STAT 441: Lecture 22
Classification: finale

Summary of classification methods we have learned
but also something new (the best at the end)
Venables and Ripley, Chapters 9, 12, and other sources

Properties of classification methods I

Predictive power: how well the method classifies?

- all methods compete for that, and new ones are introduced because they (are supposed to) perform better than old ones
- but this does not mean that the old ones are necessarily worse (LDA, for instance)

Interpretability: can we make some sense out of the rule?

- some methods are really of black box type: support vector machines, neural networks
- but of course, interpretability may be not the most important virtue

Number of classes capable to handle: many or only two?

- there is multiclass version of logistic regression, multinomial regression; only we didn't cover it (yet)
- support vector extensions for multiple classes are clumsy (there were several of them proposed)

Properties of classification methods II

Computational difficulty: feasible also for large datasets?

- this changes with time: new algorithms, new hardware
- is there a need to tune parameters (often a hidden feature)?

Special aspects

- handling of missing data
- adaptivity to unequal priors and misclassification costs
- handling of linear combinations
- type of regions (linear boundaries? more fancy?)
- invariance with respect to transformations?
- robustness to outliers?

Overview of classification methods

We covered actively:

Linear and quadratic discriminant analysis

k-nearest neighbors

Logistic regression

Classification trees

Passively:

Neural networks

Support vector machines

These are not all classification methods

What else is there: quite a few things... for instance:

Kernel discriminant analysis - similar idea like k-nearest neighbors, but using kernel density estimates

Methods that were "nonparametrized" in a similar fashion as support vector machines: flexible discriminants, regularized logistic regression

Learning vector quantization: a ramification of k-nearest neighbors, with an objective to overcome the need to store all training data

Multiclass extensions of some methods mentioned above

"Recycling" methods, or "committee" methods, like boosting, bagging or stacking: they combine results of other classifiers

Boosting

An algorithm that takes a classification method and repeatedly applies it to the reweighted data points - with the objective to get improved classification

1. Apply a classifier, store the result.
2. Reweight items: the misclassified ones receive higher weight
3. Go to point 1 and repeat.

After stopping:

4. Combine all the results obtained on the road.

AdaBoost.M1

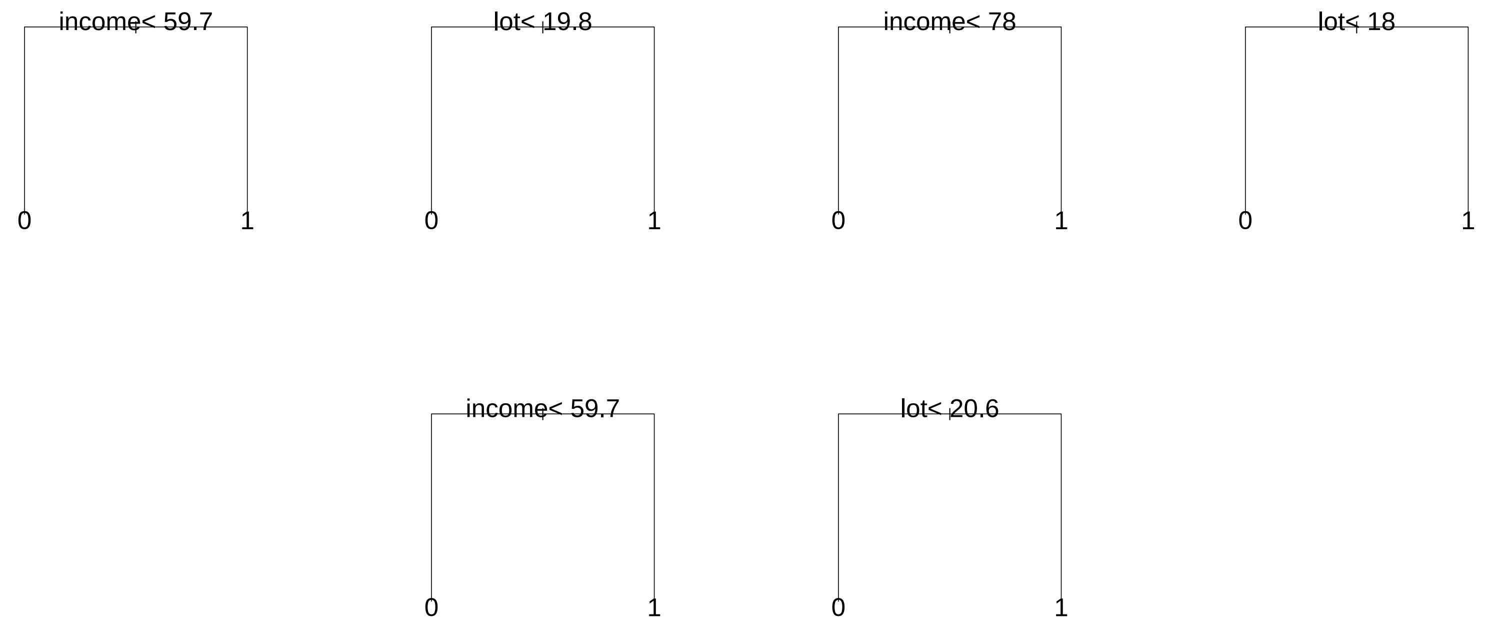
Suppose that the two classes are coded as -1 and $+1$.

1. Initialize weights to $w_i = 1/n$ (start with equal weights)
2. For $k = 1$ to K
(a) Classify training data with weights w_i; predictions are $G_k(x)$
(b) $I_i = 1$, if i-th item not classified correctly, and 0 if yes
(c) $e_k = \frac{\sum_{i=1}^{n} w_i I_i}{\sum_{i=1}^{n} w_i}$
(d) $\alpha_k = \log \frac{1 - e_k}{e_k}$
(e) set new w_i to be $w_i e^{\alpha_k I_i}$
3. After the loop, the final prediction is: sign $\left(\sum_{k=1}^{K} \alpha_k G_k(x)\right)$

A realization for riding mowers data

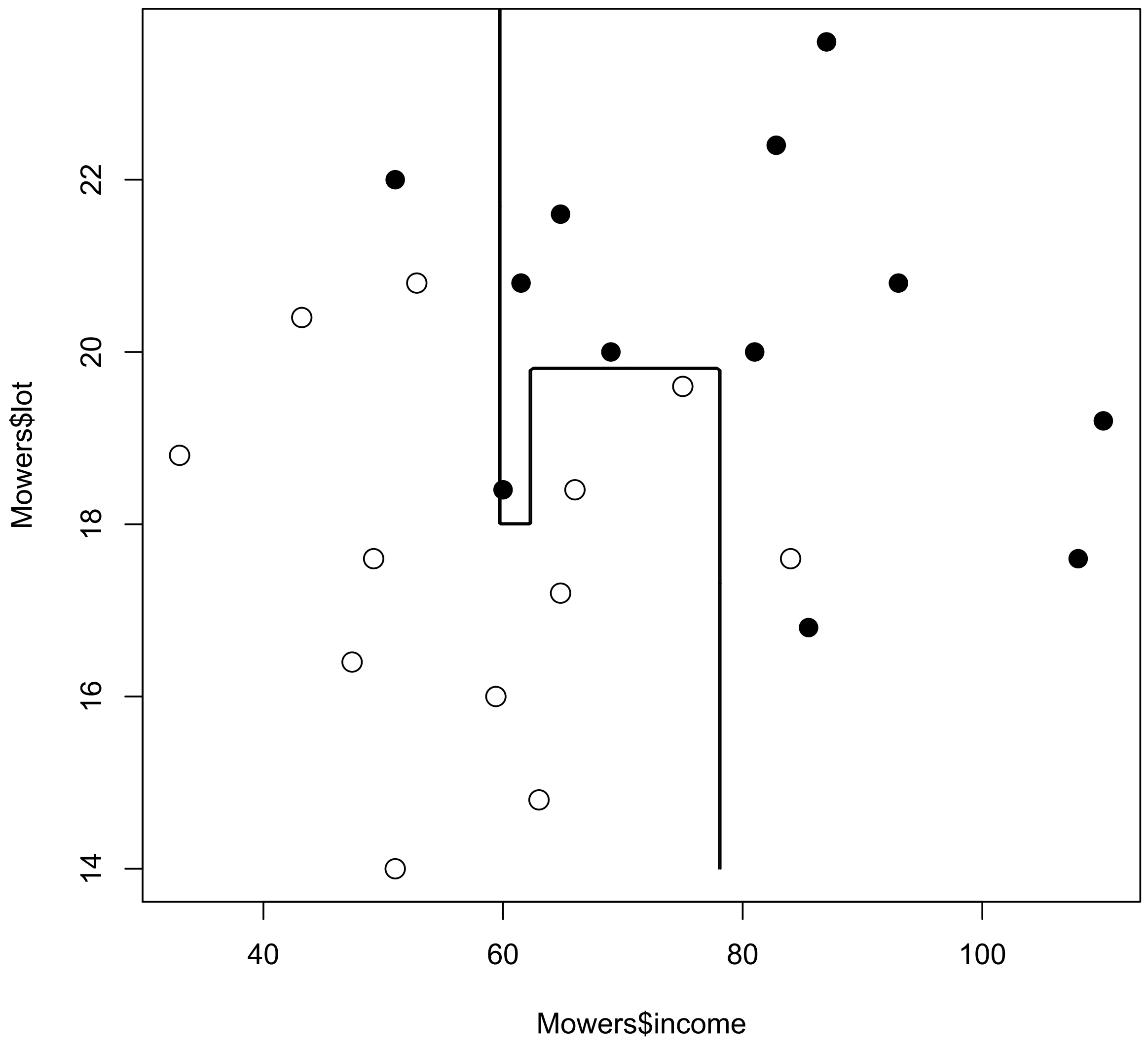
```
> boowei=rep(1/nrow(Mowers),nrow(Mowers))
> boocla=vector("list",33)
> booalp=rep(0,33)
> for (k in 1:33)
+ {
+ boocla[[k]] = rpart(factor(riding)~income+lot,
+ data=Mowers,weights=boowei,maxdepth=1)
+ boopr = predict(boocla[[k]])
+ boomis = (Mowers$riding != as.numeric(boopr[,2] > boopr[,1]))
+ booerr = crossprod(boowei,boomis)/sum(boowei)
+ booalp[k] = log((1-booerr)/booerr)
+ boowei = boowei*exp(booalp[k]*boomis)
+ print(k)
+ }
```

The classifier used: stumps



(trees with only one split)

The result

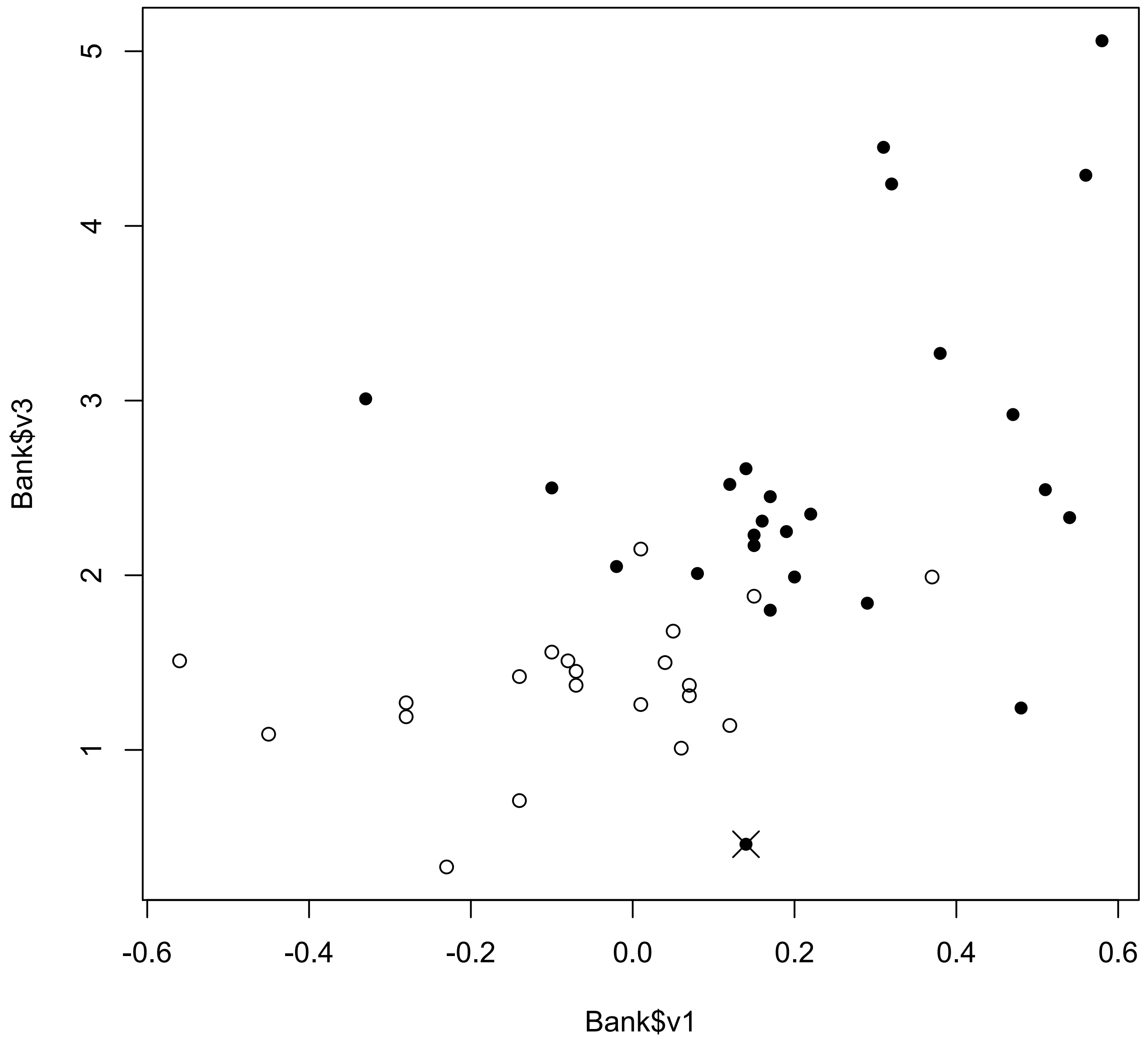


"The best off-shelf classifier (boosting with trees)"
Catch: do you know when to stop? (Hidden tuning parameter.)

Finally, bank data (as usual)

```
> boowei=rep(1/nrow(Bank),nrow(Bank))
> boocla=vector("list",15)
> booalp=rep(0,15)
> for (k in 1:15)
+ {
+ boocla[[k]] = rpart(factor(k)~.,data=Bank,
+ weights=boowei,maxdepth=1)
+ boopr = predict(boocla[[k]],type='class')
+ boomis = (Bank$k != boopr)
+ booerr = crossprod(boowei,boomis)/sum(boowei)
+ booalp[k] = log((1-booerr)/booerr)
+ boowei = boowei*exp(booalp[k]*boomis)
+ }
> boocum=rep(0,nrow(Bank))
> for (k in 1:15)
+ {
+ boocum = boocum + booalp[k]*2
+ *(predict(boocla[[k]])[,2] > 0.5)-1
+ }
> wrong=(1+sign(boocum))/2 != Bank$k
> plot(Bank$v1,Bank$v3,pch=15*Bank$k+1)
> points(Bank$v1[wrong],Bank$v3[wrong],pch=4,cex=2)
```

Boosting bank data


Bank$v3
Bank$v1