Homework 1

Yue Guo

January 22, 2020

Rewrite code with lm()

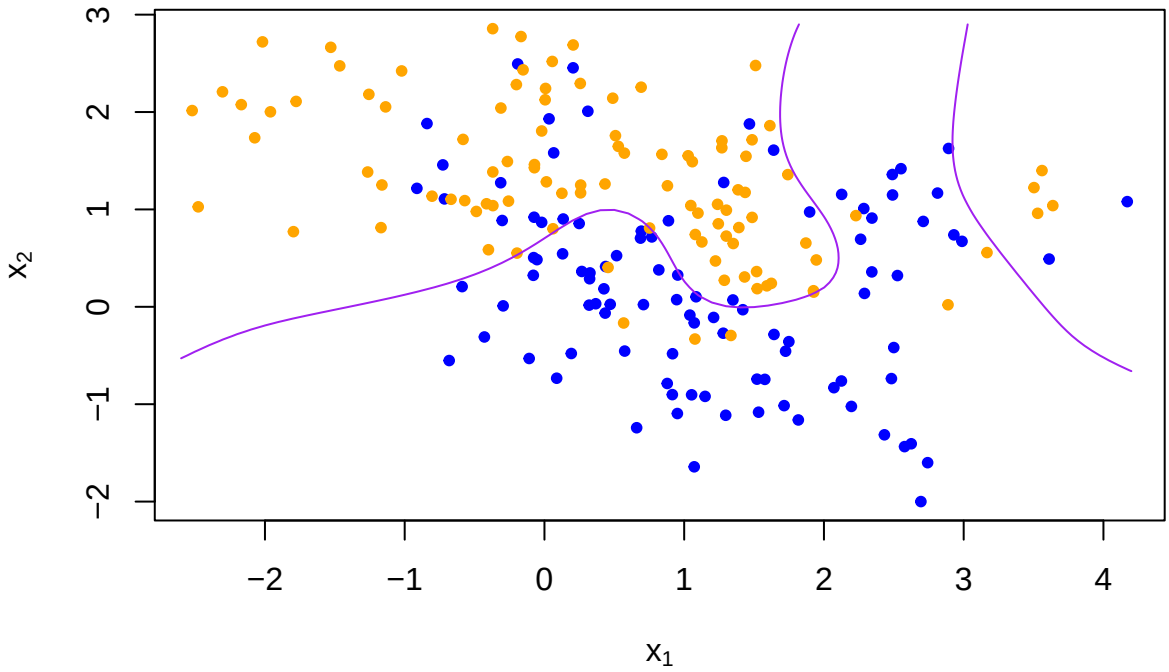
```
library('class')
library('dplyr')
```

```
##
## Attaching package: 'dplyr'
```

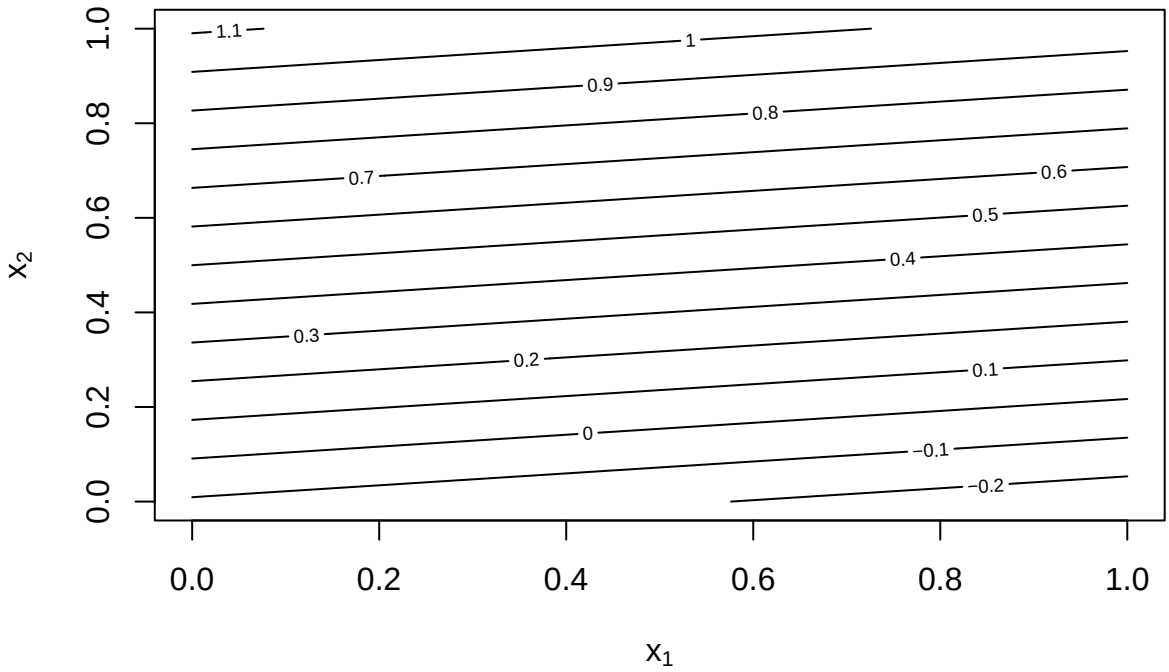
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

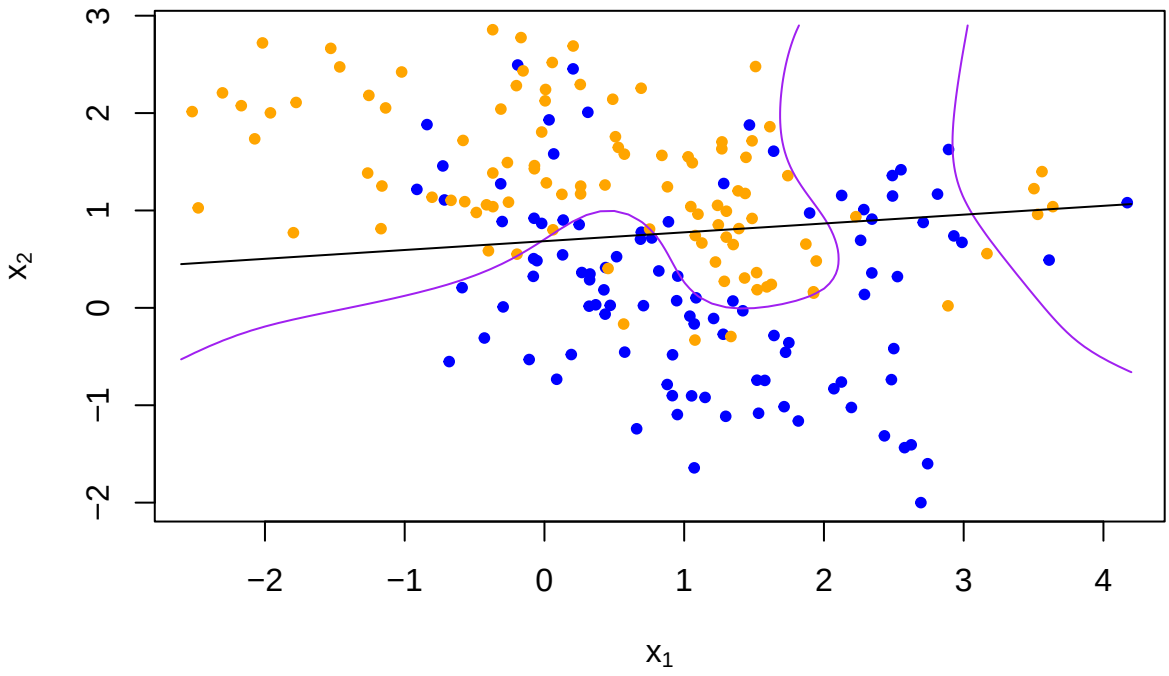
```
## load binary classification example data from author website
## 'ElemStatLearn' package no longer available
load(url('https://web.stanford.edu/~hastie/ElemStatLearn/datasets/ESL.mixture.rda'))
dat <- ESL.mixture

plot_mix_data <- expression({
  plot(dat$x[,1], dat$x[,2],
       col=ifelse(dat$y==0, 'blue', 'orange'),
       pch=20,
       xlab=expression(x[1]),
       ylab=expression(x[2]))
  ## draw Bayes (True) classification boundary
  prob <- matrix(dat$prob, length(dat$px1), length(dat$px2))
  cont <- contourLines(dat$px1, dat$px2, prob, levels=0.5)
  rslt <- sapply(cont, lines, col='purple')
})

eval(plot_mix_data)
```




```
## fit linear classifier
fit_lc <- function(y,x){
  return(lm(y~x+1))
}

## make predictions from linear classifier
predict_lc <- function(x, fun) {
  cbind(1, x) %*% fun$coefficients
}
## fit model to mixture data and make predictions
lc_beta <- fit_lc(dat$y, dat$x)
lc_pred <- predict_lc(dat$xnew, lc_beta)
## reshape predictions as a matrix
lc_pred <- matrix(lc_pred, length(dat$px1), length(dat$px2))
contour(lc_pred,
      xlab=expression(x[1]),
      ylab=expression(x[2]))
```



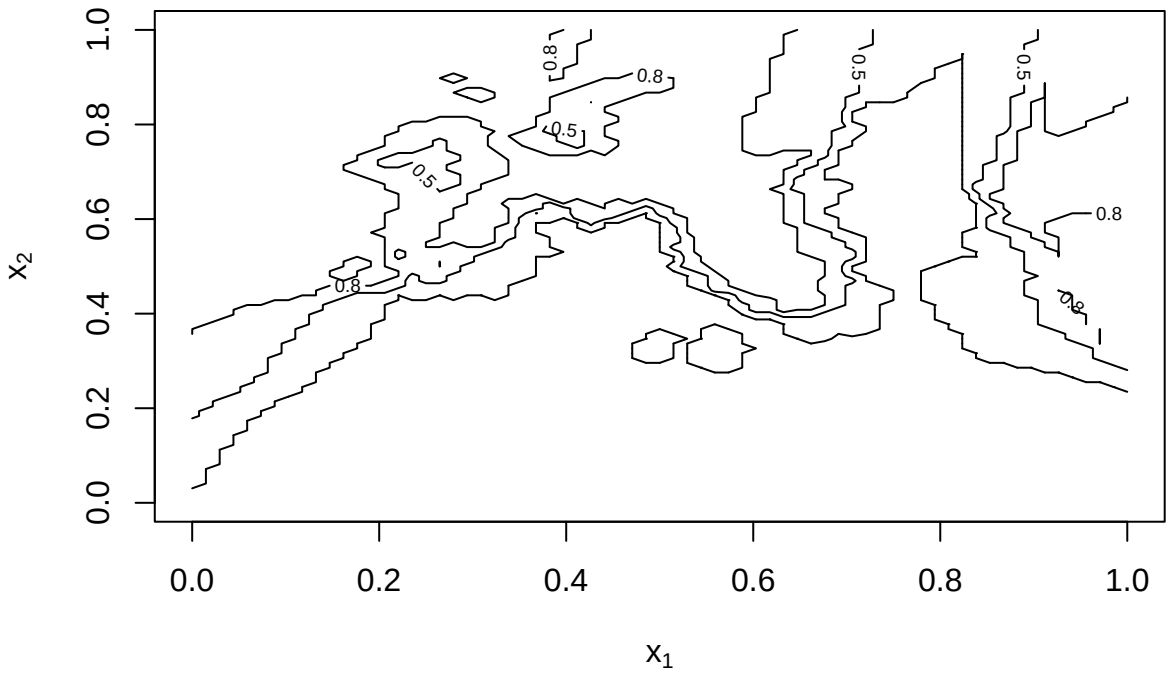

```
## find the contours in 2D space such that lc_pred == 0.5
lc_cont <- contourLines(dat$px1, dat$px2, lc_pred, levels=0.5)

## plot data and decision surface
eval(plot_mix_data)
sapply(lc_cont, lines)
```



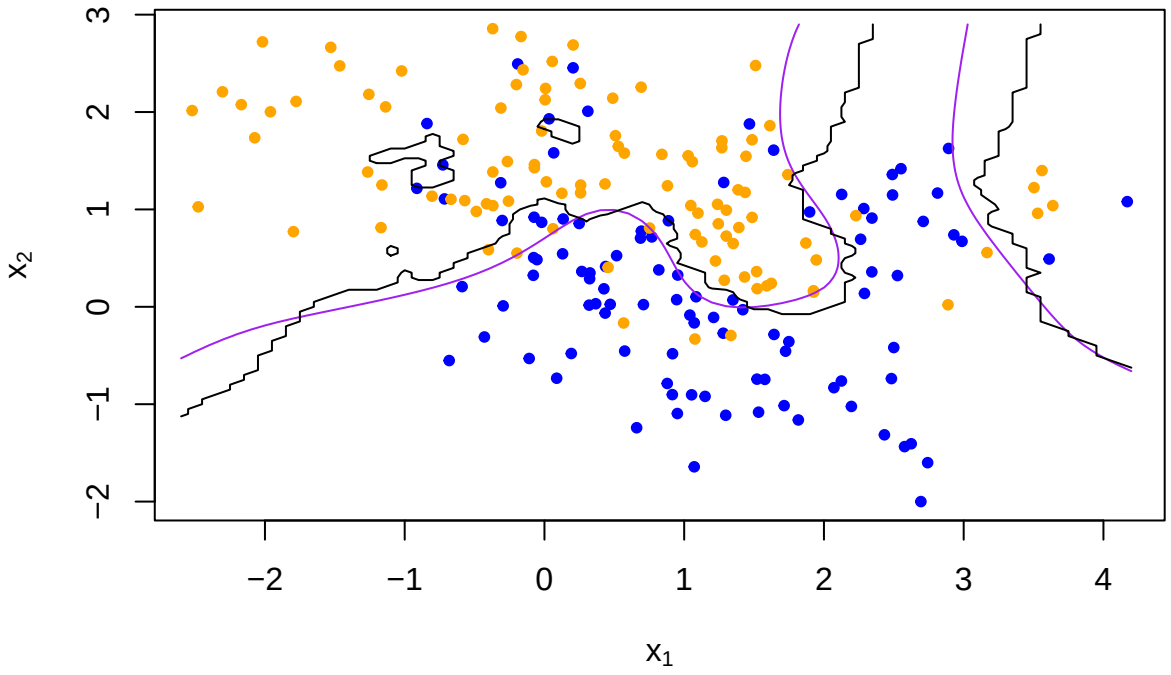

```
## [[1]]
## NULL
```

```
## fit knn classifier
## use 5-NN to estimate probability of class assignment
```

```
knn_fit <- knn(train=dat$x, test=dat$xnew, cl=dat$y, k=5, prob=TRUE)
knn_pred <- attr(knn_fit, 'prob')
knn_pred <- ifelse(knn_fit == 1, knn_pred, 1-knn_pred)

## reshape predictions as a matrix
knn_pred <- matrix(knn_pred, length(dat$px1), length(dat$px2))
contour(knn_pred,
        xlab=expression(x[1]),
        ylab=expression(x[2]),
        levels=c(0.2, 0.5, 0.8))
```



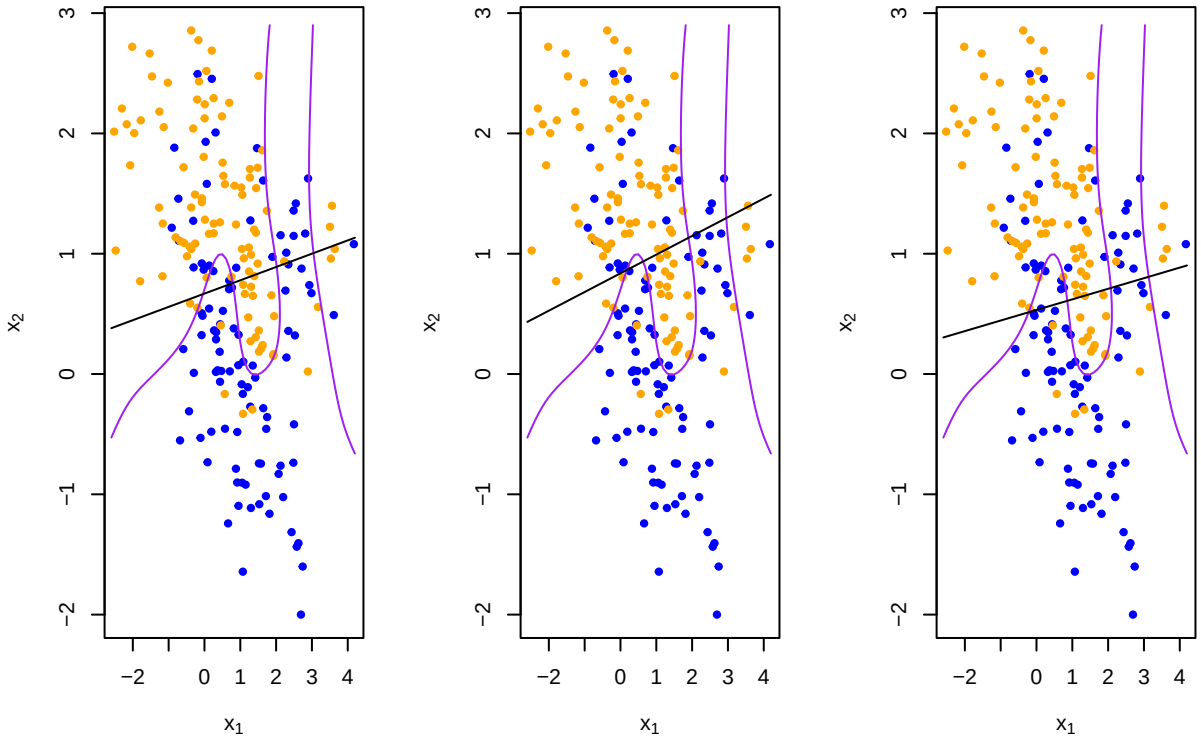

```
## find the contours in 2D space such that knn_pred == 0.5
knn_cont <- contourLines(dat$px1, dat$px2, knn_pred, levels=0.5)


## plot data and decision surface
eval(plot_mix_data)
sapply(knn_cont, lines)
```



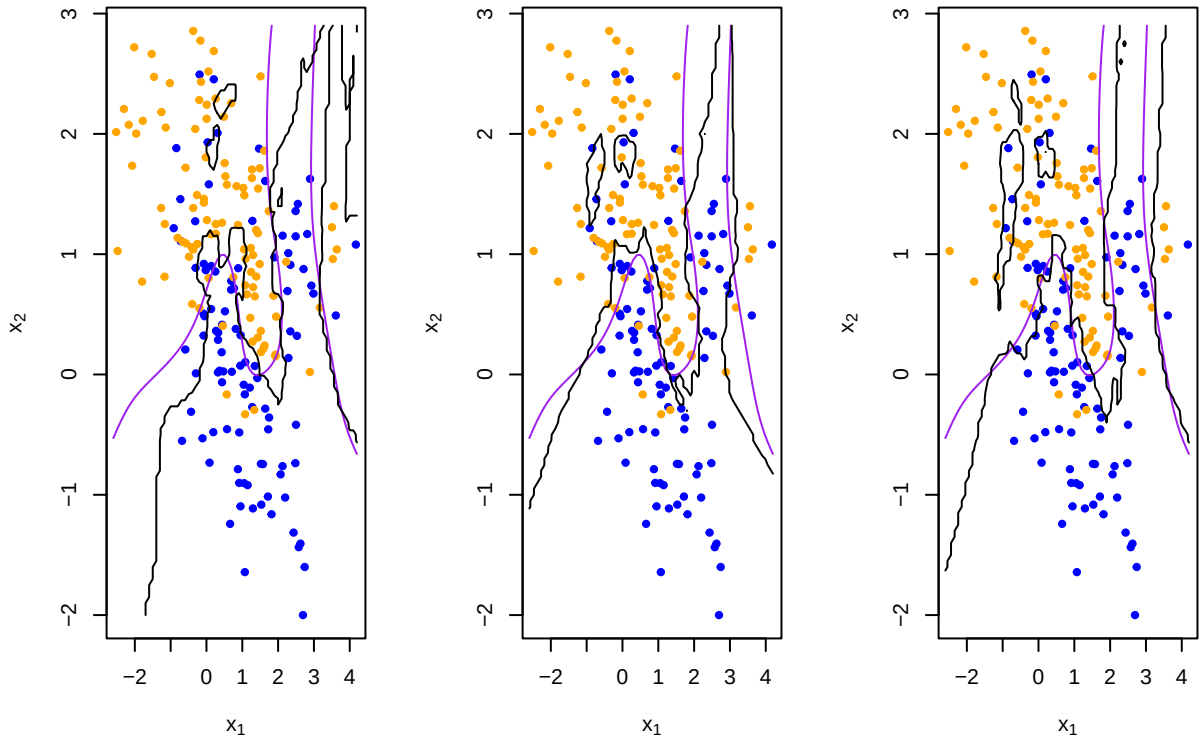

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
##
## [[5]]
## NULL
```

```
## do bootstrap to get a sense of variance in decision surface
resample <- function(dat) {
  idx <- sample(1:length(dat$y), replace = T)
  dat$y <- dat$y[idx]
  dat$x <- dat$x[idx,]
  return(dat)
}

## plot linear classifier for three bootstraps
par(mfrow=c(1,3))
for(b in 1:3) {
  datb <- resample(dat)
  ## fit model to mixture data and make predictions
  lc_beta <- fit_lc(datb$y, datb$x)
  lc_pred <- predict_lc(datb$xnew, lc_beta)

  ## reshape predictions as a matrix
  lc_pred <- matrix(lc_pred, length(datb$px1), length(datb$px2))
```

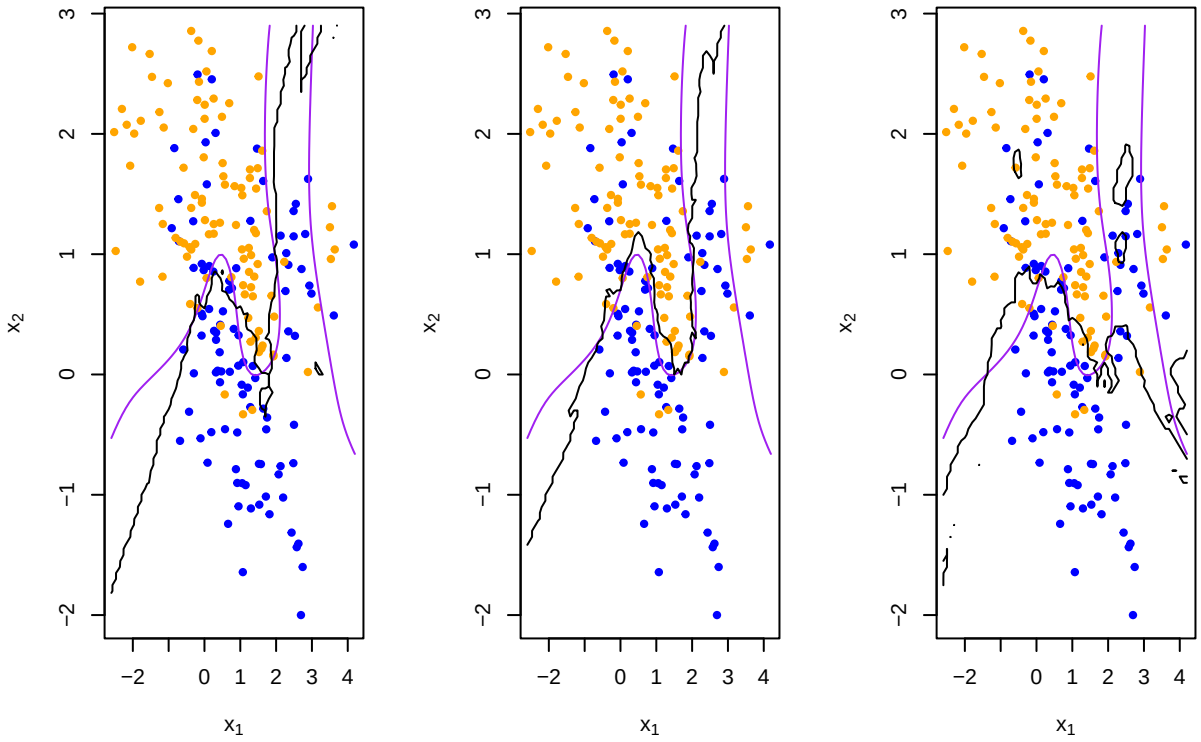
```
  ## find the contours in 2D space such that lc_pred == 0.5
  lc_cont <- contourLines(datb$px1, datb$px2, lc_pred, levels=0.5)

  ## plot data and decision surface
  eval(plot_mix_data)
  sapply(lc_cont, lines)
}
```




```
## plot 5-NN classifier for three bootstraps
par(mfrow=c(1,3))
for(b in 1:3) {
  datb <- resample(dat)

  knn_fit <- knn(train=datb$x, test=datb$xnew, cl=datb$y, k=5, prob=TRUE)
  knn_pred <- attr(knn_fit, 'prob')
  knn_pred <- ifelse(knn_fit == 1, knn_pred, 1-knn_pred)

  ## reshape predictions as a matrix
  knn_pred <- matrix(knn_pred, length(datb$px1), length(datb$px2))

  ## find the contours in 2D space such that knn_pred == 0.5
  knn_cont <- contourLines(datb$px1, datb$px2, knn_pred, levels=0.5)

  ## plot data and decision surface
  eval(plot_mix_data)
  sapply(knn_cont, lines)
}
```

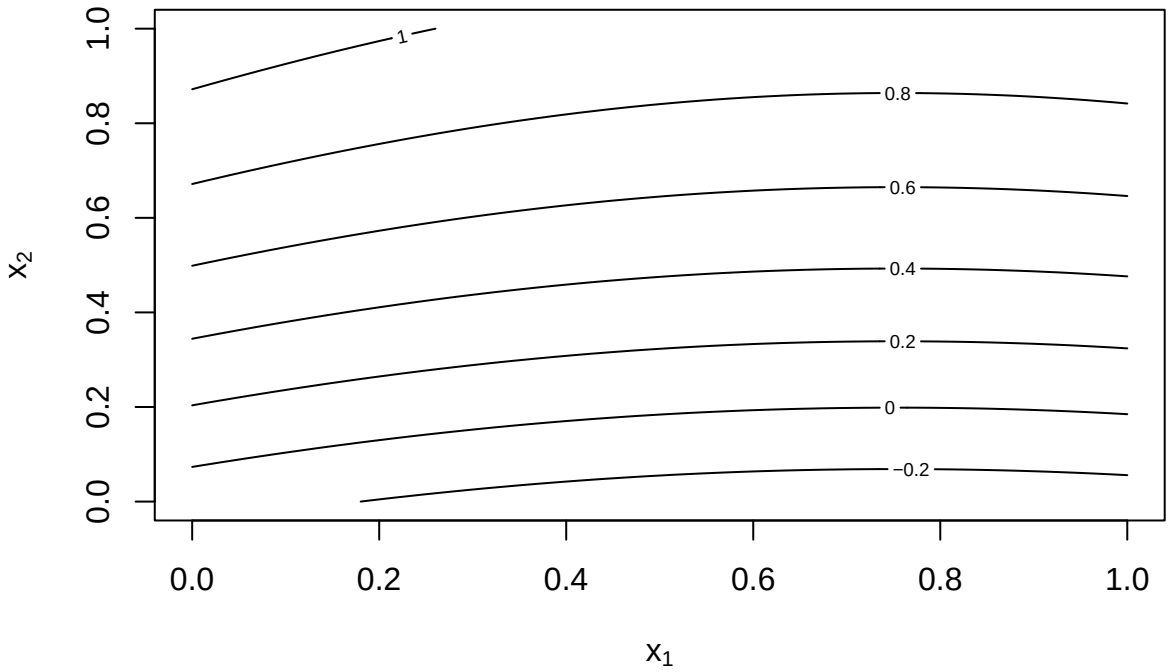



```
## plot 20-NN classifier for three bootstraps
par(mfrow=c(1,3))
for(b in 1:3) {
  datb <- resample(dat)

  knn_fit <- knn(train=datb$x, test=datb$xnew, cl=datb$y, k=20, prob=TRUE)
  knn_pred <- attr(knn_fit, 'prob')
  knn_pred <- ifelse(knn_fit == 1, knn_pred, 1-knn_pred)

  ## reshape predictions as a matrix
  knn_pred <- matrix(knn_pred, length(datb$px1), length(datb$px2))

  ## find the contours in 2D space such that knn_pred == 0.5
  knn_cont <- contourLines(datb$px1, datb$px2, knn_pred, levels=0.5)

  ## plot data and decision surface
  eval(plot_mix_data)
  sapply(knn_cont, lines)
}
```

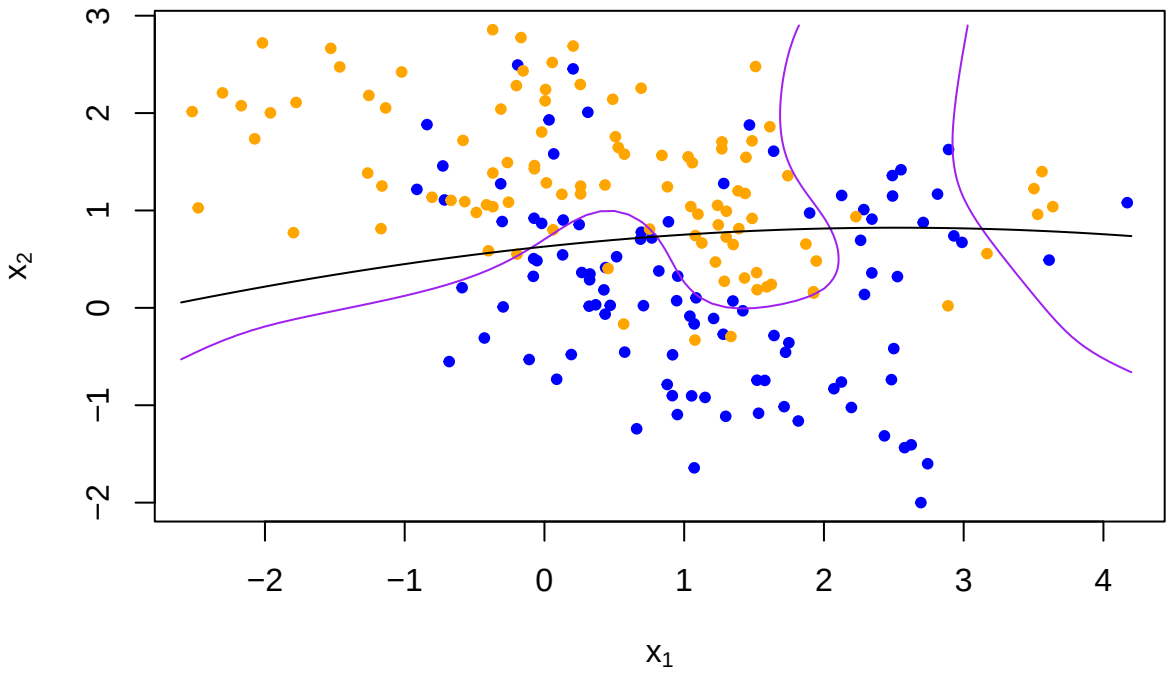


Use squared x to fit the data

```
## fit linear classifier
fit_lc <- function(y,x){

  return(lm(y~I(x^2) + I(x)+1))
}
## make predictions from linear classifier
predict_lc <- function(x, fun) {
  cbind(cbind(1, x*x),x) %*% fun$coefficients
}

## fit model to mixture data and make predictions
lc_beta_new <- fit_lc(dat$y, dat$x)
lc_pred_new <- predict_lc(dat$xnew, lc_beta_new)
## reshape predictions as a matrix
lc_pred_new <- matrix(lc_pred_new, length(dat$px1), length(dat$px2))
contour(lc_pred_new,
      xlab=expression(x[1]),
      ylab=expression(x[2]))
```



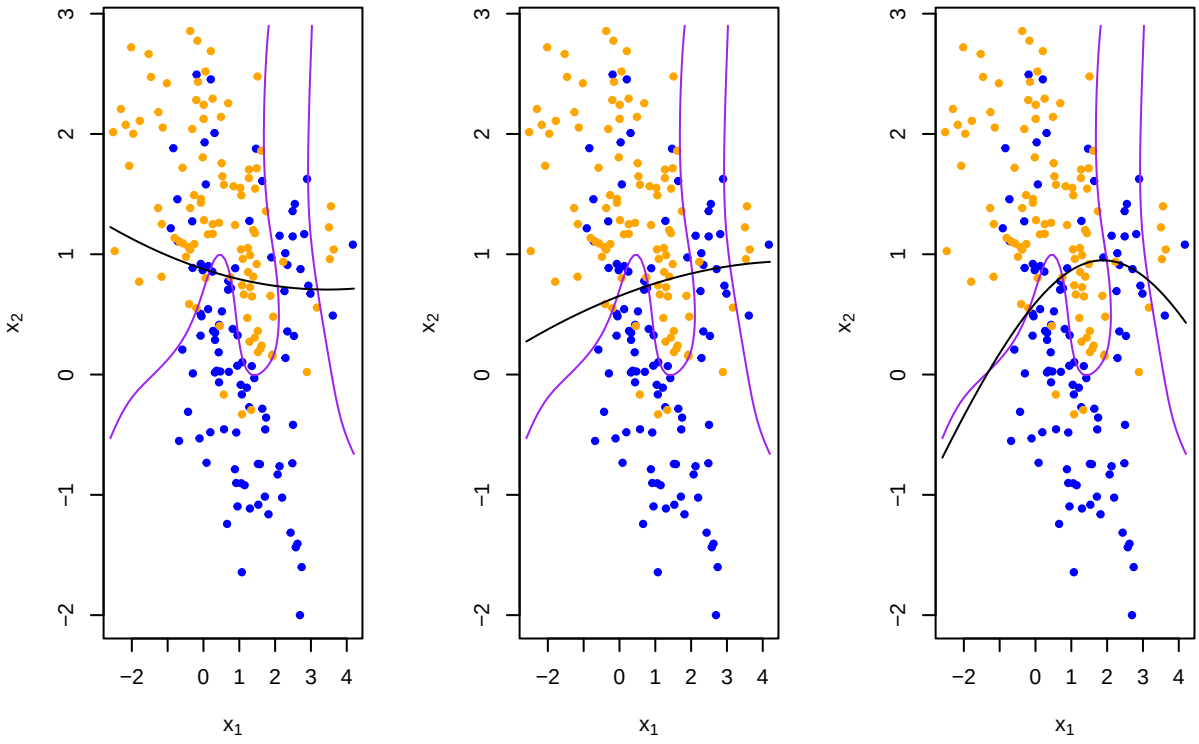

```
## find the contours in 2D space such that lc_pred == 0.5
lc_cont_new <- contourLines(dat$px1, dat$px2, lc_pred_new, levels=0.5)

## plot data and decision surface
eval(plot_mix_data)
sapply(lc_cont_new, lines)
```




```
## [[1]]
## NULL
```

```
## do bootstrap to get a sense of variance in decision surface
resample <- function(dat) {
```

```
  idx <- sample(1:length(dat$y), replace = T)
  dat$y <- dat$y[idx]
  dat$x <- dat$x[idx,]
  return(dat)
}

## plot linear classifier for three bootstraps
par(mfrow=c(1,3))
for(b in 1:3) {
  datb <- resample(dat)
  ## fit model to mixture data and make predictions
  lc_beta_new <- fit_lc(datb$y, datb$x)
  lc_pred_new <- predict_lc(datb$xnew, lc_beta_new)

  ## reshape predictions as a matrix
  lc_pred_new <- matrix(lc_pred_new, length(datb$px1), length(datb$px2))

  ## find the contours in 2D space such that lc_pred == 0.5
  lc_cont_new <- contourLines(datb$px1, datb$px2, lc_pred_new, levels=0.5)

  ## plot data and decision surface
  eval(plot_mix_data)
  sapply(lc_cont_new, lines)
}
```



From the result we can see that the more flexible model fits the curve of the real classification better, but from the results of bootstrap, the curvature of different training sets varies greatly, more unstable. Therefore, it has smaller bias and larger variance