Stats 102B - Week 6, Lecture 3

Miles Chen, PhD

Department of Statistics

Week 6 Friday

UCLA

Section 1

Classification - K-nearest neighbors

K Nearest Neighbors

Training data is provided. All of the training data is labeled with a class.

Summary of the algorithm for a new test case

1. Measure the distance (generally Euclidean) from the test case to every single other point in the training data (this can be very computationally expensive)
2. Sort the resulting distances
3. Select the k observations with the smallest distances (the k-nearest-neighbors)
4. The class with the highest representation is selected

How to pick K? Use Cross Validation

Strategies for Resolving Ties

If there are only two classes, and k is odd, no ties will result.

If k is even or if there are more than two classes, its possible to have a scenario where there is a tie among the k-nearest-neighbors.

There's no 'hard rule' about how ties should be resolved.

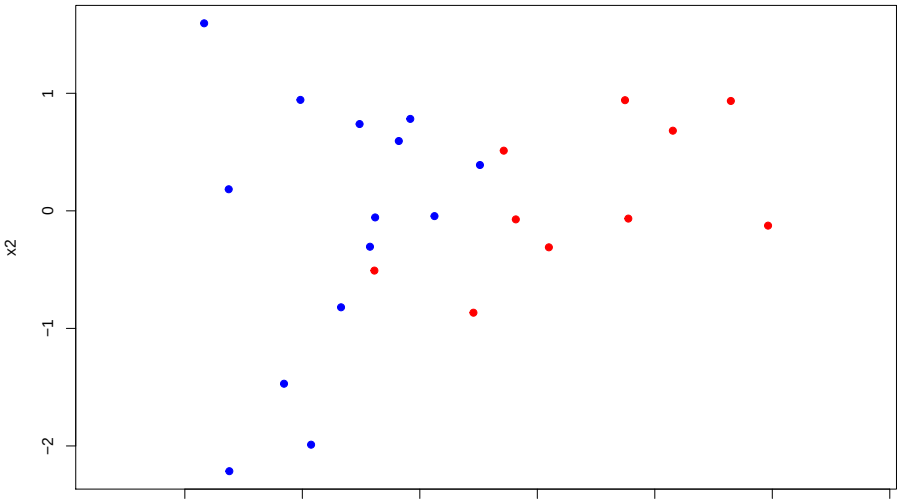
Some options to dealing with ties:

- designate no class: "unclassified"
- switch to 1-nearest-neighbor
- select randomly among the top tied groups
- among the top tied groups, select the group with the closest neighbor (not necessarily the same as the class of the 1-nearest-neighbor)

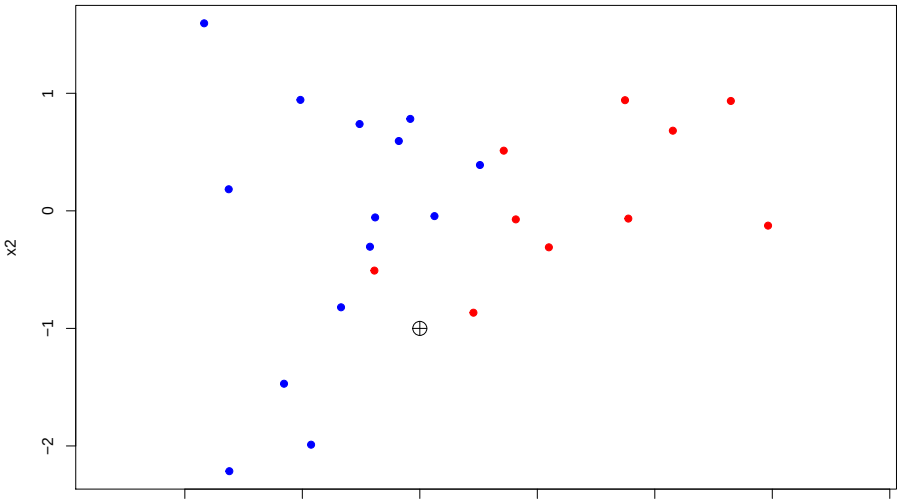
KNN - training data

```
library(mvtnorm)
set.seed(1)
xa <- rmvnorm(14, mean = c(-1,0), sigma = diag(2)) # class a
xb <- rmvnorm(10, mean = c(1.3,0), sigma = 1.5*diag(2)) # class b
x <- rbind(xa,xb) # training data
left <- min(x[,1]); right <- max(x[,2]); bot <- min(x[,2]); top <- max(x[,2])
plot(xa, ylim = c(bot,top), xlim = c(left,right), asp = 1, pch = 19, col = 'bl
points(xb, col = 'red', pch = 19)
```

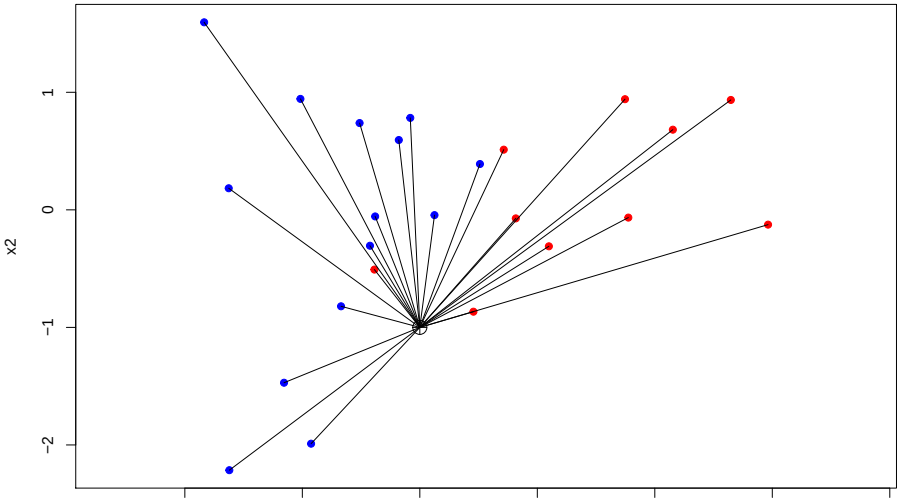
KNN - training data



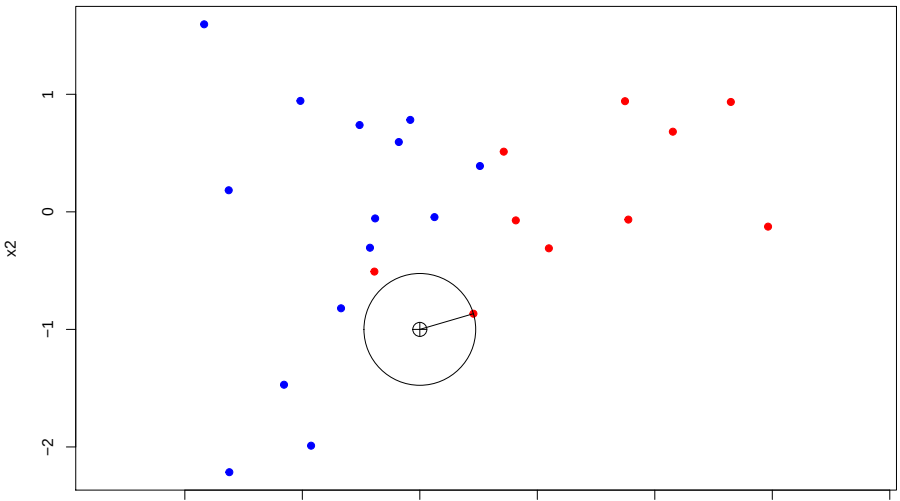
K Nearest Neighbors - test case at (0,-1)



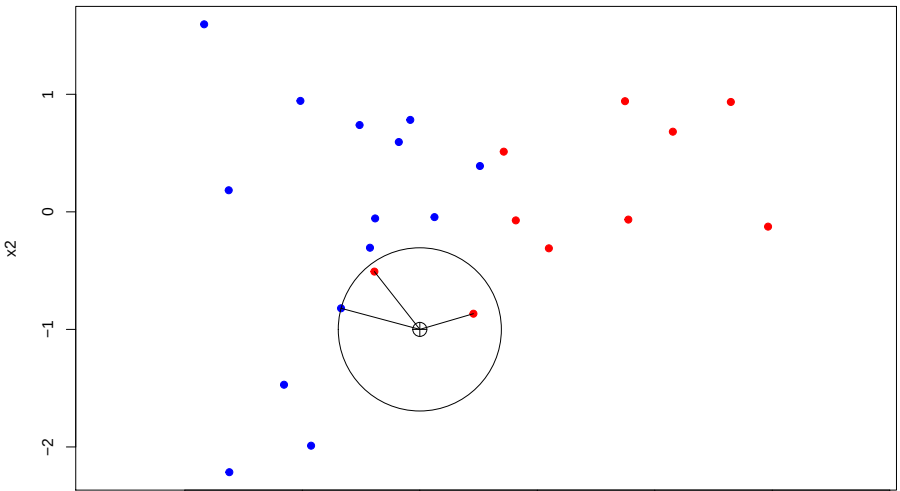
Calculate Distances



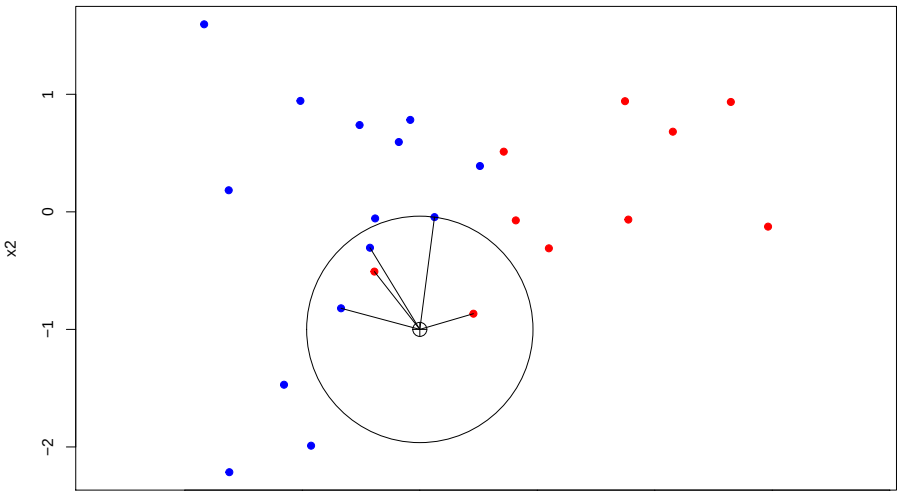
k=1 chooses Red



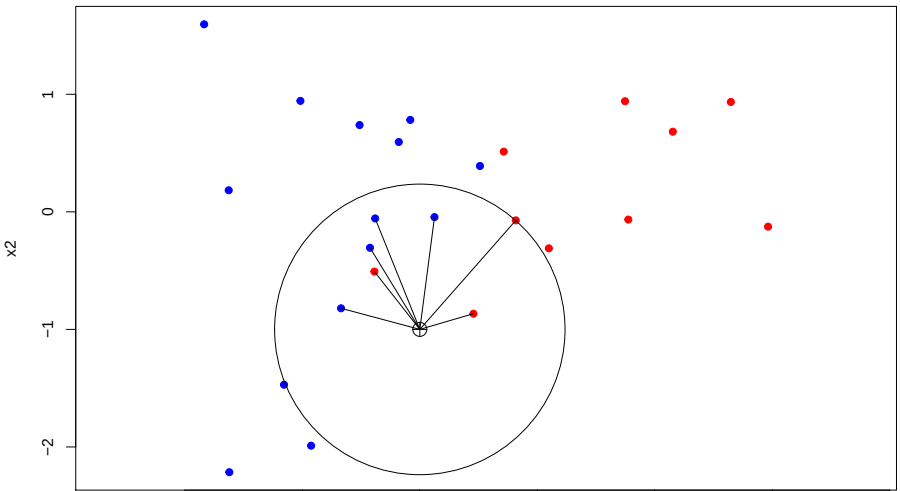
k=3 chooses Red (2 Red vs 1 Blue)



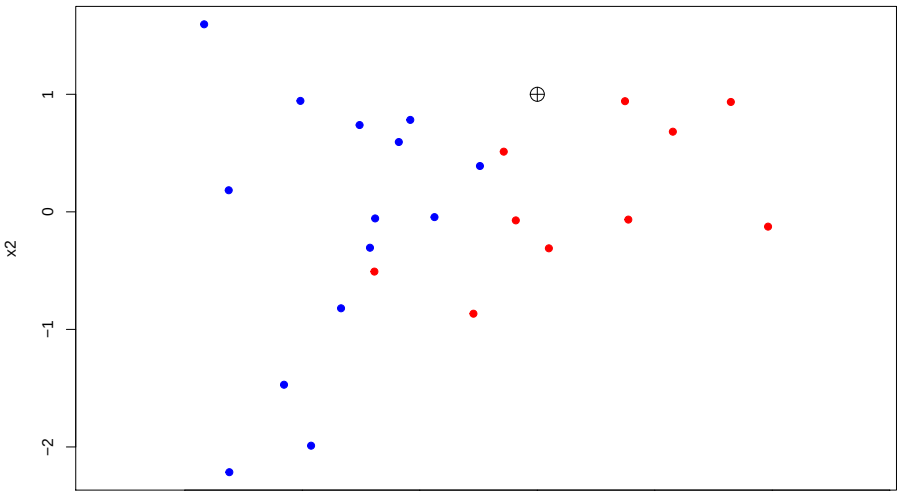
k=5 chooses Blue (3 Blue vs 2 Red)



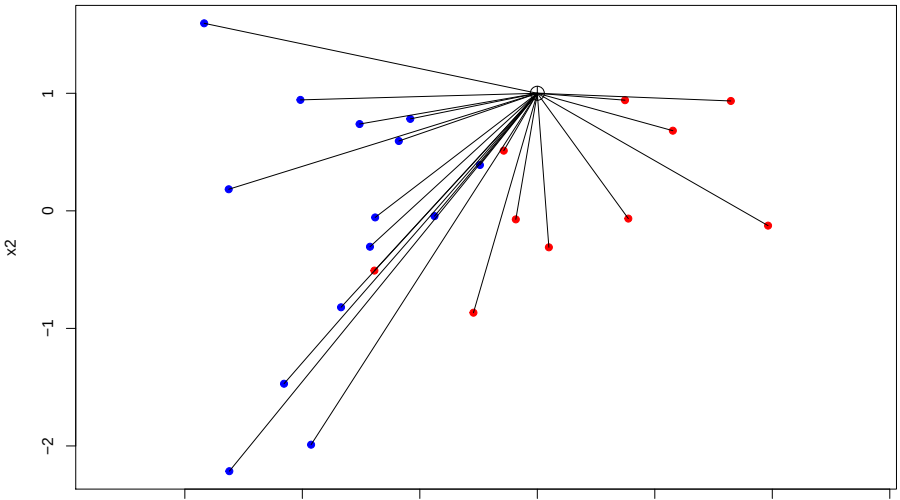
k=7 chooses Blue (4 Blue vs 3 Red)



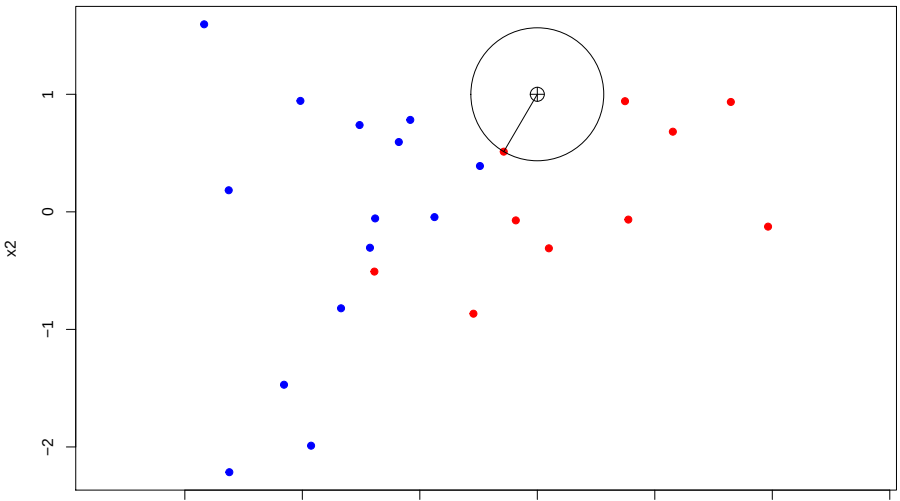
K Nearest Neighbors - test case at (1,1)



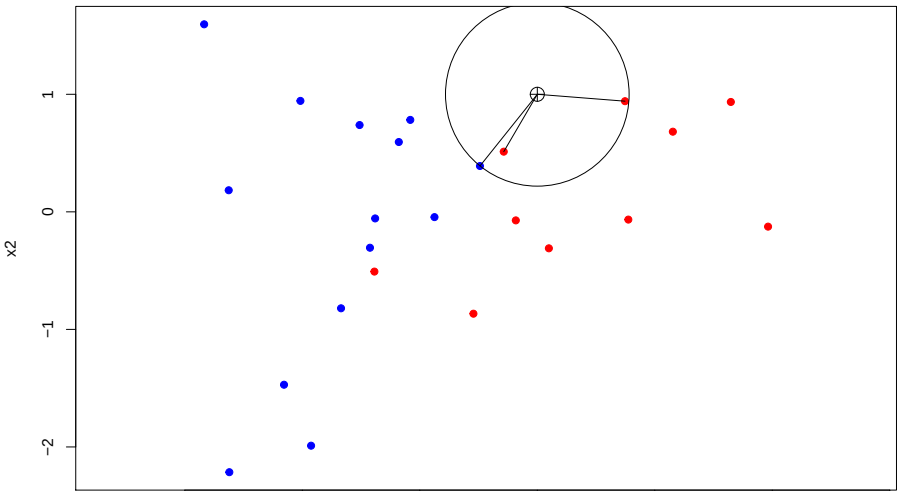
Calculate Distances



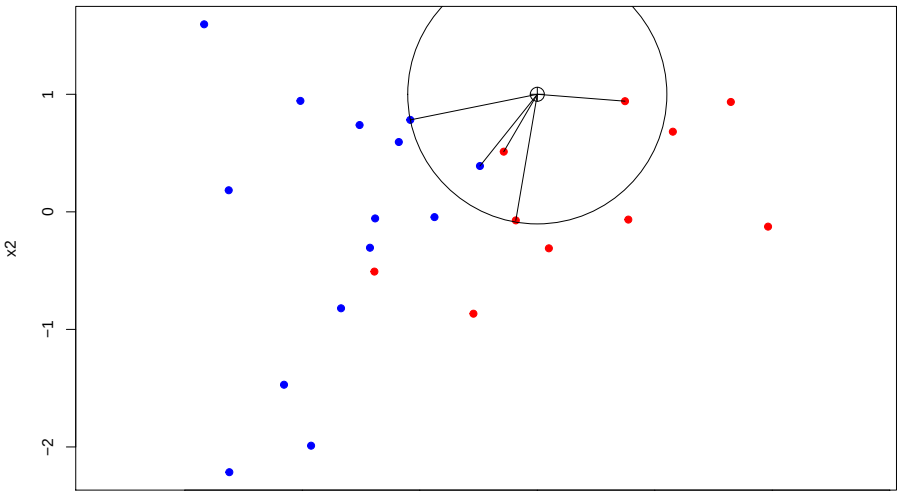
k=1 chooses Red



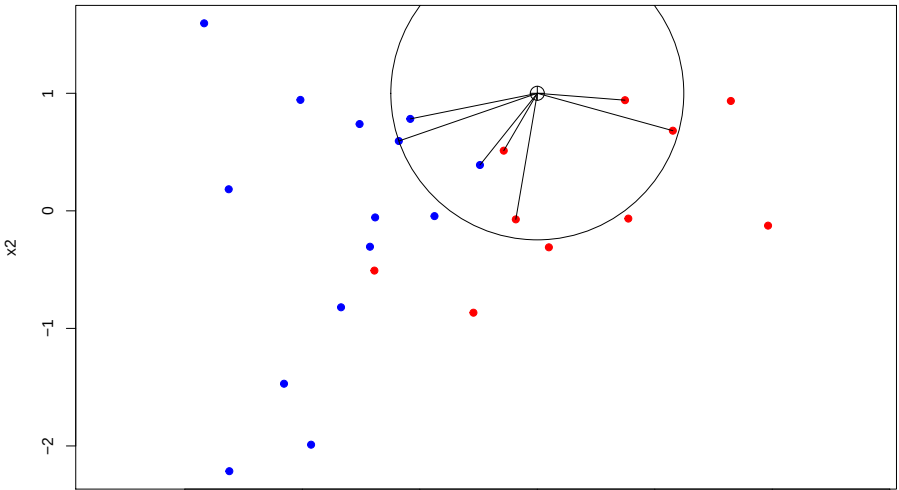
k=3 chooses Red



k=5 chooses Red



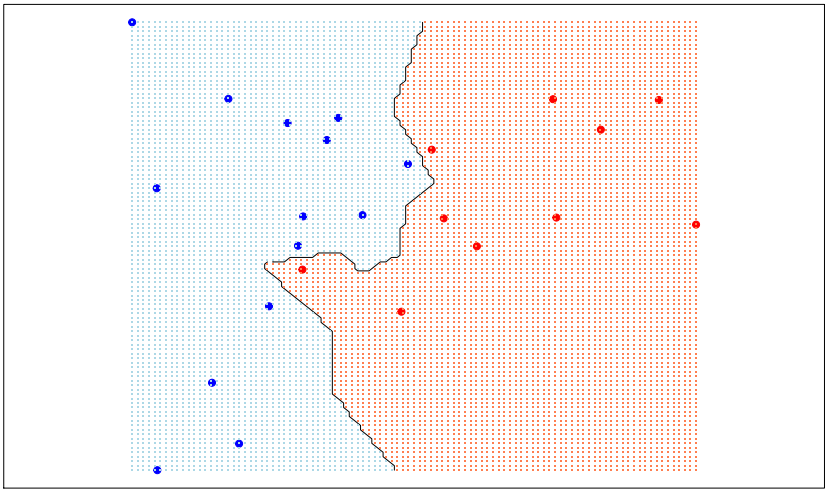
k=7 chooses Red



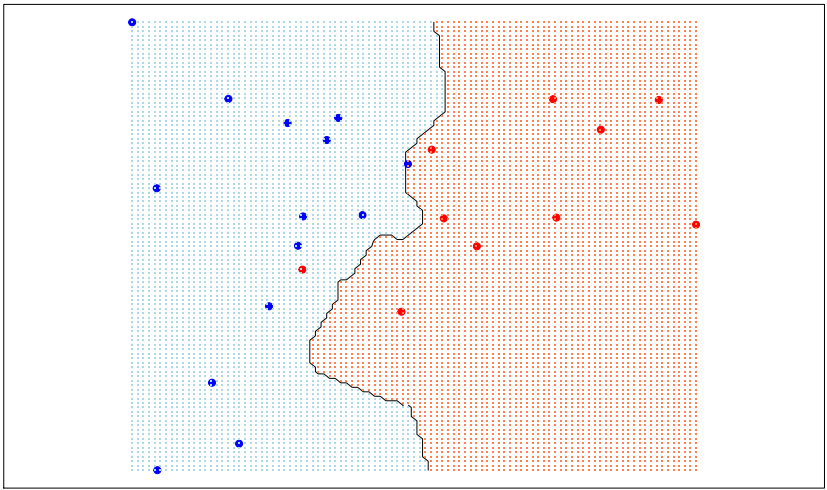
KNN Boundary Regions

```
# https://stats.stackexchange.com/questions/21572/how-to-plot-decision-bounda
require(class) # has the knn() function
x <- x # training data
g <- c(rep(0,14), rep(1,10)) # class labels
px1 <- seq(  -3, 3, length.out = 101) # range of x1
px2 <- seq(-2.5, 2, length.out = 101) # range of x2
xnew <- expand.grid(px1,px2) # grid of points
mod1 <- knn(x, xnew, g, k= 1 , prob=TRUE) # the knn model
prob <- attr(mod1, "prob"); prob <- ifelse(mod1=="1", prob, 1-prob) # extract
prob1 <- matrix(prob, length(px1), length(px2))
contour(px1, px2, prob1, levels=0.5, labels="", xlab="", ylab="", axes=FALSE)
points(x, col=ifelse(g==1, "red", "blue"), pch = 19)
points(xnew, pch=".", cex=1.2, col=ifelse(prob1>0.5, "coral", "lightblue")); b
```

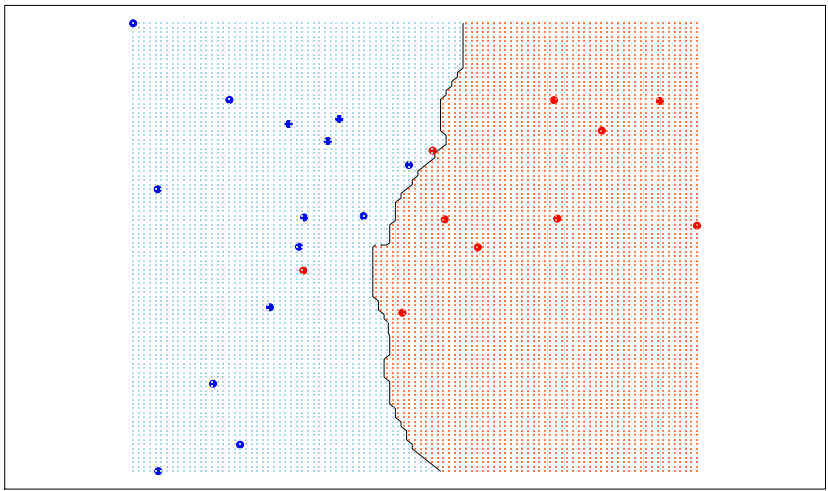
KNN Boundary Regions k = 1



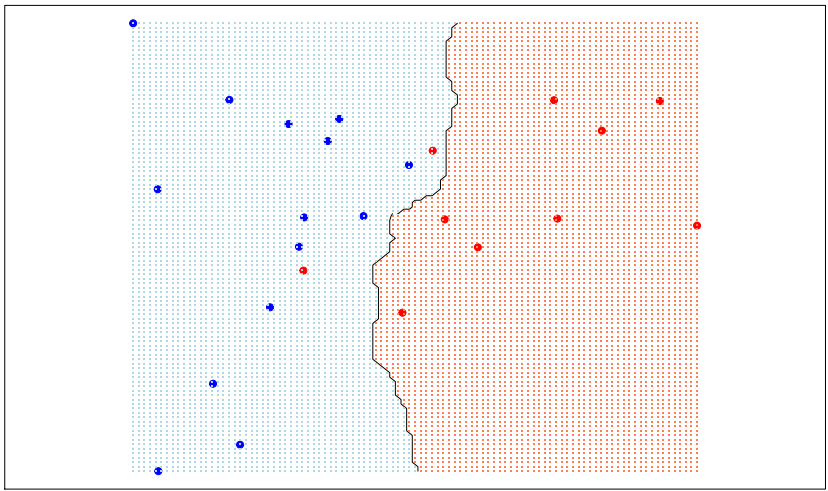
KNN Boundary Regions k = 3



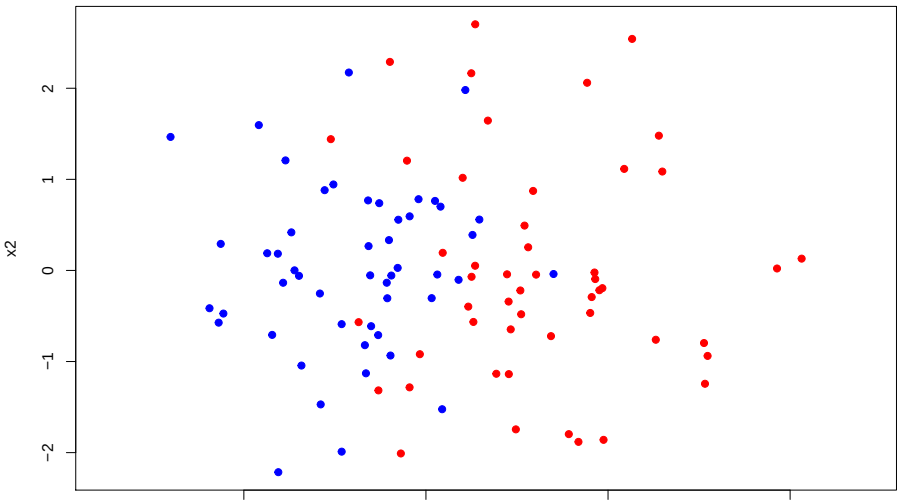
KNN Boundary Regions k = 5



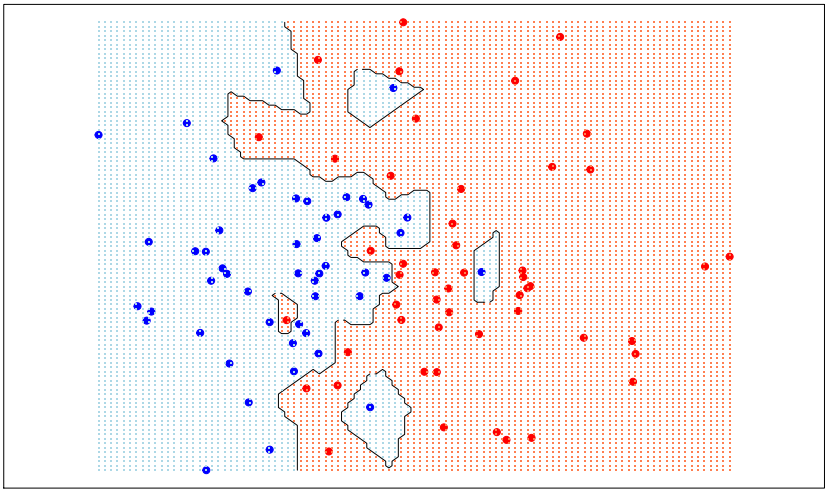
KNN Boundary Regions k = 7



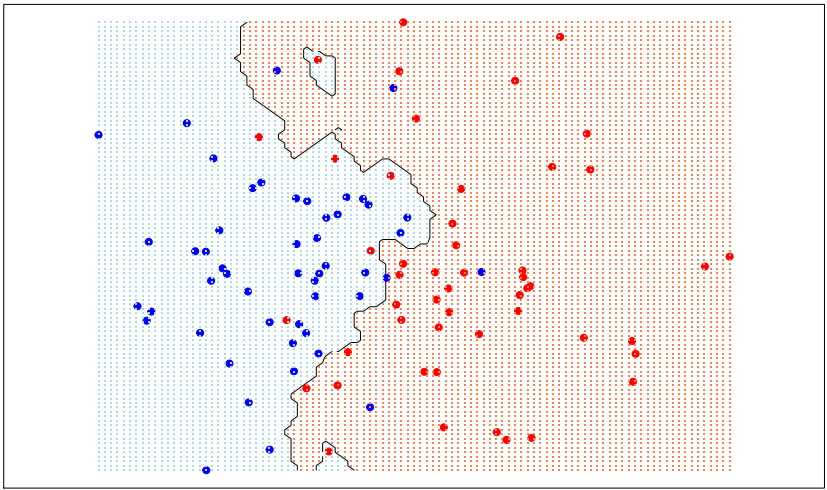
Another Example - more data



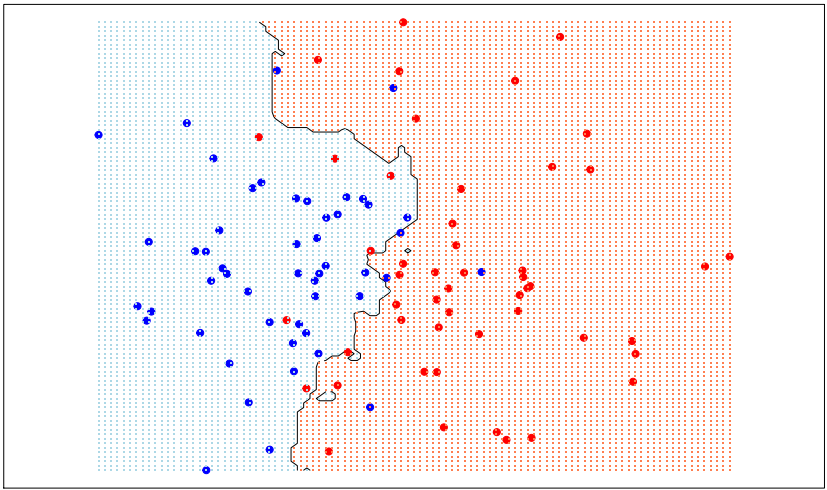
KNN Boundary Regions k = 1



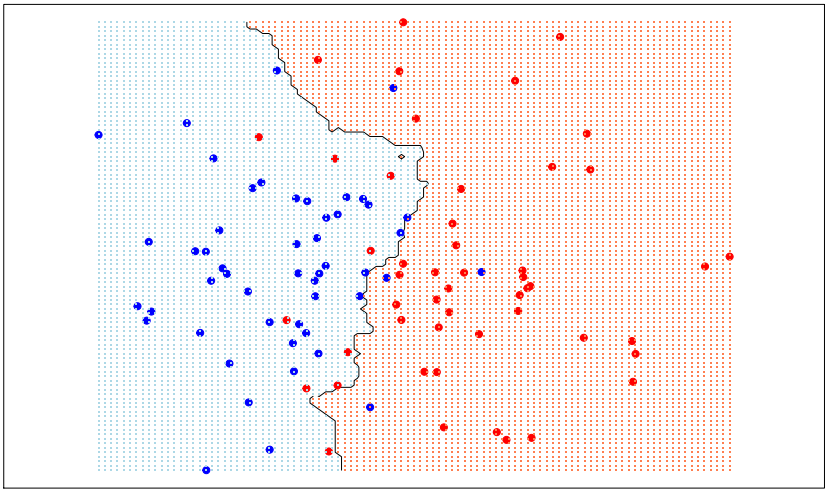
KNN Boundary Regions k = 3



KNN Boundary Regions k = 5



KNN Boundary Regions k = 7



KNN Boundary Regions k = 20