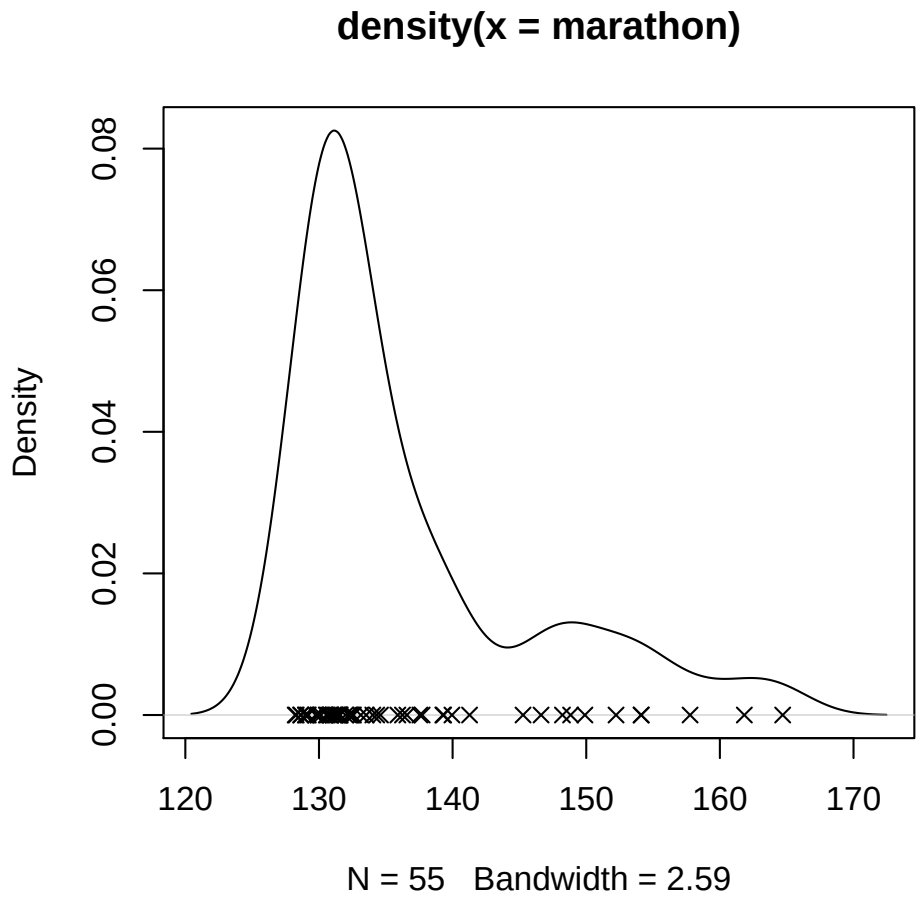
STAT 441: Lecture 14
Classification
via nonparametric density estimation

Once we know the probability density of the groups
we can write the optimal classification rule
Thus the only missing step is to estimate the density

The probability density can be estimated?

```
> attach(Trackmen)
> plot(density(marathon))
> points(marathon,rep(0,length(marathon)),pch=4)
```

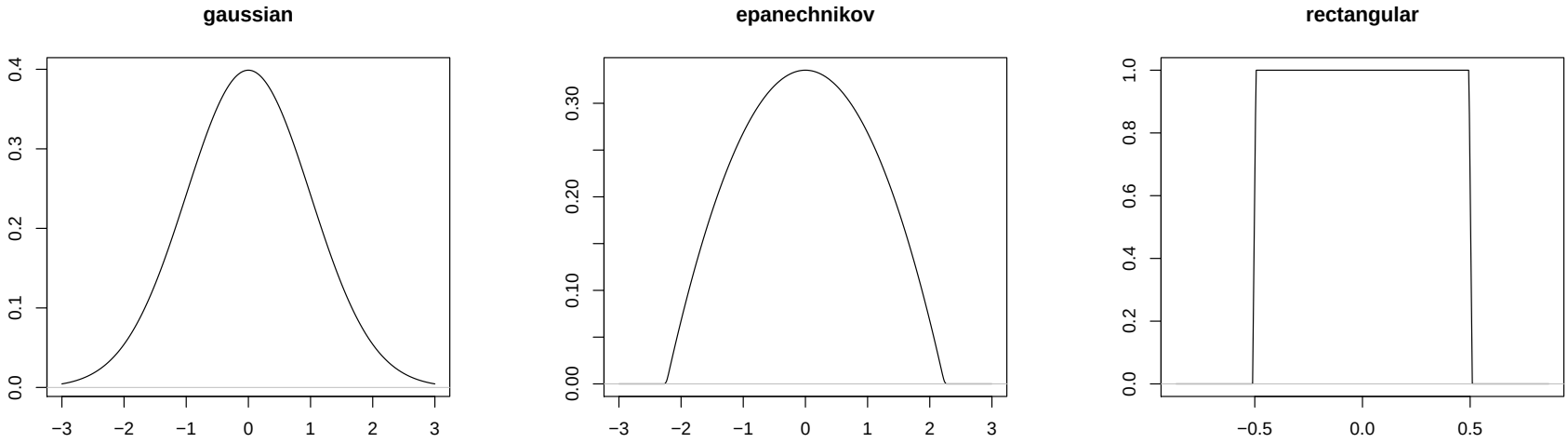


Kernel density estimator

$$\widehat{f}(x) = \frac{1}{nb} \sum_i K\left(\frac{x_i - x}{b}\right)$$

kernel: $\int K(u)\,du = 1$ and also $K(u) \geqslant 0$

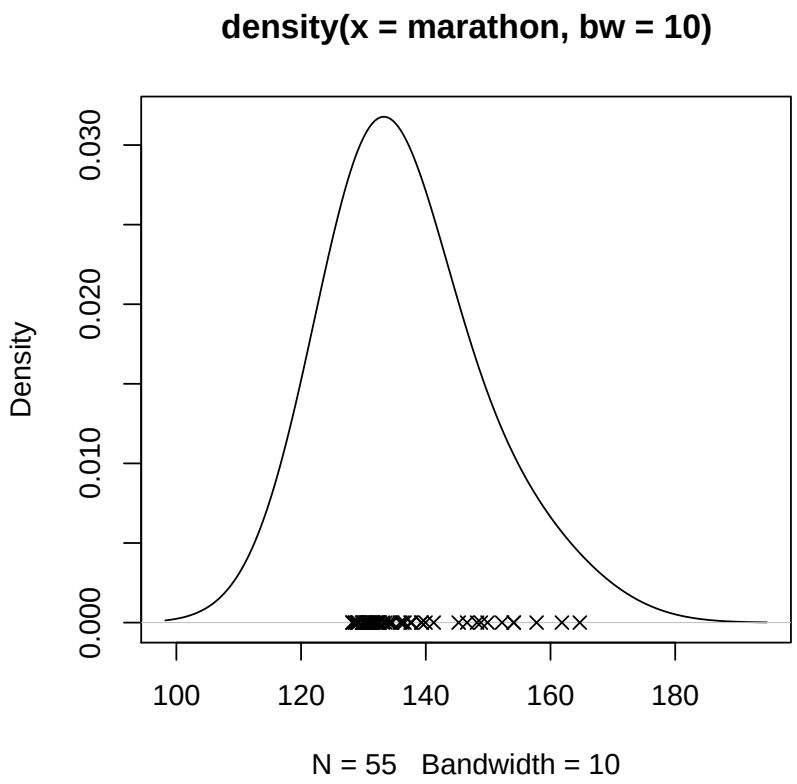
Examples: Gaussian (standard normal density), Epanechnikov, Rectangular (Parzen), and others

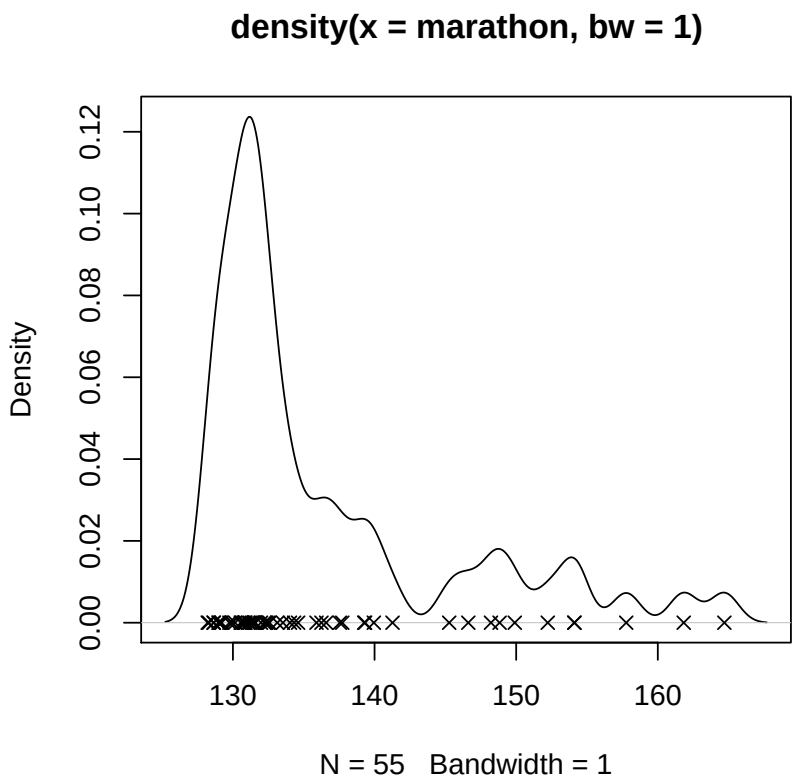


What does rectangular kernel mean? For $b = 1$, $\frac{1}{n}K(x_i - x)$ is the relative proportion number of points falling into $[x-1/2, x+1/2]$; for general b, we obtain the relative proportion of points falling into $[x - b/2, x + b/2]$, divided by the length b of the interval.

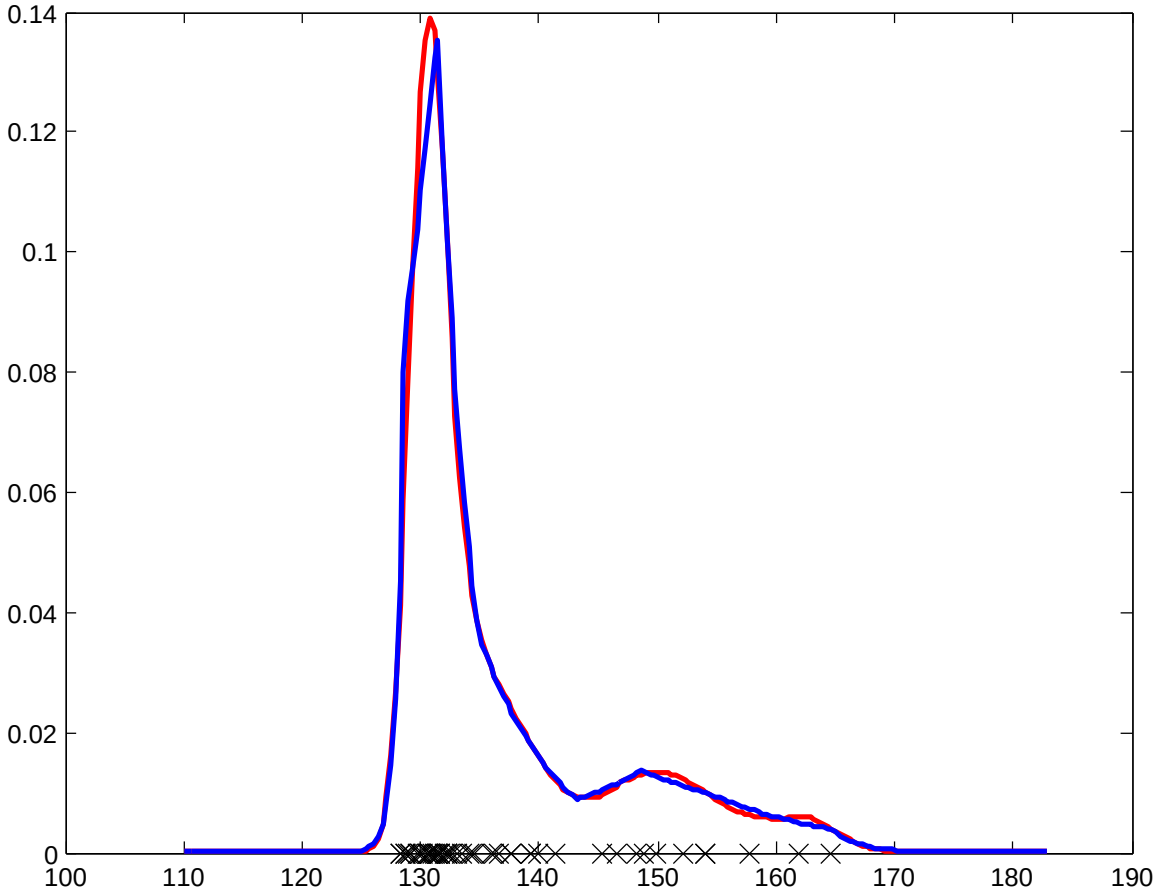
Different bandwidth

The same *bandwidth* $\mathfrak{b}$ may not equally adapt to all parts of the data





Digression: there may be better estimators...



Modification (inversion?) of the kernel idea

For fixed b and x, we take the relative proportion of points x_i falling into $[x - b/2, x + b/2]$ divided by the length of the interval...

Invert: for fixed x, take $k/(nb_k)$ where b_k is the length of an interval $[x - b_k/2, x + b_k/2]$ containing k *nearest neighbors*, neighboring data points x_i of x (this including x itself, if x is a datapoint, equal to some x_i).

Such modification is particularly good for classification, and leads to the

Method of k nearest neighbors:

(0. Scale all variables, so that they have unit standard deviation.)

1. Fix k

2. Given an item $\mathbf{x}_0$ to be classified, find k points in the training sample that are closest to $\mathbf{x}_0$. The posterior probability for the i-th class is the proportion of elements from i-th class among the k nearest neighbors. The classification decision is done by the rule of simple majority (sometimes more elaborate voting schemes may be used).

Properties

Not that bad:

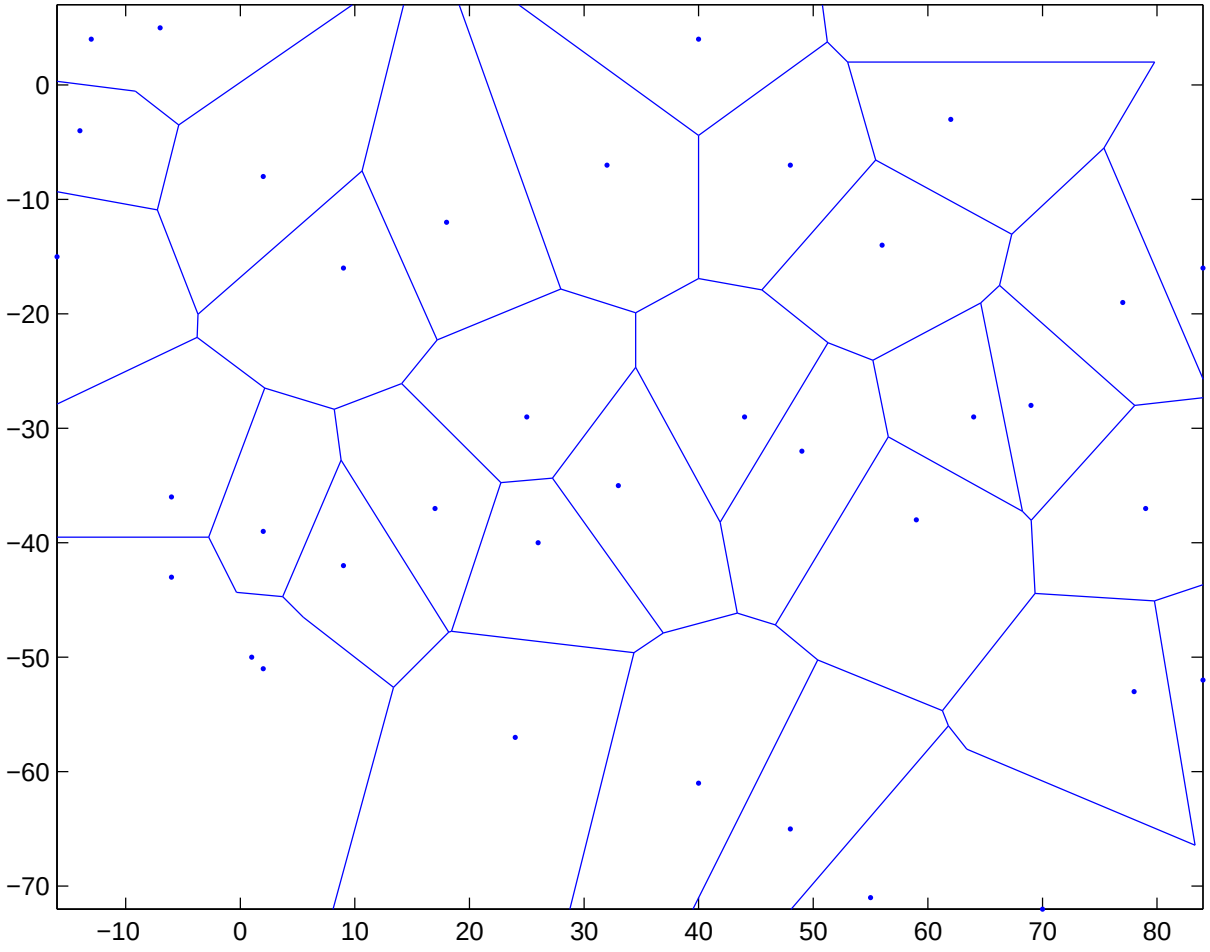
- nice interpretability ("out of k past items with closest features, ℓ belonged to category i")
- well adapts to irregular shapes ("leopard skin") of classification regions

Not that good:

- computationally, need to store all the training set (but then, it can constantly learn)
- somewhat unstable
- not scale (affine) equivariant: scaling is usually a must
- needs to specify k

Aspects of k

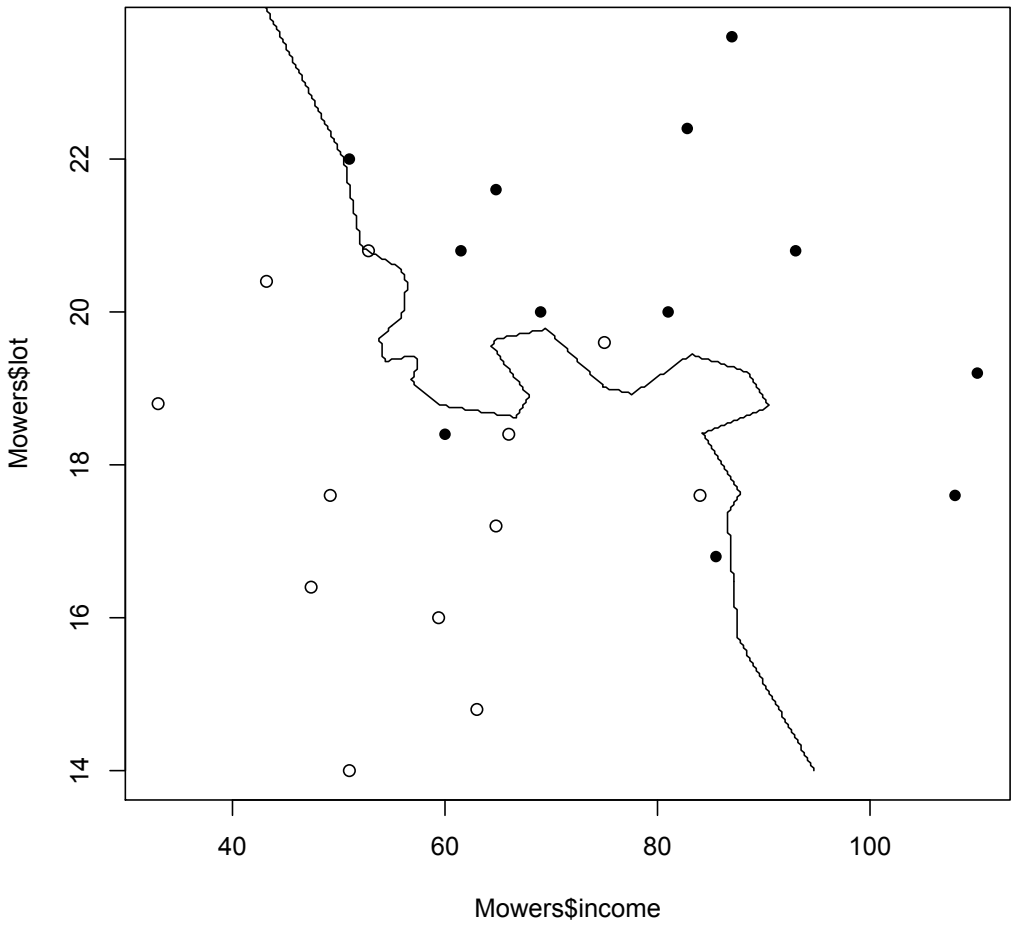
Can be used with $k = 1$; then it divides the feature space to Voronoi (Dirichlet, Thiessen) polygons.



Selection of k by (leave-one-out) cross-validation

```
> library(class)   ## note: a package is needed
> mow=scale(mowers)
> cltab=array(0,dim=c(2,2,10))
> clerr=rep(0,10)
> for (k in 1:10)
+ {
+ cl=rep(0,nrow(mowers))
+ for (j in 1:nrow(mowers))
+ {
+ cl[j]=knn(mow[-j,1:2],mow[j,1:2],mowers[-j,3],k)
+ }
+ cltab[,,k]=table(cl,mowers[,3])
+ clerr[k]=1-(cltab[1,1,k]+cltab[2,2,k])/sum(cltab[,,k])
+ }
> signif(clerr,3)
0.375 0.333 0.167 0.292 0.208 0.208 0.250 0.417 0.250 0.208
1     2     3     4     5     6     7     8     9     10
8 5   7 3   10 2  8 3   9 2    9 2  8 2   7 5   8 2   9 2
4 7   5 9    2 10 4 9   3 10   3 10 4 10  5 7   4 10  3 10
```

Mowers



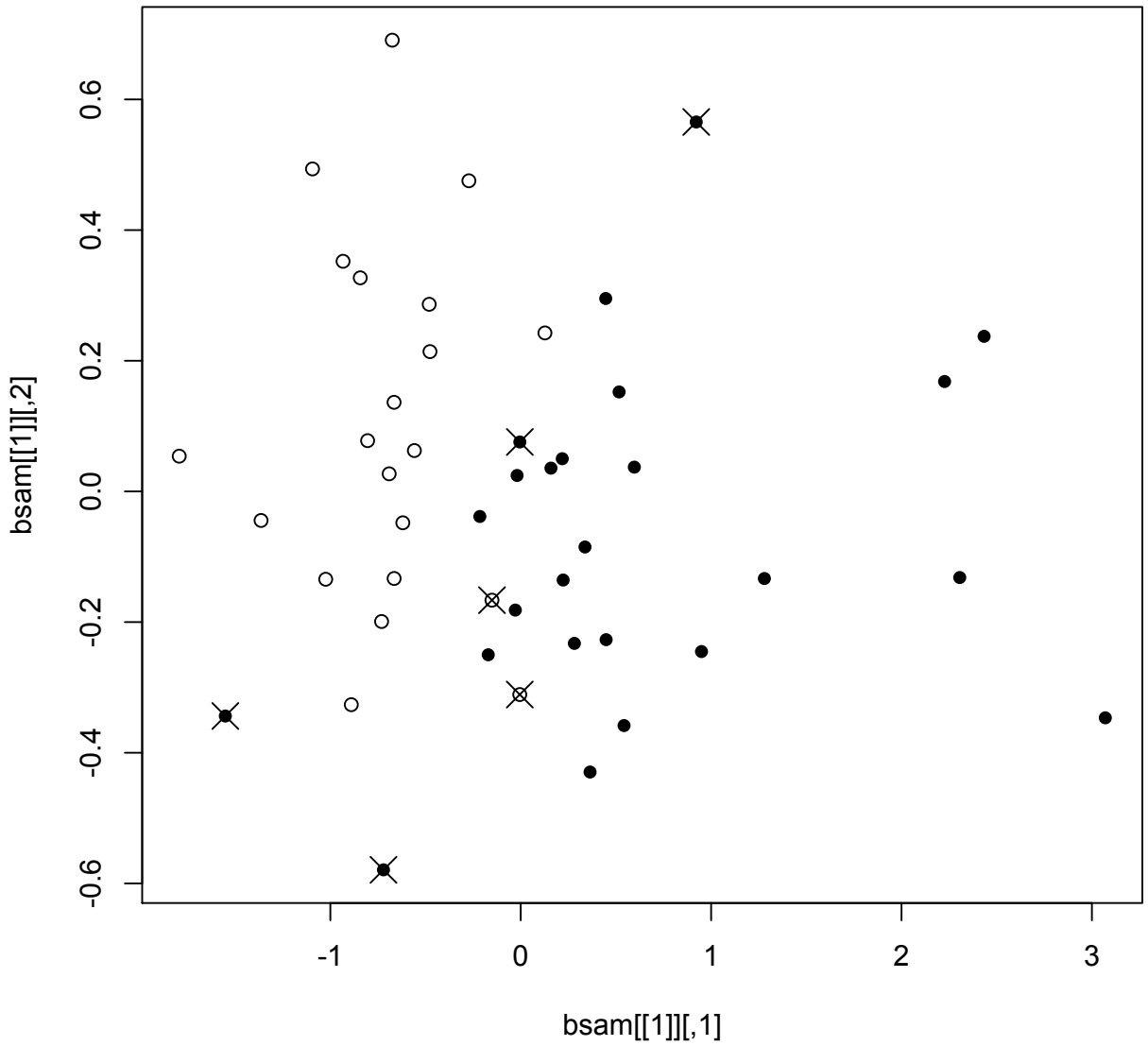

```
> table(knn(mow[,1:2],mow[,1:2],mowers[,3],3),
+ mowers[,3])
    0  1
  0 10  2
  1  2 10
```

Bank data: code

```
> signif(clerr,3)
0.196 0.196 0.152 0.152 0.152 0.109 0.196 0.217 0.196 0.217
> ban = scale(Bank[,1:4])
> banknn=knn(ban,ban,Bank[,5],6,prob=TRUE)
> table(banknn,Bank[,5])

banknn  0  1
     0 19  4
     1  2 21
> bsam = sammon(dist(Bank[,1:4]))
> plot(bsam[[1]],pch=15*Bank[,5]+1)
> points(bsam[[1]][banknn!=Bank[,5],],pch=4,cex=2)
```

And the indicative plot of the result



Problems

```
> table(knn(ban,ban,bankk[,5],6),bankk[,5])
    0  1
  0 18  3
  1  3 22
> table(knn(ban,ban,bankk[,5],6),bankk[,5])
    0  1
  0 19  2
  1  2 23
> table(knn(ban,ban,bankk[,5],6),bankk[,5])
    0  1
  0 19  4
  1  2 21
> table(knn(ban,ban,bankk[,5],6),bankk[,5])
    0  1
  0 19  4
  1  2 21
> table(knn(ban,ban,bankk[,5],6),bankk[,5])
    0  1
  0 19  3
  1  2 22
```

Classification probabilities

```
> banknn$prob
NULL
> signif(attributes(banknn)$prob,3)
 [1] 1.000 1.000 0.833 1.000 0.833 0.833 0.667 1.000 0.833 1.000
     0     0     0     0     0     0     0     0     0     0
[11] 1.000 0.833 0.500 1.000 0.667 0.833 0.833 0.833 0.667 0.667
     0     0     0     0     1     1     0     0     0     0
[21] 1.000 0.833 0.833 0.833 0.833 1.000 1.000 0.833 0.667 0.833
     0     1     1     1     1     1     1     1     0     1
[31] 1.000 1.000 0.833 0.833 1.000 1.000 0.667 0.667 0.833 0.500
     1     1     1     0     1     1     1     1     1     1
[41] 0.500 1.000 0.833 0.833 0.667 1.000
     1     1     1     1     1     1
```