Homework 1

Yue Guo

January 22, 2020

Rewrite code with lm()

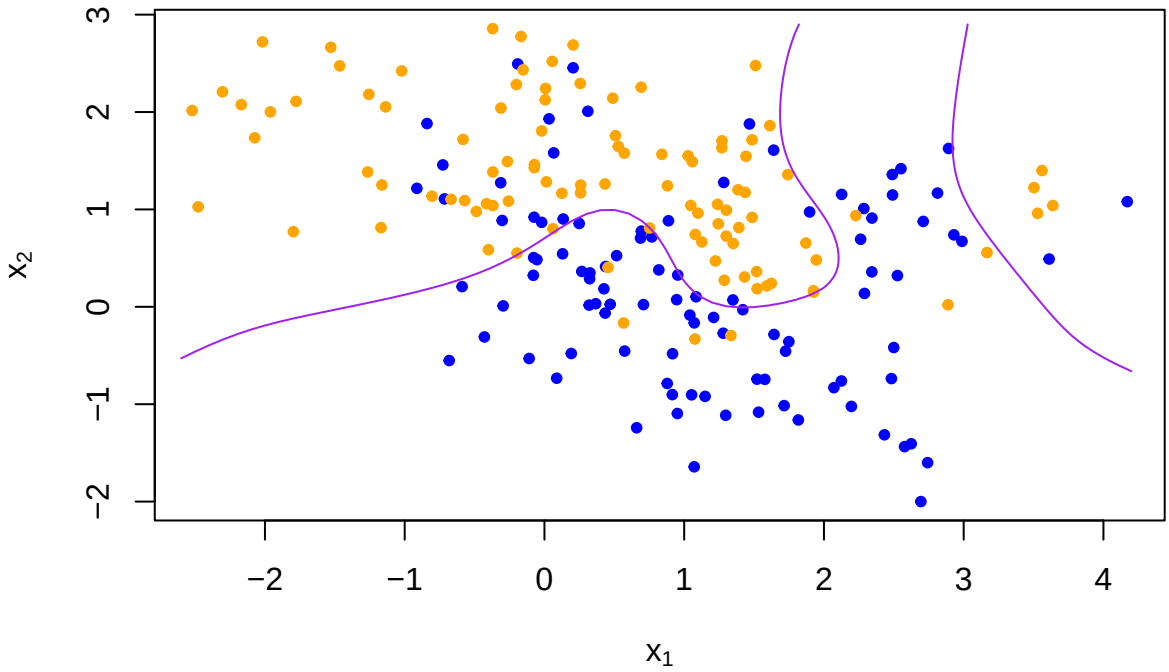
```
library('class')
library('dplyr')
```

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

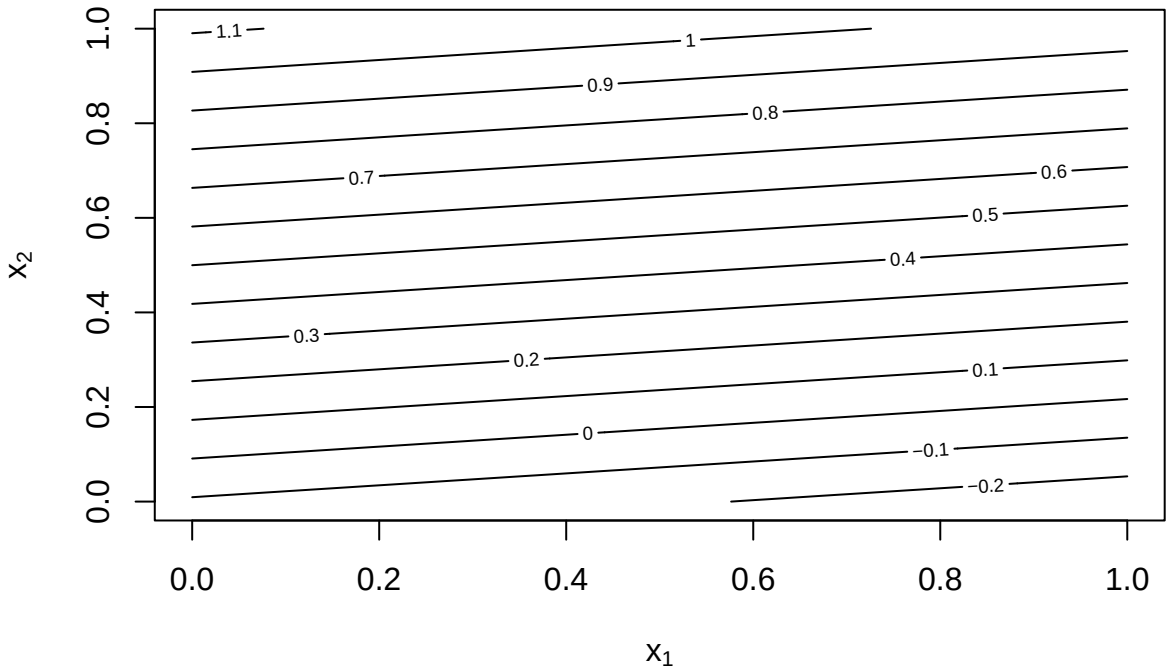
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

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

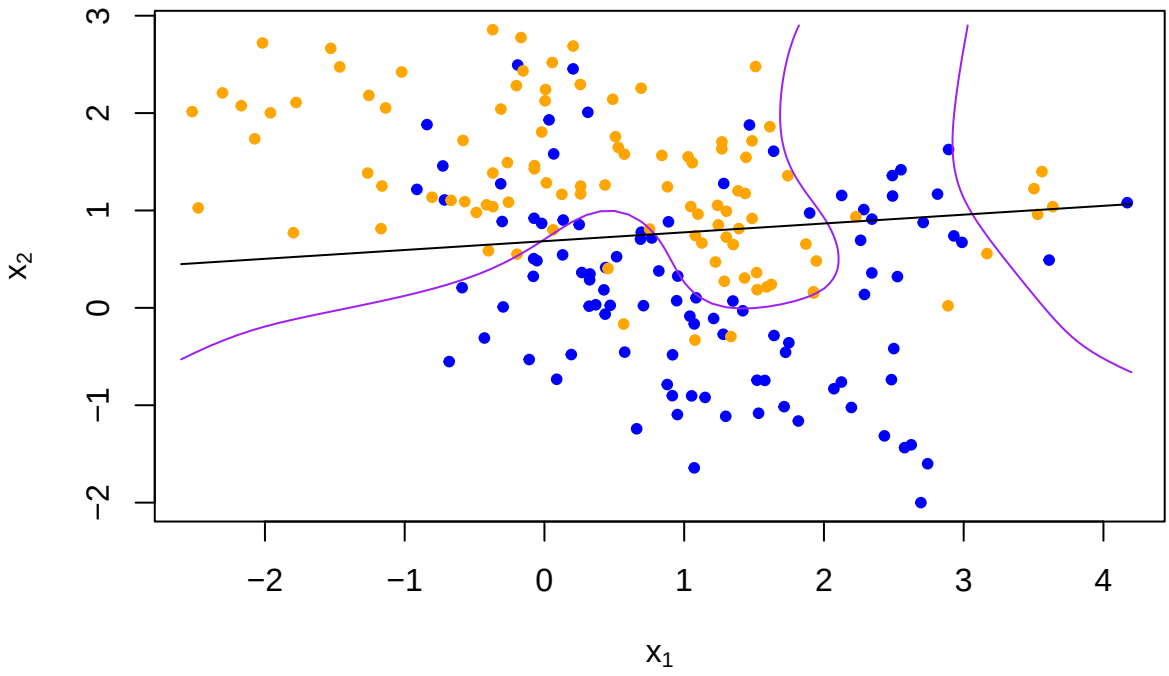
```
## load binary classification example data from author website
## 'ElemStatLearn' package no longer available
load(url('https://web.stanford.edu/~hastie/ElemStatLearn/datasets/ESL.mixture.rda'))
dat <- ESL.mixture

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]))
```



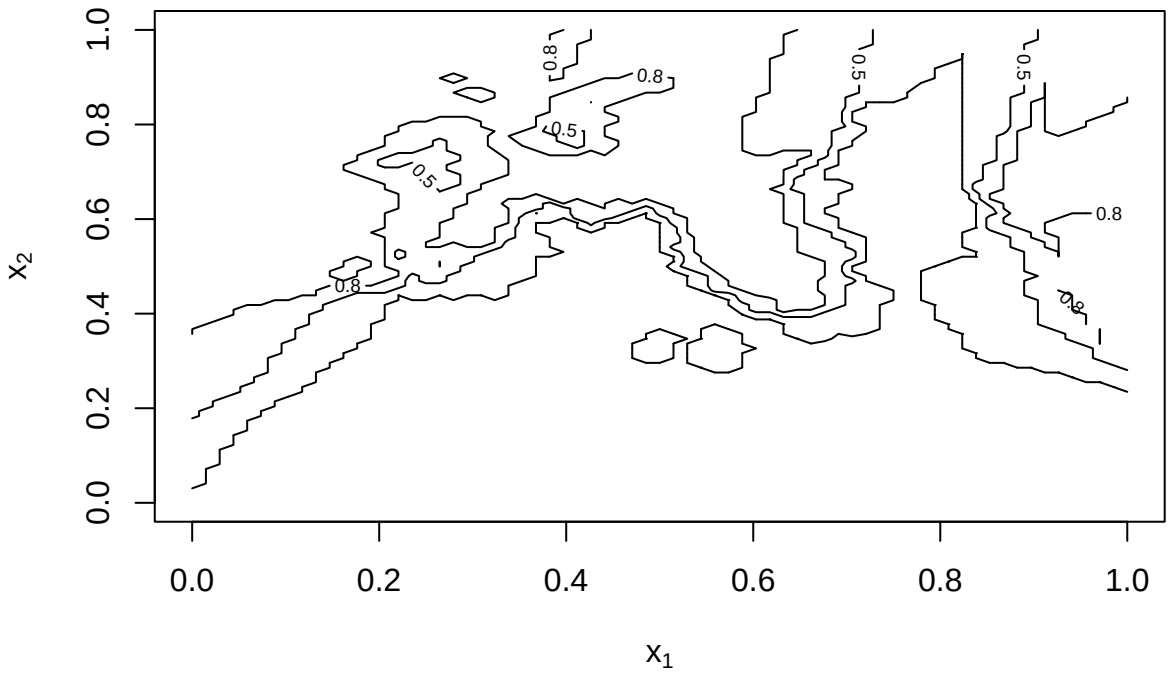

```
## find the contours in 2D space such that lc_pred == 0.5
lc_cont <- contourLines(dat$px1, dat$px2, lc_pred, levels=0.5)

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



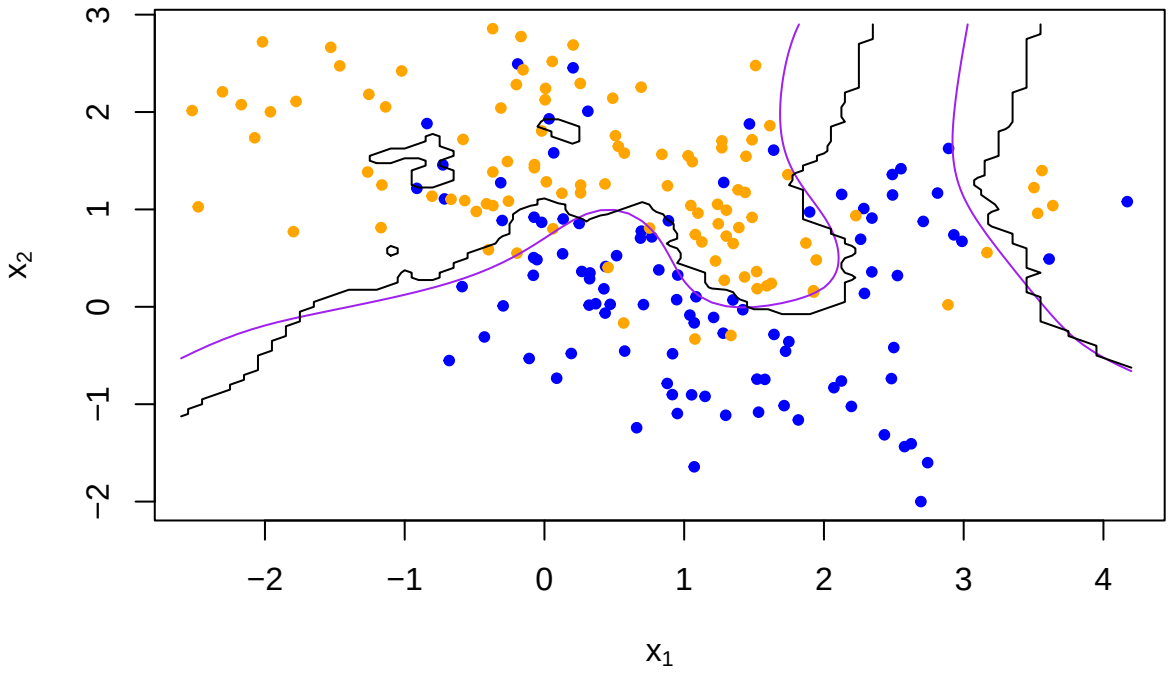

```
## [[1]]
## NULL
```

```
## 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))
```



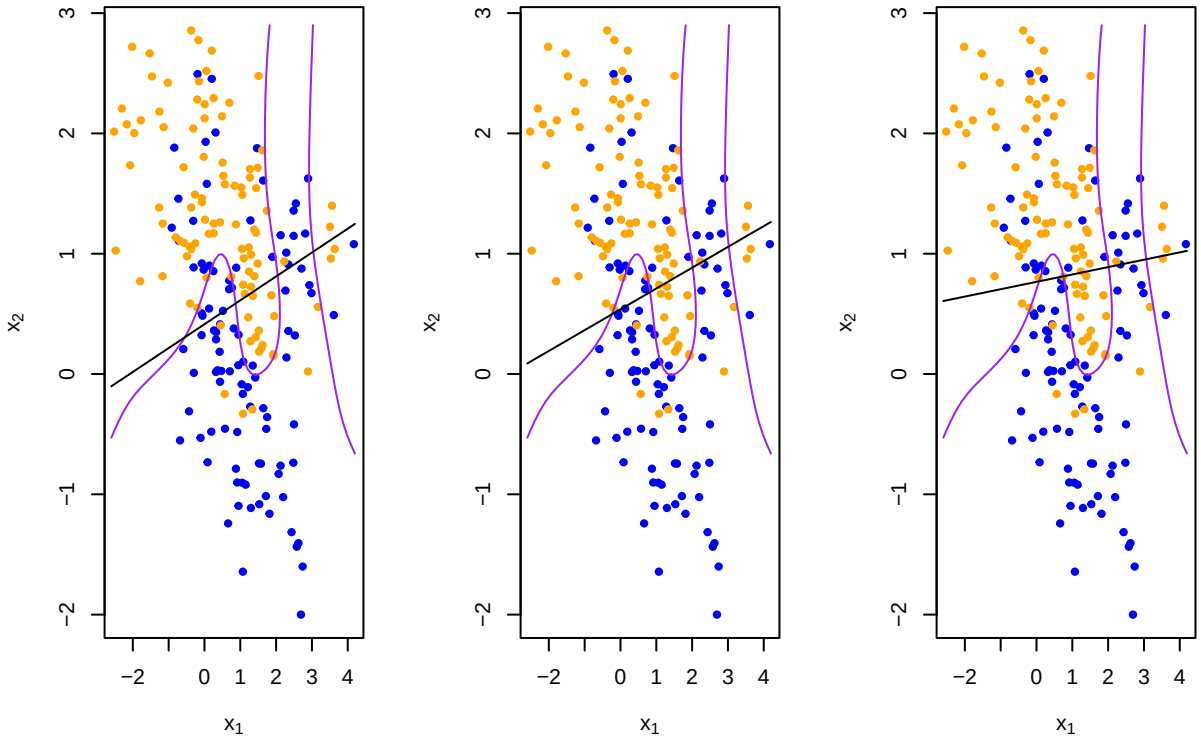

```
## find the contours in 2D space such that knn_pred == 0.5
knn_cont <- contourLines(dat$px1, dat$px2, knn_pred, levels=0.5)


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



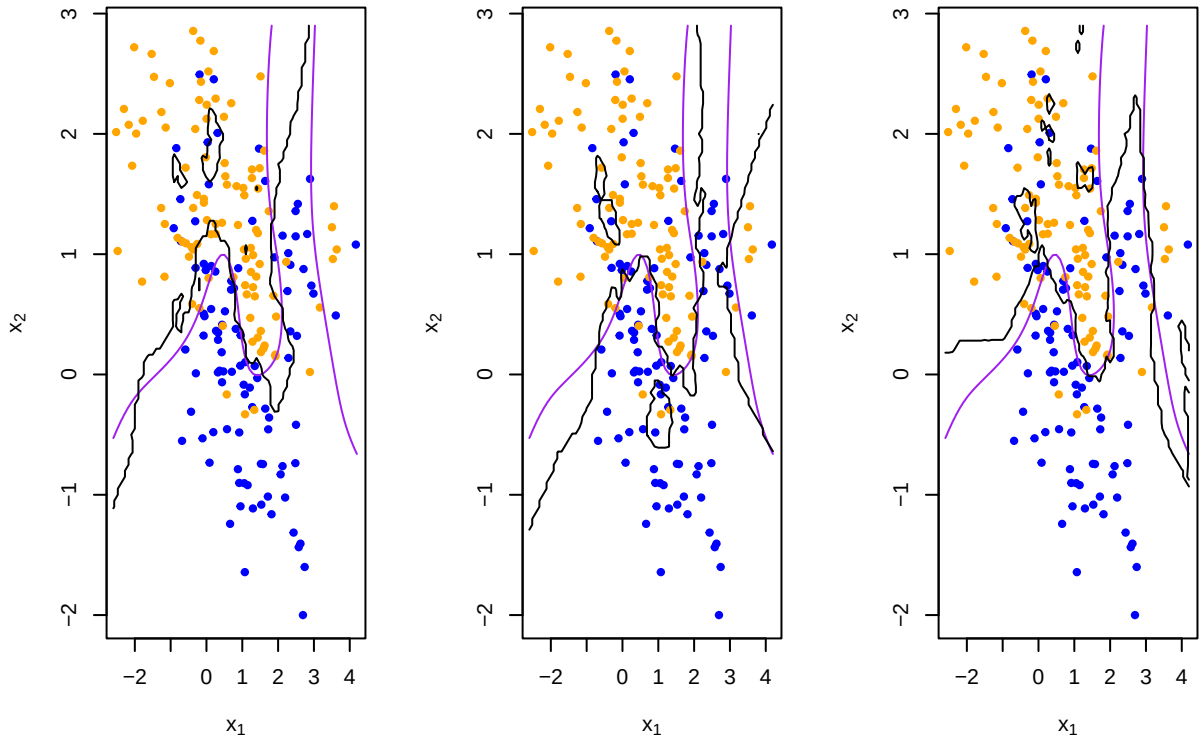

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
##
## [[5]]
## NULL
```

```
## 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))
```

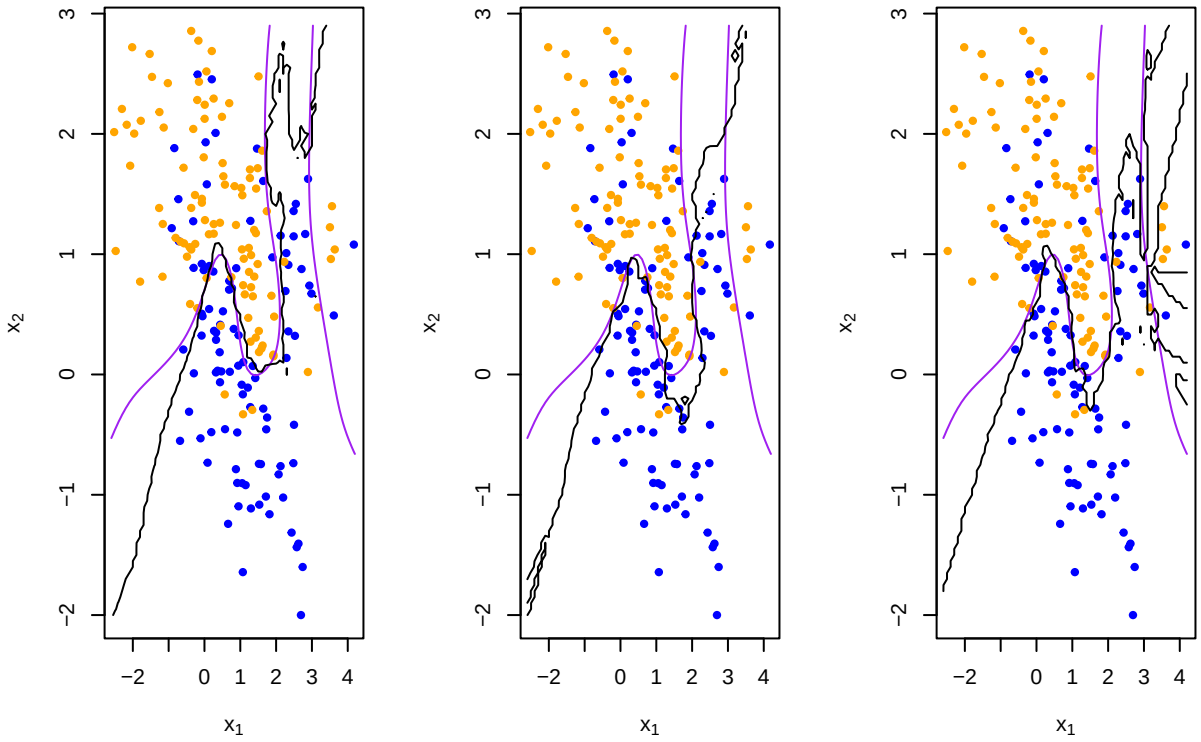
```
  ## find the contours in 2D space such that lc_pred == 0.5
  lc_cont <- contourLines(datb$px1, datb$px2, lc_pred, levels=0.5)

  ## 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))
}
(dat$x)^2
```

```
##                [,1]         [,2]
##   [1,] 6.381146e+00 1.030734e-01
##   [2,] 1.346556e-01 9.898637e-04
##   [3,] 5.901605e-01 5.147864e-01
##   [4,] 4.808530e-01 6.040306e-01
##   [5,] 3.934913e-04 7.521290e-01
##   [6,] 4.824810e+00 1.046558e+00
##   [7,] 1.191572e-02 2.815587e-01
##   [8,] 8.328754e-01 1.479182e+00
##   [9,] 6.249213e+00 1.749564e-01
##  [10,] 3.601300e+00 9.481998e-01
##  [11,] 8.723892e-02 9.357017e-05
##  [12,] 4.645727e-01 3.043180e-01
##  [13,] 6.204975e+00 1.318628e+00
##  [14,] 2.854829e-03 2.343552e-01
##  [15,] 4.728180e-01 4.970318e-01
##  [16,] 1.637273e+00 7.345904e-02
##  [17,] 9.797368e-02 1.622432e+00
##  [18,] 7.913186e+00 1.363663e+00
##  [19,] 1.023924e-01 2.958967e-04
##  [20,] 1.914433e-01 1.711047e-01
```

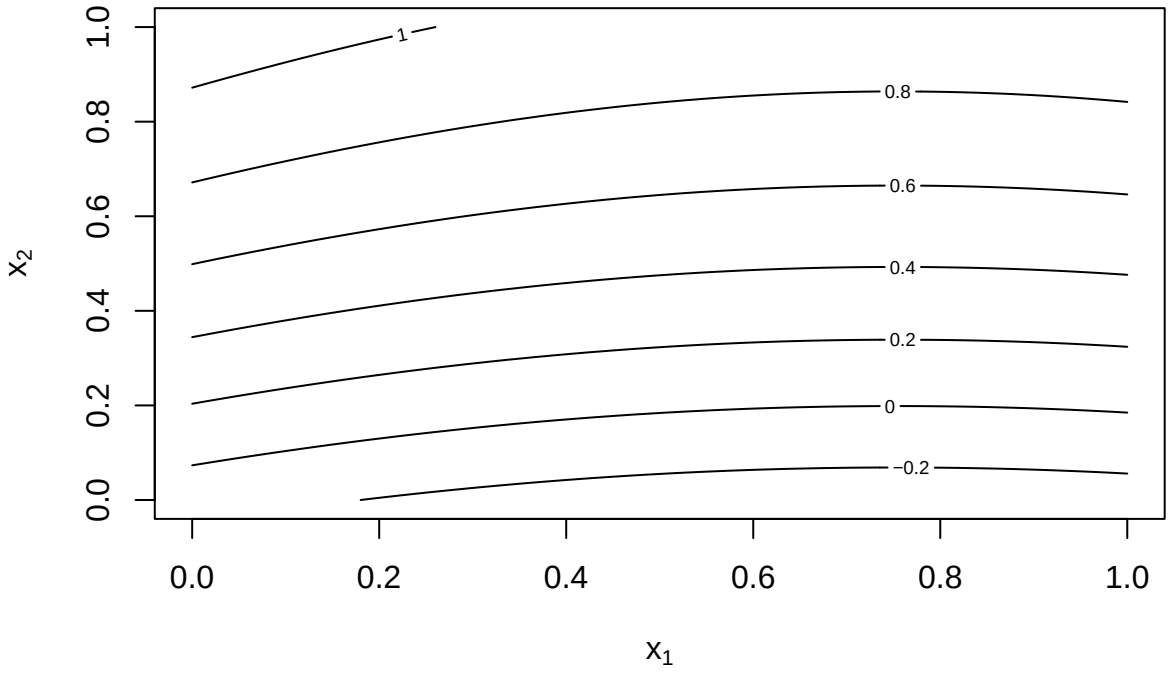
```
##  [21,] 5.919194e+00 1.727550e+00
##  [22,] 1.845988e-01 9.598299e-02
##  [23,] 6.186582e-02 7.319143e-01
##  [24,] 7.257295e+00 3.999414e+00
##  [25,] 2.152410e+00 3.525347e+00
##  [26,] 1.083051e+00 7.256922e-03
##  [27,] 1.110120e-03 3.725894e+00
##  [28,] 2.977976e+00 2.078987e-01
##  [29,] 1.321306e+00 8.460142e-01
##  [30,] 4.523570e+00 1.332131e+00
##  [31,] 8.928900e+00 4.520554e-01
##  [32,] 1.148897e+00 2.699510e+00
##  [33,] 1.176850e+00 1.060392e-02
##  [34,] 7.152536e-02 1.316430e-01
##  [35,] 1.645778e+00 1.630477e+00
##  [36,] 1.689626e-02 2.961294e-01
##  [37,] 6.199134e+00 1.845549e+00
##  [38,] 6.698276e-01 1.436413e-01
##  [39,] 2.347814e+00 1.171080e+00
##  [40,] 5.020331e-01 5.046340e-04
##  [41,] 2.313180e+00 5.518283e-01
##  [42,] 7.877606e-01 7.806644e-01
##  [43,] 2.486855e+00 5.549261e-01
##  [44,] 5.488550e+00 1.285063e-01
##  [45,] 3.658512e-02 6.218468e+00
##  [46,] 3.059659e+00 1.281038e-01
##  [47,] 7.062201e-01 3.540950e+00
##  [48,] 8.952639e-01 5.371953e-03
##  [49,] 8.377324e-01 8.131699e-01
##  [50,] 1.818993e+00 4.963562e-03
##  [51,] 8.584101e+00 5.459621e-01
##  [52,] 1.044625e-01 8.292053e-02
##  [53,] 4.290760e+00 6.895887e-01
##  [54,] 9.136997e-02 7.854082e-01
##  [55,] 3.697257e-02 2.298690e-01
##  [56,] 5.286412e-01 2.123901e+00
##  [57,] 1.804924e-02 8.130192e-01
##  [58,] 5.946640e-03 2.551479e-01
##  [59,] 7.348239e+00 7.685061e-01
##  [60,] 1.739512e+01 1.166041e+00
##  [61,] 1.061254e-01 1.211011e-01
##  [62,] 4.210299e-02 6.021565e+00
##  [63,] 2.690117e+00 2.587391e+00
##  [64,] 6.333510e-03 1.045306e-01
##  [65,] 7.519071e+00 2.561129e+00
##  [66,] 9.658734e-02 4.031991e+00
##  [67,] 5.244294e+00 1.887594e-02
##  [68,] 4.354762e-01 1.541662e+00
##  [69,] 2.666990e-01 2.757842e-01
##  [70,] 6.165001e+00 5.434291e-01
##  [71,] 1.109207e+00 8.164308e-01
##  [72,] 2.942025e+00 1.029671e+00
##  [73,] 1.465612e+00 1.191381e-02
##  [74,] 1.149261e+00 2.729258e-02
```

```
##  [75,] 7.721253e-01 6.195814e-01
##  [76,] 9.049053e-01 1.200974e+00
##  [77,] 1.810296e-01 3.418750e-02
##  [78,] 4.516070e+00 5.804111e-01
##  [79,] 7.676351e-03 5.377984e-01
##  [80,] 5.222436e+00 1.018067e+00
##  [81,] 2.216609e-01 6.471199e-04
##  [82,] 1.304347e+01 2.406256e-01
##  [83,] 6.640122e+00 2.061183e+00
##  [84,] 3.300854e+00 1.349646e+00
##  [85,] 8.363811e+00 2.643171e+00
##  [86,] 8.405644e-01 2.322476e-01
##  [87,] 5.487713e-03 8.458522e-01
##  [88,] 3.298098e-01 2.067134e-01
##  [89,] 3.469750e-01 4.298305e-02
##  [90,] 6.888410e+00 1.977566e+00
##  [91,] 4.399960e-03 2.498946e+00
##  [92,] 2.698939e+00 8.058704e-02
##  [93,] 6.509877e+00 2.011236e+00
##  [94,] 2.015085e+00 8.848323e-04
##  [95,] 5.116310e-01 1.227406e+00
##  [96,] 1.887914e-01 4.166096e-03
##  [97,] 5.493073e+00 8.298818e-01
##  [98,] 5.119875e+00 4.810831e-01
##  [99,] 9.097983e-01 1.062984e-01
## [100,] 1.682643e+00 1.239974e+00
## [101,] 3.500266e+00 4.284546e-01
## [102,] 1.323405e+01 1.079869e+00
## [103,] 1.245571e+01 9.215819e-01
## [104,] 4.027373e-02 5.205630e+00
## [105,] 1.159836e+00 1.093858e-01
## [106,] 1.606100e-01 3.432131e-01
## [107,] 1.359956e-01 1.079360e+00
## [108,] 2.602455e+00 3.459901e+00
## [109,] 1.693128e+00 5.267851e-01
## [110,] 1.534551e+00 1.107200e+00
## [111,] 2.208210e+00 2.940744e+00
## [112,] 1.207218e+00 9.253440e-01
## [113,] 4.791686e-01 5.086232e+00
## [114,] 1.165876e+00 5.509617e-01
## [115,] 4.298651e+00 3.011696e+00
## [116,] 4.070592e+00 7.400450e+00
## [117,] 1.825481e+00 4.221173e-01
## [118,] 2.795217e-01 2.714505e+00
## [119,] 3.033666e+00 1.842485e+00
## [120,] 5.663058e-01 6.545621e-01
## [121,] 2.335469e-02 5.920052e+00
## [122,] 1.599873e+00 1.915646e+00
## [123,] 1.780653e+00 8.669548e-02
## [124,] 1.498309e+00 2.216029e-01
## [125,] 2.145319e+00 6.120291e+00
## [126,] 2.636732e+00 5.805061e-02
## [127,] 4.705435e+00 4.309379e+00
## [128,] 2.388731e-01 4.588343e+00
```

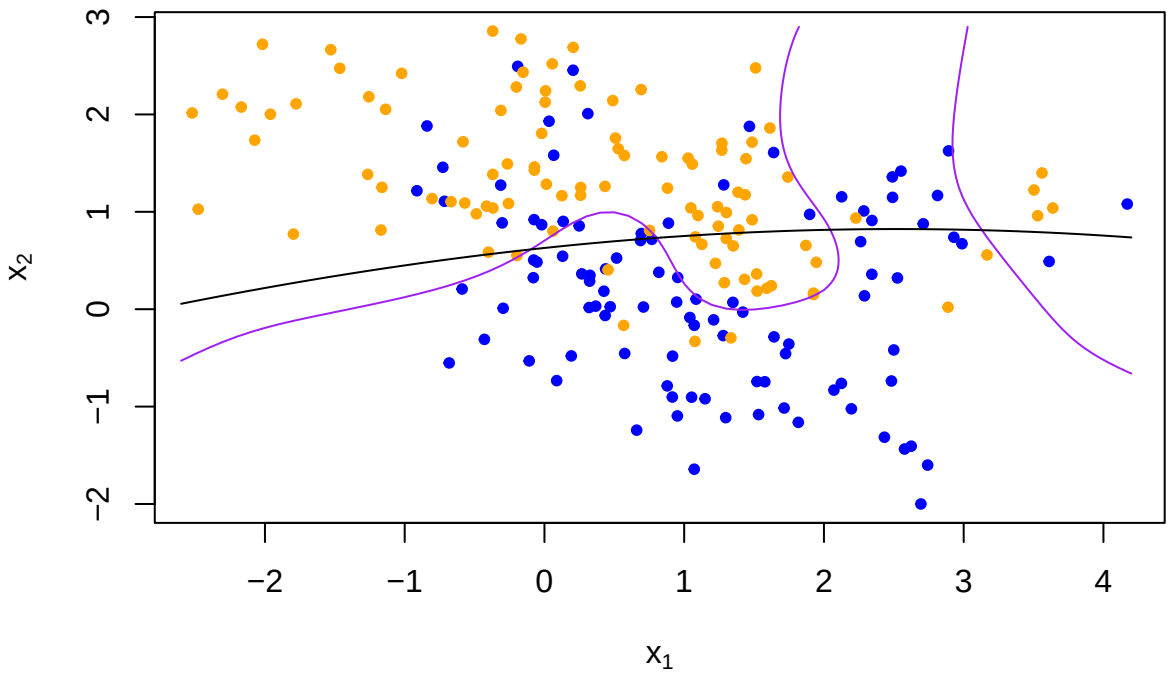
```
## [129,] 1.291721e+00 4.213521e+00
## [130,] 5.307784e+00 4.873007e+00
## [131,] 3.838863e-04 3.259106e+00
## [132,] 2.338164e+00 7.099711e+00
## [133,] 5.083001e-03 2.133187e+00
## [134,] 7.748064e-01 1.544724e+00
## [135,] 1.952484e-04 1.646566e+00
## [136,] 1.879941e-01 1.591043e+00
## [137,] 1.611165e+00 2.666893e+00
## [138,] 2.051014e+00 9.392697e-02
## [139,] 2.806424e-02 7.701693e+00
## [140,] 5.180310e-03 2.038935e+00
## [141,] 9.712446e-02 4.166083e+00
## [142,] 4.967177e+00 8.765141e-01
## [143,] 3.157466e+00 4.447058e+00
## [144,] 1.097167e+00 1.081592e+00
## [145,] 6.468886e-01 1.289743e+00
## [146,] 1.120102e+00 2.219592e+00
## [147,] 3.738777e-03 6.422784e-01
## [148,] 1.563035e-02 1.358609e+00
## [149,] 6.137598e+00 1.053715e+00
## [150,] 2.080641e+00 2.388498e+00
## [151,] 1.057739e+00 2.405556e+00
## [152,] 3.391393e-01 2.954567e+00
## [153,] 3.844657e+00 4.009102e+00
## [154,] 8.340238e+00 4.282004e-04
## [155,] 1.935860e+00 6.621430e-01
## [156,] 3.226646e+00 5.955695e-01
## [157,] 1.002971e+01 3.101895e-01
## [158,] 2.374541e-01 9.586574e-01
## [159,] 2.581840e-01 3.087864e+00
## [160,] 2.532315e+00 4.631605e-02
## [161,] 1.920363e+00 1.442934e+00
## [162,] 3.212819e-01 2.759022e-02
## [163,] 1.371292e-01 8.155622e+00
## [164,] 3.711679e+00 2.276005e-02
## [165,] 3.253898e-01 1.188785e+00
## [166,] 1.227189e+01 1.498132e+00
## [167,] 2.282600e+00 6.139816e+00
## [168,] 1.365253e-01 1.917118e+00
## [169,] 2.317429e+00 3.465411e-02
## [170,] 3.156690e-03 6.346661e+00
## [171,] 3.270380e-01 2.489902e+00
## [172,] 2.750670e-05 4.518206e+00
## [173,] 1.269329e+00 4.434393e-01
## [174,] 3.784696e+00 2.314384e-01
## [175,] 1.548627e+00 7.264540e-01
## [176,] 1.717619e-01 1.119675e+00
## [177,] 6.611907e-02 1.177624e+00
## [178,] 1.657455e+00 7.430245e-02
## [179,] 2.062895e+00 1.380088e+00
## [180,] 1.578646e+00 4.759159e+00
## [181,] 1.613636e+00 2.904092e+00
## [182,] 6.701925e-02 1.369029e+00
```

```
## [183,] 2.305217e+00 1.308516e-01
## [184,] 7.067722e-01 2.450712e+00
## [185,] 7.009610e-02 2.225564e+00
## [186,] 4.468194e-01 1.218627e+00
## [187,] 2.208932e+00 8.430554e-01
## [188,] 6.730577e-02 1.563395e+00
## [189,] 1.267796e+01 1.957170e+00
## [190,] 1.352018e+00 1.564928e+00
## [191,] 1.044043e+00 5.865680e+00
## [192,] 1.367166e+00 6.609163e-01
## [193,] 4.221222e-02 7.228342e+00
## [194,] 6.354532e+00 4.060359e+00
## [195,] 2.073361e-01 1.632467e-01
## [196,] 6.592068e-02 5.260622e+00
## [197,] 3.706293e+00 2.724237e-02
## [198,] 1.695050e+00 9.844601e-01
## [199,] 6.610594e-05 5.027747e+00
## [200,] 3.851262e-02 3.040459e-01
```

```
## 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]))
```



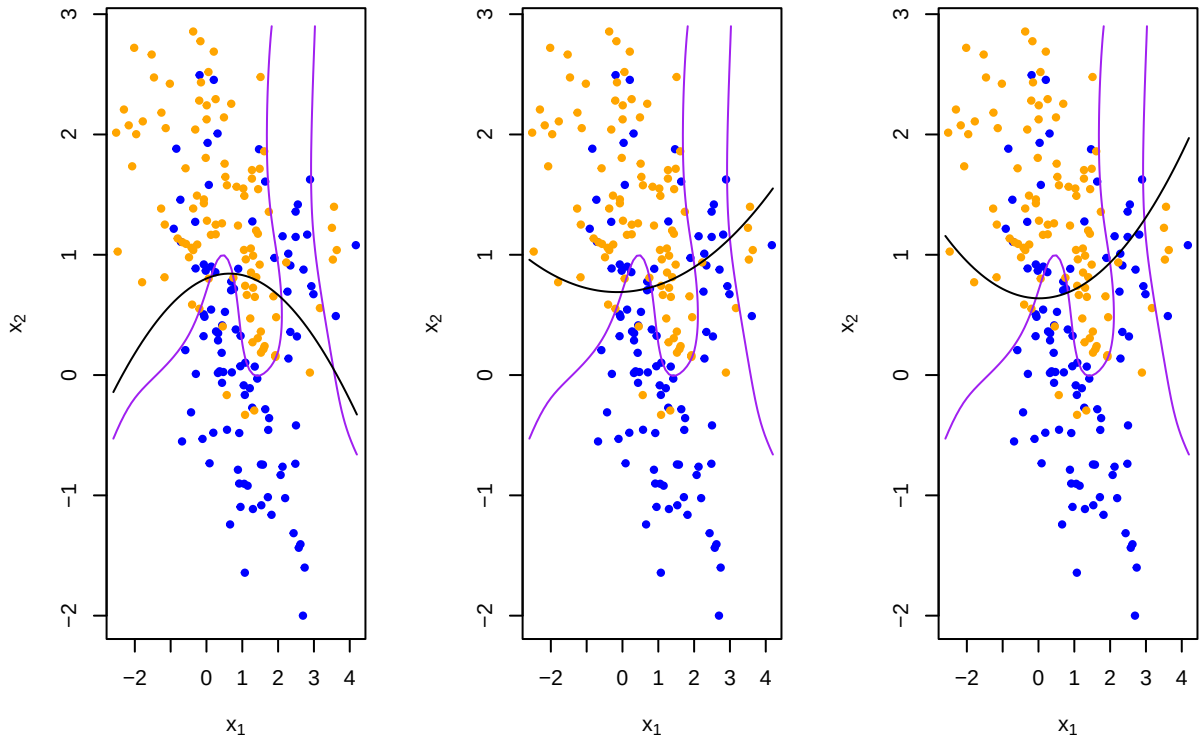
```
## find the contours in 2D space such that lc_pred == 0.5
lc_cont_new <- contourLines(dat$px1, dat$px2, lc_pred_new, levels=0.5)

## 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)
}
```




```
var(as.vector(lc_pred))
```

```
## [1] 0.1133352
```

```
var(as.vector(lc_pred_new))
```

```
## [1] 0.151983
```

```
result1 <- as.vector(lc_pred)
result2 <- as.vector(lc_pred_new)
p1<-rep(0,1)
p2<-rep(0,1)
for (i in 1:length(result1)){
  if ((result1[i] >= 0.5 & dat$marginal[i]>= 0.5)|(result1[i] <= 0.5 & dat$marginal[i]<= 0.5)){
    p1[i] <- 0
  }
  else{
    p1[i] <- 1
  }

  if ((result2[i] >= 0.5 & dat$marginal[i]>= 0.5)|(result2[i] <= 0.5 & dat$marginal[i]<= 0.5)){
    p2[i] <- 0
  }
  else{
    p2[i] <- 1
  }
}
mean(p1)
```

```
## [1] 0.4261455
```

```
mean(p2)
```

```
## [1] 0.3917435
```

From the result we can see that the variance of the fit model which has squared x has smaller bias and variance