Homework 1

Yue Guo

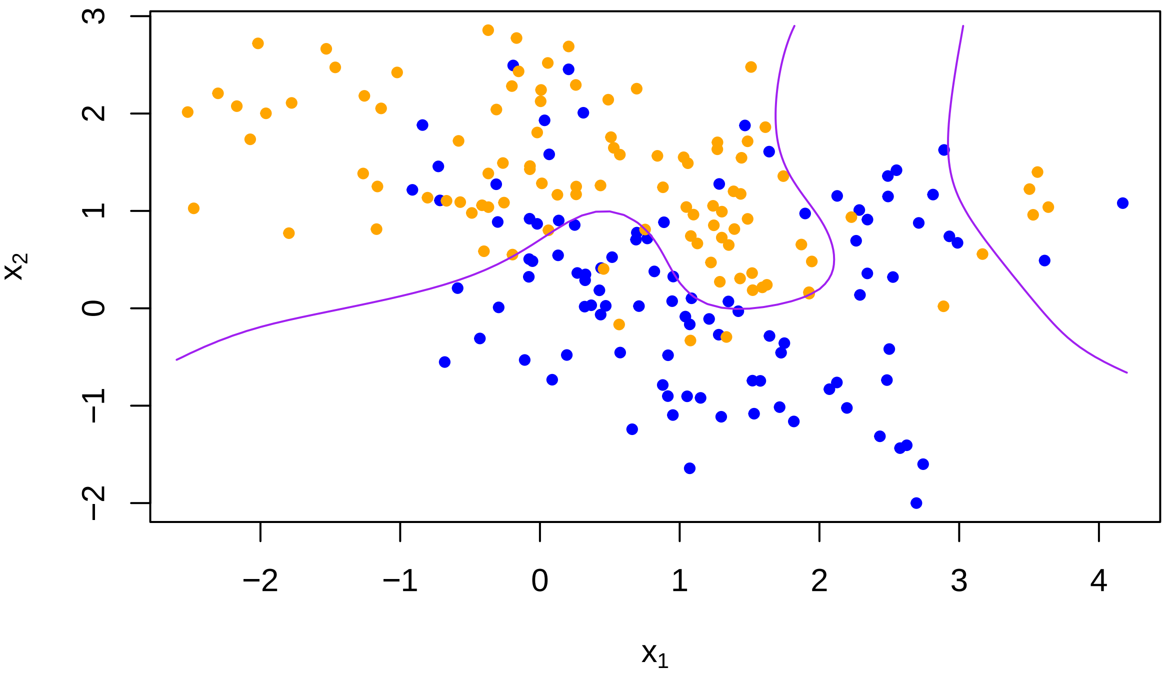
January 22, 2020

Rewrite code with lm()

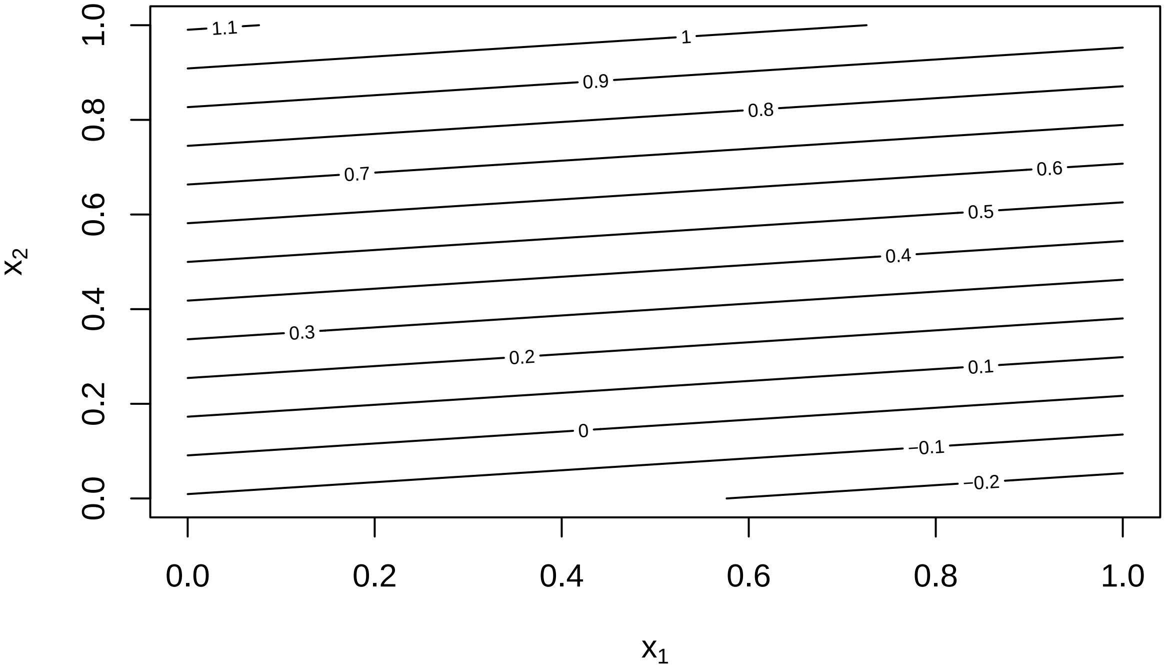
```
library('class')
library('dplyr')
```

```
##
## 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
```

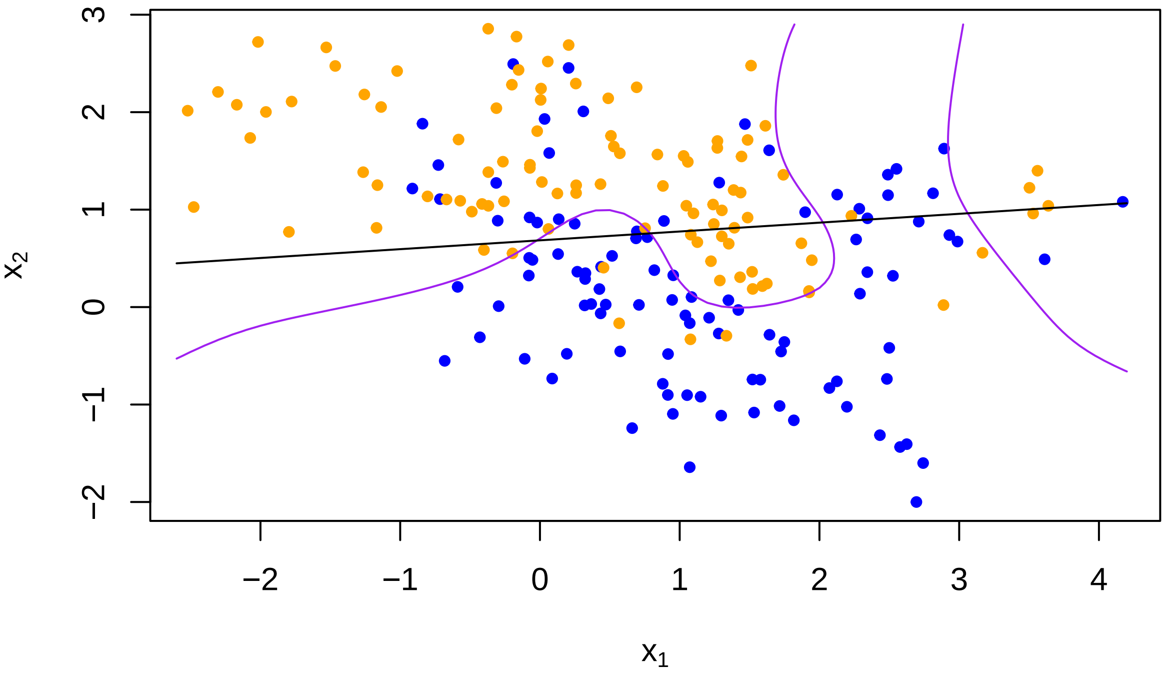
```
## load binary classification example data from author website
## 'ElemStatLearn' package no longer available
load(url('https://web.stanford.edu/~hastie/ElemStatLearn/datasets/ESL.mixture.rda'))
dat <- ESL.mixture

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]))
```



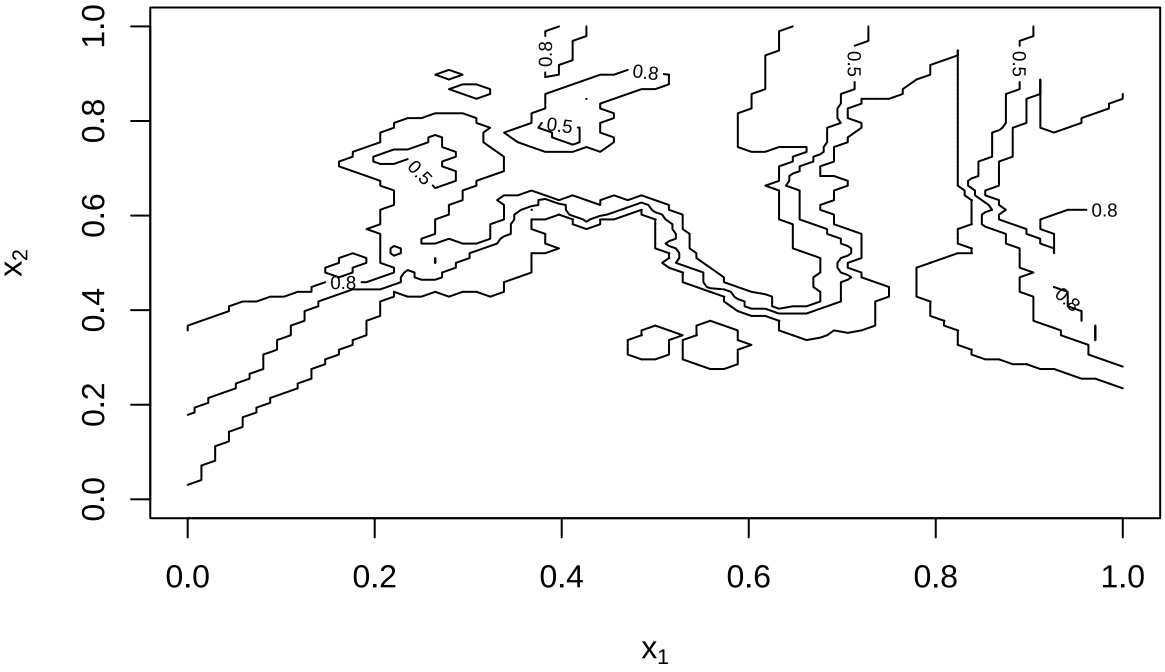
```
## find the contours in 2D space such that lc_pred == 0.5
lc_cont <- contourLines(dat$px1, dat$px2, lc_pred, levels=0.5)

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



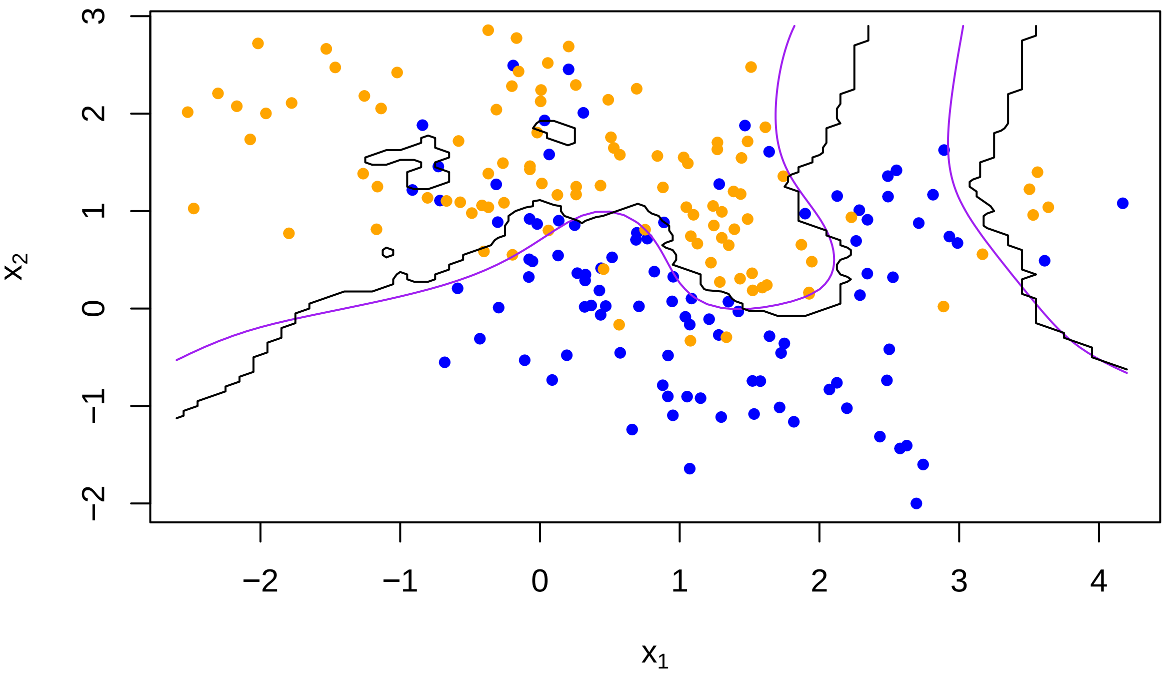
```
## [[1]]
## NULL
```

```
## 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))
```

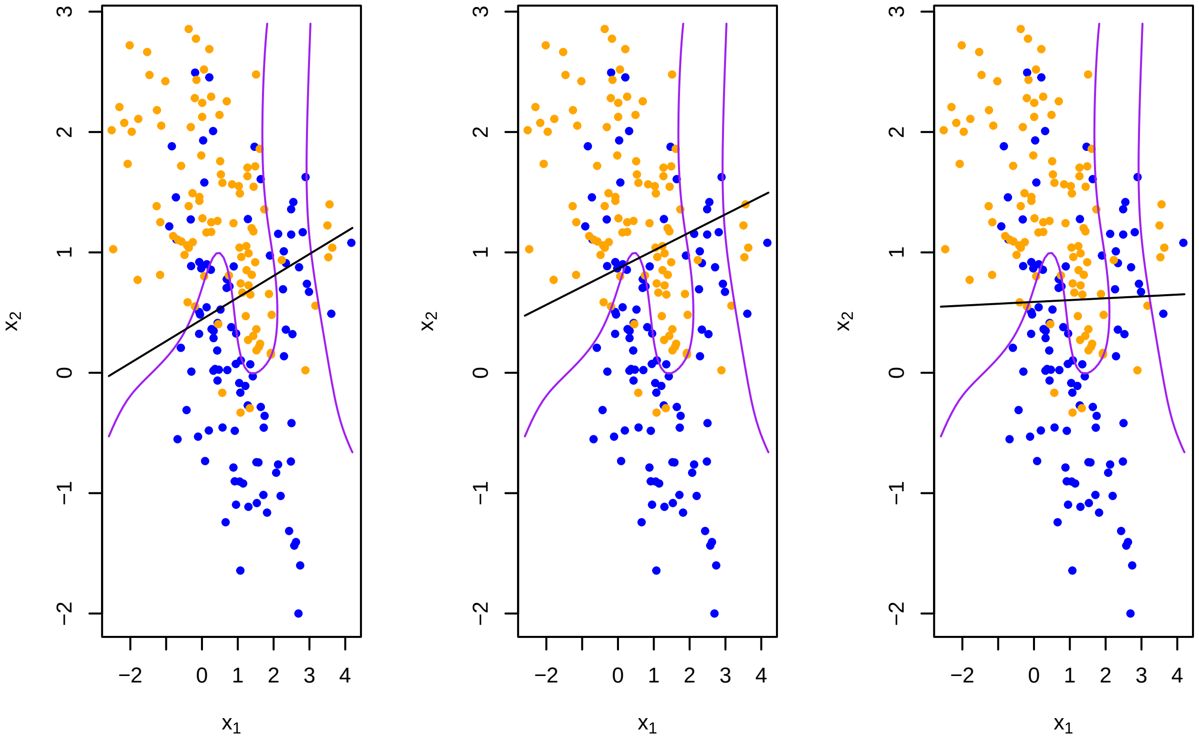


```
## find the contours in 2D space such that knn_pred == 0.5
knn_cont <- contourLines(dat$px1, dat$px2, knn_pred, levels=0.5)


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

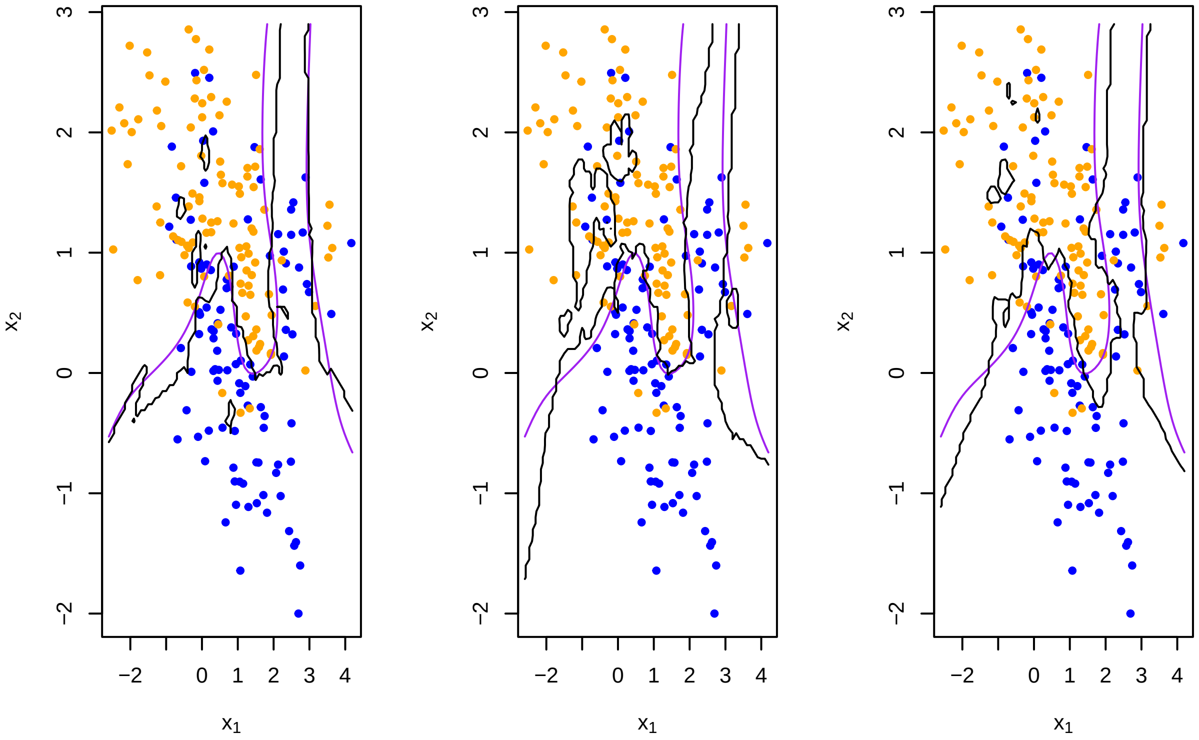


```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
##
## [[5]]
## NULL
```

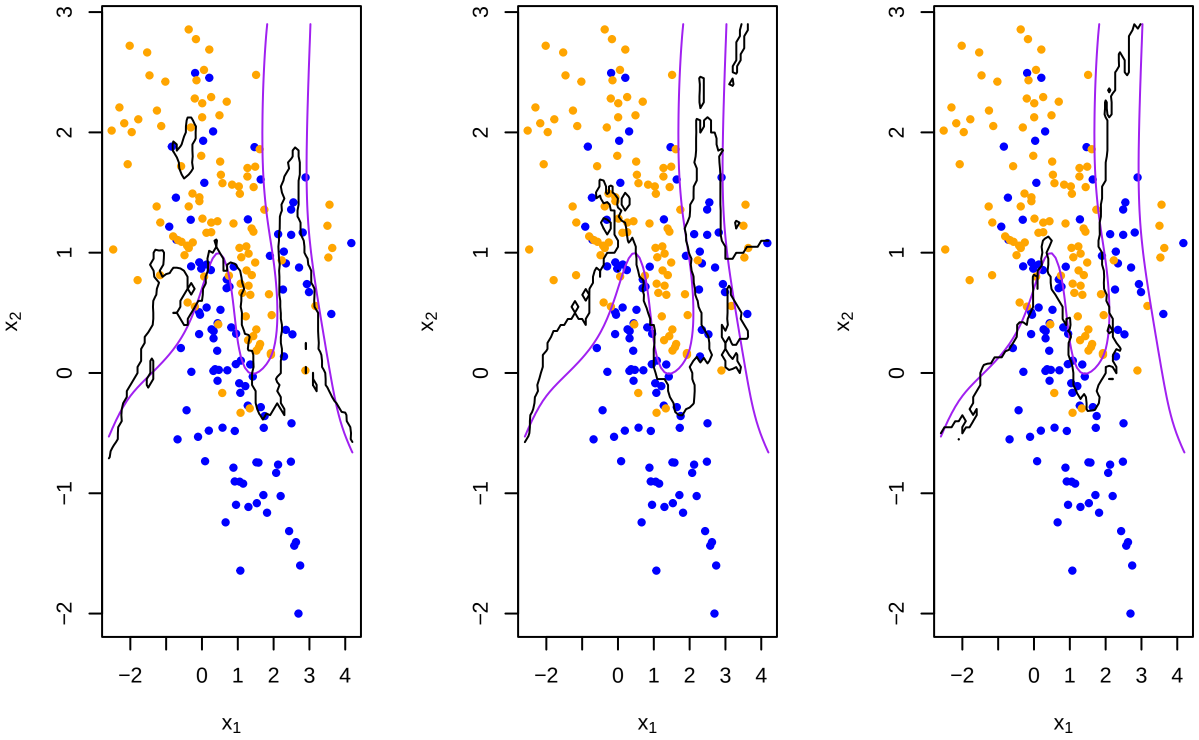
```
## do bootstrap to get a sense of variance in decision surface
resample <- function(dat) {
  idx <- sample(1:length(dat$y), replace = T)
  dat$y <- dat$y[idx]
  dat$x <- dat$x[idx,]
  return(dat)
}

## 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)
}
```

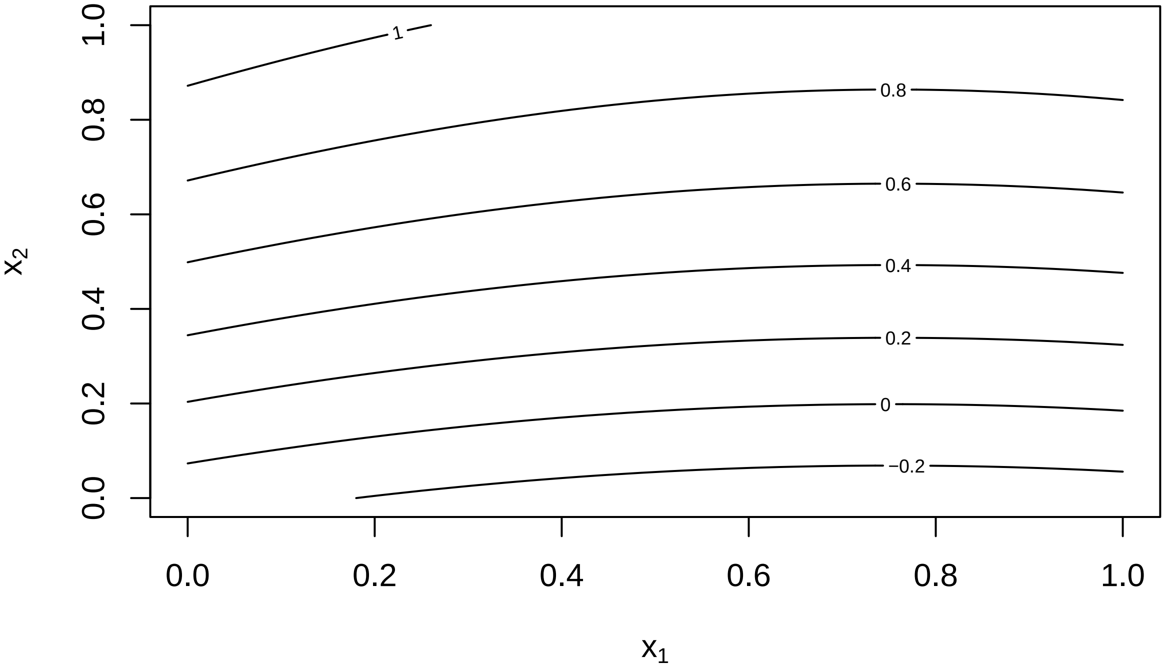


```r
## plot 20-NN classifier for three bootstraps
par(mfrow=c(1,3))
for(b in 1:3) {
  datb <- resample(dat)

  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)
}
```

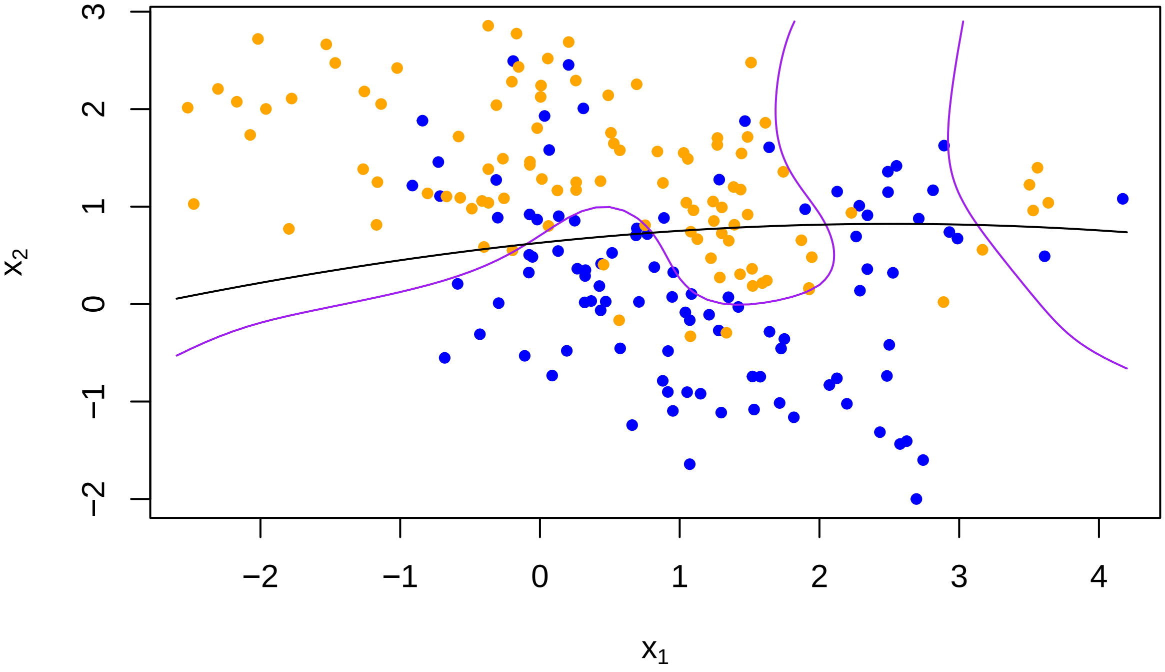


Use squared x to fit the data

```r
## 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]))
```

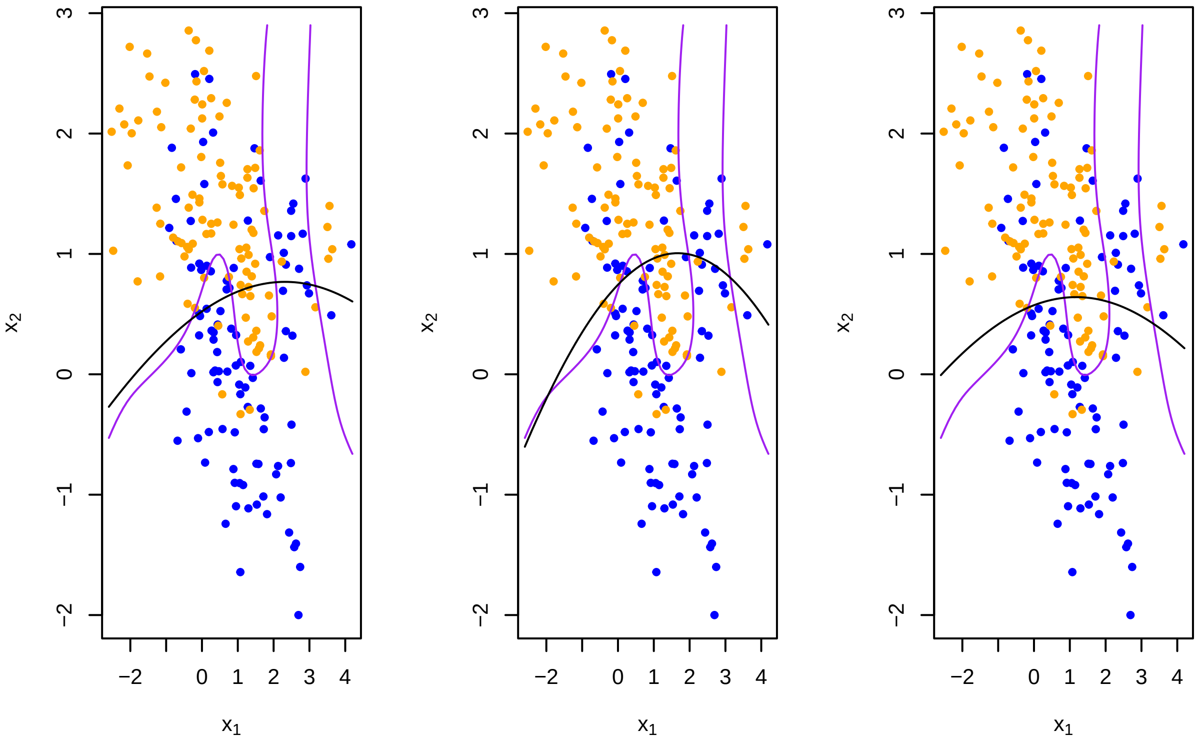


```
## find the contours in 2D space such that lc_pred == 0.5
lc_cont_new <- contourLines(dat$px1, dat$px2, lc_pred_new, levels=0.5)

## 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)
}
```



```
summary(lc_beta)
```

```
##
## Call:
## lm(formula = y ~ x + 1)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.02904 -0.32259  0.07101  0.33856  0.75045
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.336807   0.046728   7.208 1.19e-11 ***
## x1          -0.004206   0.025000  -0.168    0.867
## x2           0.277270   0.031542   8.791 7.37e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4054 on 197 degrees of freedom
## Multiple R-squared:  0.3473, Adjusted R-squared:  0.3406
## F-statistic:  52.4 on 2 and 197 DF,  p-value: < 2.2e-16
```

```
summary(lc_beta_new)
```

```
##
## Call:
## lm(formula = y ~ I(x^2) + I(x) + 1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.89457 -0.38400  0.07979  0.34586  0.62578
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.34084    0.05321   6.405 1.10e-09 ***
## I(x^2)1      0.01251    0.01409   0.887    0.376
## I(x^2)2     -0.03006    0.02246  -1.338    0.182
## I(x)1       -0.02948    0.02978  -0.990    0.324
## I(x)2        0.29456    0.04406   6.685 2.37e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4176 on 195 degrees of freedom
## Multiple R-squared:  0.3115, Adjusted R-squared:  0.2974
## F-statistic: 22.06 on 4 and 195 DF,  p-value: 4.893e-15
```

From the result we can see that the more flexible model has squared x has smaller bias and larger variance