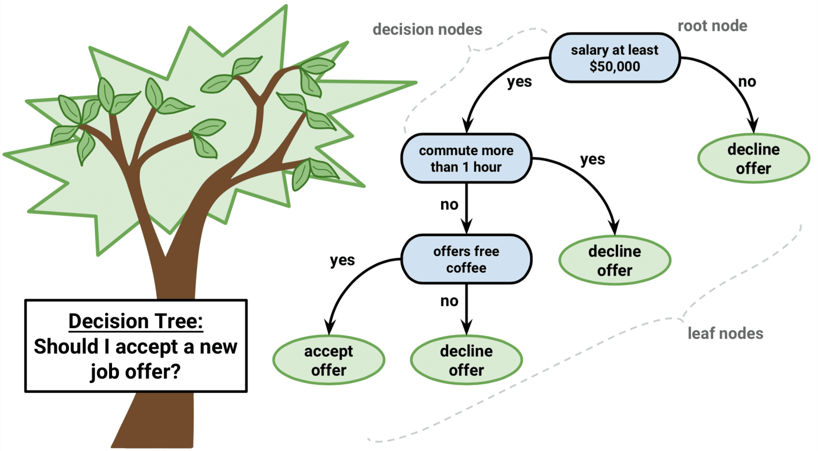
Statistical learning and Visualization

Boosting and Support Vector Machines

Maarten Cruyff

Content

Classifiaction tree



Classification trees

1. Recursive binary splitting algorithm
2. Splits features on basis of node *purity*

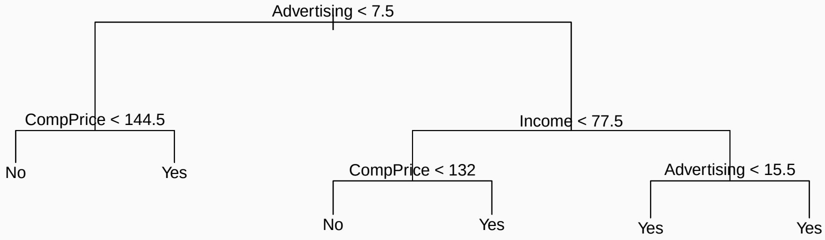


Figure 1: Sale of car seats (Yes/No)

Recusrsive binary splitting

Algorithm

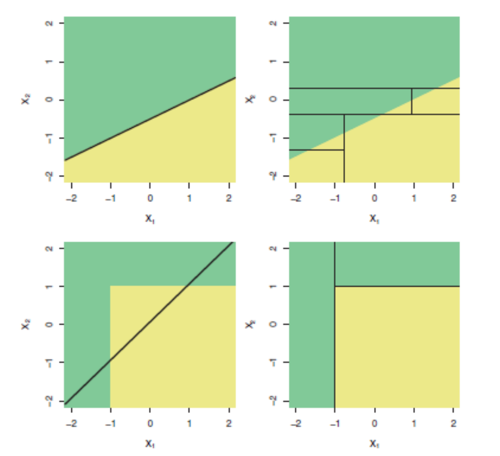
1. Divide feature space in non-overlapping, rectangular regions
2. Choose splits that minimize *node impurity* (homogeneity of nodes)
3. Assign region to class with highest mode
4. Stop when node purity no longer increases

Algorithm is top-down and greedy, so

- high variance

Classification with trees or regression?

Depends on nature of relationship between classes and features



Package `tree`

Growing and plotting trees with function `tree()`

```
fit_tree <- tree(formula, data, split = c("deviance", "gini"))

plot(fit_tree)
text(fit_tree)
```

- minimization of `deviance` or `gini impurity`
- `text()` for adding labels to nodes

Methods to reduce variance:

1. Pruning
   - cut branches with cross-validation and regularization

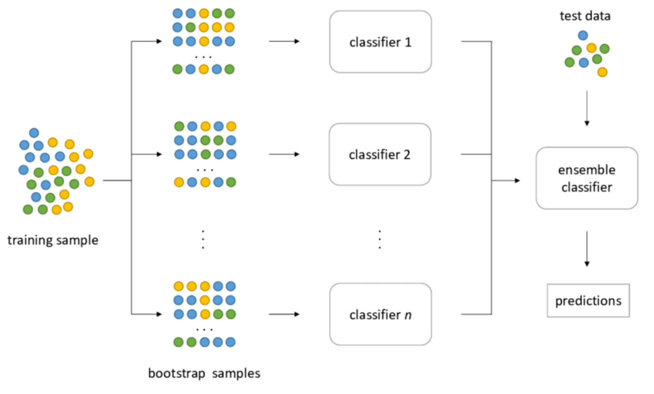
2. Bagging
   - average predictions of bootstrapped trees

3. Random forests
   - average predictions of decorrelated bootstrapped trees

4. Boosting
   - weighted combination of weak classifiers (small trees)

Random forest

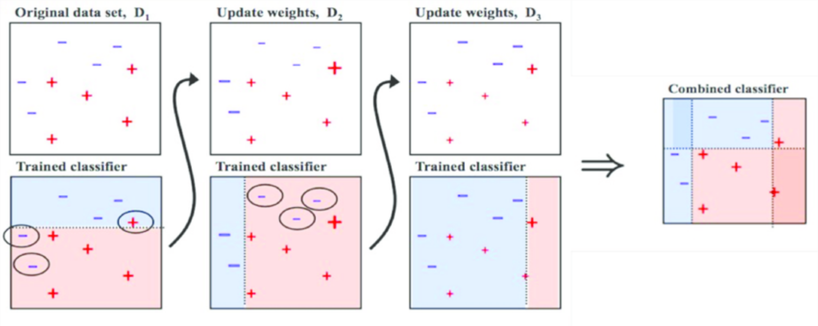
1. Fit classification trees to B bootstrap samples
2. Average the predictions
3. Out-Of-Bag (OOB) as estimate validation error



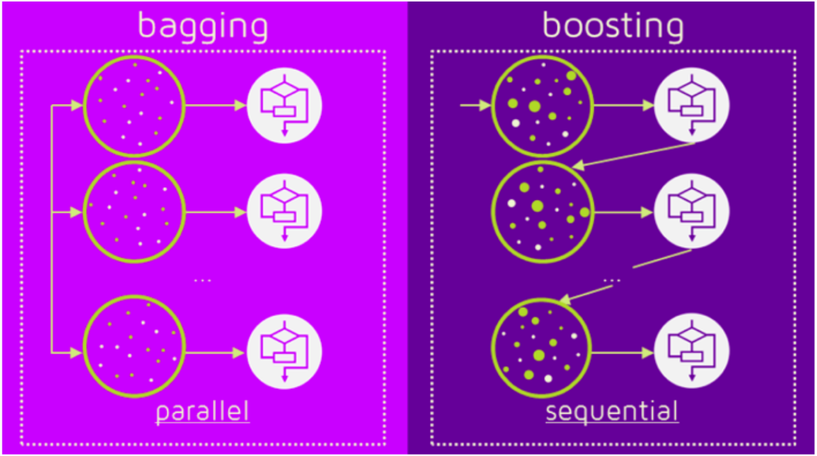
Boosting

Algorithm

1. Apply a weak classifier (e.g. stump) to training data
2. Increase weights for incorrect classifications, and repeat
3. Classifier is linear combination of weak classifiers



Boosting vs bagging/random forest



Boosting with package gbm

Boosting a single model

```
boost <- gbm(formula, data, distribution)

predict(boost, newdata,  type = c("link", "response"))
```

Boosting with package caret

Simple example

- distribution depends on response variable

```
gbm <- train(formula,
             data,
             distribution  = "bernoulli",
             method = "gbm")

predict(gbm, newdata, type = "prob")
```

Support Vector Machines (SVM)

SVM for binary classification

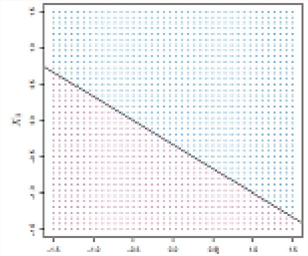
Classifiers using support vectors

1. *maximal margin classifier*
    - classes perfectly separable by hyperplane
2. *support vector classifier*
    - allows for non-separable cases
3. *support vector machine*
    - allows for non-linear boundaries

Hyperplane

Divides the feature space in two

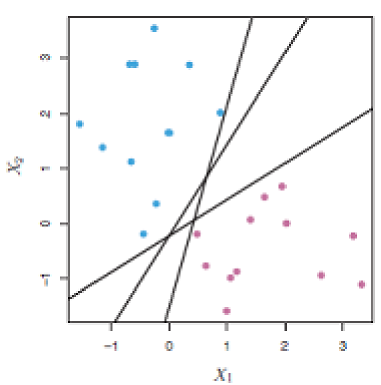
- in two dimensions hyperplane is simply a line



Separating hyperplane

Perfectly separates the two classes of the outcome variable

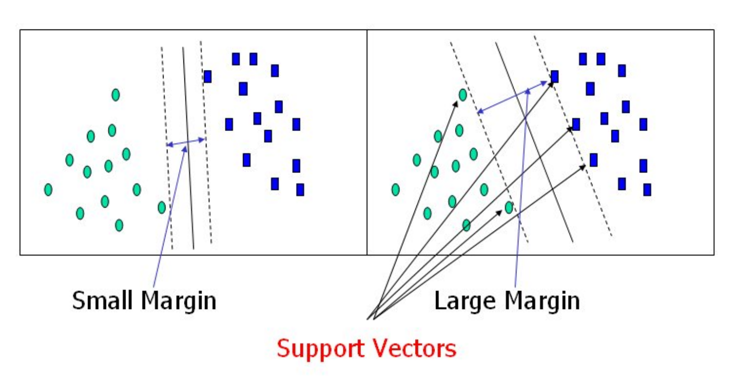
- hyperplane not uniquely identified
- high variance



Maximal Margin Classifier

Identifies hyperplane by specification of a maximal marging

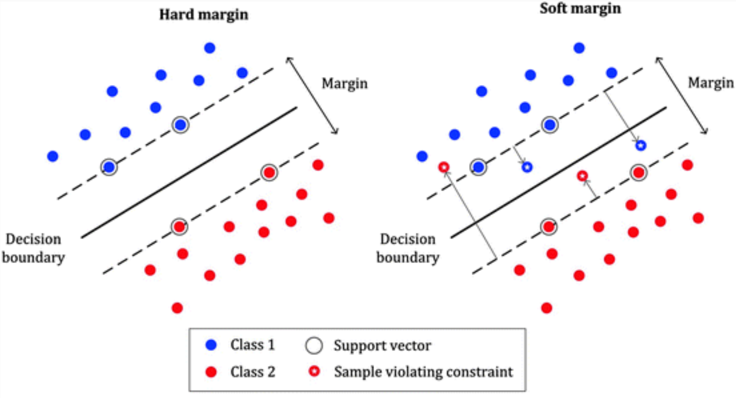
- points on margin are support vectors
- only works if cases are *separable*



Support Vector Classifier (SVC)

Allows for violations of the margin (soft margin)

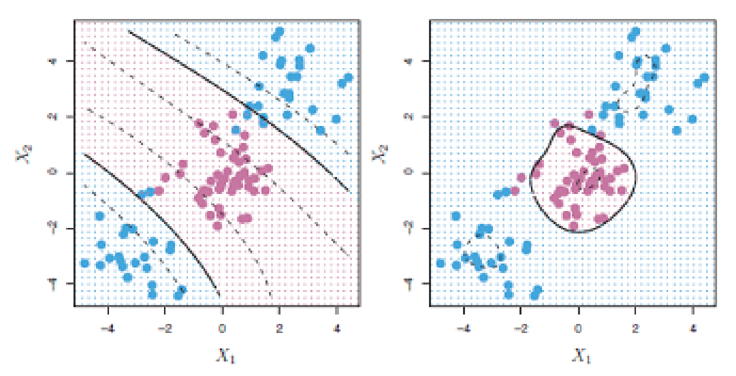
- budget for violations is called *cost* (C)
- cases the wrong side of hyperplane contribute to the cost



Support Vector Machines (SVM)

Kernels allow for nonlinear hyperplanes, e.g.

- polynomial kernel (left)
- radial kernel (right)



SVM with pacakge e1071

```
svm_train <- tune(svm, formula, data,
                  degree = 3, #default
                  coef0  = 0, #default
                  cost   = 1, #default
                  kernel = c("linear", "polynomial", "radial"),
                  ranges = list(cost = <sequence>), etc.)

svm_train$best.model # performance summary

svm_class <- predict(svm_train, newdata, probability = TRUE)
svm_prob  <- attr(svm_class, "probabilities")
```

- `cost`, `degree` and `coef0` are tuning parameters
- `ranges` works similar as `tuneGrid()`

SVM classification plot

Compression hyperplane two dimensions

```r
plot(svm_train$best.model, data, x1 ~ x2)
```