Seoul National University
Dept. of Civil and Environmental Engineering

Instructor: Junho Song
junhosong@snu.ac.kr

M1586.002500 Information Engineering for CE Engineers
In-Class Material: Class 21
Tree-Based Methods (ISL Chapter 8)

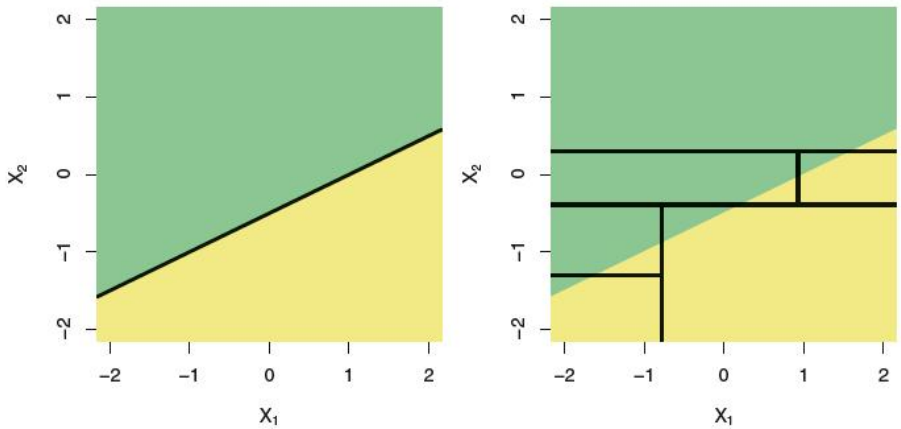
1. Trees Versus Linear Models

(a) Model forms:

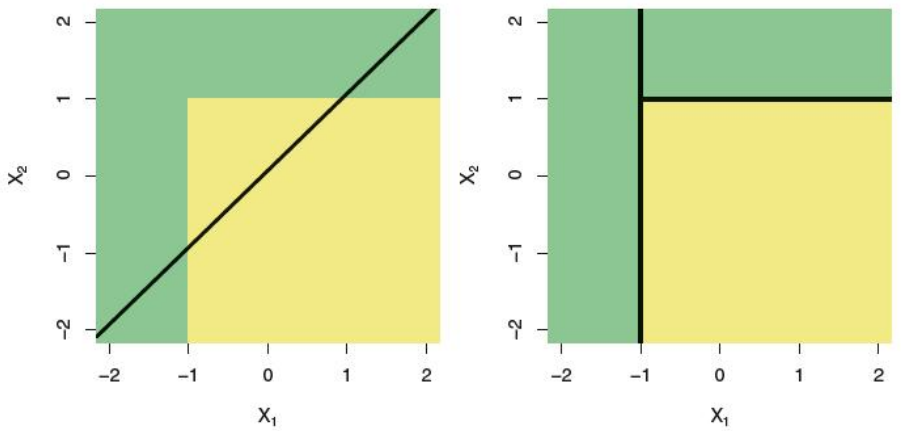
Linear model: $f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$

Regression Tree: $f(X) = \sum_{m=1}^{M} \hat{y}_{R_m} \cdot 1_{(X \in R_m)}$
where $1_{(X \in R_m)} = 1$ if $x \in R_m$, and 0 otherwise.

(b) Linear models are preferable when the given features and the response can be well approximated by linear relationship



(c) Trees are preferable when there is a highly non-linear and complex relationship between the features and the response



or for the sake of interpretability and visualization

Instructor: Junho Song
junhosong@snu.ac.kr

2. Advantages and Disadvantages of Trees

(a) Advantages

- Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if the trees are small)
- Some people believe that decision trees more closely mirror human decision-making
- Trees can easily handle qualitative predictor without the need to create dummy variables

(b) Disadvantages

- Trees generally do not have the same level of predictive accuracy as other advanced approaches for regression and classification
- Trees can be very non-robust, i.e. a small change in data can cause a large change in the final estimated tree

Note: the predictive performance of trees can be substantially improved by a_____ many decision trees, using methods like *bagging*, *random forests*, and *boosting*.

3. Techniques to Improve Trees (1): Bagging ("Bootstrap Aggregation")

(a) Strategy: By averaging a set of observations, reduce the high variance of a single tree, i.e. avoid overfitting

Note: Given a set of n independent observations $Z_1, \cdots, Z_n$, each with variance σ^2, the variance of their mean is given by σ^2/n

(b) Procedure:

(1) Generate *B* different bootstrapped training data set.
Recall: Bootstrap artificially generates a large set of data from a given smaller data set by randomly sampling the data points with replacement.

(2) For each of the *B* training sets, evaluate a deep, not pruned (i.e. high variance and low bias) tree, T_b, $b = 1, \cdots, B$.

(3) Make predictions $\hat{f}^{*b}(x)$ using T_b and aggregate to obtain the final solution $\hat{f}_{\text{bag}}(x)$:
For regression, take the average, i.e. $\hat{f}_{\text{bag}}(x) = \left(\sum_{b=1}^{B} \hat{f}^{*b}(x)\right)/B$
For classification, determine by majority vote among the *B* predictions

(c) Out-of-Bag (OOB) error estimation

- Efficient way to estimate the test error of a bagged model

Instructor: Junho Song
junhosong@snu.ac.kr

- Useful especially for large data-set for which cross-validation would be computationally onerous

- Motivation: One can show that on average, each bagged tree uses around two-thirds of the observations (Why? **Homework**)

- Method: Predictions are made on each observation using around $B/3$ trees that have not used the observation, and aggregate the result for all observations to get OOB error

- OOB error is a valid estimate of the test error (For *B* sufficiently large, OOB error is virtually equivalent to leave-one-out cross-validation error)

(d) Variable importance measure

- In contrast to a single tree, utilizing multiple trees makes it hard to interpret the resulting model

- Alternatively, by for each predictor, one can quantify the relative importance measure by computing the mean decrease in reference error measure, e.g. RSS during split across the *B* trees, i.e. the larger the mean value, the more important the predictor is

4. Techniques to Improve Trees (2): Random Forests

(a) Strategy: By **de-correlating** the trees in bagging, obtain higher reduction in variance than bagging

Note: If the observations $Z_1, \cdots, Z_n$ are NOT independent, the variance of their mean is greater than by σ^2/n. Bootstrapping generates highly correlated samples.

(b) Method: For each split in each tree, only a fresh sample of m predictors are considered among *p* predictors
- Typically, $m \approx \sqrt{p}$ is chosen
- Equivalent to bagging when $m = p$
- Small m is desirable when a large number of predictors are correlated

Large *B* allows us to avoid overfitting. In practice, it is typical to choose *B* sufficiently large for the error rate to have settled down

5. Techniques to Improve Trees (3): Boosting

(a) Strategy: Instead of fitting a tree directly into a given set of data (i.e. fitting the data hard), sequentially fit a decision tree to the r_____ of current tree (i.e. learn slowly)

(b) Procedure (in regression):

(1) Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
(y_i: the response of *i*th observation)

Instructor: Junho Song
junhosong@snu.ac.kr

(2) For $b = 1, \cdots, B$, repeat:

(2-1) Fit a tree $\hat{f}^b$ with d splits ($d + 1$ terminal nodes) to the training data $(\boldsymbol{X}, \boldsymbol{r})$.

(2-2) Update $\hat{f}$ by adding in a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$

(2-3) Update the residuals,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$$

until $\boldsymbol{r}$ is sufficiently small to meet the stopping criterion.

(3) Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

(c) Parameters:

(1) The number of trees *B*: Unlike bagging and random forests, boosting can lead to overfitting if *B* is too large, although overfitting tends to occur slowly if at all.
→ Cross-validation to select *B* is required

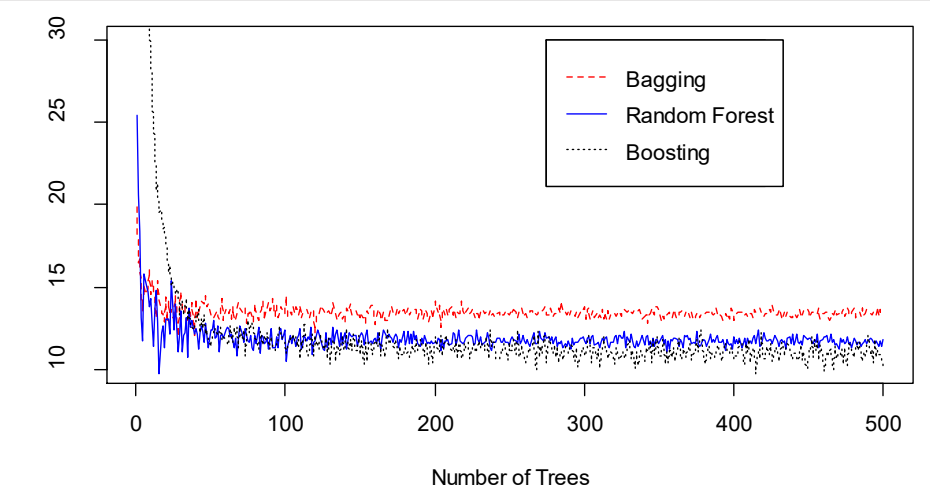
(2) The shrinkage parameter λ, a small positive number: This controls the rate at which boosting learns. Typical values are 0.01 or 0.001.
→ The smaller λ, the larger *B*

(3) The number *d* of splits in each tree (interaction depth): This controls the complexity of the boosted ensemble. Often $d = 1$ works well, i.e. **stump**, consisting of a single split (→ leads to an *additive* model)

Instructor: Junho Song
junhosong@snu.ac.kr

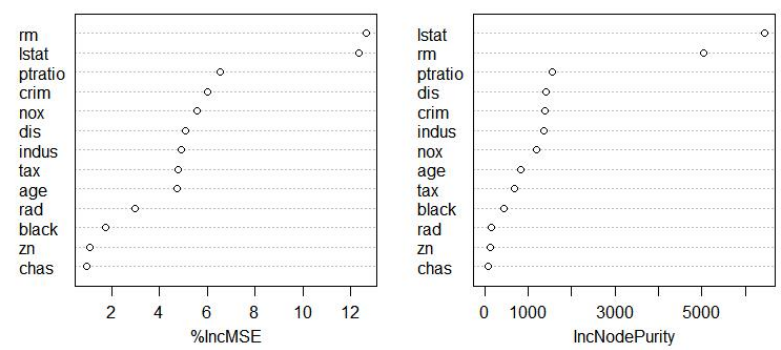
Example 3 (Bagging, random forests, and boosting): Using the Boston data set, predict medv (median value of owner-occupied homes) based on other 13 predictors.

Using R package "randomForest" for bagging and random forests, and "gbm" for boosting, the sets of trees with size from 5 to 2,500 have been examined. For random forests, $m = 4 \approx \sqrt{13}$ is used, and for boosting, $d = 4$ and $\lambda = 0.1$ are used. The mean squared errors of test data are obtained as



In the figure, for all of the three approaches, 100 trees appear to be enough for the error measure to settle down. Throughout the experiments, random trees and boosting show comparative performance to each other while both outperforming bagging.

The importance of each variable can be measured using bagging or random forests. Using the random forest with 100 trees, the importance of variables is quantified as



On the right-hand side is the mean decrease of accuracy in predictions on the OOB samples when a given variable is excluded from the model. On the left-hand side is a measure of the total decrease in node impurity that results from splits over that variable, averaged over all trees. While the two measures agree overall, there is a slight difference in the order and the relative magnitude of quantified importance between variables.

Instructor: Junho Song
junhosong@snu.ac.kr

```
library(randomForest)
library(gbm)
library(MASS)
attach(Boston)

## Example 3 (Bagging, random forests, boosting)

set.seed(1)
train=sample(1:nrow(Boston),nrow(Boston)/2)
boston.test = Boston[-train,"medv"]

ntrees = 1:500
for (i in ntrees) {
  bag2.boston =
   randomForest(medv~.,data=Boston,subset=train,mtry=13,ntree=i)
  rf2.boston =
   randomForest(medv~.,data=Boston,subset=train,mtry=4,ntree=i)
  boost2.boston =
   gbm(medv~.,data=Boston[train,],distribution="gaussian",n.trees=i,intera
   ction.depth=4) # default value for option shrinkage = 0.1

  yhat.bag = predict(bag2.boston,newdata=Boston[-train,])
  yhat.rf = predict(rf2.boston,newdata=Boston[-train,])
  yhat.boost = predict(boost2.boston,newdata=Boston[-train,],n.trees=i)

  if (i==ntrees[1]) {
    mse.bag = mean((yhat.bag-boston.test)^2)
    mse.rf = mean((yhat.rf-boston.test)^2)
    mse.boost = mean((yhat.boost-boston.test)^2)
  } else {
    mse.bag = c(mse.bag,mean((yhat.bag-boston.test)^2))
    mse.rf = c(mse.rf,mean((yhat.rf-boston.test)^2))
    mse.boost = c(mse.boost,mean((yhat.boost-boston.test)^2))
  }
}

# figure(1)
plot(ntrees,mse.bag,type='l',lty=2,ylab='Mean Squared Error (Test
   data)',xlab='Number of Trees',ylim=c(10,30),col='red')
lines(ntrees,mse.rf,type='l',lty=1,col='blue')
lines(ntrees,mse.boost,type='l',lty=3,col='black')

legend(x=275,y=30,legend=c("Bagging","Random
   Forest","Boosting"),lty=c(2,1,3),col=c("red","blue","black"))

# Importance measure
rf.boston =
   randomForest(medv~.,data=Boston,subset=train,mtry=4,ntree=100,importanc
   e=TRUE)
# figure(2)
varImpPlot(rf.boston)
```

Seoul National University
Dept. of Civil and Environmental Engineering

Instructor: Junho Song
junhosong@snu.ac.kr

M1586.002500 Information Engineering for CE Engineers
In-Class Material: Class 22
Support Vector Machine (ISL Chapter 9)

Maximal margin classifier: Separate the classes by a linear boundary, i.e. hyperplane

Support vector classifier: Allow some observations to be on the incorrect side of the margin, or even the incorrect side of the hyperplane

Support vector machine: Handle non-linear decision boundary by employing kernels

1. Maximal Margin Classifier

(a) Definition of hyperplane

In a p-dimensional space, a hyperplane is a flat affine subspace of dimension $(p-1)$

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$$

(b) Classification using a separating hyperplane

$$\beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip} > 0 \text{ if } y_i = 1$$

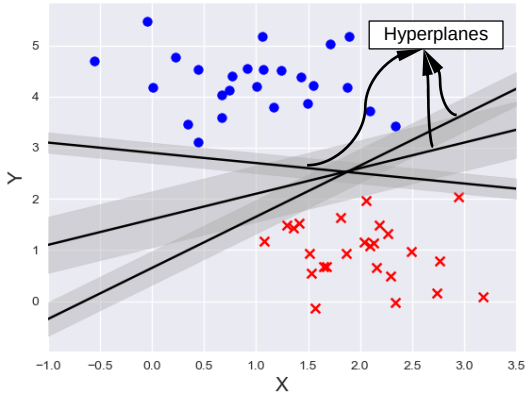
$$\beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip} < 0 \text{ if } y_i = -1$$

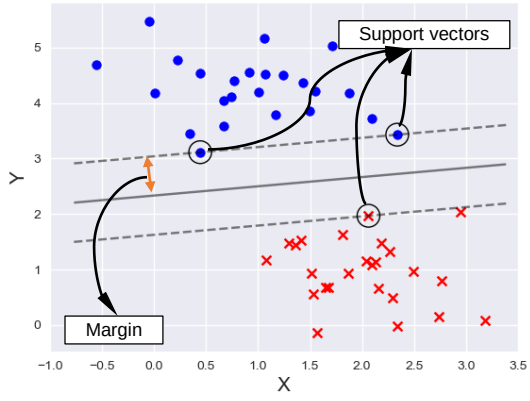
$$y_i \cdot (\beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip} > 0 \text{ for all } i = 1, \ldots, n$$

(c) Maximal margin hyperplane (optimal separating hyperplane)

Perfectly separating hyperplane for which the margin is largest, i.e. the hyperplane that has the farthest minimum distance to the training observations

Margin: minimal distance from the observations to the hyperplane





Instructor: Junho Song
junhosong@snu.ac.kr

(d) Construction of the maximal margin hyperplane – optimization problem: find the hyperplane maximizing the minimum distance
(How? Using so-called Lagrange multipliers)

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{maximize}}\, M$$

$$\text{subject to } \sum_{j=1}^{p}\beta_j^2 = 1,$$

$$y_i \cdot \left(\beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip}\right) \geq M, \;\; \forall\, i = 1, 2, \ldots, n$$

where $y_i \cdot \left(\beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip}\right)$ is the distance from a point $\left(X_{i1}, X_{i2}, \ldots, X_{ip}\right)$ to the hyperplane $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$.

Recall perpendicular distance d from a point (x_0, y_0) to the line $ax + by + c = 0$:

$$d = \frac{|ax_0 + bx_0 + c|}{\sqrt{a^2 + b^2}}$$

Therefore, distance d from a point $\left(X_{i1}, X_{i2}, \ldots, X_{ip}\right)$ to the hyperplane $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$ is

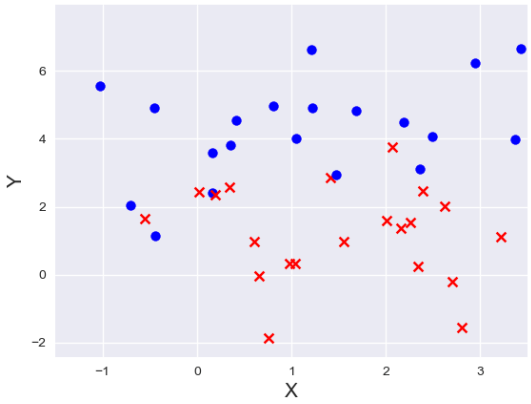
$$d = \frac{y_i(\beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip})}{\sqrt{\sum_{j=1}^{p}\beta_j^2}} = y_i(\beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip})$$

Note: Only observations that lie on the margin affect the hyperplane during optimization, i.e. lies strictly on the correct side of the margin does not affect the support vector classifier

→ **"Support vector"**

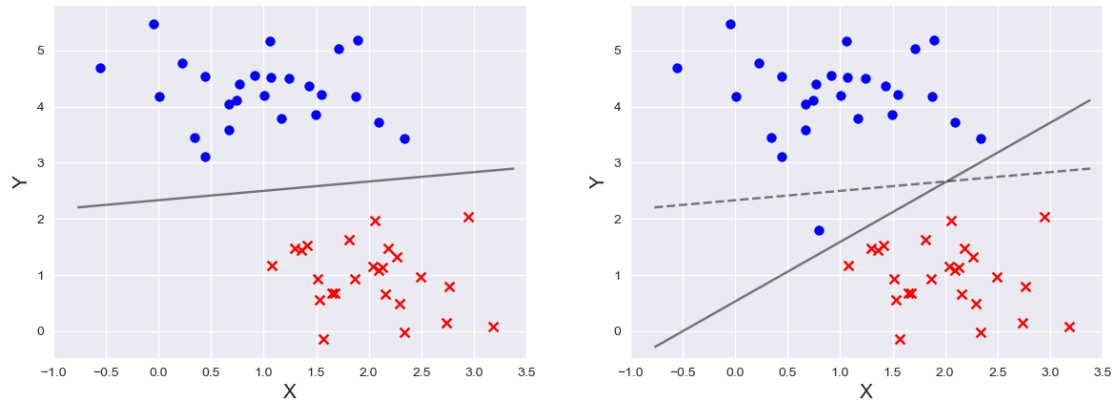
(e) Limitations of the maximal margin classifier

Case1: Non-separable



Instructor: Junho Song
junhosong@snu.ac.kr

Case2: Sensitive to the individual observations



2. Support Vector Classifier (Soft margin classifier)

(a) Goals

Greater robustness to individual observations

Better classification for most of the training observations

➔ Correctly separate most of the training observations into the two classes, but may misclassify a few observations

(b) Optimization problem for construction of support vector classifier

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \; M$$

subject to $\sum_{j=1}^{p} \beta_j^2 = 1$,

$$y_i \cdot \left(\beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip}\right) \geq M \cdot (1 - \epsilon_i),$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C$$

where

ε: Slack variable

C: Nonnegative tuning parameter

(c) Slack variable ε

Indicate where the observation is located

e.g. $\epsilon_i > 0$: ith observation is on the wrong side of the margin
$\epsilon_i > 1$: ith observation is on the wrong side of the hyperplane

Instructor: Junho Song
junhosong@snu.ac.kr

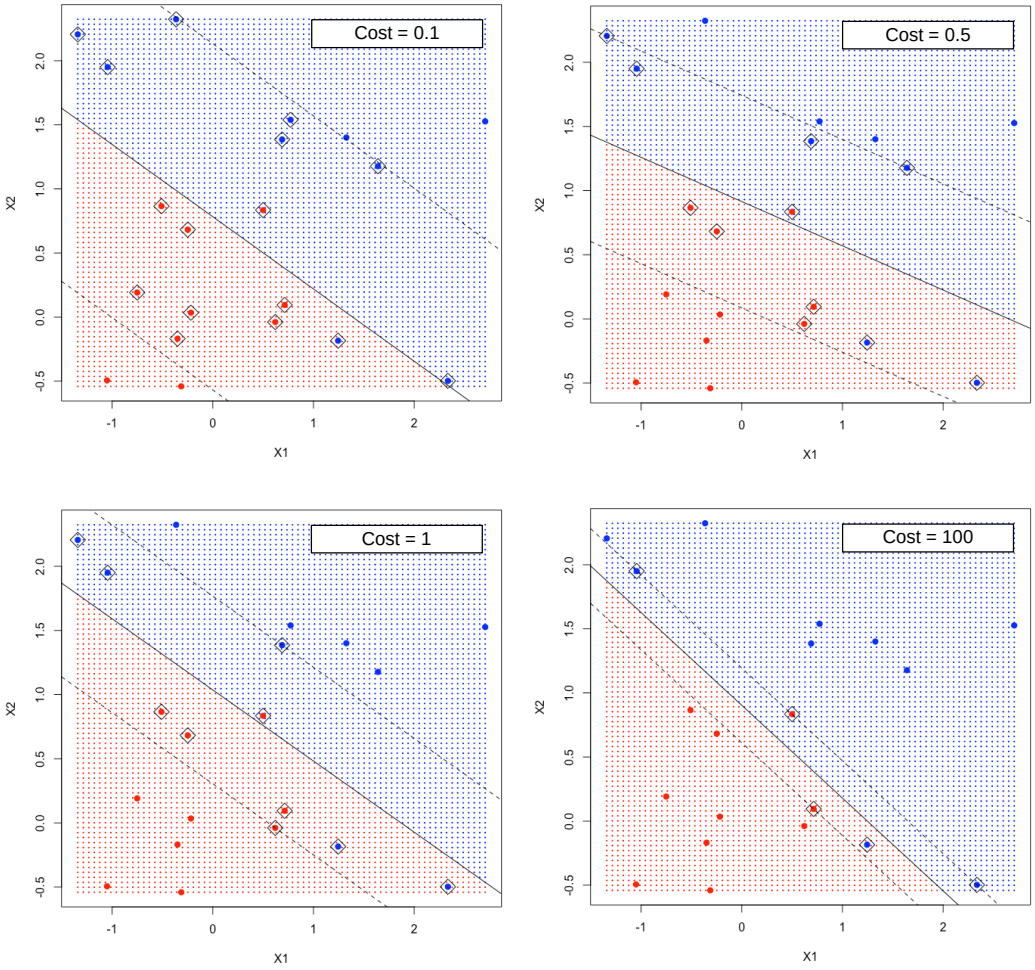
(d) Nonnegative tuning parameter C

Determines the number and severity of the violations to the margin and to the hyperplane that will be tolerated

Bias-variance trade-off: can be chosen via cross-validation technique

- Small C: seeking narrow margins that are rarely violated → highly fit to the data → low bias but high variance
- Large C: seeking wide margins and allow more violations → fitting the data less hard → more biased but have lower variance

By "cost" command in R (inverse value of C)



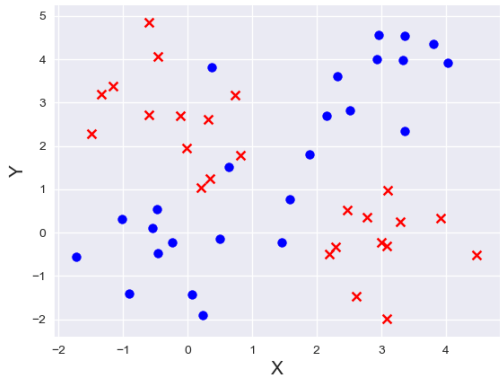

```
# Random number generation
set.seed(10111)
x = matrix(rnorm(40), 20, 2)
y = rep(c(-1, 1), c(10, 10))
x[y == 1,] = x[y == 1,] + 1
plot(x, col = y + 3, pch = 19)

# Classification using SVM
```

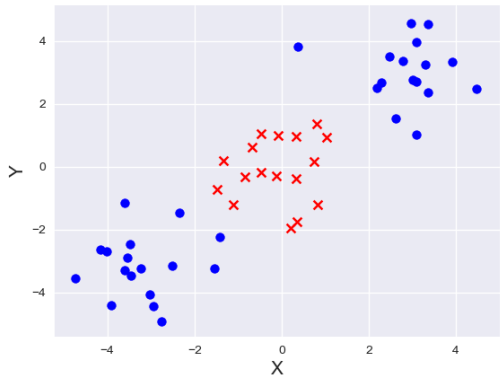
Instructor: Junho Song
junhosong@snu.ac.kr

```
# install.packages("e1071")
dat=data.frame(x=x, y=as.factor(y))
library(e1071)
dat = data.frame(x, y = as.factor(y))
svmfit = svm(y ~ ., data = dat, kernel = "linear", cost = 100, scale =
   FALSE)
print(svmfit)

# Plot SVM
make.grid = function(x, n = 75) {
  grange = apply(x, 2, range)
  x1 = seq(from = grange[1,1], to = grange[2,1], length = n)
  x2 = seq(from = grange[1,2], to = grange[2,2], length = n)
  expand.grid(X1 = x1, X2 = x2)
}
xgrid = make.grid(x)
xgrid[1:10,]

ygrid = predict(svmfit, xgrid)
plot(xgrid, col = c("red","blue")[as.numeric(ygrid)], pch = 20, cex = .2)
points(x, col = y + 3, pch = 19)
points(x[svmfit$index,], pch = 5, cex = 2)

beta = drop(t(svmfit$coefs)%*%x[svmfit$index,])
beta0 = svmfit$rho # 'rho' of svm object is the "negative intercept"

plot(xgrid, col = c("red", "blue")[as.numeric(ygrid)], pch = 20, cex = .2)
points(x, col = y + 3, pch = 19)
points(x[svmfit$index,], pch = 5, cex = 2)
abline(beta0 / beta[2], -beta[1] / beta[2])
abline((beta0 - 1) / beta[2], -beta[1] / beta[2], lty = 2)
abline((beta0 + 1) / beta[2], -beta[1] / beta[2], lty = 2)
```

(e) Limitation of support vector classifier

Case: non-linear decision boundaries





3. Support Vector Machine

(a) Mapping to higher dimensional space

Analogous to the attempts to move beyond nonlinearity (CM 18, CM 19)

For example, using $X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2$ instead of $X_1, X_2, \ldots, X_p$

Huge computational cost is accompanied

Instructor: Junho Song
junhosong@snu.ac.kr

(b) Support vector machine (SVM)

The linear support vector classifier can be represented as follows (See p. 420 of ESL for the proof):

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i \langle x, x_i \rangle$$

where,

$\langle x, x_i \rangle = \sum_{j=1}^{p} x_j x_{ij}$

S represents the collection of support vectors

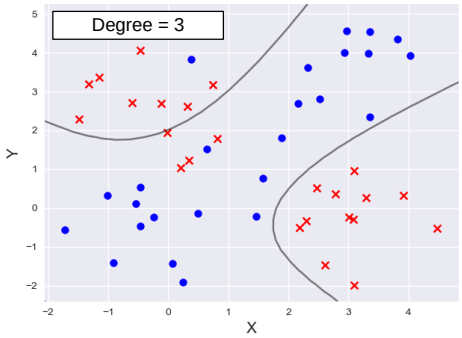
Can replace the inner product $\langle x, x_i \rangle$ with a *generalization* $K(x_i, x_{i'})$ called a **kernel**

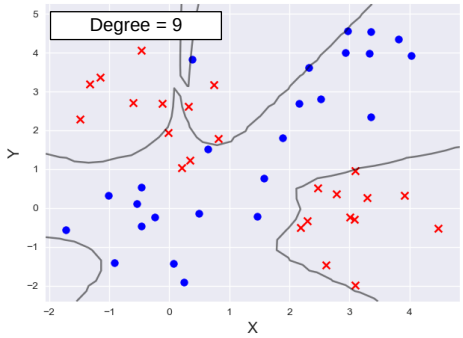
(c) Common kernel functions

- Linear: $K(x, x_i) = \sum_{j=1}^{p} x_j x_{ij}$
- Polynomial (of degree d): $K(x, x_i) = \left(1 + \sum_{j=1}^{p} x_j x_{ij}\right)^d$
- Radial: $K(x, x_i) = \exp\left(-\gamma \sum_{j=1}^{p} (x_j - x_{ij})^2\right)$

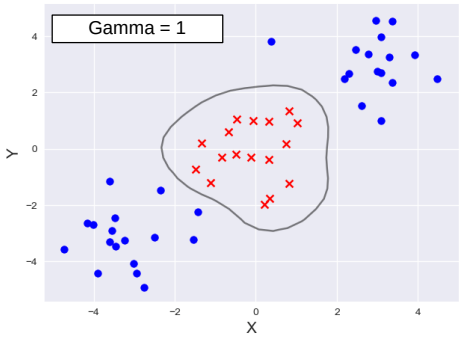
(d) Example

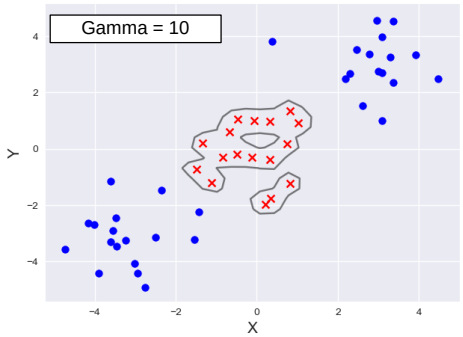
Polynomial kernel





Radial kernel





Instructor: Junho Song
junhosong@snu.ac.kr

(e) Pros and cons

Pros

- Effective in high dimensional spaces
- Can utilize different kernel function for various decision functions
- Add kernel function together to take into account even more complex behaviors

Cons

- Poor performance when $p > n$
- Do not directly provide probability estimates