BIOS362: Lab 1

==========

Lecturer: Guanhua Chen, Date: 01/09/2015

Machine learning focuses on outcome prediction and data exploration: unsupervised learning (without label) and supervised learning (with label). To cover topics in the lab as follows (order is not fixed):

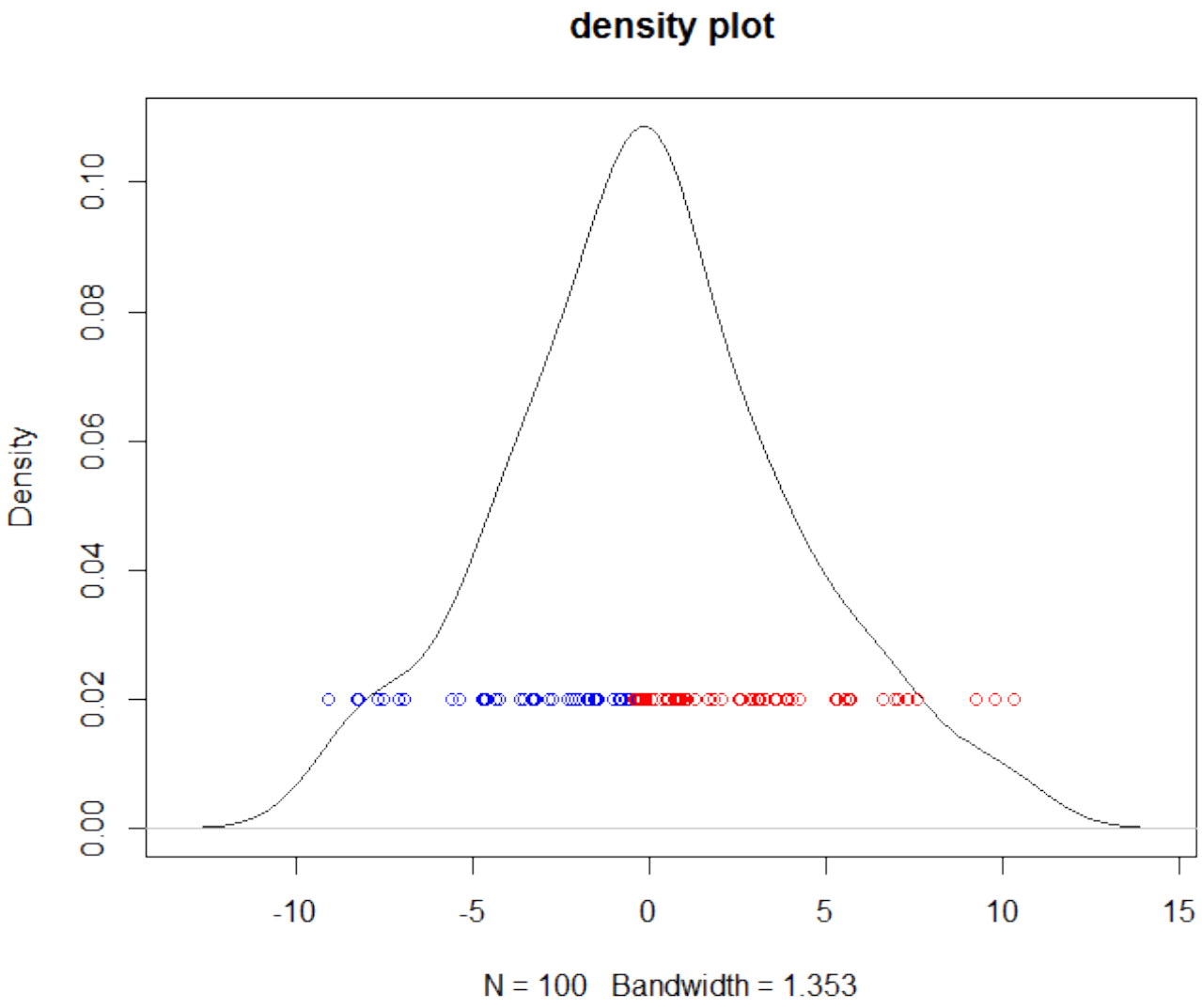
1. Penalized regression.
2. Cross-validation and bootstrap.
3. Convex optimization, MM algorithm.
4. Clustering and related topics.
5. Valid post selection inference.
6. Supporting vector machine.
7. PCA,CCA.
8. LDA, QDA, Variable screening
9. Multiple testing
10. Sliced inverse regression, dimensional reduction
11. Generalized measure of association.
12. Graphical Model, community detection.
13. Boosting, Ensemble method.
14. Kernel based density estimation or regression.
15. Neural network and Deep Learning

A. Motivating example: Significance of a clustering:

```
set.seed(100)
library(sigclust)
x <- rnorm(100,0,4)
result <- kmeans(x,2)
pvalue1 <- t.test(x[result$cluster==1],x[result$cluster==2])
pvalue2 <- sigclust(data.frame(x),nsim=1000,icovest=3)
```

X is 100 observations from a Gaussian distribution, we groups x by k-means (k=2) clustering. We want to access the significance of the clustering output. The p-value from the t-test for comparing the two outcomes from K-means is = 4.510^{-21}. The p-value from sigclust is 0.922. The density plot is as follows:

```
plot(density(x),main="density plot")
color = c("red","blue")
points(x,rep(0.02,100),col=color[result$cluster])
```

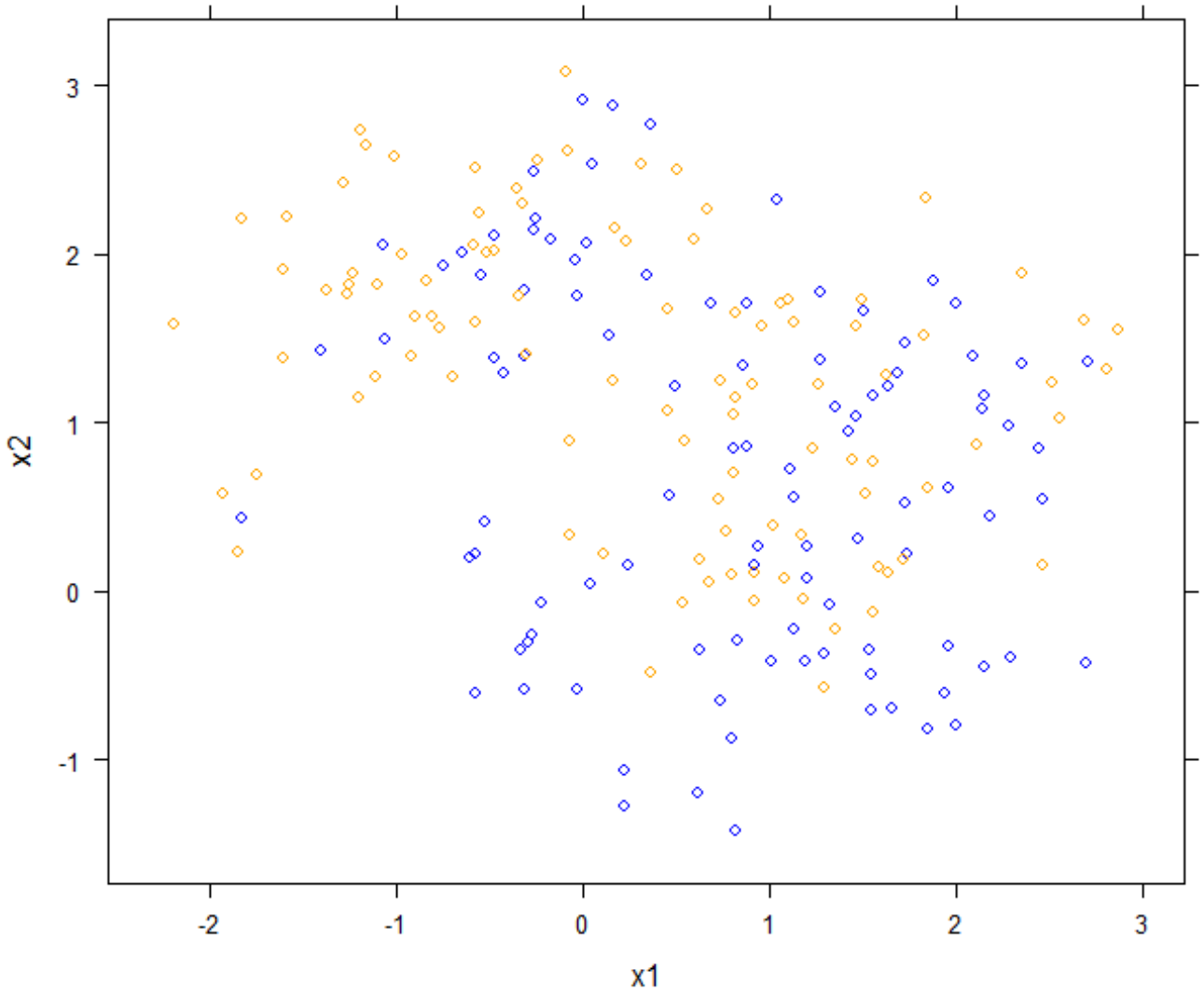


Take home message: do not be exited too soon if the output of a machine learning method is 'significant'.

B. Reproduce figure 2.2 (code is from http://stats.stackexchange.com/questions/21572/how-to-plot-decision-boundary-of-a-k-nearest-neighbor-classifier-from-elements-o)

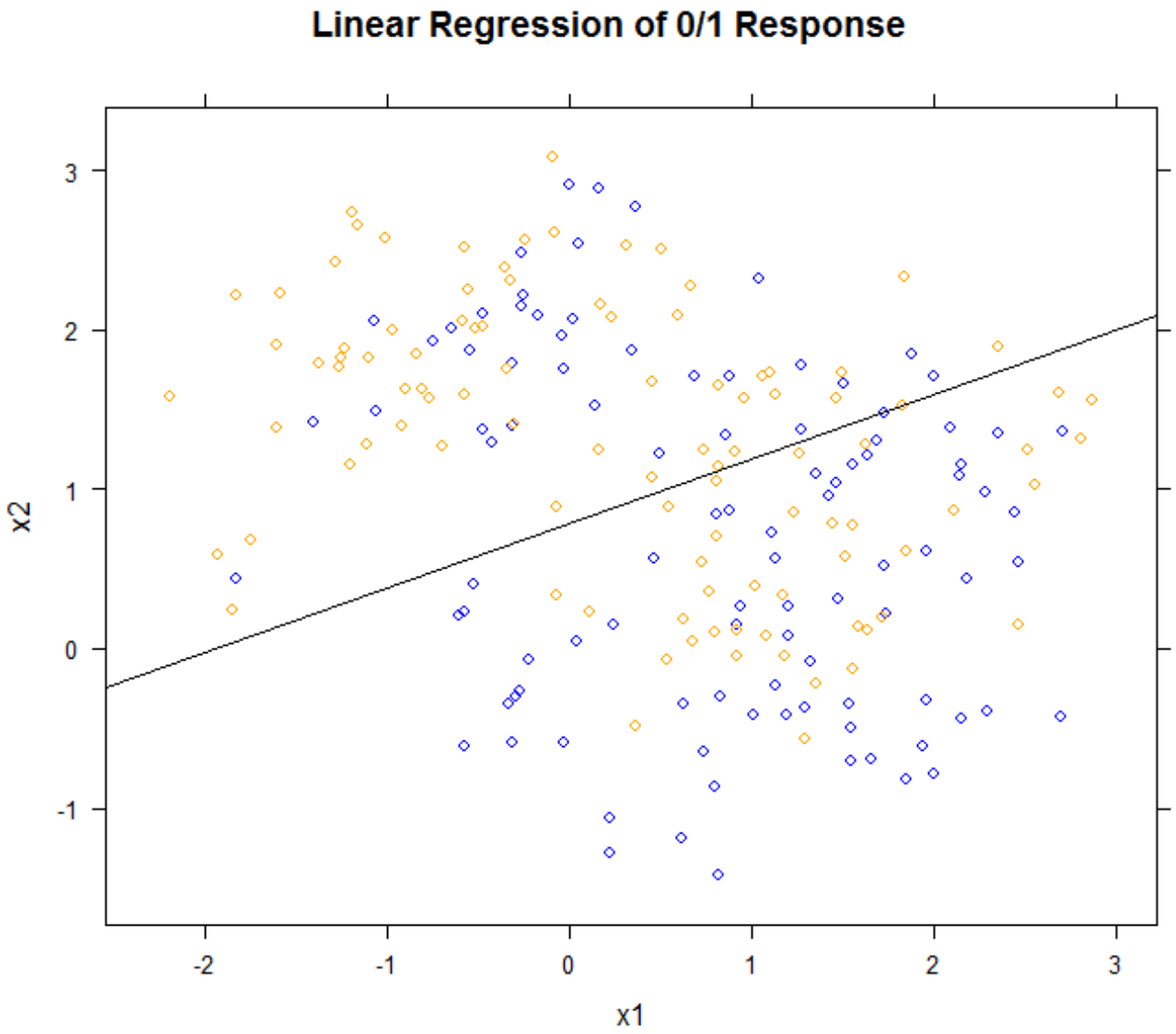
```
library(MASS)
# set the seed to reproduce data generation in the future
seed <- 123456
set.seed(seed)

# generate two classes means
Sigma <- matrix(c(1,0,0,1),nrow = 2, ncol = 2)
means_1 <- mvrnorm(n = 10, mu = c(1,0), Sigma)
```

```r
means_2 <- mvrnorm(n = 10, mu = c(0,1), Sigma)

# pick an m_k at random with probability 1/10
# function to generate observations
genObs <- function(classMean, classSigma, size, ...)
{
  # check input
  if(!is.matrix(classMean)) stop("classMean should be a matrix")
  nc <- ncol(classMean)
  nr <- nrow(classMean)
  if(nc != 2) stop("classMean should be a matrix with 2 columns")
  if(ncol(classSigma) != 2) stop("the dimension of classSigma is
wrong")

  # mean for each obs
    # pick an m_k at random
  meanObs <- classMean[sample(1:nr, size = size, replace = TRUE),]
  obs <- t(apply(meanObs, 1, function(x) mvrnorm(n = 1, mu = x, Sigma =
classSigma )) )
  colnames(obs) <- c('x1','x2')
  return(obs)
}


obs100_1 <- genObs(classMean = means_1, classSigma = Sigma/5, size =
100)
obs100_2 <- genObs(classMean = means_2, classSigma = Sigma/5, size =
100)

# generate label
y <- rep(c(0,1), each = 100)

# training data matrix
trainMat <- as.data.frame(cbind(y, rbind(obs100_1, obs100_2)))

# plot them
library(lattice)
with(trainMat, xyplot(x2 ~ x1,groups = y, col=c('blue', 'orange')))
```




```
# now fit two models

# model 1: linear regression
lmfits <- lm(y ~ x1 + x2 , data = trainMat)

# get the slope and intercept for the decision boundary
intercept <- -(lmfits$coef[1] - 0.5) / lmfits$coef[3]
slope <- - lmfits$coef[2] / lmfits$coef[3]

# Figure 2.1
xyplot(x2 ~ x1, groups = y, col = c('blue', 'orange'), data = trainMat,
       panel = function(...)
       {
        panel.xyplot(...)
        panel.abline(intercept, slope)
```

```
        },
       main = 'Linear Regression of 0/1 Response')
```




```
# model2: k nearest-neighbor methods
library(class)
# get the range of x1 and x2
rx1 <- range(trainMat$x1)
    rx2 <- range(trainMat$x2)
# get lattice points in predictor space
px1 <- seq(from = -1.6, to = 4.0, by = 0.1 )
px2 <- seq(from = -2, to = 3.2, by = 0.1 )
xnew <- expand.grid(x1 = px1, x2 = px2)

# get the contour map
knn15 <- knn(train = trainMat[,2:3], test = xnew, cl = trainMat[,1], k
```

```
= 15, prob = TRUE)
prob <- attr(knn15, "prob")
prob <- ifelse(knn15=="1", prob, 1-prob)
prob15 <- matrix(prob, nrow = length(px1), ncol = length(px2))

# Figure 2.2
par(mar = rep(2,4))
contour(px1, px2, prob15, levels=0.5, labels="", xlab="", ylab="",
main=
    "15-nearest neighbour", axes=FALSE)
points(trainMat[,2:3], col=ifelse(trainMat[,1]==1, "coral",
"cornflowerblue"))
points(xnew, pch=".", cex=1.2, col=ifelse(prob15>0.5, "coral",
"cornflowerblue"))
box()
```