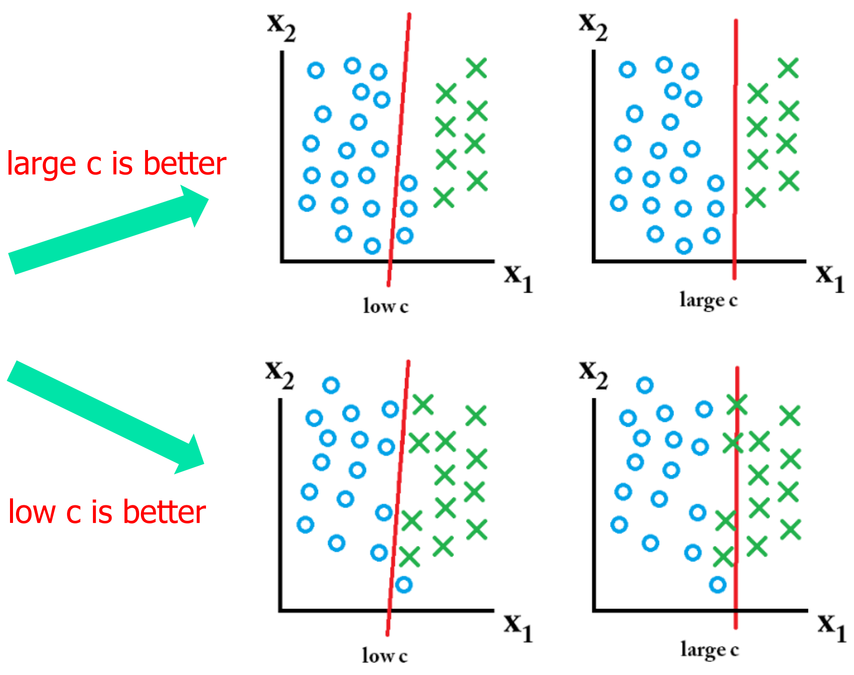
Source: http://tomaszkacmajor.pl/index.php/2016/04/24/svm-model-selection/

SVM – considered heuristics

- Complexity parameter (**C**) $\frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i$
  - It tells us how many points are allowed to be on the wrong side of the margin.
  - **C** is high, we get a hard margin. Otherwise, if **C** is close to 0, the SVM margin is very soft and almost not caring about proper division of data.
  - Therefore, the decision about the adequate complexity parameter is very important.



training data

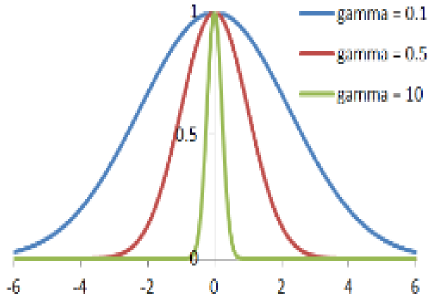


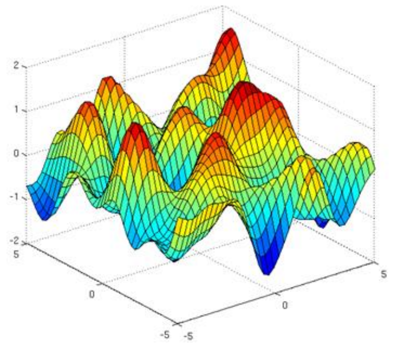
future data

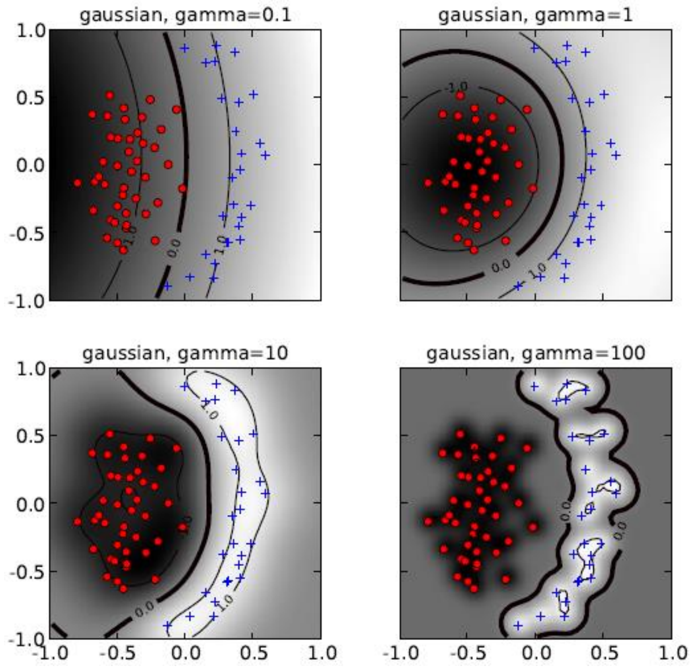
Source: http://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel

SVM – considered heuristics

- Gamma parameter (γ)
  - The larger gamma the thinner gaussian curve is.
  - Gaussian kerneling for a point x consists in calculation of a sum of Gaussian "bumps" centered around each support vector in a training dataset.
  - For small values of γ (upper left) the decision boundary is nearly linear.
  - As γ increases the flexibility of the decision boundary increases.
  - Large values γ of lead to overftting (bottom).







Source: *"A User's Guide to Support Vector Machines" by Ben-Hur & Weston*

Related Packages

- Using "e1071" packages
- Related functions
    - svm()
    - predict()
    - plot()
    - tune.svm()

Classification using SVM

- install.packages("e1071")
- library(e1071)

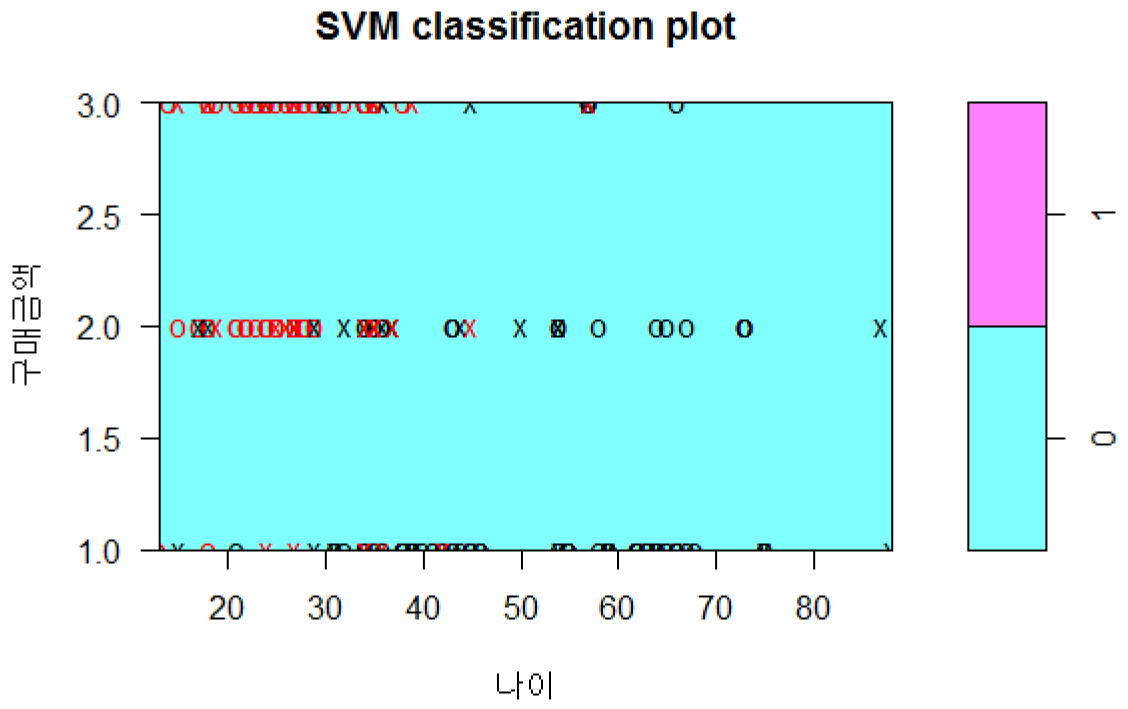
- svm_model <- svm(반품여부 ~ 성별+나이+구매금액+출연자, data=cb.train, cost=100, gamma=1, probability = TRUE)

- summary(svm_model)

```
Call:
svm(formula = 반품여부 ~ 성별 + 나이 + 구매금액 + 출연자, data = cb.train, cost = 100, gamma = 1,
    probability = TRUE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  100
      gamma:  1

Number of Support Vectors:  77

 ( 45 32 )


Number of Classes:  2

Levels:
 0 1
```

Classification using SVM

- `plot(svm_model, data=cb.train, 구매금액~나이)`



x: support vector

Classification using SVM

- `cb.test$svm_pred <- predict(svm_model, cb.test)`
- `confusionMatrix(cb.test$rf_pred, cb.test$반품여부)`

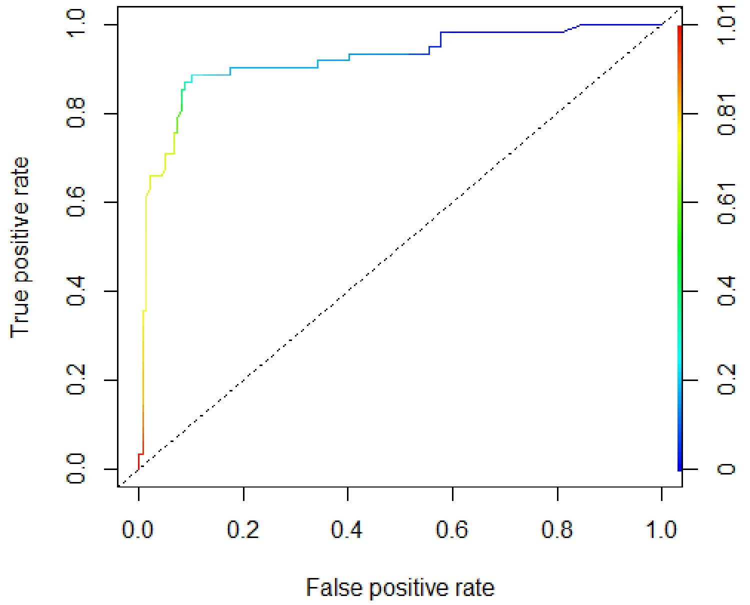
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 126  12
         1  11  50

               Accuracy : 0.8844
                 95% CI : (0.8316, 0.9253)
    No Information Rate : 0.6884
    P-Value [Acc > NIR] : 7.026e-11

                  Kappa : 0.7294
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9197
            Specificity : 0.8065
         Pos Pred Value : 0.9130
         Neg Pred Value : 0.8197
             Prevalence : 0.6884
         Detection Rate : 0.6332
   Detection Prevalence : 0.6935
      Balanced Accuracy : 0.8631

       'Positive' Class : 0
```

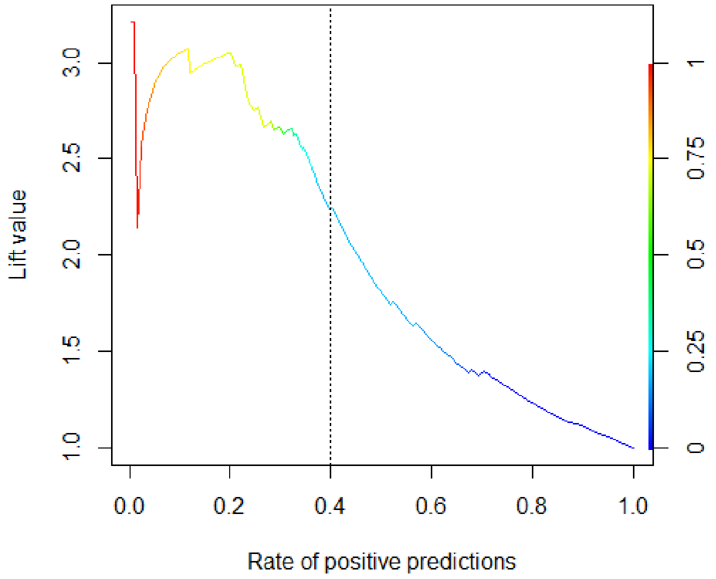
- `postResample(cb.test$svm_pred, cb.test$반품여부)`

```
 Accuracy     Kappa
0.8844221 0.7293798
```

Classification using SVM

```
➢ cb.test$svm_pred_prob <- attr(predict(svm_model, cb.test, probability=
  TRUE), "probabilities")[,2]
➢ svm_pred <- prediction(cb.test$svm_pred_prob, cb.test$반품여부)
➢ svm_model.perf1 <- performance(svm_pred, "tpr", "fpr") # ROC-chart
➢ svm_model.perf2 <- performance(svm_pred, "lift", "rpp")
➢ plot(svm_model.perf1, colorize=TRUE); abline(a=0, b=1, lty=3)
➢ plot(svm_model.perf2, colorize=TRUE); abline(v=0.4, lty=3)
```





```
➢ performance(svm_pred, "auc")@y.values[[1]]
  [1] 0.9210031
```

Classification using SVM

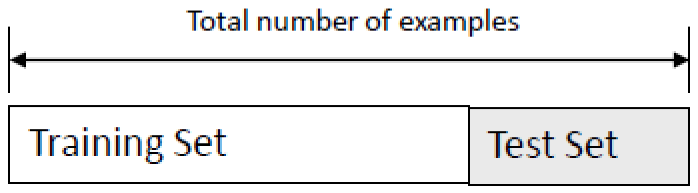
- tune.svm(반품여부~성별+나이+구매금액+출연자, data=cb.train, gamma=seq(.5, .9, by=.1), cost=seq(100,1000, by=100))

```
Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 gamma cost
   0.9  300

- best performance: 0.1062366
```

Cross Validation

The holdout method

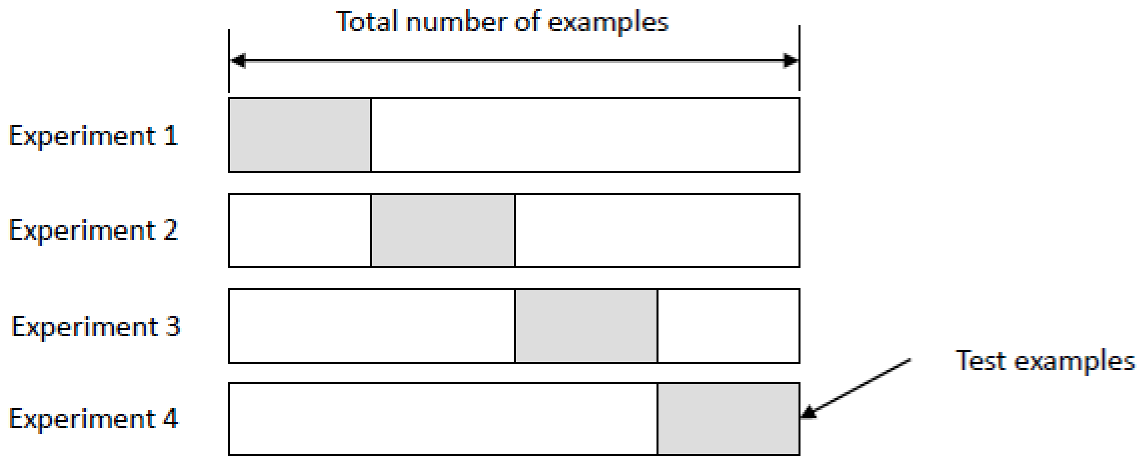
- **Split dataset into two groups**
  - Training set: used to train the classifier
  - Test set: used to estimate the error rate of the trained classifier



- **Drawbacks**
  - In problems where we have a sparse dataset we may not be able to afford the "luxury" of setting aside a portion of the dataset for testing
  - Since it is a single train-and-test experiment, the holdout estimate of error rate will be misleading if we happen to get an "unfortunate" split

K–fold cross validation (KFCV)

- **Create a K-fold partition of the dataset**
  - For each of K experiments, use $K-1$ folds for training and a different fold for testing
  - This procedure is illustrated in the following figure for $K=4$



- **Advantages**
  - The advantage of KFCV is that all the examples in the dataset are eventually used for both training and testing
  - The error is estimated as the average error rate on test examples

Related Packages

- Using "caret" packages
- Related functions

createDataPartition()	for holdout method
createFolds()	for KFCV
createMultiFolds()	for repetition of KFCV

5-fold cross validation process

```
# Create a 5-fold partition using the caret package

set.seed(1)
flds <- createFolds(cb$반품여부, k=5, list=TRUE, returnTrain=FALSE)
str(flds)
```

```
List of 5
 $ Fold1: int [1:99] 5 7 12 14 18 19 21 25 30 32 ...
 $ Fold2: int [1:101] 11 33 39 41 45 80 85 86 90 94 ...
 $ Fold3: int [1:100] 10 15 16 22 24 31 34 35 36 48 ...
 $ Fold4: int [1:100] 2 3 4 8 9 13 20 23 26 29 ...
 $ Fold5: int [1:100] 1 6 17 27 28 43 46 47 53 54 ...
```

```
# Perform 5 experiments

experiment <- function(train, test, m) {
  rf <- randomForest(반품여부 ~ .-ID, data=train, ntree=50)

  rf_pred <- predict(rf, test, type="response")
  m$acc = c(m$acc, confusionMatrix(rf_pred, test$반품여부)$overall[1])

  rf_pred_prob <- predict(rf, test, type="prob")
  rf_pred <- prediction(rf_pred_prob[,2], cb.test$반품여부)
  m$auc = c(m$auc, performance(rf_pred, "auc")@y.values[[1]])

  return(m)
}
```

5–fold cross validation process

```
measure = list()
for(i in 1:5){
  inTest <- flds[[i]]
  cb.test <- cb[inTest, ]
  cb.train <- cb[-inTest, ]

  measure = experiment(cb.train, cb.test, measure)
}
```

```
# Summarize model performance
```

```
measure
```

```
$acc
 Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
0.8888889 0.8712871 0.9200000 0.9100000 0.9000000

$auc
[1] 0.9656072 0.9585598 0.9630669 0.9614306 0.9794296
```

```
mean(measure$acc); sd(measure$acc)
```

```
[1] 0.8980352
[1] 0.0188983
```

```
mean(measure$auc); sd(measure$auc)
```

```
[1] 0.9656188
[1] 0.008133616
```