STAT 441: Lecture 21
Classification via regression III: Neural networks

Neural networks: some principles

A somewhat actual way of specifying nonlinear models is via superposition of functions - *neural networks*. The simplest variation on this theme is the "feed-forward one hidden layer neural network"

$$f(x) = \varphi_0\left(\alpha + \sum_h \beta_h \varphi_h\left(\alpha_h + \sum_i \beta_{ih} x_i\right)\right).$$

As a rule, the "hidden layer activation functions" φ_h are logistic; the output function φ_0 is in classification logistic or treshold ($\varphi_0(x) = I(x > 0)$). (Neural networks may be used also for usual regression with quantitative response, where φ_0 may be also linear.) Estimation ("training") of α's and β's is usually done by least squares method.

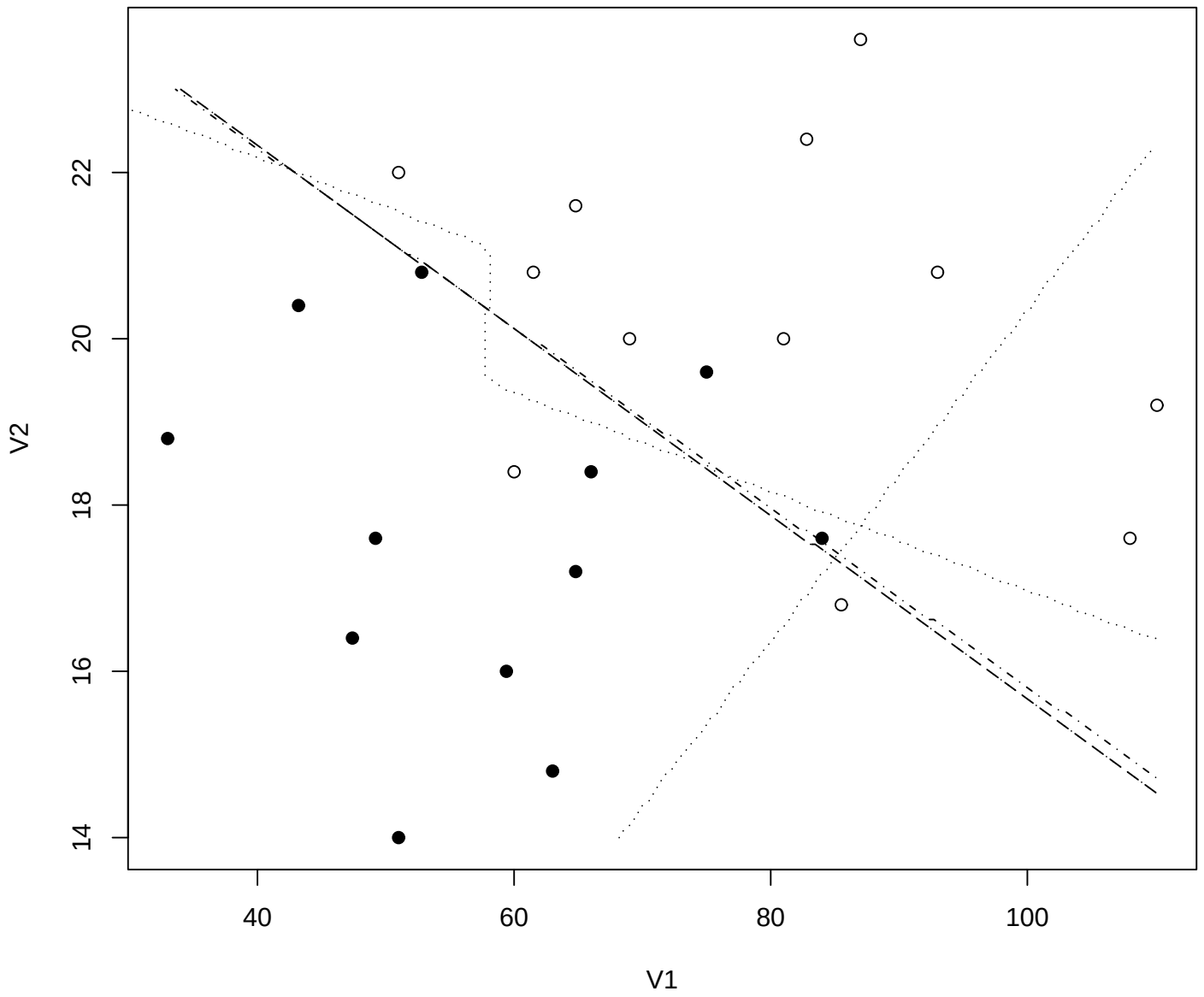
Expressed graphically

Neural networks: some details

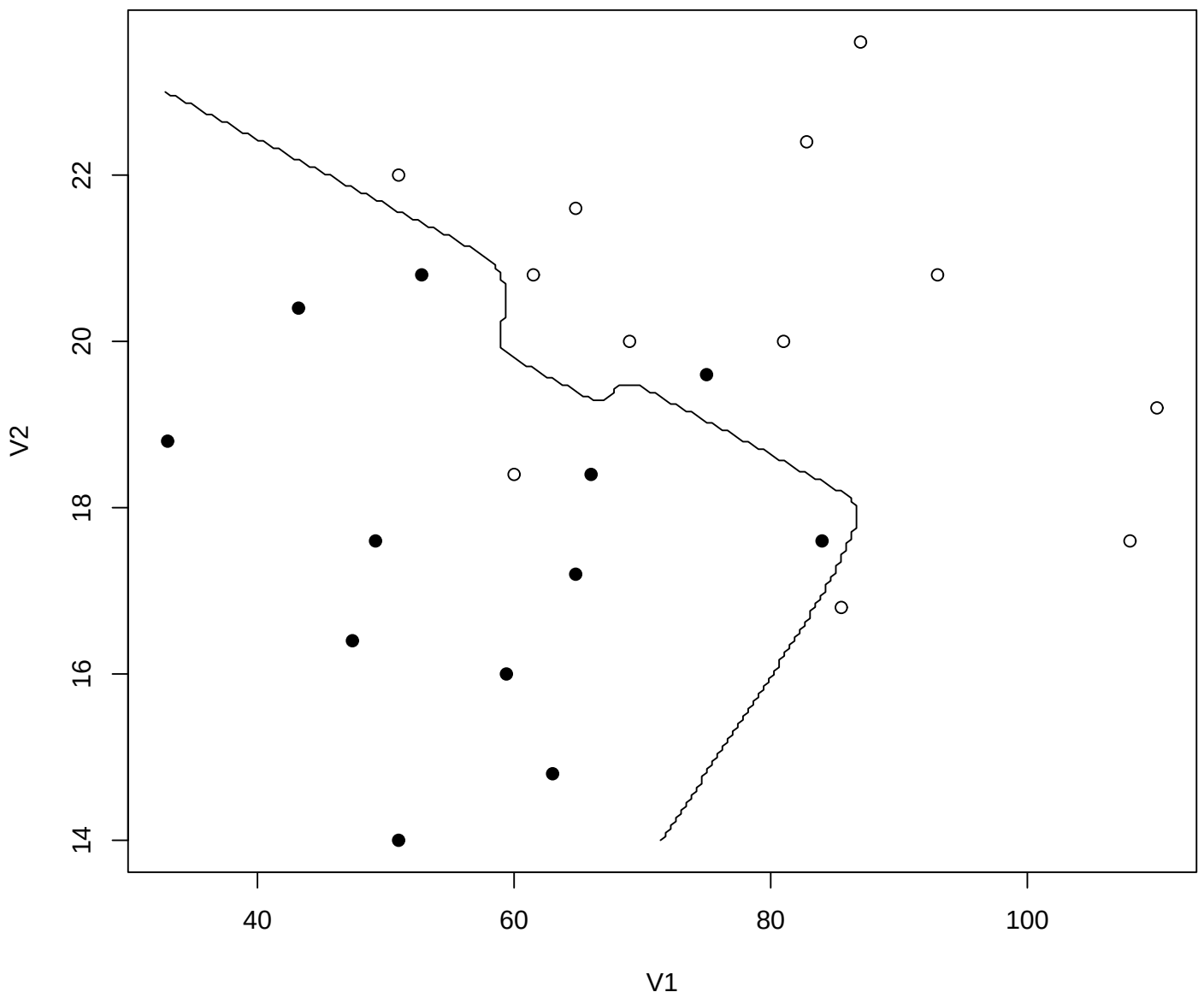
The number of layers and hidden units is chosen in advance. The number of parameters grows quickly, so the main concern is overfitting: the classification fits very well the training data, but performs poorly on the next sample. There are techniques proposed to cope with this; in particular, splitting to training and validation sample almost a must (leave-one-out cross-validation is in most cases computationally too expensive).

The algorithmic side is nontrivial; methods often lead to different local minima, so alternative predictive strategies (averaging) have to be used.

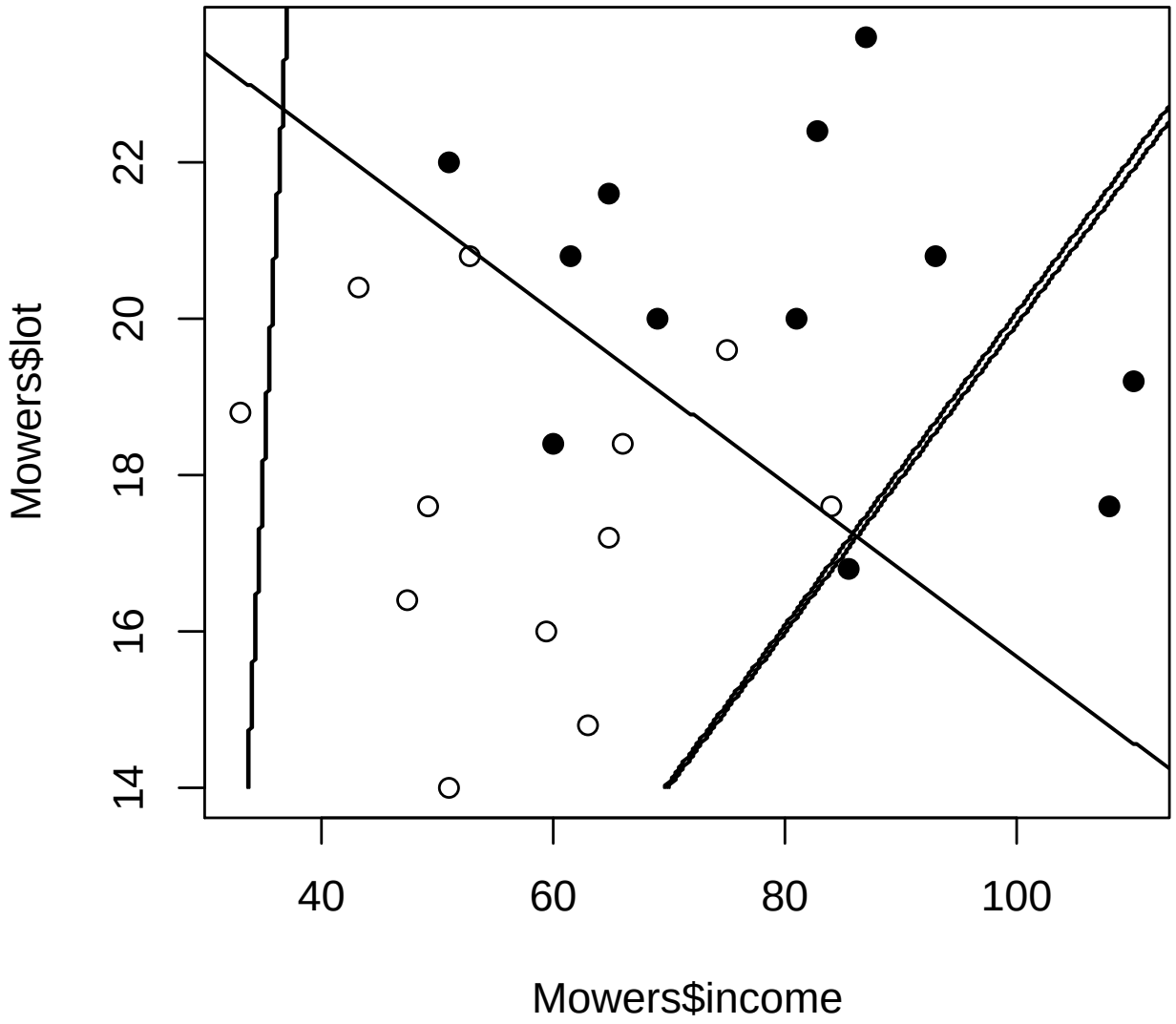
Mowers: neural networks



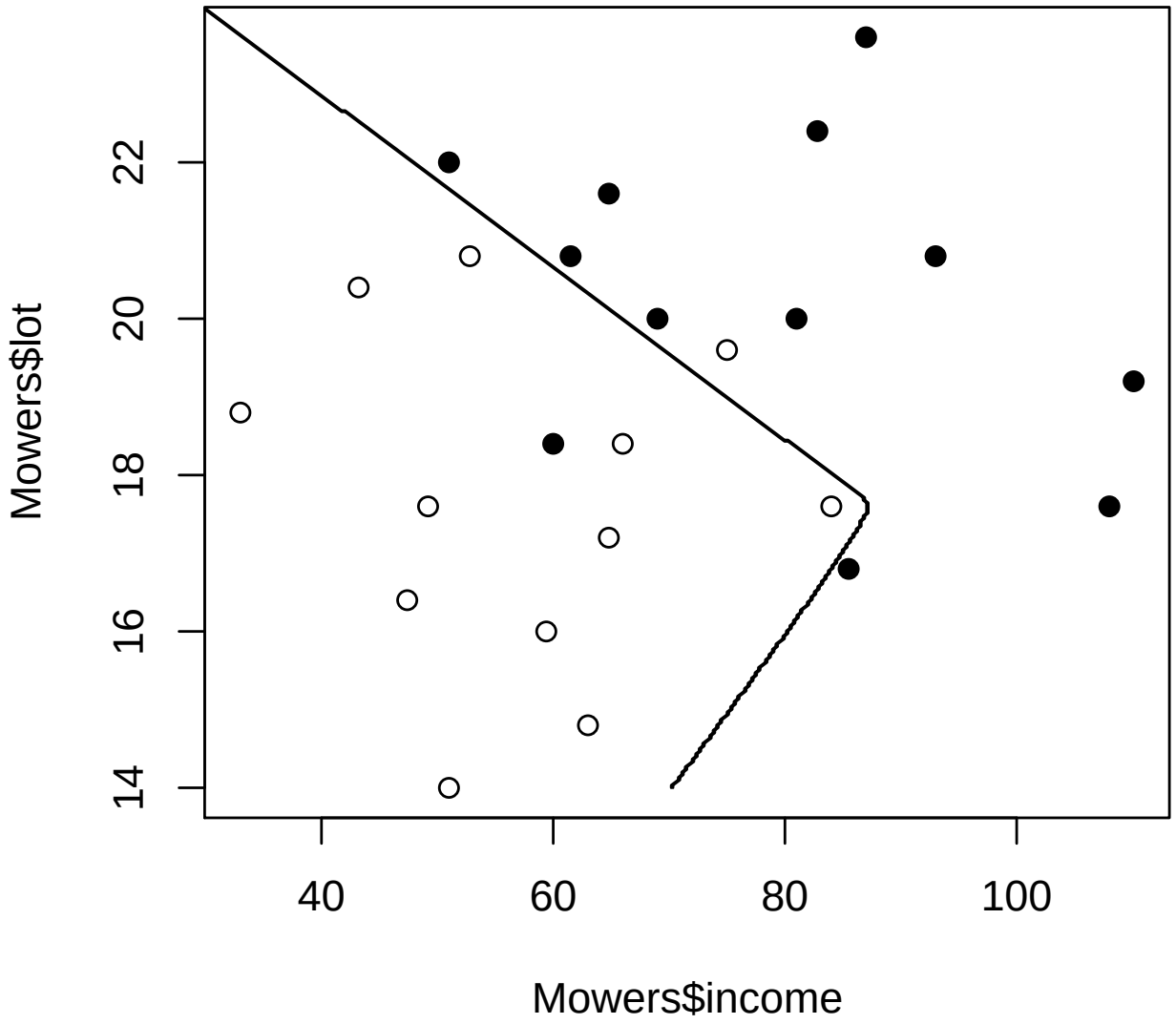
Mowers: averaged neural networks



Reproducibility?



Mowers: averaged neural networks again



Final remarks

Neural networks offer a lot of flexibility, but they are not that transparent regarding their nature.

An objective to find a classifier modeling somewhat human perception is probably lost.

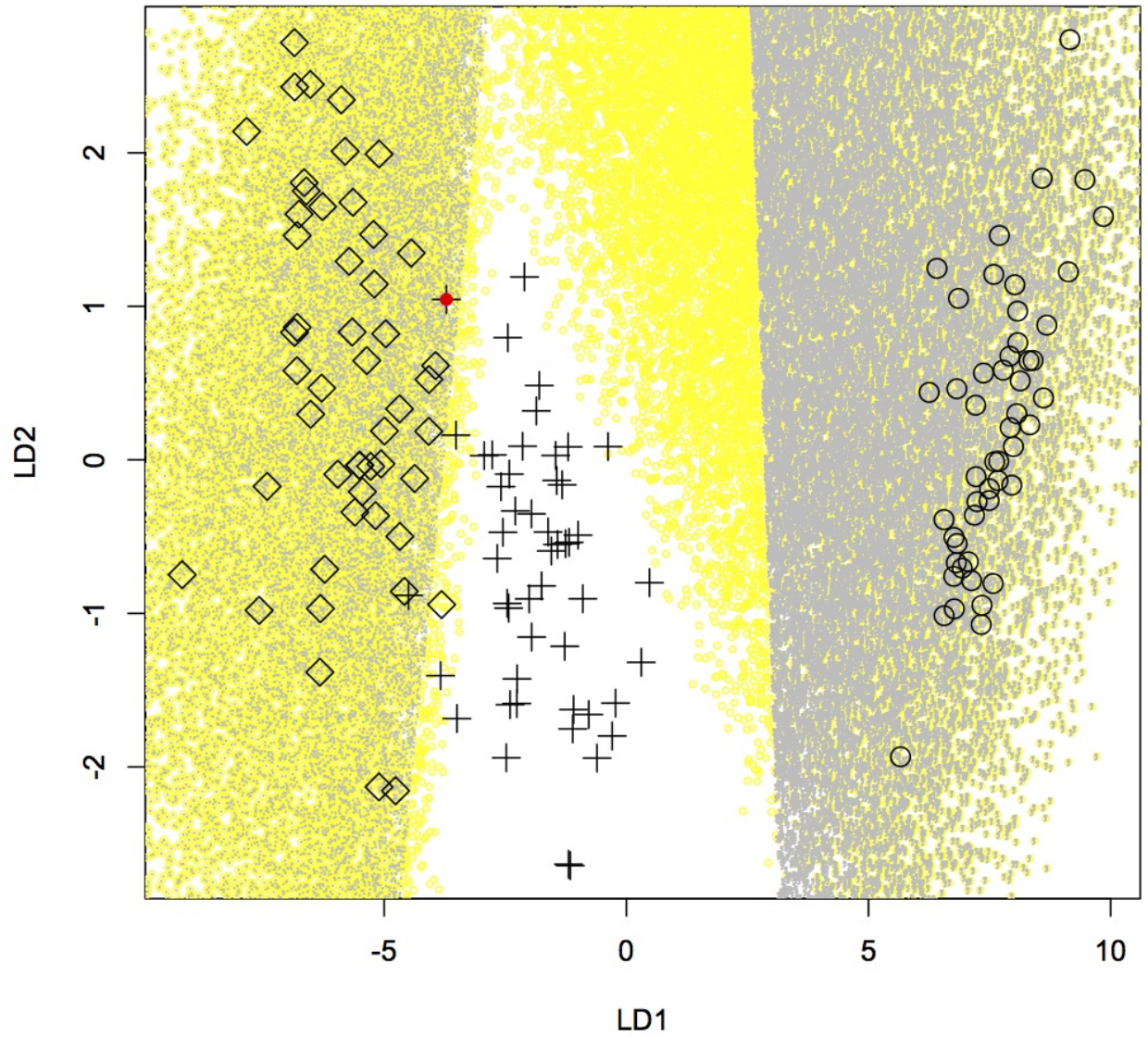
What remains are serious algorithmic problems.

Nevertheless, who knows.

Appendix: also from package nnet

```
require(MASS)
require(nnet)
iris.lda=lda(Species~.,data=iris)
iris.mlr=multinom(Species~.,data=iris)
iris.rng=apply(iris[,1:4],2,range)
iris.pr=matrix(,nrow=100000,ncol=4)
for (k in 1:4)
    iris.pr[,k]=runif(nrow(iris.pr),iris.rng[1,k],iris.rng[2,k])
iris.pr=data.frame(iris.pr)
names(iris.pr)=names(iris)[1:4]
plot(predict(iris.lda)$x,pch=as.numeric(iris$Species),type='n')
iris.prd=predict(iris.lda,newdata=iris.pr)
iris.trt=predict(iris.mlr,newdata=iris.pr)
points(iris.prd$x[iris.trt=="virginica",1],
       iris.prd$x[iris.trt=="virginica",2],col='yellow',cex=0.5)
points(iris.prd$x[iris.trt=="setosa",1],
       iris.prd$x[iris.trt=="setosa",2],col='yellow',cex=0.5)
points(iris.prd$x[iris.prd$class=="virginica",1],
       iris.prd$x[iris.prd$class=="virginica",2],pch='.',col='grey')
points(iris.prd$x[iris.prd$class=="setosa",1],
       iris.prd$x[iris.prd$class=="setosa",2],pch=',',col='grey')
points(predict(iris.lda)$x,pch=as.numeric(iris$Species)*2-1,cex=1.5)
dff=which(predict(iris.mlr) != predict(iris.lda)$class)
points(predict(iris.lda)$x[dff,1],predict(iris.lda)$x[dff,2],pch=16,col="red")
```

Comparison of LDA and multinomial regression



(Unlike for LDA, the plot of logistic regression is not exact!)