R based Introduction to Classification as a Part of Bigdata Course

John Aa. Sørensen, lektor
Section for Information Technology, DTU Diplom

16 March 2017

A selection of examples on classification methods

The references [Kabacoff, 2015], [Meyer, 2015], and the R code

`6_R_Intro_Classification.R`

form the basis for the following examples:
Using Univ. of Wisconsin breast cancer data (11 variables, 699 observations) for designing a

- malign/benign logistic classifier (binary classifier) from R package: "core" and a
- decision tree classifier from R package: "rpart" and a
- conditional inference tree classifier from R package: "party" and a
- random forest classifier from R package: "randomForest" and a
- support vector machine (svm) from R package: "e1071".

Distance measures for heterogeneous data

Examples on distance measures on heterogeneous data (mixtures of ratio, interval, ordinal and nominal scales) in [Greenacre, 2013].

The application areas of these measures are e.g. the clustering and classification of data which might contain mixtures of

- ratio scaled data (e.g. temperature in Kelvin, not Celsius or weight)
- interval data from e.g. temperature or time observations
- ordinal data from e.g. Likert scales
- nominal data from e.g. counting the number of elements in a category (male/female).

Examples on Classification Preparation Steps

```
#
install.packages("rpart",lib=loc) # Classification trees
install.packages("rpart.plot",lib=loc)
install.packages("partykit",lib=loc)
install.packages("randomForest",lib=loc) # Random forest
install.packages("e1071",lib=loc) # SVM etc.
install.packages("ISLR",lib=loc) # [James, 2013]
install.packages("tree",lib=loc)
install.packages("MASS",lib=loc)
install.packages("rattle",lib=loc) # GUI for data mining, etc.
install.packages("kernlab",lib=loc) # SVM
install.packages("proxy",lib=loc)
install.packages("class",lib=loc) # k-nearest neighbor etc.
```

Loading Wisconsin dataset

```
#######################################################
# 6.1: Preparing data for classification examples.
#      p. 391 [Kabacoff, 2015]
loc <- "http://archive.ics.uci.edu/ml/
                  machine-learning-databases/"
ds  <- "breast-cancer-wisconsin/breast-cancer-wisconsin.data"
url <- paste(loc, ds, sep="")
#
breast <- read.table(url, sep=",", header=FALSE, na.strings="?")
str(breast)
names(breast) <- c("ID", "clumpThickness", "sizeUniformity",
    "shapeUniformity", "maginalAdhesion",
"singleEpithelialCellSize",
    "bareNuclei", "blandChromatin", "normalNucleoli", "mitosis",
    "class")
#
```

Dataset structure: 11 attributes $\times$ 699 observations

```
##############################################################
str(breast)
'data.frame': 699 obs. of  11 variables:
 $ V1 : int  1000025 1002945 1015425 1016277 1017023 1017122
 $ V2 : int  5 5 3 6 4 8 1 2 2 4 ...
 $ V3 : int  1 4 1 8 1 10 1 1 1 2 ...
 $ V4 : int  1 4 1 8 1 10 1 2 1 1 ...
 $ V5 : int  1 5 1 1 3 8 1 1 1 1 ...
 $ V6 : int  2 7 2 3 2 7 2 2 2 2 ...
 $ V7 : int  1 10 2 4 1 10 10 1 1 1 ...
 $ V8 : int  3 3 3 3 3 9 3 3 1 2 ...
 $ V9 : int  1 2 1 7 1 7 1 1 1 1 ...
 $ V10: int  1 1 1 1 1 1 1 1 5 1 ...
 $ V11: int  2 2 2 2 2 4 2 2 2 2 ...
```

Save dataset in file, clear dataset and reload dataset. Not needed - only for illustrating the operations

```
#####################################################
#
# Save dataset in file.
ls()           # List objects.
save(breast, file="dataset_breast")
rm(breast)     # Clear dataset.
ls()           # List objects.
# Reload the dataset using
load("dataset_breast")
ls()    # List objects.
#
```

Docum. from the Wisconsin dataset

```
6. Number of Attributes: 10 plus the class attribute

7. Attribute Information: (class attribute to last column)

   #  Attribute                     Domain
   -- -----------------------------------------
   1. Sample code number            id number
   2. Clump Thickness               1 - 10
   3. Uniformity of Cell Size       1 - 10
   4. Uniformity of Cell Shape      1 - 10
   5. Marginal Adhesion             1 - 10
   6. Single Epithelial Cell Size   1 - 10
   7. Bare Nuclei                   1 - 10
   8. Bland Chromatin               1 - 10
   9. Normal Nucleoli               1 - 10
  10. Mitoses                       1 - 10
  11. Class:                (2 for benign, 4 for malignant)

8. Missing attribute values: 16

   There are 16 instances in Groups 1 to 6 that contain
        a single missing (i.e., unavailable) attribute value,
        now denoted by "?".

9. Class distribution:

   Benign:    458 (65.5%)
   Malignant: 241 (34.5%)
```

Split the breast cancer data into training and test set

```
#
set.seed(1234)
#
# Use the sample() function for extracting rows for
#     training and validation.
# Use 70% of dataset for training.
train <- sample(nrow(df),0.7*nrow(df))
df.train <- df[train,]    # Form training set.
# Use the rest of dataset (30%) for validation.
df.validate <- df[-train,]
table(df.train$class)
table(df.validate$class)
#
```

Console output of train and validate tables sum to 699 observations

```
#####################################################
#
> df.train <- df[train,]      # Form training set.
> df.validate <- df[-train,] # Use the rest for validation.
> table(df.train$class)

   benign malignant
      329       160
> table(df.validate$class)

   benign malignant
      129        81
> #
```

Logistic regression

```
#####################################################
# 6.2: Logistic regression ex. p. 392 p. [Kabacoff, 2015].
#
# Logistic regression is for:
#     binary output ("benign", "malignant").
#
fit.logit <- glm(class~., data=df.train, family=binomial())
summary(fit.logit)
prob <- predict(fit.logit, df.validate, type="response")
logit.pred <- factor(prob > .5, levels=c(FALSE, TRUE),
                     labels=c("benign", "malignant"))
logit.perf <- table(df.validate$class, logit.pred,
                     dnn=c("Actual", "Predicted"))
logit.perf
```

Logistic regression, some console output

```
#####################################################
> summary(fit.logit)
Call:
glm(formula = class ~ ., family = binomial(), data = df.train)
Deviance Residuals:
   Min        1Q    Median        3Q       Max
-2.75813  -0.10602  -0.05679   0.01237   2.64317
Coefficients:
                  Estimate Std. Error z value Pr(>z—) —
(Intercept)      -10.42758    1.47602  -7.065 1.61e-12 ***
clumpThickness     0.52434    0.15950   3.287  0.00101 **
sizeUniformity    -0.04805    0.25706  -0.187  0.85171
shapeUniformity    0.42309    0.26775   1.580  0.11407
maginalAdhesion    0.29245    0.14690   1.991  0.04650 *
singleEpithel...   0.11053    0.17980   0.615  0.53871
bareNuclei         0.33570    0.10715   3.133  0.00173 **
blandChromatin     0.42353    0.20673   2.049  0.04049 *
normalNucleoli     0.28888    0.13995   2.064  0.03900 *
mitosis            0.69057    0.39829   1.734  0.08295 .
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
(Dispersion parameter for binomial family taken to be 1)
    Null deviance: 612.063  on 482  degrees of freedom
Residual deviance:  71.346  on 473  degrees of freedom
  (6 observations deleted due to missingness)
AIC: 91.346
Number of Fisher Scoring iterations: 8
```

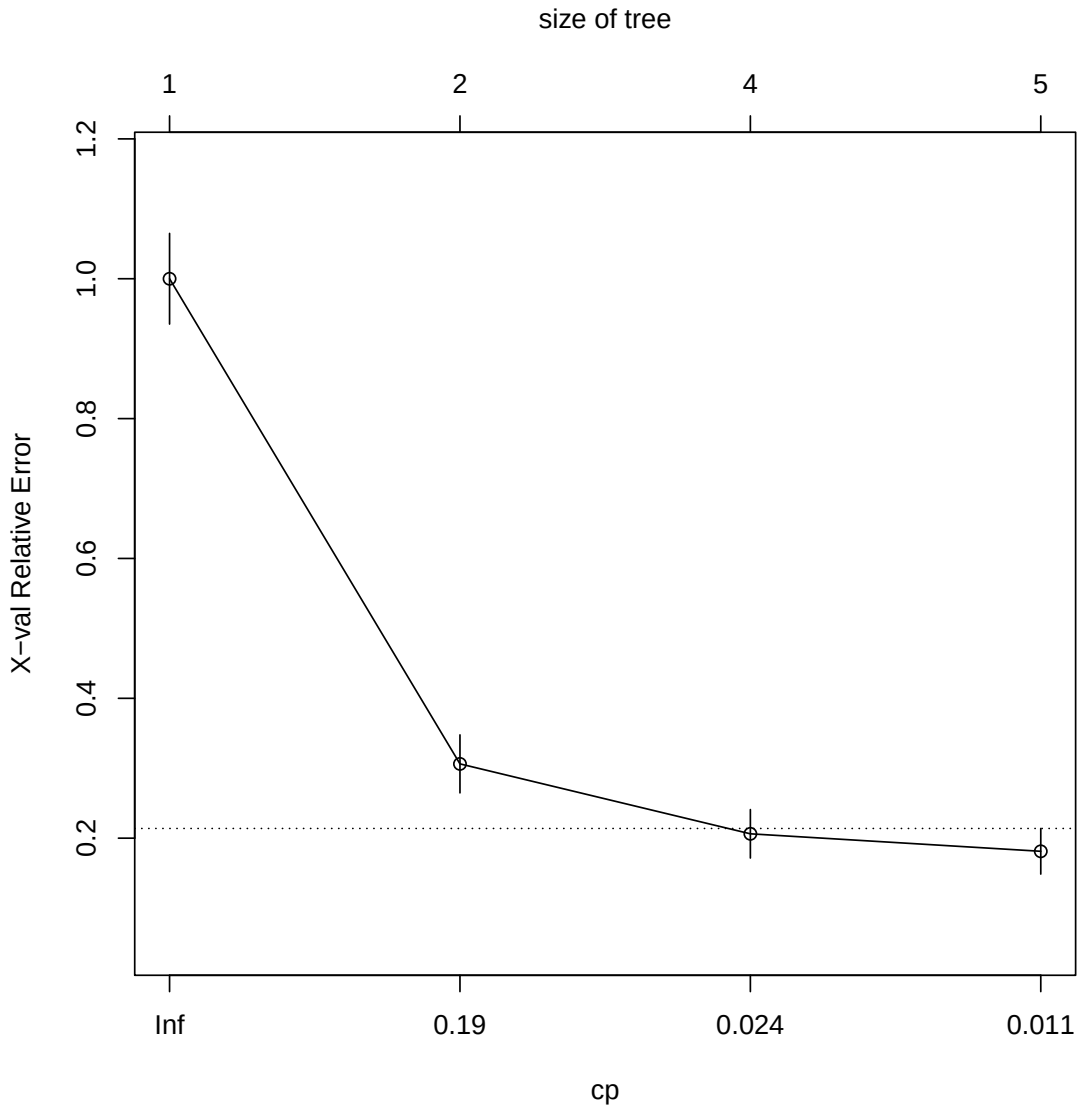
Logistic regression, performance

```
>
> prob <- predict(fit.logit, df.validate, type="response")
> logit.pred <- factor(prob > .5, levels=c(FALSE, TRUE),
+                      labels=c("benign", "malignant"))
> logit.perf <- table(df.validate$class, logit.pred,
+                     dnn=c("Actual", "Predicted"))
> logit.perf
           Predicted
Actual      benign malignant
  benign       118         2
  malignant      4        76
>
```

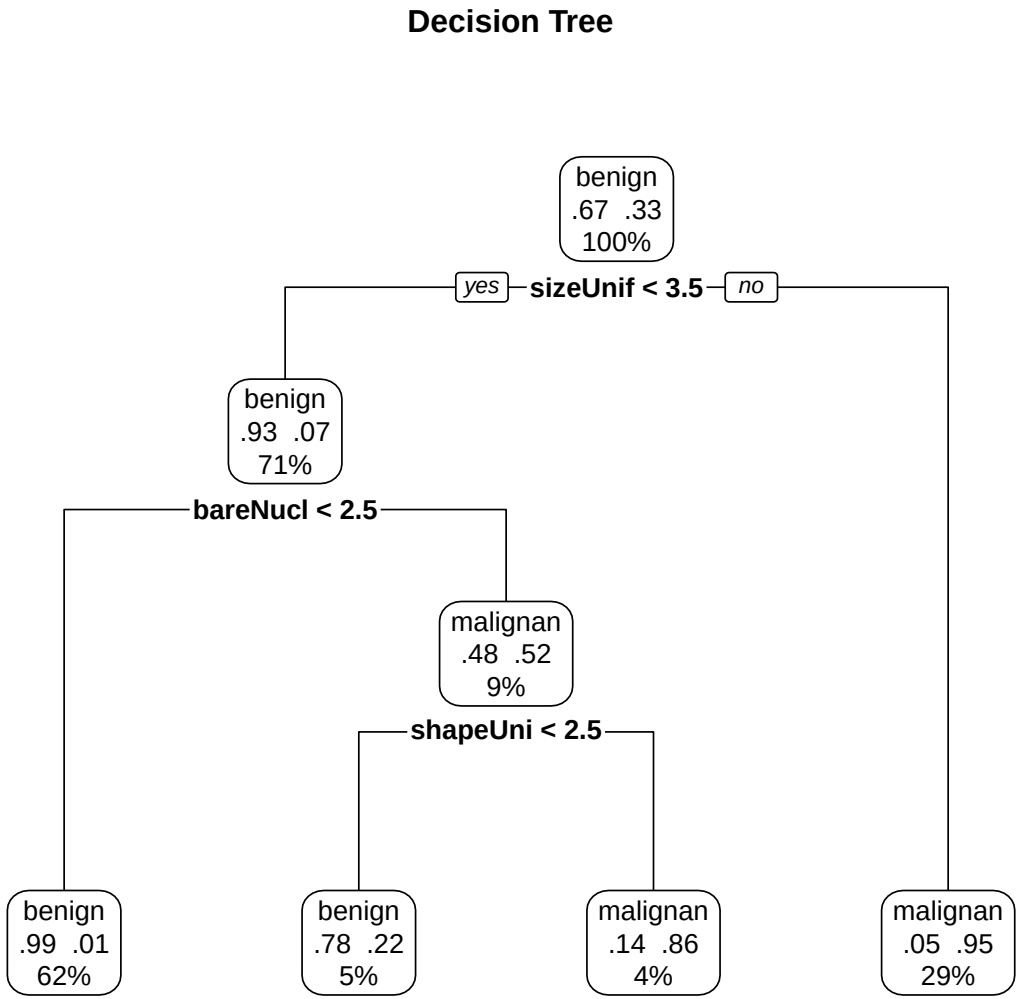
Creating and validating a decision tree

```
#############################################################
# 6.3: Creating a decision tree, p. 394 [Kabacoff, 2015].
#
# library(rpart)  moved to top of script.
#
set.seed(1234)
dtree <- rpart(class ~ ., data=df.train, method="class",
               parms=list(split="information"))
dtree$cptable
plotcp(dtree)  # Plot x-validated error vers. complexity cp.
#
pdf("fig_6_1_dectree.pdf") # Plot x-valid. error vs. complex.
par(opar)
plotcp(dtree)  # Plot x-validated error vers. complexity cp.
dev.off()
par(opar)
#
dtree.pruned <- prune(dtree, cp=.0125)  # prune the tree
#
```

Decision tree, cross validated error vs. complexity



Decision tree, breast cancer data set



Decision tree performance

```
#
>dtree.pred <- predict(dtree.pruned, df.validate, type="class")
>dtree.perf <- table(df.validate$class, dtree.pred,
+                   dnn=c("Actual", "Predicted"))
>dtree.perf
               Predicted
 Actual       benign malignant
  benign         122         7
  malignant        2        79
#
```

Random forest

```
############################################################
# 6.5: Random forest, p. 399 [Kabacoff, 2015].
#
#library("randomForest") moved to top of script.
#
>set.seed(1234)
>fit.forest <- randomForest(class~., data=df.train,
+                            na.action=na.roughfix,
+                            importance=TRUE)
>fit.forest

Call:
 randomForest(formula = class ~ ., data = df.train,
    importance = TRUE, na.action = na.roughfix)
    Type of random forest: classification
          Number of trees: 500
    No. of variables tried at each split: 3
    OOB estimate of  error rate: 3.68%
 Confusion matrix:
          benign malignant class.error
 benign       319        10  0.03039514
 malignant      8       152  0.05000000
```

Random forest performance

```
> importance(fit.forest, type=2)
                        MeanDecreaseGini
clumpThickness                 12.504484
sizeUniformity                 54.770143
shapeUniformity                48.662325
maginalAdhesion                 5.969580
singleEpithelialCellSize       14.297239
bareNuclei                     34.017599
blandChromatin                 16.243253
normalNucleoli                 26.337646
mitosis                         1.814502
> forest.pred <- predict(fit.forest, df.validate)
> forest.perf <- table(df.validate$class, forest.pred,
+                      dnn=c("Actual", "Predicted"))
> forest.perf
               Predicted
Actual       benign malignant
  benign        117         3
  malignant       1        79
>
```

SVM for linearly separated data, [James, 2013]
Support Vectors

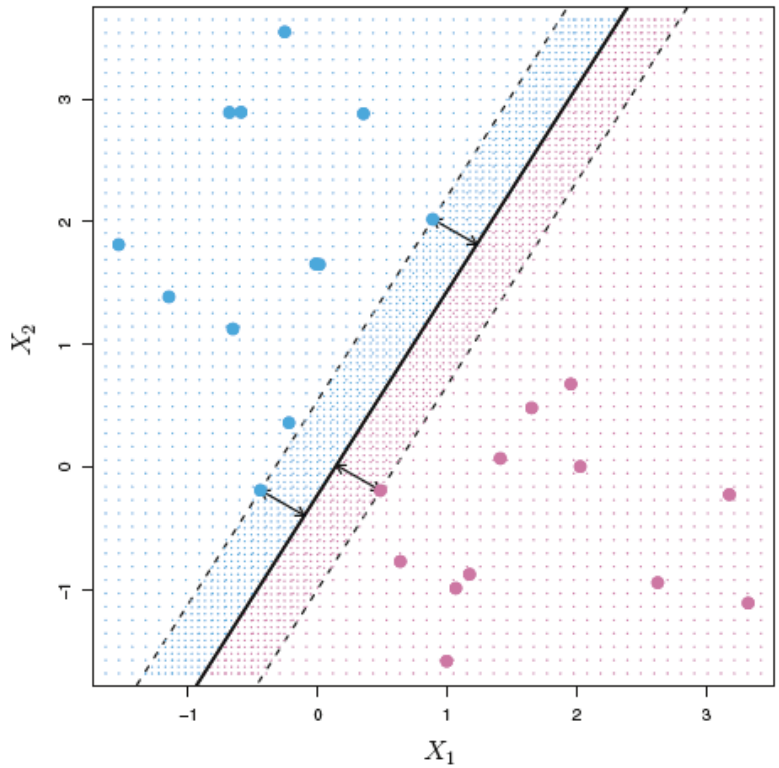


FIGURE 9.3. *There are two classes of observations, shown in blue and in purple. The maximal margin hyperplane is shown as a solid line. The margin is the distance from the solid line to either of the dashed lines. The two blue points and the purple point that lie on the dashed lines are the support vectors, and the distance from those points to the margin is indicated by arrows. The purple and blue grid indicates the decision rule made by a classifier based on this separating hyperplane.*

SVM for linearly separated data, [James, 2013]
Notice sensitivity of separting planes.

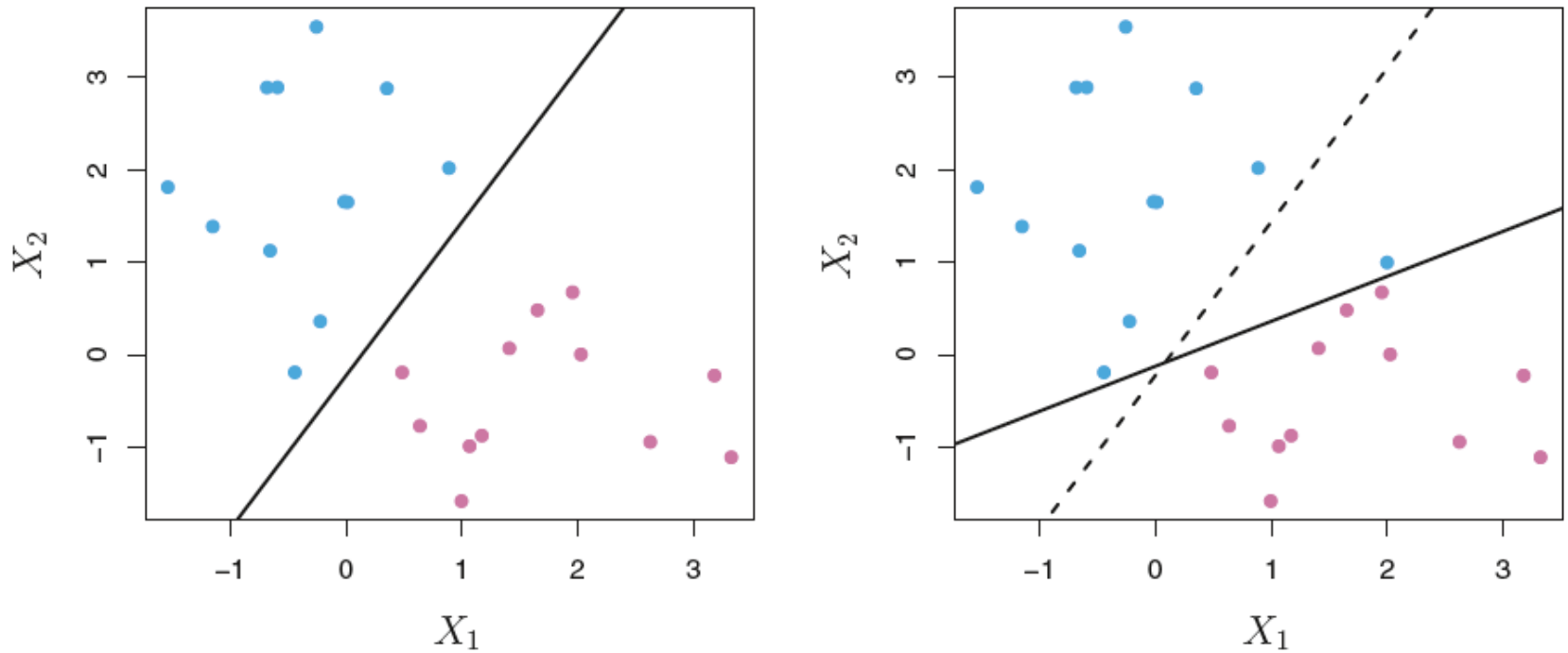


FIGURE 9.5. Left: *Two classes of observations are shown in blue and in purple, along with the maximal margin hyperplane.* Right: *An additional blue observation has been added, leading to a dramatic shift in the maximal margin hyperplane shown as a solid line. The dashed line indicates the maximal margin hyperplane that was obtained in the absence of this additional point.*

Soft boundaries [James, 2013]

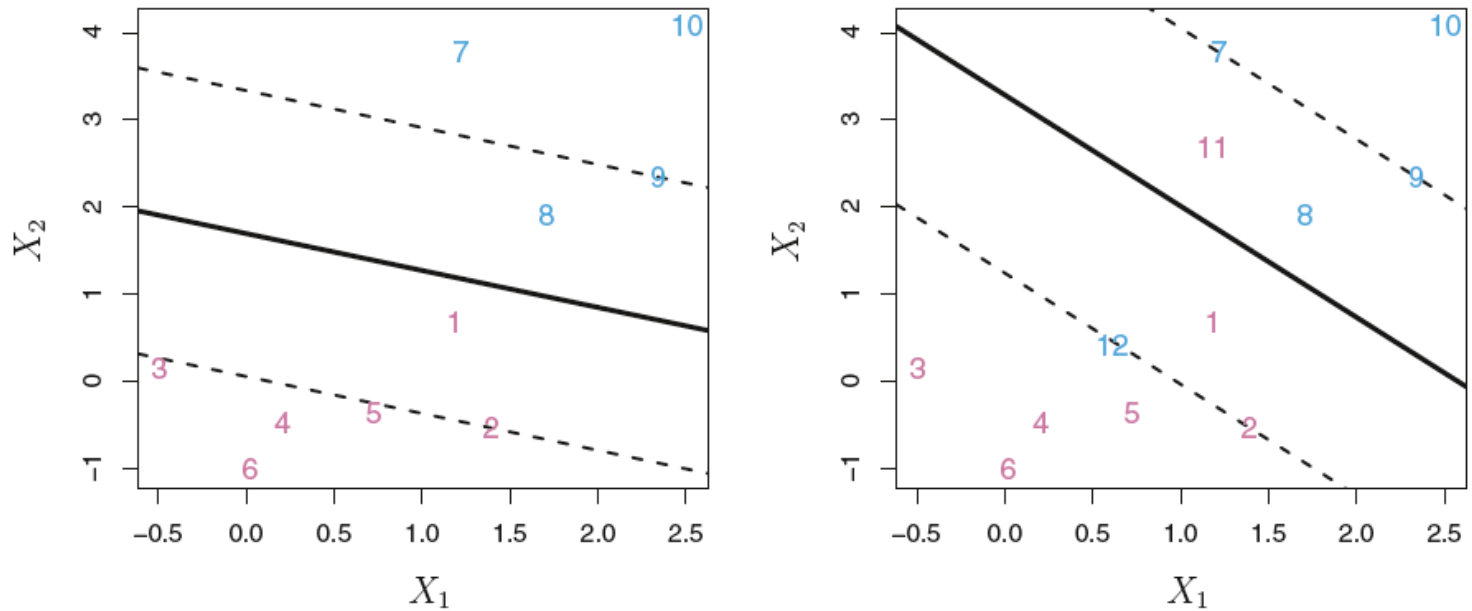


FIGURE 9.6. Left: *A support vector classifier was fit to a small data set. The hyperplane is shown as a solid line and the margins are shown as dashed lines.* Purple observations: *Observations* 3, 4, 5, *and* 6 *are on the correct side of the margin, observation* 2 *is on the margin, and observation 1 is on the wrong side of the margin.* Blue observations: *Observations* 7 *and* 10 *are on the correct side of the margin, observation* 9 *is on the margin, and observation* 8 *is on the wrong side of the margin. No observations are on the wrong side of the hyperplane.* Right: *Same as left panel with two additional points,* 11 *and* 12. *These two observations are on the wrong side of the hyperplane and the wrong side of the margin.*

SVM Optimization Problem [James, 2013] page 353

Use:
Coefficents on observations $\beta = (\beta_0 \ldots, \beta_p)$.
Positive margins $\epsilon = (\epsilon_0 \ldots, \epsilon_n)$
Tuning parameter C of observation being on the "wrong" side of hyperplane.
A high C corresponds to a high acceptance of beeing on the "wrong" side of hyperplane.
Total cost of all margins M.
Then

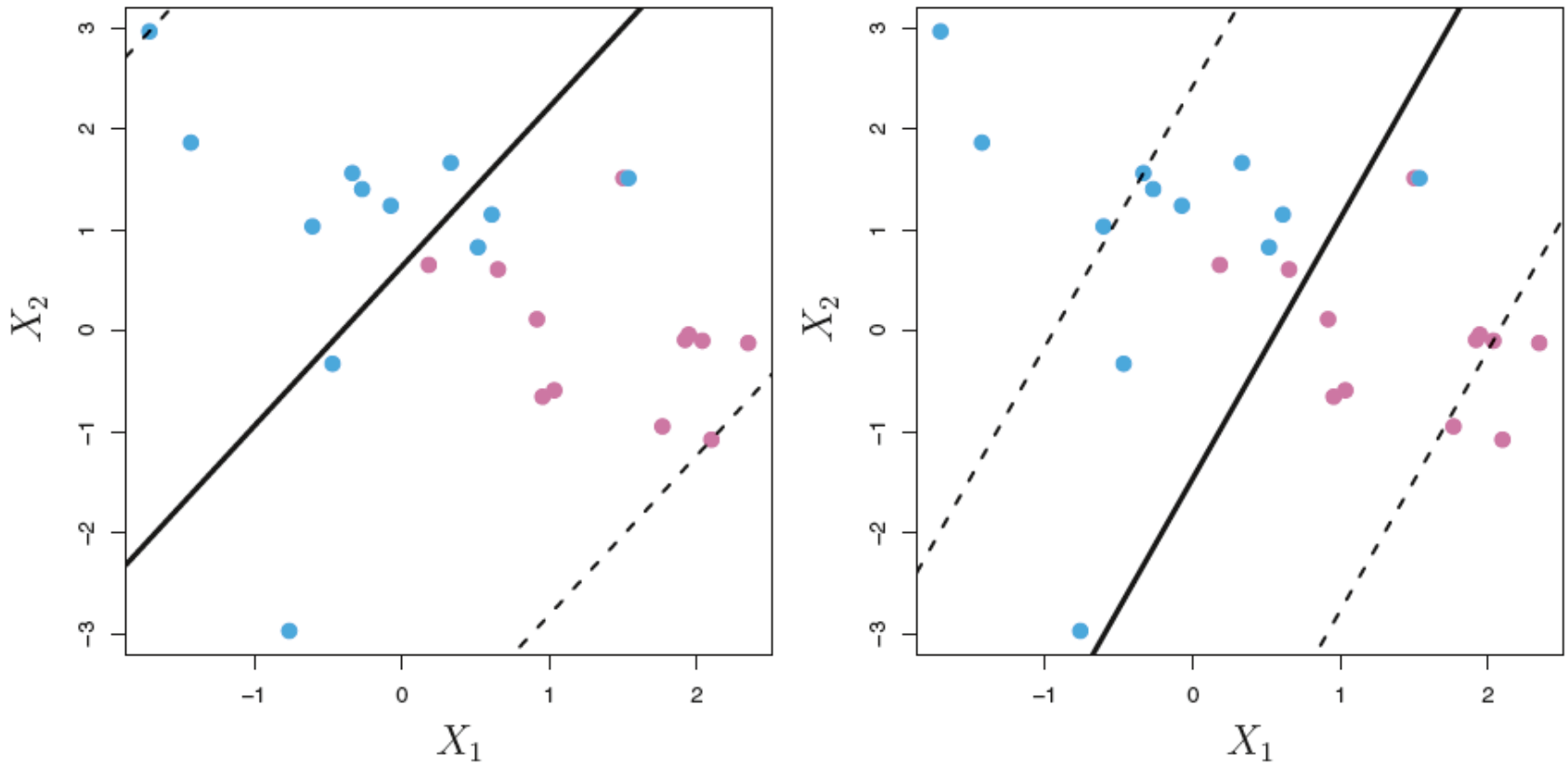
$$\text{maximize} \quad M(\beta, \epsilon)$$

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

$$y_i(\beta_0 + \beta_i x_{i,1} + \cdots + \beta_p x_{i,p}) \geq M(1 - \epsilon_i)$$

$$\epsilon_i \geq 0 \text{ and } \sum_{i=1}^{n} \epsilon_i \leq C \text{ high value, very tolerant}$$

Soft boundaries [James, 2013]



Soft boundaries [James, 2013]

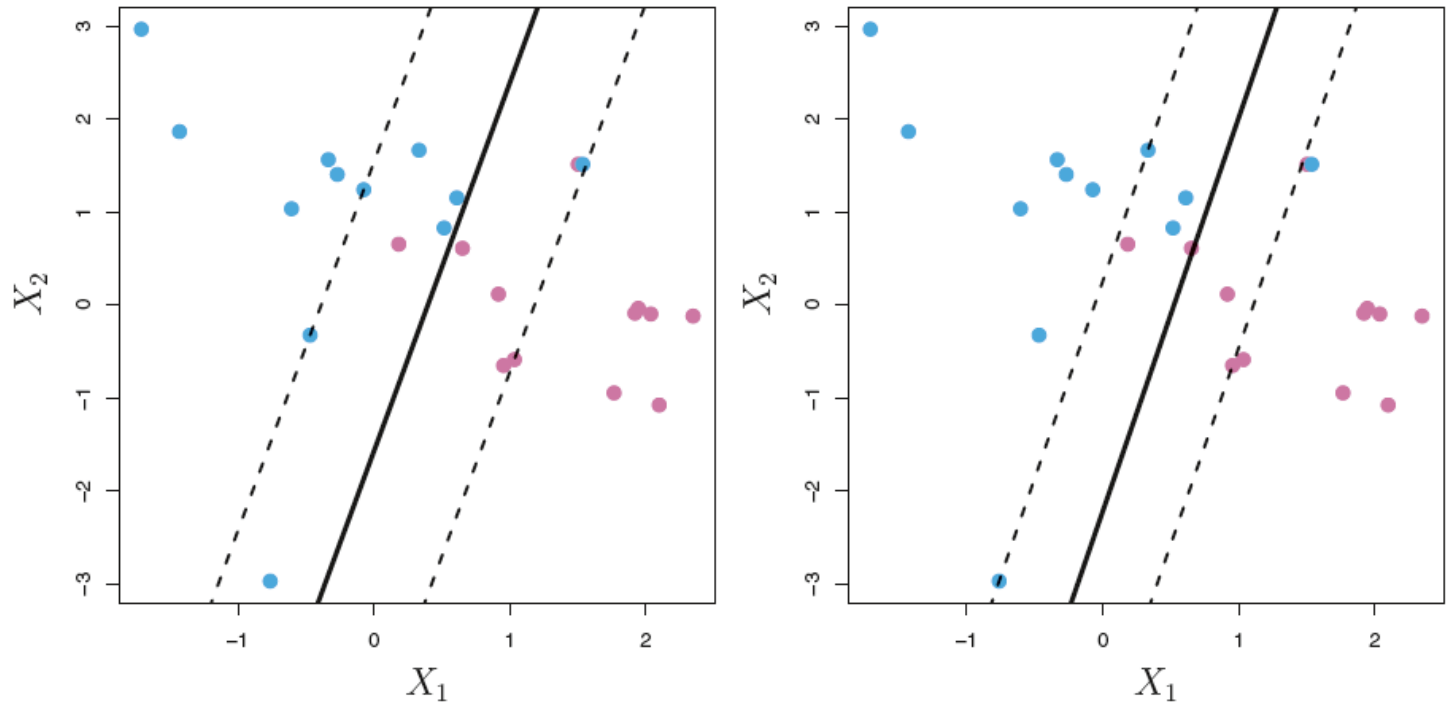


FIGURE 9.7. *A support vector classifier was fit using four different values of the tuning parameter C in (9.12)–(9.15). The largest value of C was used in the top left panel, and smaller values were used in the top right, bottom left, and bottom right panels. When C is large, then there is a high tolerance for observations being on the wrong side of the margin, and so the margin will be large. As C decreases, the tolerance for observations being on the wrong side of the margin decreases, and the margin narrows.*

Support Vector Machines

```
#####################################################
# 6.6.0: Support vector machines, p. 401 [Kabacoff, 2015]
#
# Prepare data, cf. [Kabacoff, 2015] p. 391
#
# library("e1071") moved to top of script.
#
loc <- "http://archive.ics.uci.edu/ml/
                        machine-learning-databases/"
ds  <- "breast-cancer-wisconsin/
                       breast-cancer-wisconsin.data"
url <- paste(loc, ds, sep="")
>
```

Support Vector Machines

```
> breast <- read.table(url, sep=",", header=FALSE,
                                        na.strings="?")
> names(breast) <- c("ID", "clumpThickness", "sizeUniformity",
+       "shapeUniformity", "maginalAdhesion",
+       "singleEpithelialCellSize", "bareNuclei",
+       "blandChromatin", "normalNucleoli", "mitosis", "class")
> #
df <- breast[-1]
df$class <- factor(df$class, levels=c(2,4),
           labels=c("benign", "malignant"))
```

Support Vector Machines

```
set.seed(1234)
 #
 # Use the sample() function for extracting rows
 #   for training and validation.
 # Use 70% of dataset for training.
 train <- sample(nrow(df), 0.7*nrow(df))
 df.train <- df[train,]    # Form training set.
 # Use the rest of dataset (30%) for validation.
 df.validate <- df[-train,] # Remove train obs.

 table(df.train$class)
   benign malignant
      329       160

> table(df.validate$class)
   benign malignant
      129        81
> #
```

Support Vector Machines

```
set.seed(1234)
fit.svm <- svm(class~., data=df.train)
fit.svm

Call:
svm(formula = class ~ ., data = df.train)

Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.1111111

Number of Support Vectors:  76

svm.pred <- predict(fit.svm, na.omit(df.validate))
svm.perf <- table(na.omit(df.validate)$class,
+                 svm.pred, dnn=c("Actual", "Predicted"))
```

Support Vector Machines

```
svm.perf
           Predicted
Actual      benign malignant
  benign       116         4
  malignant      3        77
```

Compare performance of ctree, dtree, forest & svm

From [Kabacoff, 2015] p. 405 Table 17.1

Sensitivity Probability of getting a positive classification, when the true outcome is positive. Also denoted "True positive rate".

Specificity Probability of getting a negative classification, when the true outcome is negative. Also denoted "True negative rate".

Positive predictive value Probability that an observation with a positive classification is correctly identified as positive. Also denoted "Precision" .

Negative predictive value Probability that an observation with a negative classification is correctly identified as negative.

Accuracy Proportions of observations correctly identified.

Compare performance of ctree, dtree, forest & svm

```
performance <- function(table, n=2){
  if(!all(dim(table) == c(2,2)))
    stop("Must be a 2 x 2 table")
  tn = table[1,1]
  fp = table[1,2]
  fn = table[2,1]
  tp = table[2,2]
  sensitivity = tp/(tp+fn)
  specificity = tn/(tn+fp)
  ppp = tp/(tp+fp)
  npp = tn/(tn+fn)
  hitrate = (tp+tn)/(tp+tn+fp+fn)
  result <- paste("Sensitivity = ",
     round(sensitivity, n) ,
   "\nSpecificity = ", round(specificity, n),
   "\nPositive Predictive Value = ", round(ppp, n),
   "\nNegative Predictive Value = ", round(npp, n),
   "\nAccuracy = ", round(hitrate, n), "\n", sep="")
    cat(result)
```

Compare performance of ctree, dtree, forest & svm, I

```
#
# Load performance of classifiers:
#
load("perf_dtree.perf")    # -> dtree.perf
load("perf_ctree.perf")    # -> ctree.perf
load("perf_forest.perf")   # -> forest.perf
load("perf_svm.perf")      # -> svm.perf

# List performance of the classifiers.
performance(dtree.perf)
performance(ctree.perf)
performance(forest.perf)
performance(svm.perf)
```

Compare performance of ctree, dtree, forest & svm, II

```
> performance(dtree.perf)
Sensitivity = 0.98  Probability of getting a positive
    classification, when the true outcome is positive.
Specificity = 0.95  Probability of getting a negative
    classification, when the true outcome is negative.
Positive Predictive Value = 0.92  Probability that an
    observation with a positive classification is correctly
    identified as positive.
Negative Predictive Value = 0.98  Probability that an
    observation with a negative classification is correctly
    identified as negative.
Accuracy = 0.96 Proportions of observations correctly identified
```

Compare performance of ctree, dtree, forest & svm, II

```
> performance(ctree.perf)
Sensitivity = 0.91 Probability of getting a positive
    classification, when the true outcome is positive.
Specificity = 0.96  Probability of getting a negative
    classification, when the true outcome is negative.
Positive Predictive Value = 0.94  Probability that an
    observation with a positive classification is correctly
    identified as positive.
Negative Predictive Value = 0.95  Probability that an
    observation with a negative classification is correctly
    identified as negative.
Accuracy = 0.94 Proportions of observations correctly identified
```

Compare performance of ctree, dtree, forest & svm, III

```
> performance(forest.perf)
Sensitivity = 0.99
Specificity = 0.98
Positive Predictive Value = 0.96
Negative Predictive Value = 0.99
Accuracy = 0.98

> performance(svm.perf)
Sensitivity = 0.96
Specificity = 0.97
Positive Predictive Value = 0.95
Negative Predictive Value = 0.97
Accuracy = 0.96
```

References I

Joseph Adler (2012)

R in a Nutshell

OReilly

Malika Charrad, Nadia Ghazzali, Vronique Boiteau, Azam Niknafs (2014)

NbClust: An R Package for Determining Relevant Number of Clusters in a Data Set

October 2014, Volume 61, Issue 6, Journal of Statistical Software

Michael Greenacre, Raul Primicerio (2013)

Multivariate Analysis of Ecological Data, Chap. 5 "Measures of distance between samples: non-Euclidean"

www.multivariatestatistics.org/publications.html

Gareth James, Daniela Witte, Trevor Hastie, Robert Tibshirani (2013)

An Introduction to Statistical learning with applications in R, Springer

http://www-bcf.usc.edu/ gareth/ISL/

Robert I. Kabacoff (2015)

R in Action

Manning Publications 2'Ed.

References II

David Meyer et al.

Package 'e1071'

cran.r-project.org/web/packages/e1071/index.html

Greet Pison, Anja Struyf, Peter J. Rousseeuw (1999)

Displaying a Clustering with CLUSPLOT.

Computational Statistics & Data Analysis 30 (1999) 381-392, Elsevier

R Core Team and contributors worldwide (2015)

The R Language Manual System

CRAN e.g. via RStudio

Tom Short, (2004)

Short Reference Card

CRAN cran.r-project.org/doc/contrib/Short-refcard.pdf

Paul Teetor

R Cookbook

O'Reilley

References III

Paul Torfs, Caludia Brauer

A (very) Short Introduction to R.

CRAN cran.r-project.org/doc/contrib/Torfs+Brauer-Short-R-Intro.pdf

Yanchang Zhao

R and Data Mining.

Elsevier 2013.

Yanchang Zhao

R Reference Card for Data Mining.

www.rdatamining.com

www.rdatamining.com/docs/r-reference-card-for-data-mining.pdf