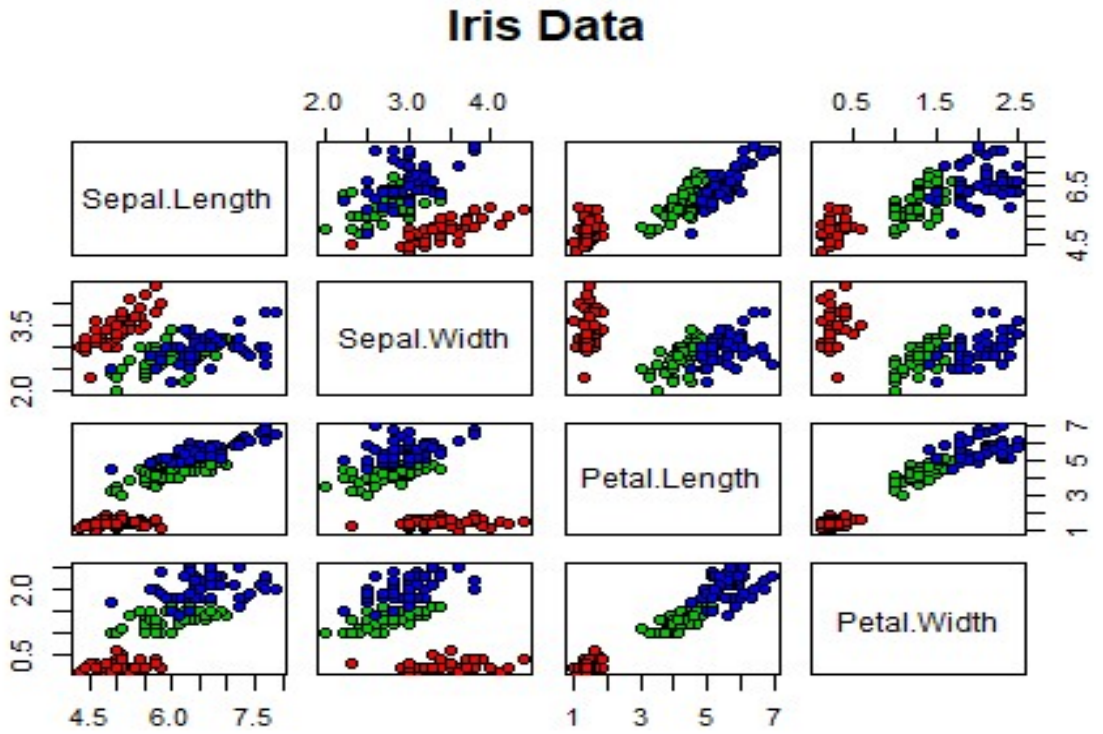
IE8990: Adv. Data Analytics for Complex Systems

- Lab 4: Classification
  - LDA & QDA
  - Logistic regression
  - SVM

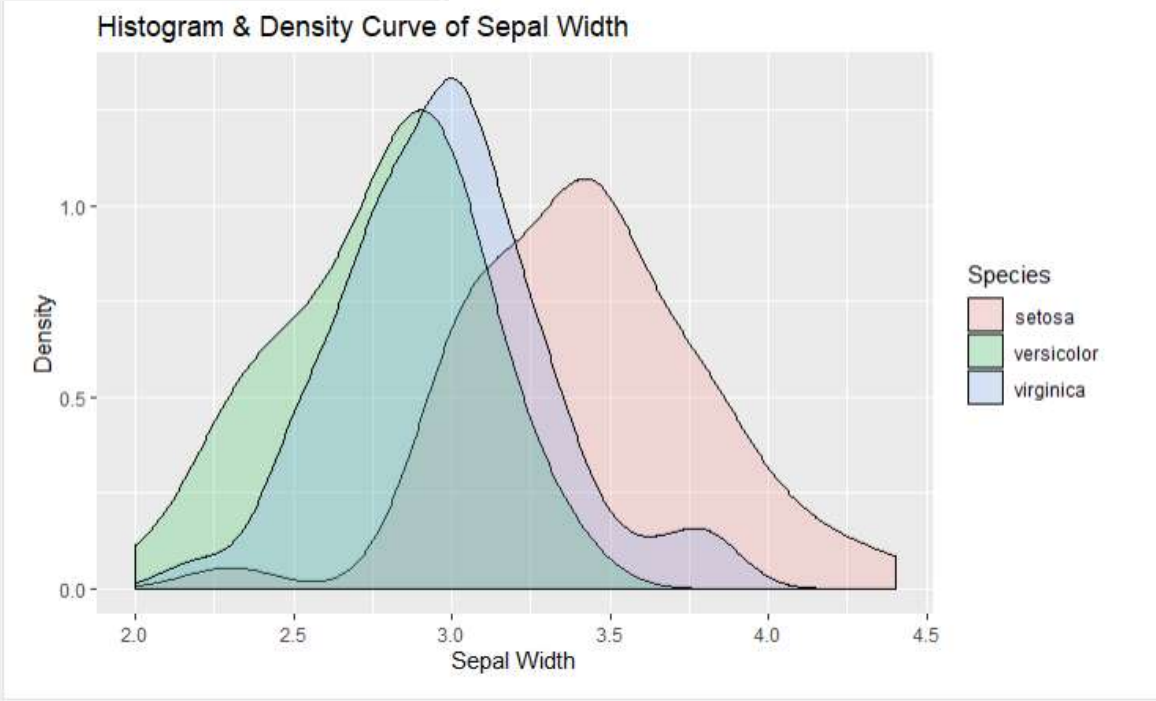
Iris Example 3: Classification

This time we try to classify the iris samples into different species using the given four inputs.



Explore functions in ggplot2 for data visualization

```
46  # try to use functions in ggplot2 to visualize data distribution
47  density2 <- ggplot(data=iris, aes(x=Sepal.Width, fill=Species))
48  density2 + geom_density(stat="density", alpha=I(0.2)) +
49    xlab("Sepal Width") +  ylab("Density") + ggtitle("Histogram &
    Density Curve of Sepal Width")
```



LDA Model

```
50 ```{r}
51 # fit lda model
52 model.lda<- lda(Species~.,data=train.data)
53 print(model.lda)
54 ```
55
56 #### 3.2 Use the testing set to evaluate the classification
   performance
57 ```{r}
58 predictions.lda = data.frame(predict(model.lda, test.data))
59 confusionM.lda<-confusionMatrix(predictions.lda$class,test.data$
   Species)
60 print(confusionM.lda$table)
61 ```
```

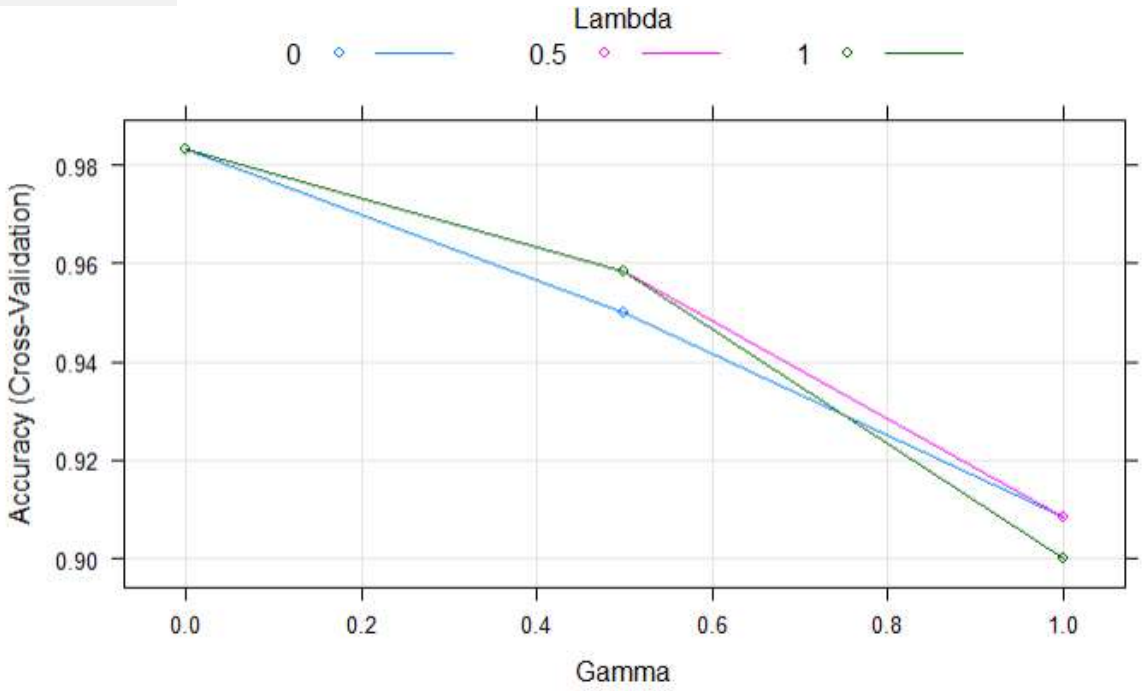
```
            Reference
Prediction   setosa versicolor virginica
  setosa         10          0         0
  versicolor      0         10         1
  virginica       0          0         9
```

QDA Model

```
63 #### 3.3 Fit a QDA model and use testing set to evaluate
   classification performance
64 ```{r}
65 model.qda<- qda(Species~.,data=train.data)
66 print(model.qda)
67 predictions.qda = data.frame(predict(model.qda, test.data))
68 confusionM.qda<-confusionMatrix(predictions.qda$class,test.data$
   Species)
69 print(confusionM.qda$table)
70 ```
```

```
            Reference
Prediction   setosa versicolor virginica
  setosa         10          0         0
  versicolor      0         10         1
  virginica       0          0         9
```

Regularized Discriminant Analysis

```
#### 3.4 Regularized discriminant analysis
```{r}
library(klaR)
cv_5_grid = trainControl(method = "cv", number = 5)
fit_rda_grid = train(Species ~ ., data = train.data,
method = "rda", trControl = cv_5_grid)

fit_rda_grid
plot(fit_rda_grid)
```
```



Logistic Regression

```
80
81 ### 4. Fit a logistic regression model to predict iris
   species
82 #### 4.1 Use the same training set in 3.1 to fit a
   logistic regression model
83 ```{r}
84 library(nnet)
85 # Fit the model
86 model.logReg <- nnet::multinom(Species ~., data =
   train.data)
87 # Summarize the model
88 summary(model.logReg)
89 # Make predictions
90 predicted.logReg <- model.logReg %>% predict(test.data)
91 head(predicted.logReg)
92 # Model accuracy
93 mean(predicted.logReg == test.data$Species)
94
95 confusionM.logReg<-confusionMatrix(predicted.logReg,
   test.data$Species)
96 ```
```

```
Call:
nnet::multinom(formula = Species ~ ., data = train.data)

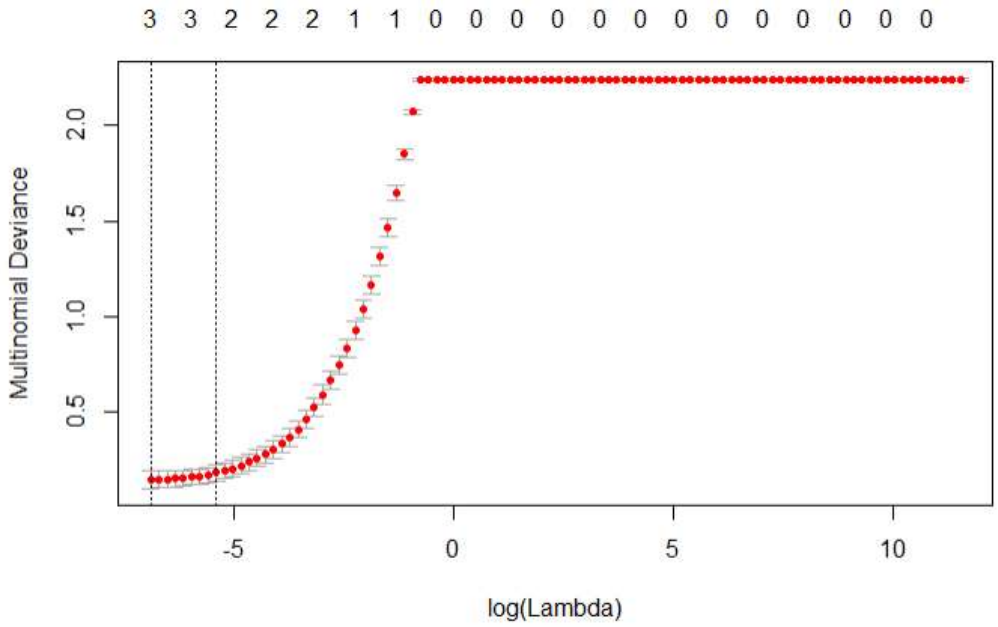
Coefficients:
           (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
versicolor    44.88584    -50.14588   -125.0783     191.7161    76.71824
virginica   -132.54466    -62.24794   -142.1385     224.9369   155.84279
```

Do you remember how to interpret your coefficients in logistics regression?

Regularized Logistic Regression

```
#### 4.2 Regularized logistic regression model

```{r}
library(glmnet)
library(fastDummies)
x<-train.data[,1:4] %>% as.matrix()

y_dummy<-dummy_columns(y)
y_dummy1<-y_dummy[,2:4] %>% as.matrix()

# Perform 10-fold cross-validation to select lambda
----------------------------
lambdas_to_try <- 10^seq(-3, 3, length.out = 100)
# Setting alpha = 1 implements lasso regression
lasso_cv <- cv.glmnet(x, y_dummy1, family="multinomial",
alpha =1, lambda = lambdas_to_try,
                      standardize = TRUE, nfolds = 10)
plot(lasso_cv)
#lasso_model<-glmnet(x, y_dummy1, family="multinomial",
alpha = 1,  standardize = TRUE)

#plot(lasso_model,xvar = "lambda")
#legend("bottomright", lwd = 1, col = 1:6, legend =
colnames(x), cex = .7)
```
```



HW 2: Q2

- Use data set `kyphosis` from R package "gam", randomly divide the data into training (75%) and testing (25%) set.

2.1 Fit a regularized discriminant analysis model, find the best tuning parameter combination, and evaluate the model using the testing set and report the confusion matrix

2.2 Fit a logistic regression model, and interpret the coefficients you estimated

2.3 Fit a regularized logistic regression model, and find the best model from your regularization

Logistic Regression – Model Interpretation

```
80
81 ### 4. Fit a logistic regression model to predict iris
   species
82 #### 4.1 Use the same training set in 3.1 to fit a
   logistic regression model
83 ```{r}
84 library(nnet)
85 # Fit the model
86 model.logReg <- nnet::multinom(Species ~., data =
   train.data)
87 # Summarize the model
88 summary(model.logReg)
89 # Make predictions
90 predicted.logReg <- model.logReg %>% predict(test.data)
91 head(predicted.logReg)
92 # Model accuracy
93 mean(predicted.logReg == test.data$Species)
94
95 confusionM.logReg<-confusionMatrix(predicted.logReg,
   test.data$Species)
96 ```
```

```
Call:
nnet::multinom(formula = Species ~ ., data = train.data)

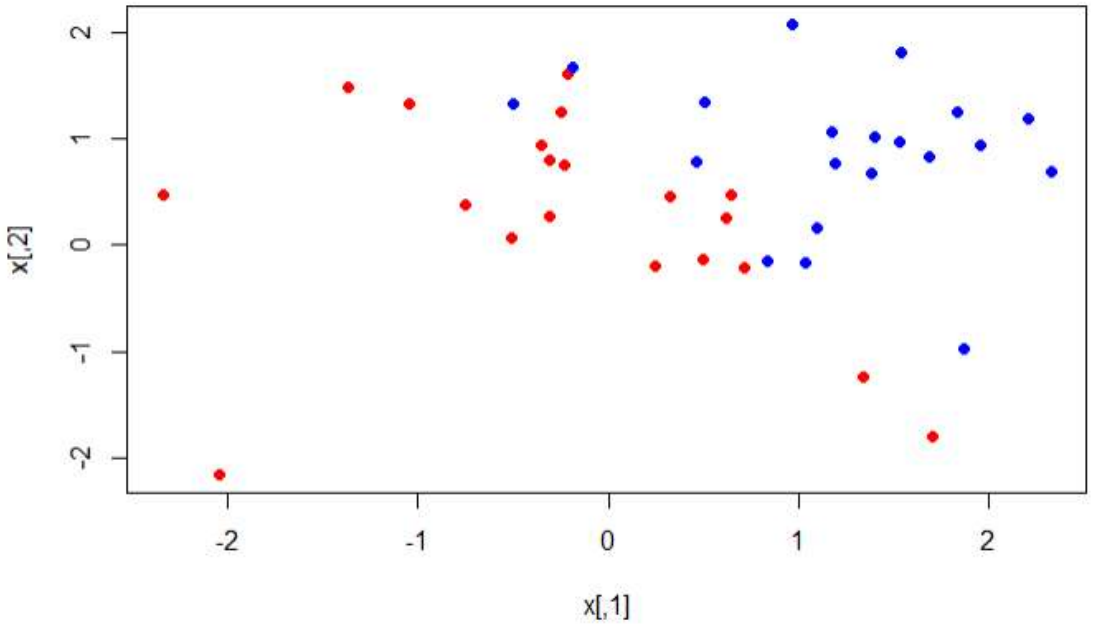
Coefficients:
           (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
versicolor    44.88584    -50.14588   -125.0783     191.7161    76.71824
virginica   -132.54466    -62.24794   -142.1385     224.9369   155.84279
```

Linear SVM Example

- Random data generation – will be involved in the exam

```r
```{r}
set.seed(10111)
x = matrix(rnorm(80), 40, 2)
y = rep(c(-1, 1), c(20, 20))
x[y == 1,] = x[y == 1,] + 1
plot(x, col = y + 3, pch = 19)

set.seed(123)
xt = matrix(rnorm(80), 40, 2)
yt = rep(c(-1, 1), c(20, 20))
xt[yt == 1,] = xt[yt == 1,] + 1
```
```



Linear SVM Example

```r
```{r}
library(e1071)
library(caret)
library(dplyr)
library(tidyverse)
dat = data.frame(x, y = as.factor(y))
dat.test=data.frame(xt, y = as.factor(yt))

svmfit = svm(y ~ ., data = dat, kernel = "linear", cost = 10, scale = FALSE)
print(svmfit)
dat.test.svm<-predict(svmfit,data=dat.test)
ConfusionM.svm<-confusionMatrix(dat.test.svm,dat.test$y)
print(ConfusionM.svm)
```
```

```
          Reference
Prediction -1  1
        -1 19  2
         1  1 18
```

- kernel = “linear”
- “cost” is the C in svm formulation

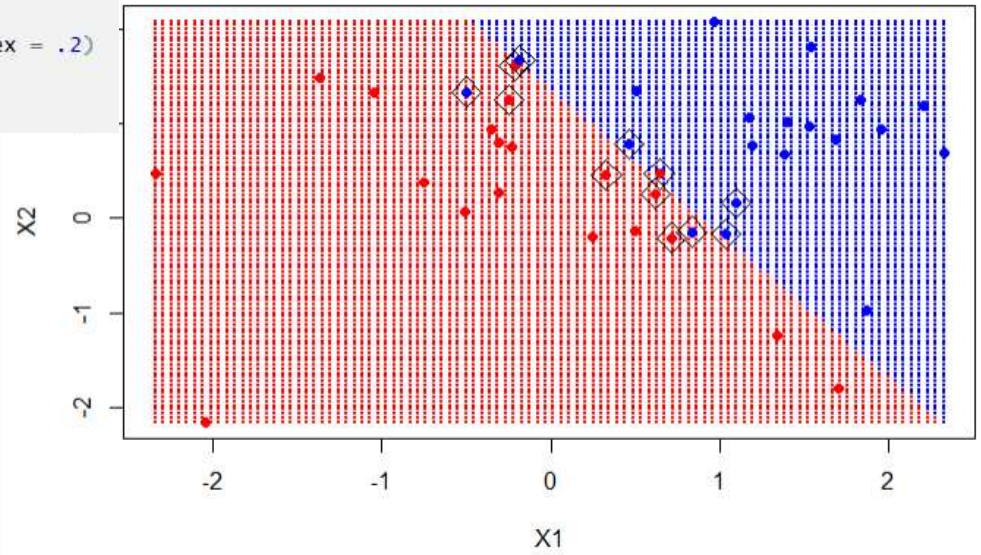
```
               Sensitivity : 0.9500
               Specificity : 0.9000
            Pos Pred Value : 0.9048
            Neg Pred Value : 0.9474
                Prevalence : 0.5000
            Detection Rate : 0.4750
      Detection Prevalence : 0.5250
         Balanced Accuracy : 0.9250

       'Positive' Class : -1
```

Linear SVM Example

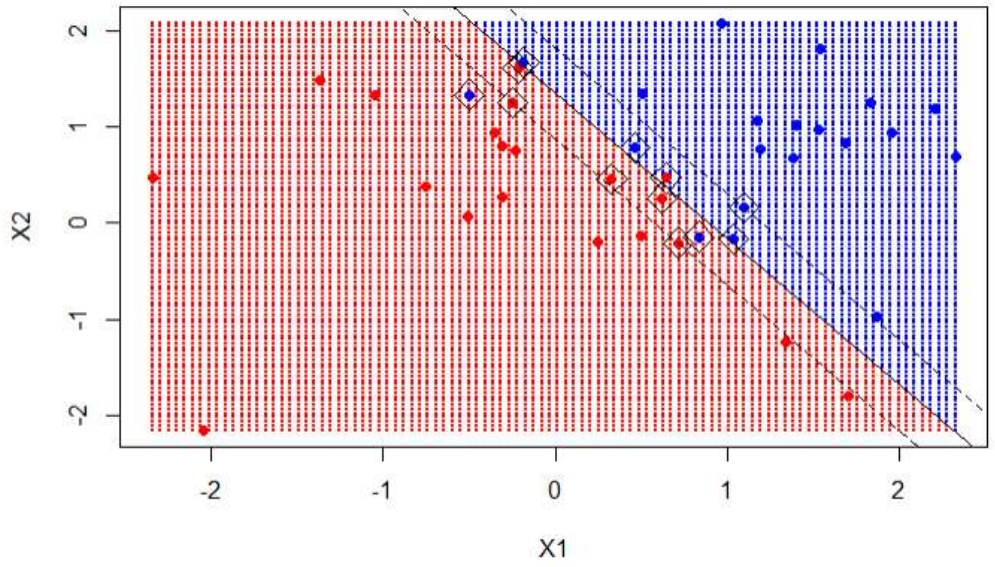
- Create a function named “make.grid”

```r
```{r}
make.grid = function(x, n = 100) {
  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)
```
```



Linear SVM Example

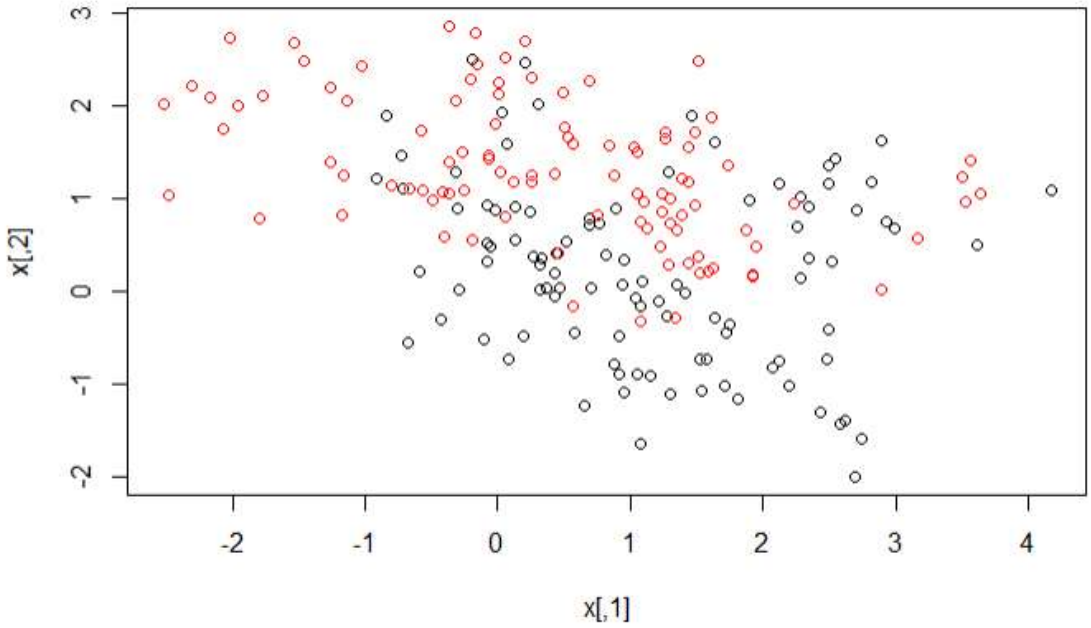
```r
beta = drop(t(svmfit$coefs)%*%x[svmfit$index,])
beta0 = svmfit$rho
plot(xgrid, col = c("red", "blue")[as.numeric(ygrid)], pch = 20, cex = .2)
points(x, col = y + 3, pch = 19)
points(x[svmfit$index,], pch = 5, cex = 2)
abline(beta0 / beta[2], -beta[1] / beta[2])
abline((beta0 - 1) / beta[2], -beta[1]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



Nonlinear SVM

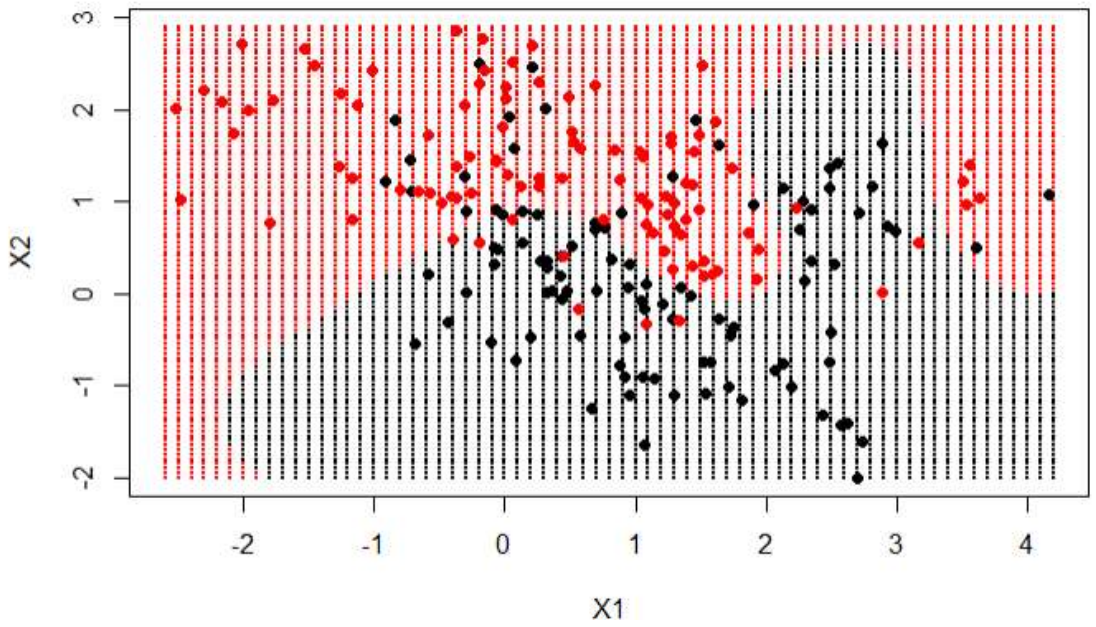
- Example from the textbook

```r
load(file = "ESL.mixture.rda")
names(ESL.mixture)
rm(x, y)
attach(ESL.mixture)
plot(x, col = y + 1)
```

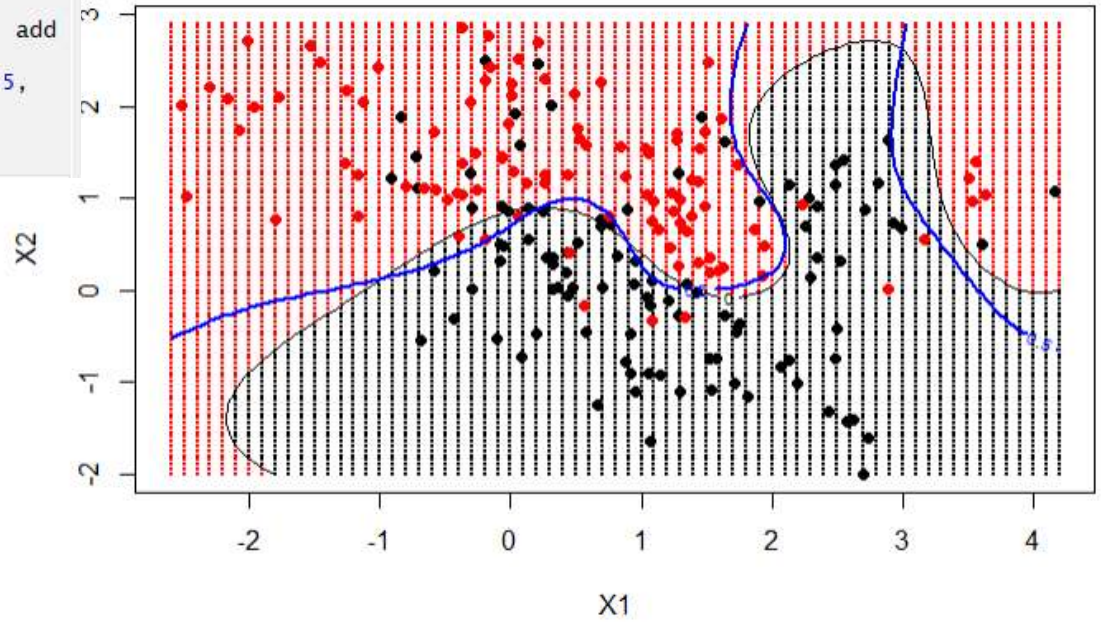


Nonlinear SVM

```r
dat = data.frame(y = factor(y), x)
fit = svm(factor(y) ~ ., data = dat, scale = FALSE, kernel = "radial", cost = 5)
xgrid = expand.grid(X1 = px1, X2 = px2)
ygrid = predict(fit, xgrid)

plot(xgrid, col = as.numeric(ygrid), pch = 20, cex = .2)
points(x, col = y + 1, pch = 19)
```



Nonlinear SVM

```r
```{r}
func = predict(fit, xgrid, decision.values = TRUE)
func = attributes(func)$decision

xgrid = expand.grid(X1 = px1, X2 = px2)
ygrid = predict(fit, xgrid)
plot(xgrid, col = as.numeric(ygrid), pch = 20, cex = .2)
points(x, col = y + 1, pch = 19)

contour(px1, px2, matrix(func, 69, 99), level = 0, add = TRUE)
contour(px1, px2, matrix(prob, 69, 99), level = 0.5, add = TRUE, col = "blue", lwd = 2)

```
```



Naïve Bayes Classifier Demonstration

```
### Naive Bayes Classifier Demonstration
```{r}
NBclassfier=naiveBayes(y~., data=dat)
print(NBclassfier)
dat.test.NB=predict(NBclassfier, newdata = dat.test, type = "class")
confusionM.svm<-confusionMatrix(dat.test.NB,dat.test$y)
print(confusionM.svm)
```
```

```
Confusion Matrix and Statistics

          Reference
Prediction -1  1
        -1 17  7
        1   3 13


               Sensitivity : 0.8500
               Specificity : 0.6500
            Pos Pred Value : 0.7083
            Neg Pred Value : 0.8125
                Prevalence : 0.5000
            Detection Rate : 0.4250
      Detection Prevalence : 0.6000
         Balanced Accuracy : 0.7500

          'Positive' Class : -1
```