STAT 441: Lecture 16
Classification via regression I
LDA and logistic regression

Classification can be also seen as a regression problem:
from the values of features predict the class
Then we no longer find posterior probabilities via plugging
estimated densities into the optimal classification rule
but rather estimate them directly
The only problem is that the class variables are qualitative

LDA as regression

Two classes: if we code them by -1 and 1, and fit the linear regression of classes on classifiers (features),

the level zero set of the fitted hyperplane

is the LDA boundary (with equal prior probabilities and missclasification costs)

when there is the same number of points in both classes

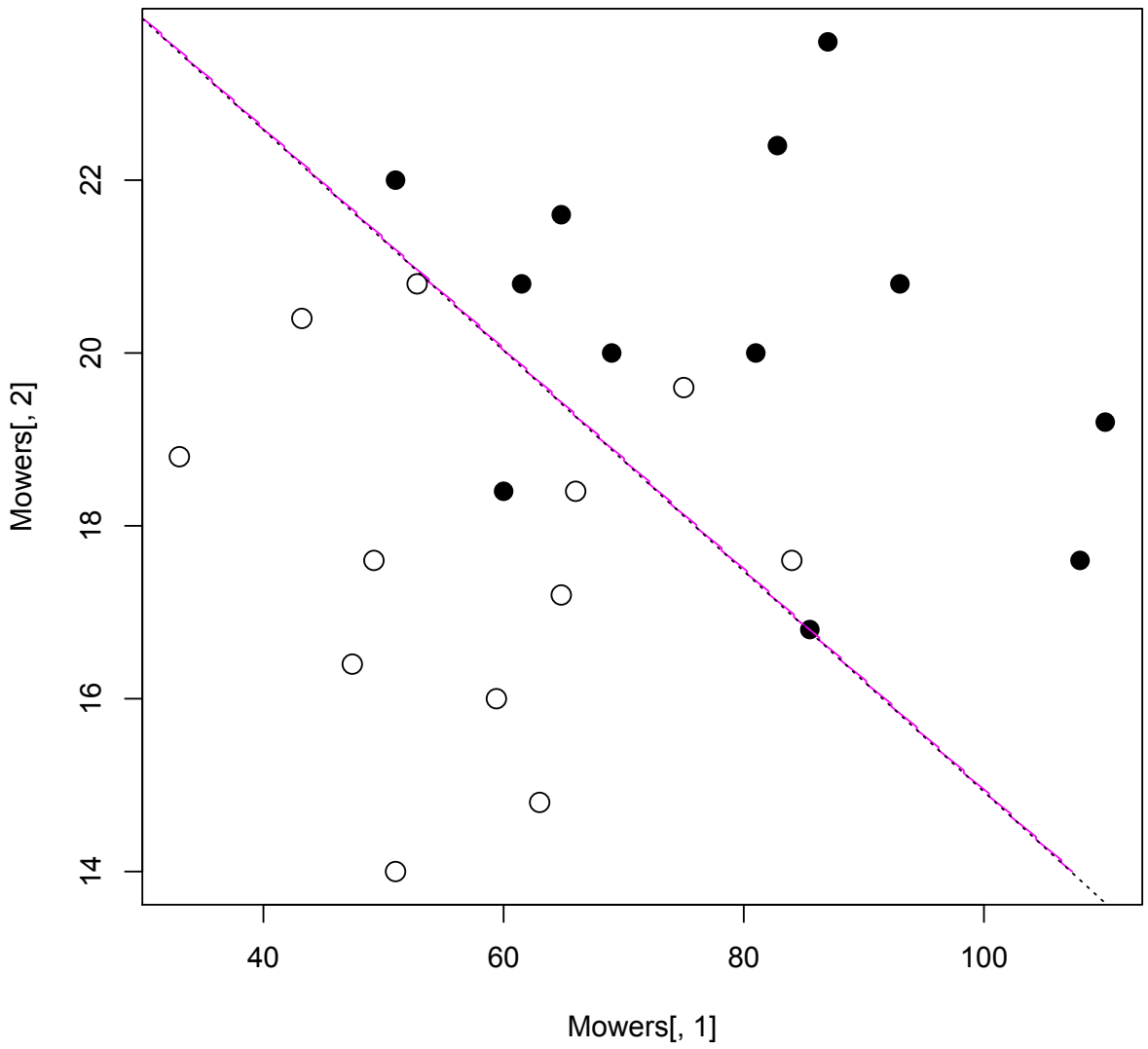
(Even if not, the level zero set is parallel)

```
> lm(2*riding-1~income+lot,data=mowers)
Coefficients:
(Intercept)       income          lot
   -5.47100      0.02522      0.19761
```

```
> plot(mowers[,1],mowers[,2],pch=15*mowers[,3]+1)
> abline(5.471/0.19761,-0.02522/0.19761)
```

The only problem with such a regression: not too handy for predicting probabilities...

LDA as regression for mobile mowers



LDA and logistic regression

Logistic regression:

regress $\log\left(\frac{c_i}{1-c_i}\right)$ linearly on the remaining variables

In the linear discriminant analysis with two categories, the expression for the posterior probabilities, *if the distributions are normal*, is

$$\frac{p_1 f_1(x)}{p_1 f_1(x) + p_2 f_2(x)} = \frac{e^{a+b^T x}}{1 + e^{a+b^T x}}$$

That indicates that both LDA and logistic regression estimate the same quantities (although they do not yield necessarily same results)

So what is the difference?

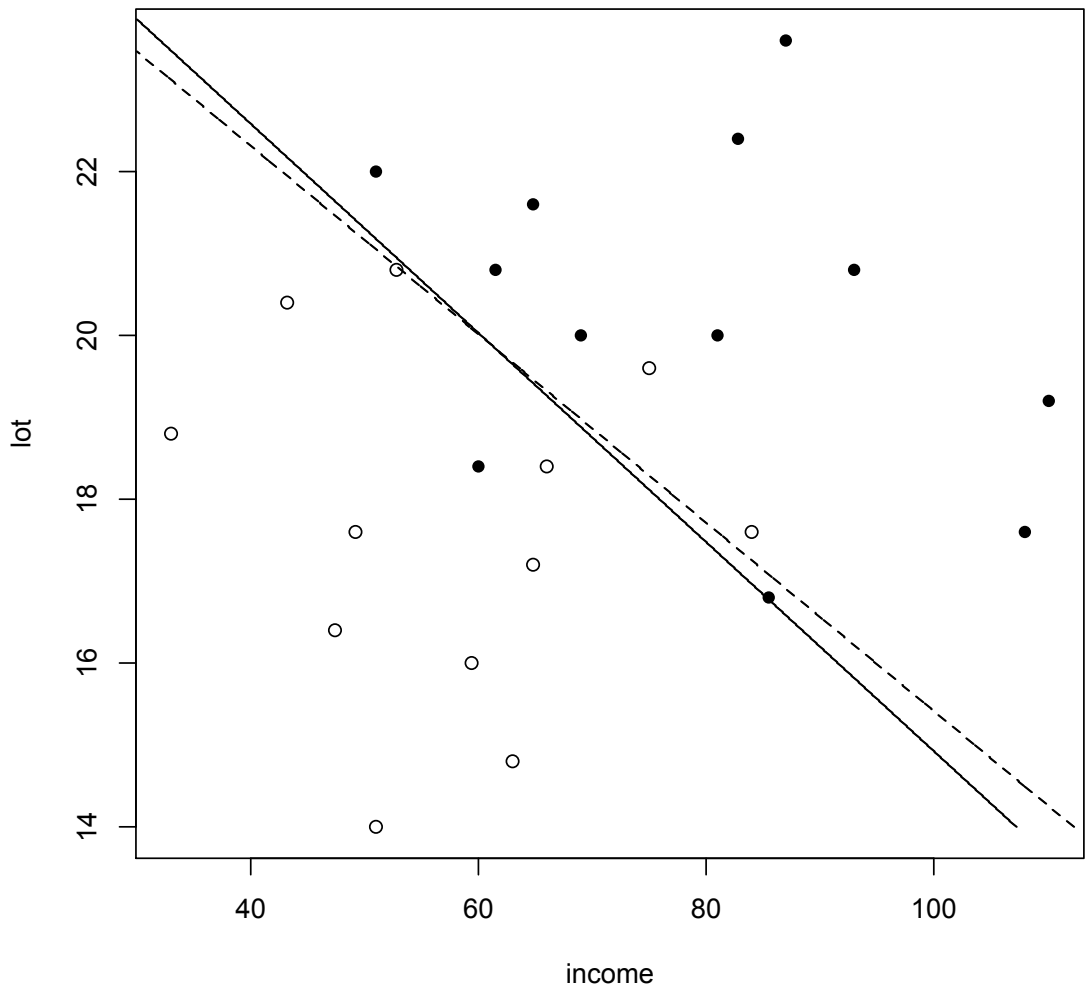
The difference is in the method: logistic regression maximizes the binomial likelihood conditionally on regressors, while LDA utilizes the covariance information in them. The results are often pretty much the same; the logistic regression is reported to be about 30% less efficient if the LDA normality assumptions hold.

On the other hand, logistic regression is more flexible, in particular with respect to incorporating qualitative classifiers, etc.

There is also a generalization to more than two classes, multinomial log-linear model. Basically, it is multivariate logistic regression; thus it is not that standard as the univariate one.

LDA and logistic regression: mobile mowers

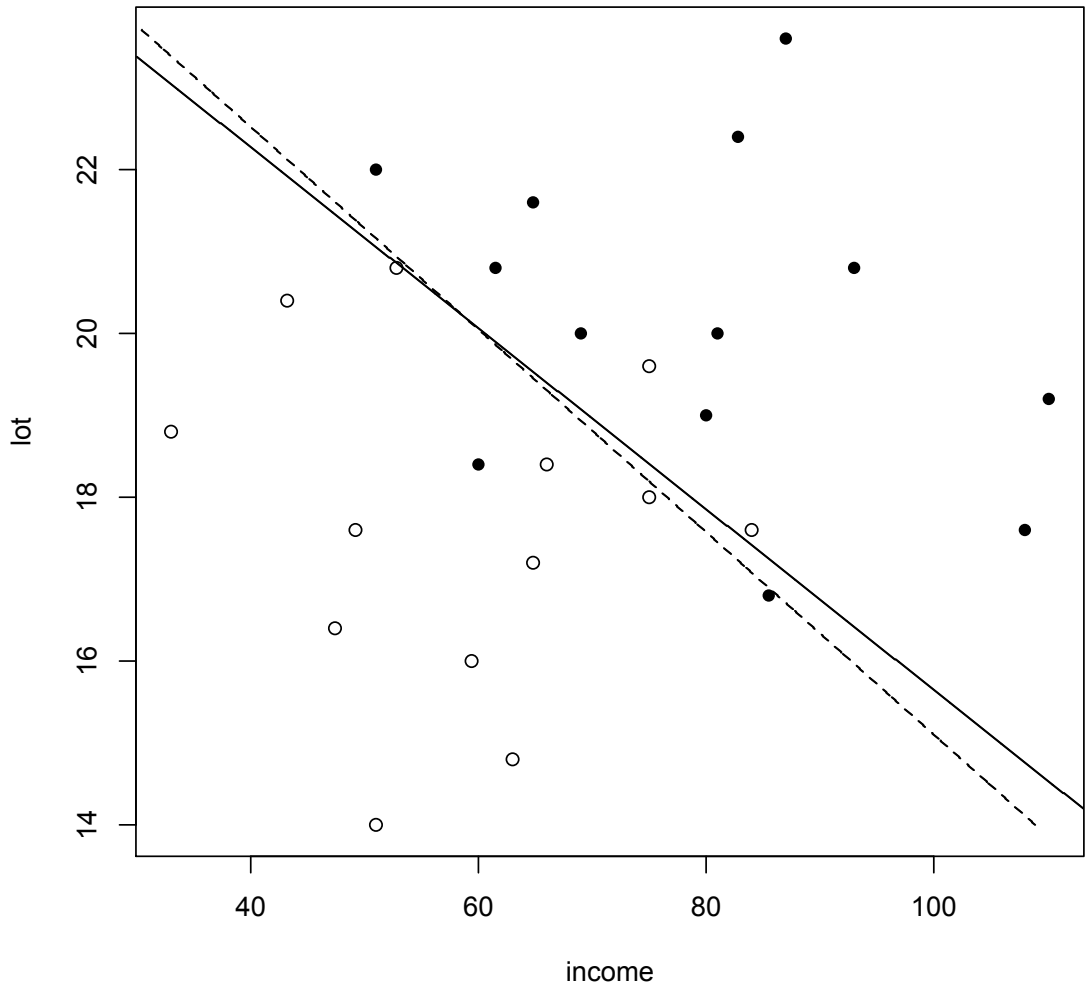
```
> mowlda=lda(riding~income+lot,data=Mowers)
> mowlog=glm(riding~income+lot,data=Mowers,family=binomial)
```



None of these has to be a "perceptron"

Neither LDA nor logistic regression yields necessarily linear separation with minimal apparent error rate (data augmented)

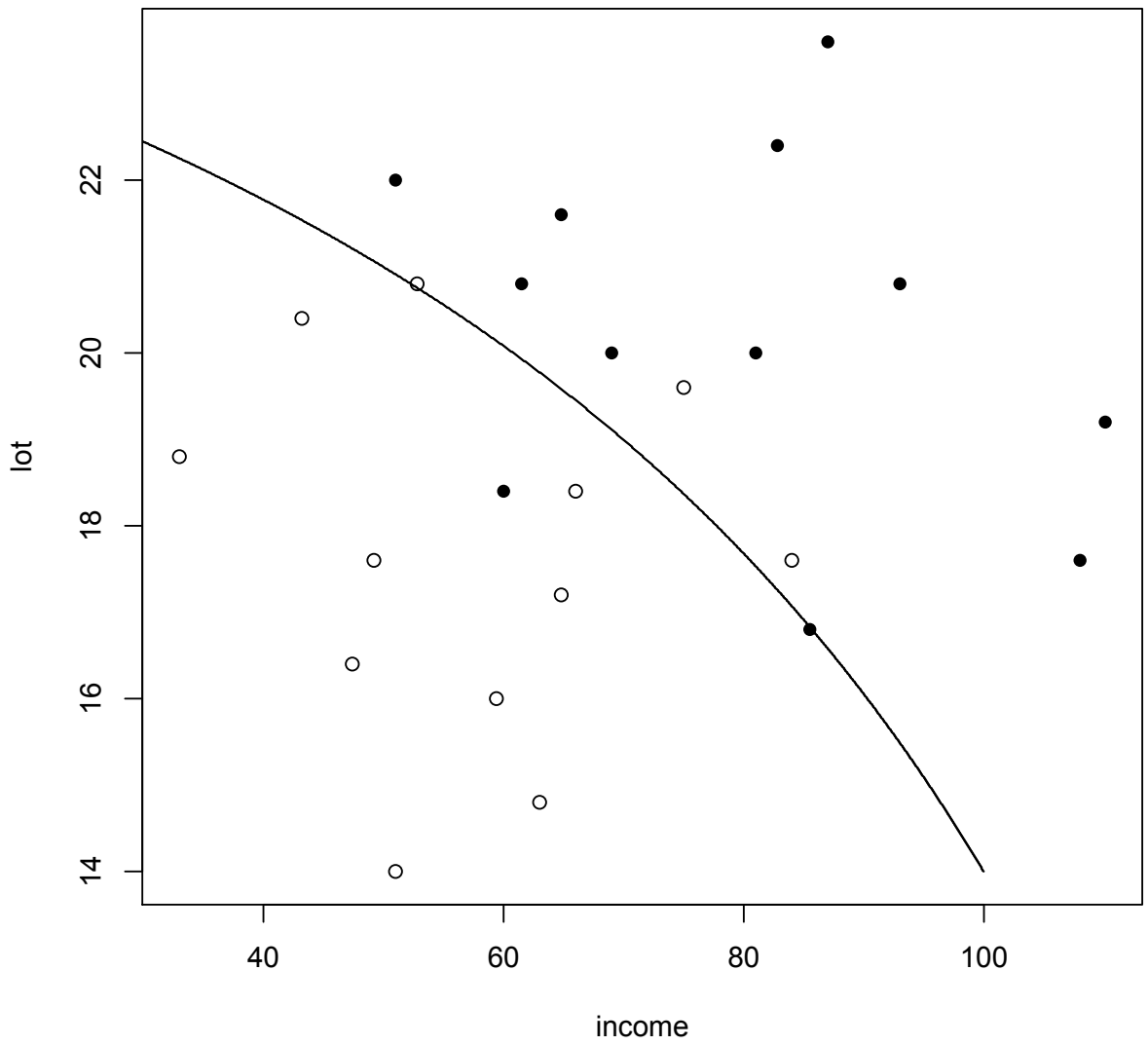
```
> mowersp = rbind(Mowers,c(80,19,1),c(75,18,0))
```



Mobile mowers: add interaction?

```
> mowlog=glm(riding~income+lot+I(income*lot),data=Mowers,family=bin
> summary(mowlog)
...
Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)     -37.433891  41.179585  -0.909    0.363
income            0.280810   0.574596   0.489    0.625
lot               1.560129   2.085627   0.748    0.454
I(income * lot)  -0.008915   0.029744  -0.300    0.764
```

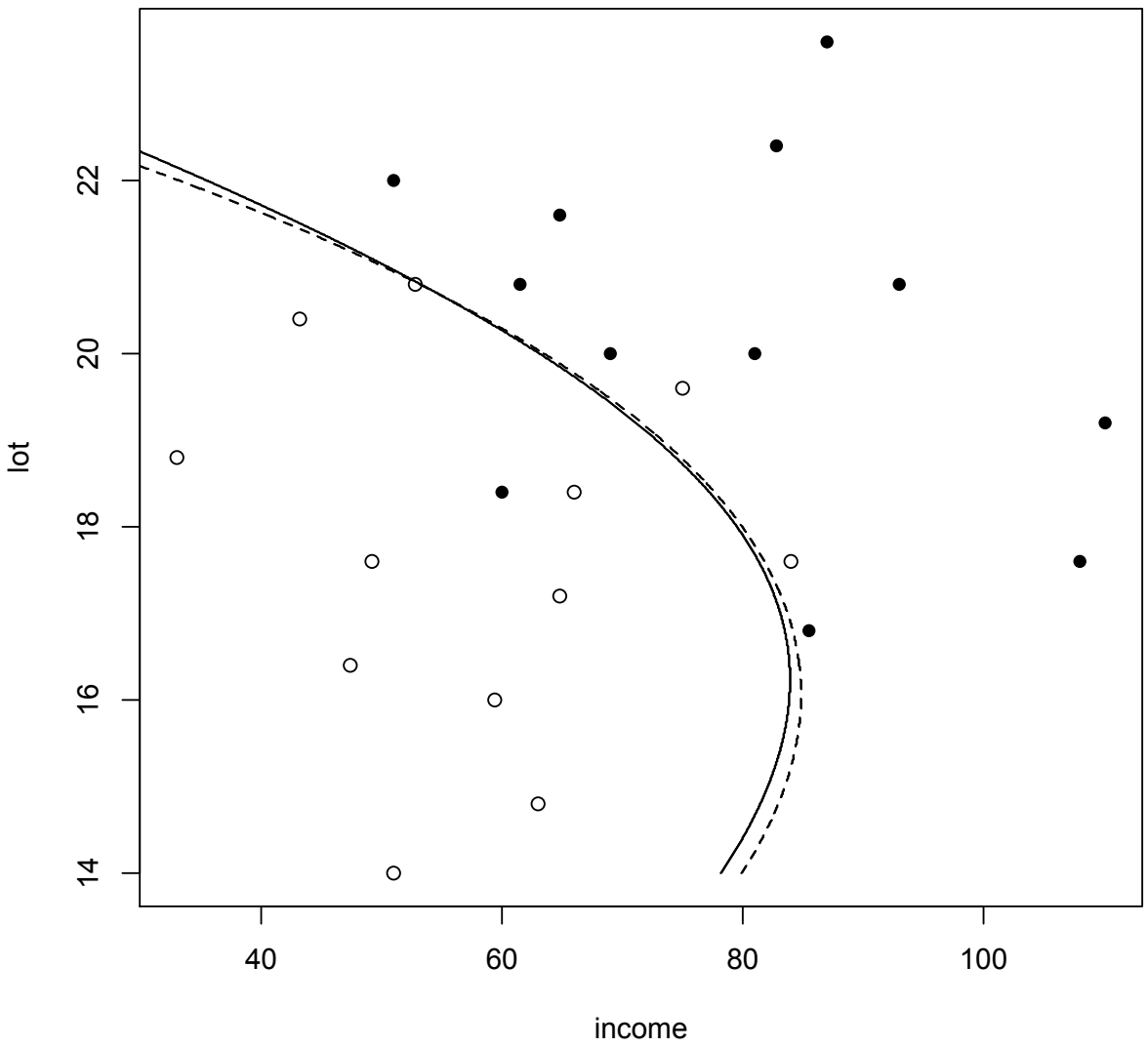
Bilinear logistic regression



Mobile mowers: add all quadratic terms?

```
> mowlog=glm(riding~income+lot+I(income*lot)+I(lot^2),
+ data=Mowers,family=binomial)
> mowlog=glm(riding~income+lot+I(income*lot)+I(lot^2)+I(income^2),
+ data=Mowers,family=binomial)
> summary(mowlog)
...
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)     20.7446485 84.6472477   0.245    0.806
income           0.3047331  0.8640822   0.353    0.724
lot             -4.7782138  8.0562174  -0.593    0.553
I(income * lot) -0.0061966  0.0336469  -0.184    0.854
I(income^2)     -0.0004715  0.0026086  -0.181    0.857
I(lot^2)         0.1632251  0.2075052   0.787    0.432
```

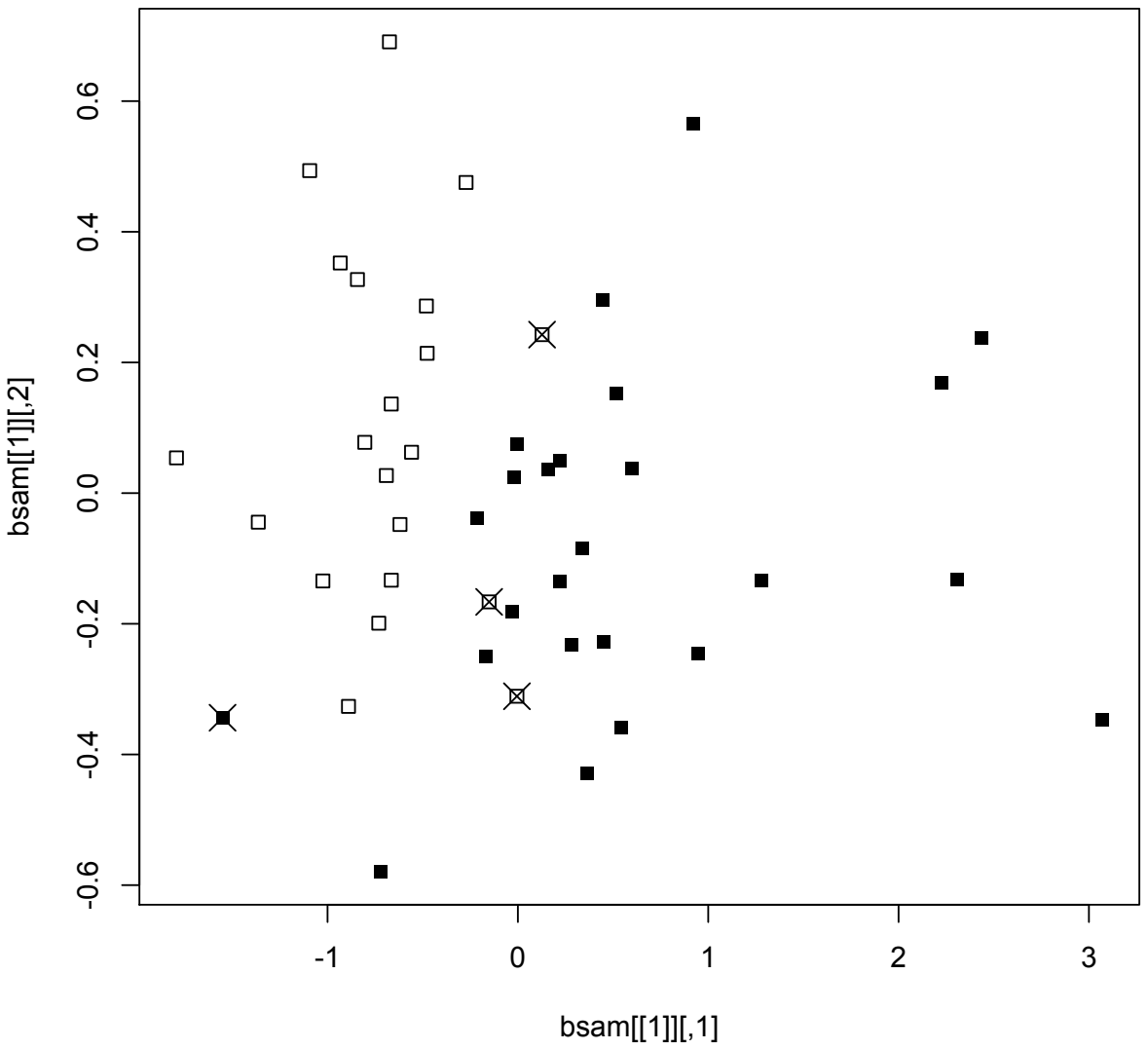
Go further?



Bank data: logistic regression

```
> library(MASS)
> bsam=sammon(dist(Bank[,1:4]))
> plot(bsam[[1]],pch=15*Bank$k)
> banlog=glm(k~.,data=Bank,family=binomial)
> wrong=(Bank$k!=(predict(banlog,type='response')>0.5))
> points(bsam[[1]][wrong,],pch=4,cex=2)
```

The result



However

```
> summary(banlog)
...
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    -5.320      2.366  -2.248  0.02459 *
v1              7.138      6.002   1.189  0.23433
v2             -3.703     13.670  -0.271  0.78647
v3              3.415      1.204   2.837  0.00455 **
v4             -2.968      3.065  -0.968  0.33286
```

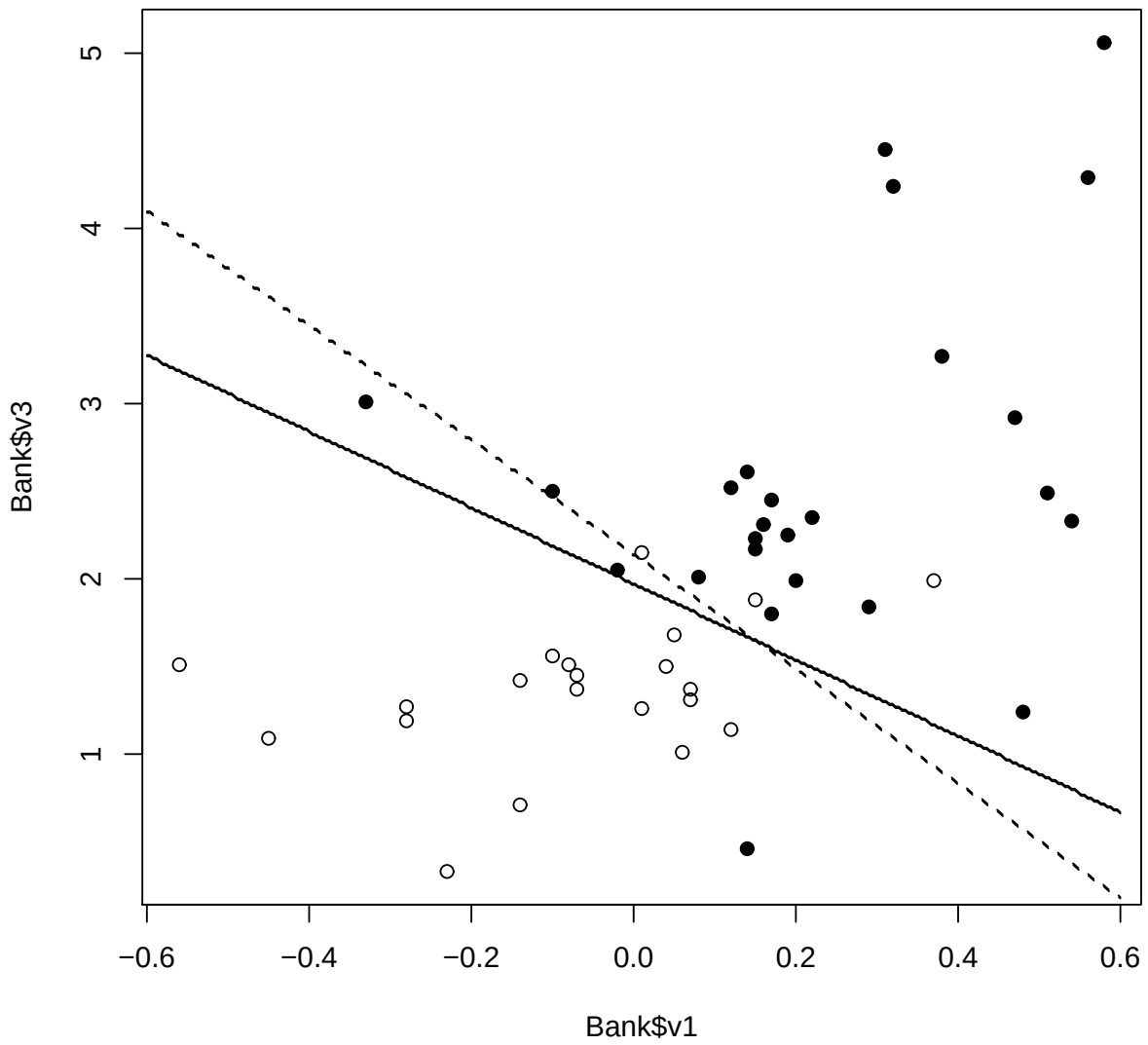
Simplify by dropping classifiers?

```
> banlog=glm(k~v3,data=Bank,family=binomial)
> table(Bank$k,(predict(banlog,type='response')>0.5))
    FALSE TRUE
  0    18    3
  1     2   23
> banlog=glm(k~v3+v1,data=Bank,family=binomial)
> table(Bank$k,(predict(banlog,type='response')>0.5))
    FALSE TRUE
  0    18    3
  1     1   24
> summary(banlog)
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -5.940      1.985  -2.992  0.00277 **
v3             3.019      1.002   3.013  0.00259 **
v1             6.556      2.905   2.257  0.02402 *
> attach(Bank)
> plot(v1,v3,pch=15*k+1)
```

Simplified prediction - and revert to LDA?

```
> banlda=lda(k~v3+v1,data=Bank) % broken line
...
> table(k,predict(banlda)$class)
k    0  1
  0 18  3
  1  3 22
```

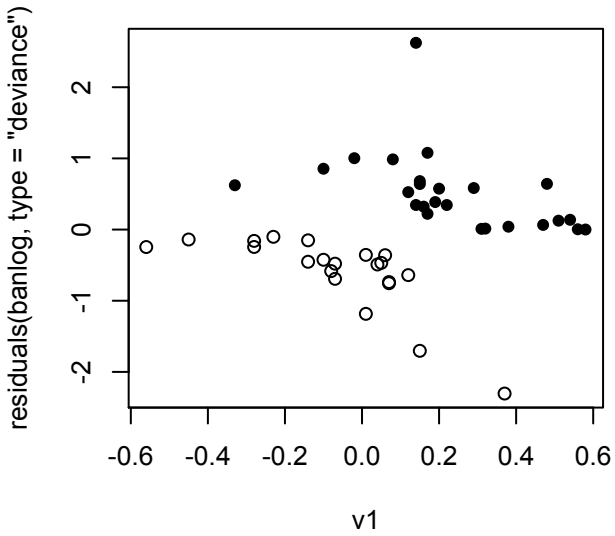
Bank data in two dimensions now

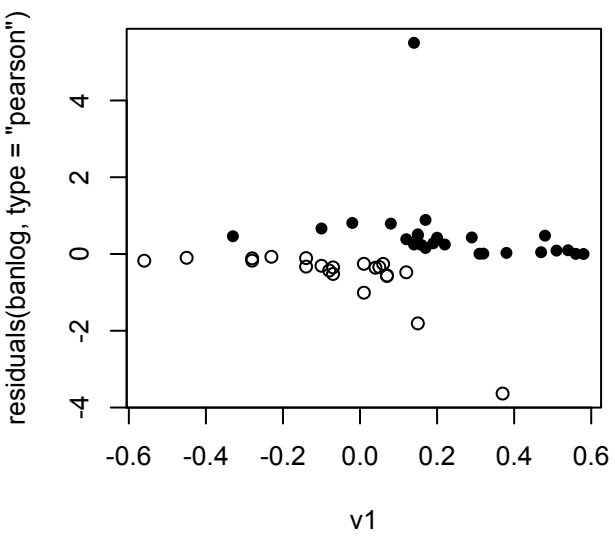


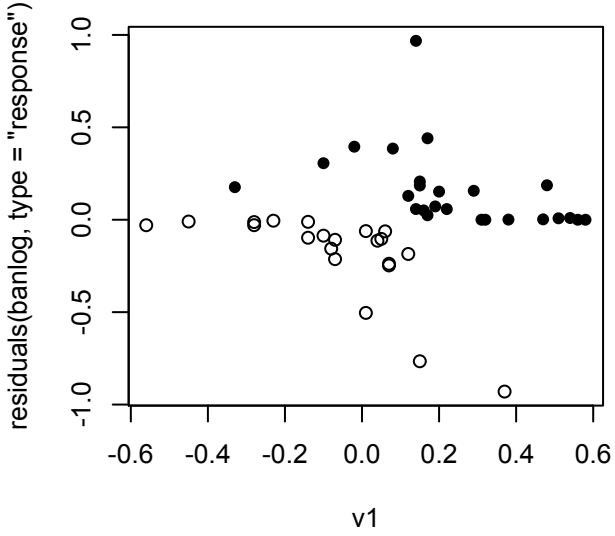
A look at residuals: code

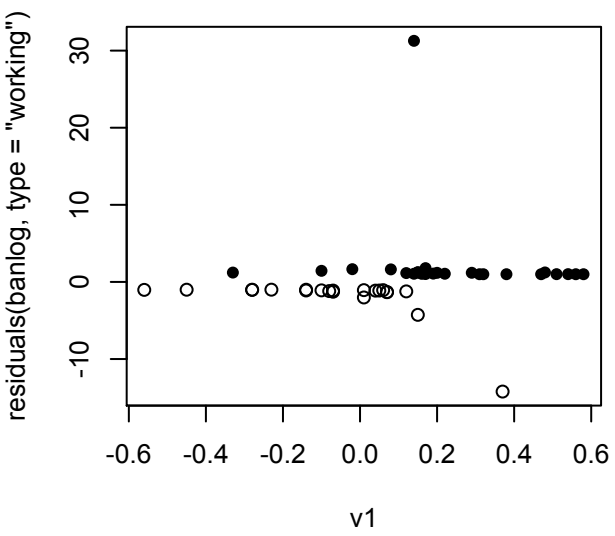
```
> par(mfrow=c(2,2))
> plot(v1,residuals(banlog,type='deviance'),pch=15*k+1)
> plot(v1,residuals(banlog,type='pearson'),pch=15*k+1)
> plot(v1,residuals(banlog,type='response'),pch=15*k+1)
> plot(v1,residuals(banlog,type='working'),pch=15*k+1)
```

A look at residuals: pictures



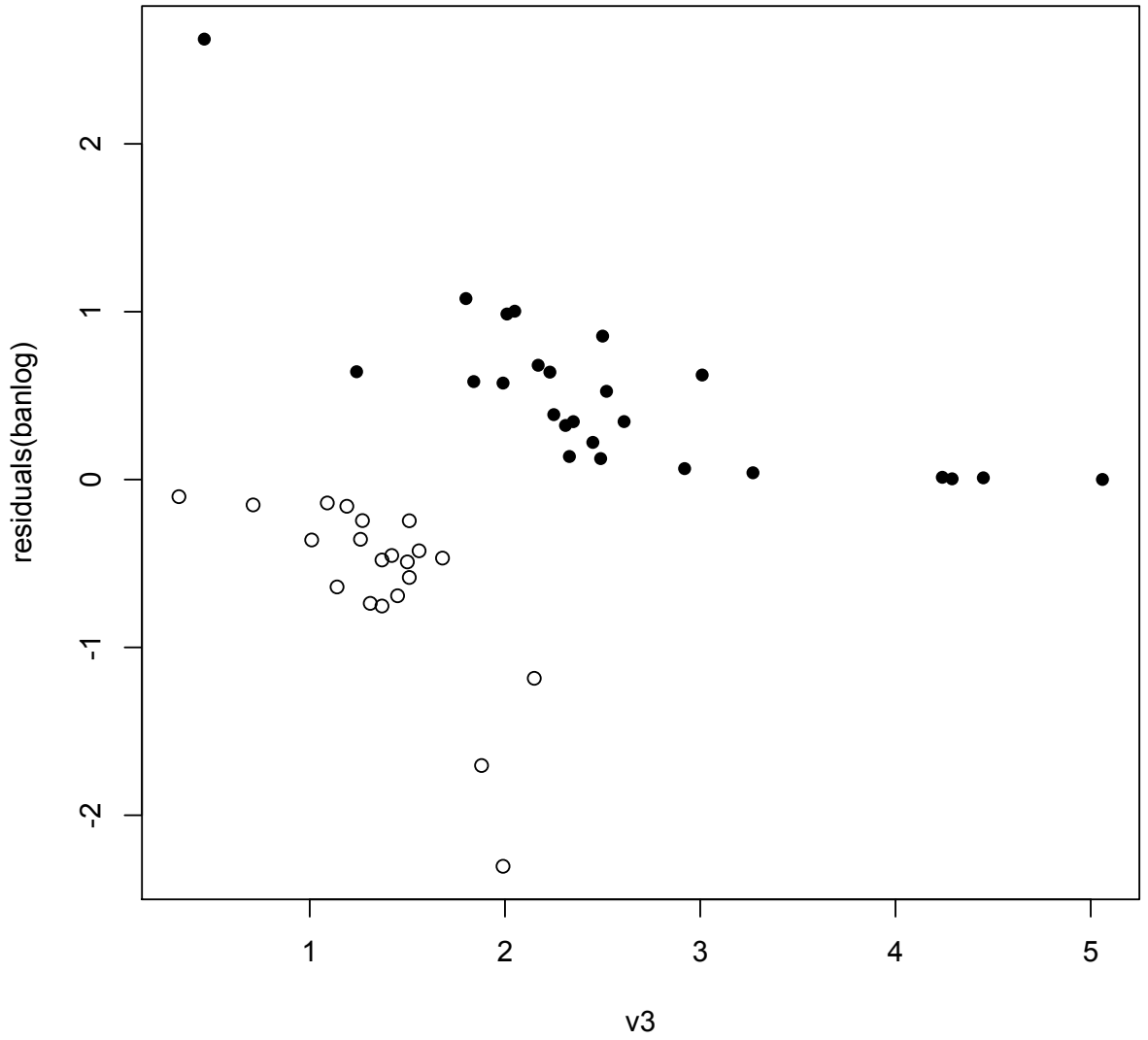






Further look at residuals

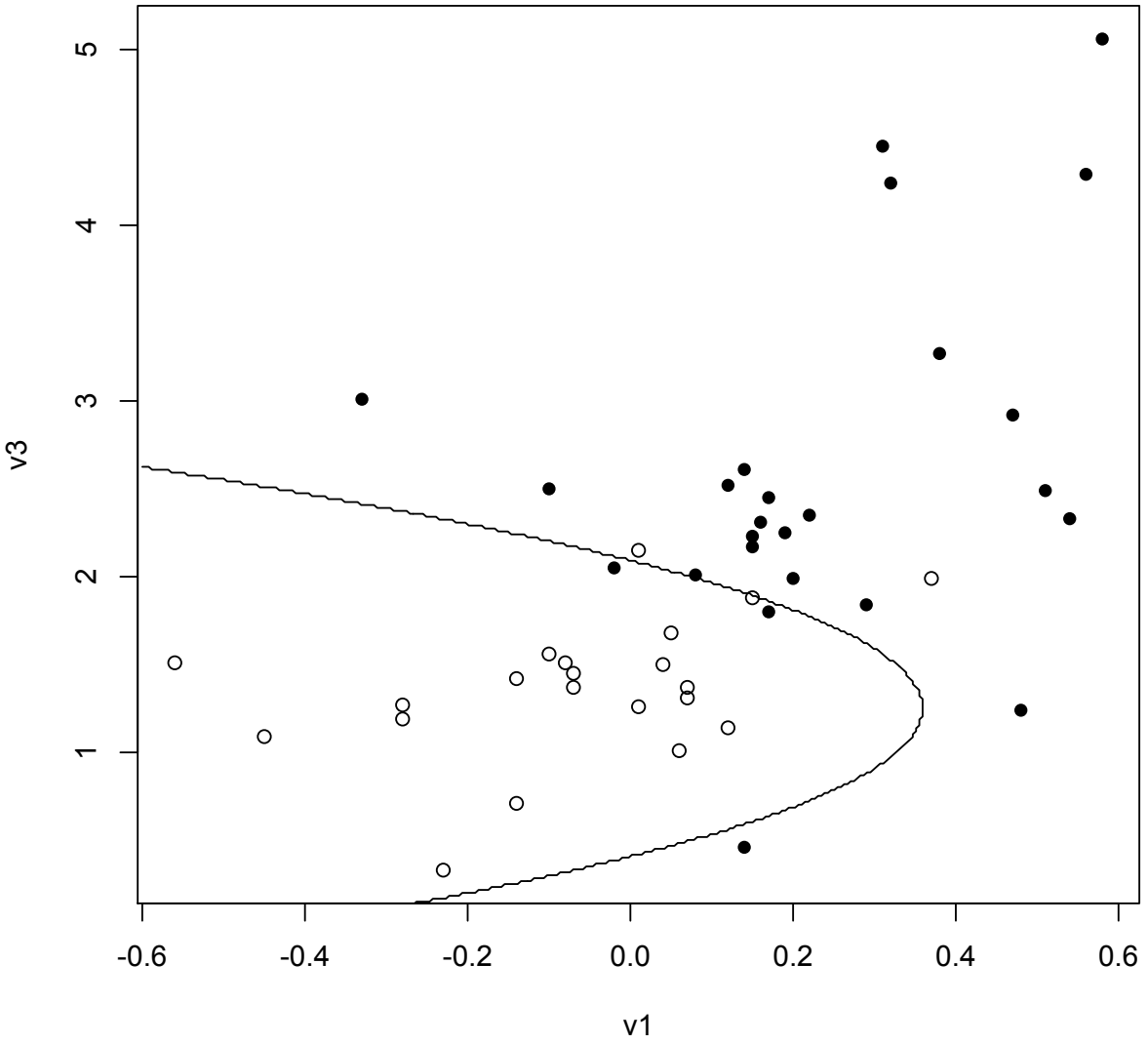
```
> plot(v3,residuals(banlog),pch=15*k+1)
```



Quadratic term in v3?

```
> banlog=glm(k~poly(v3,2)+v1,data=Bank,family=binomial)
Warning message:
fitted probabilities numerically 0 or 1 occurred in: glm.fit(x = X,
...
> table(k,predict(banlog)>0.5)
k   FALSE TRUE
  0    19    2
  1     3   22
```

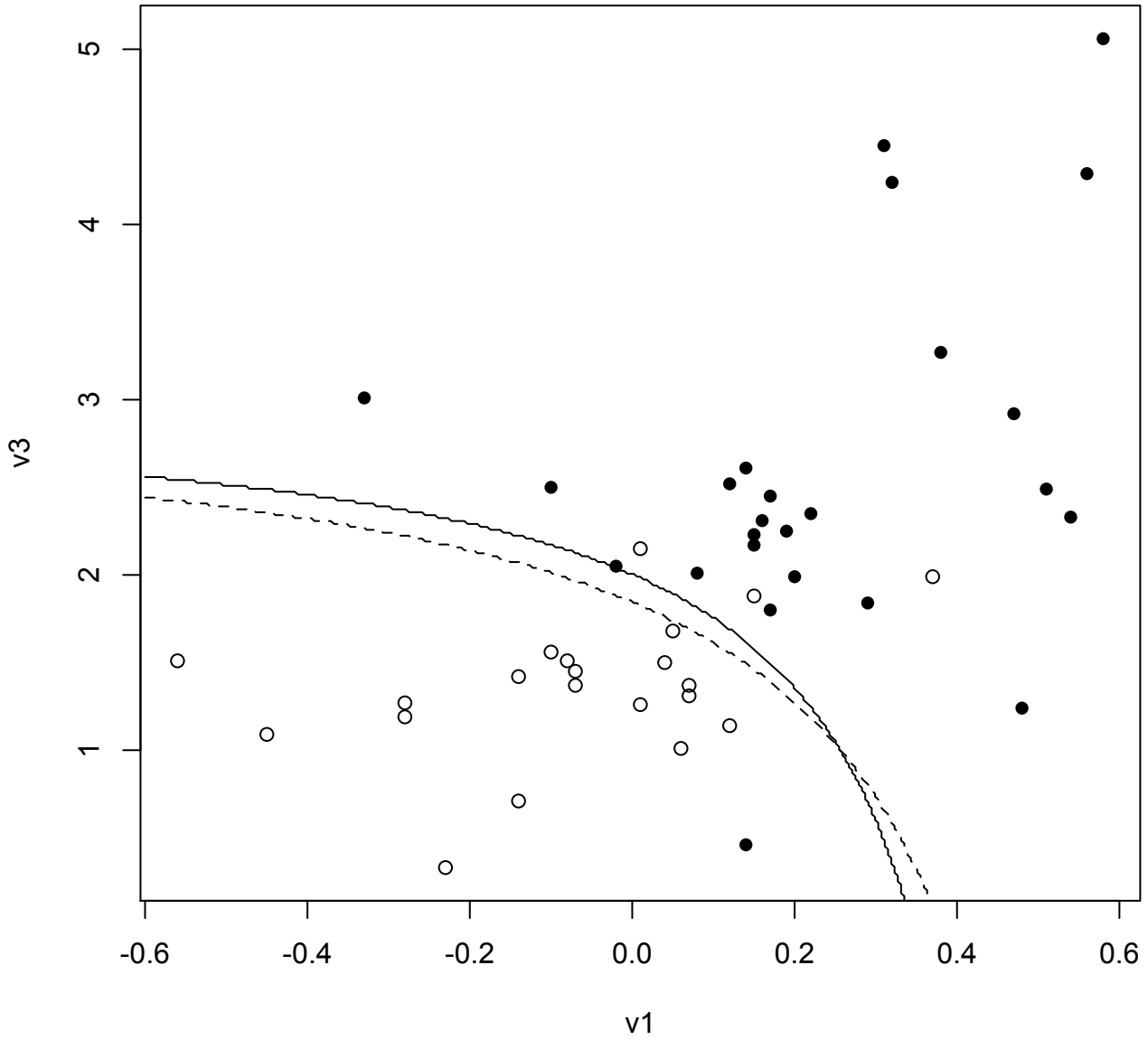
Odd?



Interaction?

```
> banlog=glm(k~v1+v3+I(v1*v3),data=Bank,family=binomial)
> table(k,predict(banlog)>0.5)
k   FALSE TRUE
  0    18    3
  1     2   23
>summary(banlog)
...
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -7.234      2.509  -2.883  0.00393 **
v1            21.141      9.201   2.298  0.02158 *
v3             3.627      1.286   2.820  0.00480 **
I(v1 * v3)    -6.953      3.476  -2.000  0.04549 *
```

The result (broken: LDA in a same way!)



(Semi) Final remarks

Not only logistic regression, but also LDA can be viewed as a regression method for classification.

Once the normality assumption for the classifiers does not bother us too much, our possibilities are expanded: not only we can do some model selection, but we can also consider transformed classifiers.

However, there is a serious danger of overfitting. Would be nice to prevent it somehow.

For instance, we could downweight higher-order terms - but is there some automatic way to do it?

That is, we would like to have some nonlinear flexibility - but how to achieve that?