Decision Trees and Random Forests

Reference: Leo Breiman, http://www.stat.berkeley.edu/~breiman/RandomForests

1. Decision trees

Example (Guerts, Fillet, et al., Bioinformatics 2005):

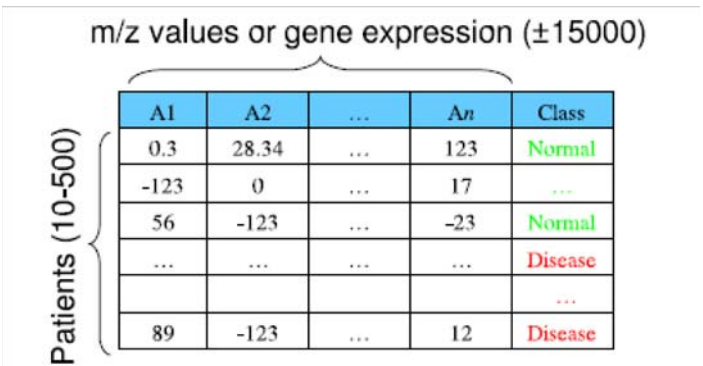
Patients to be classified: normal vs. diseased

Decision trees

Classification of biomarker data: large number of values (e.g., microarray or mass spectrometry analysis of biological sample)

Decision trees

Mass spectrometry (m/z) parameters or
gene expression parameters (around 15k values)



Decision trees

Given new patient with biomarker data, is s/he normal or ill?

Needed: selection of relevant variables from many variables

Number n of known examples in $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ is small (characteristic of machine learning/data mining problems)

Assume we have for each biological sample a feature vector **x**, and will classify it:

Decision trees

diseased: $y = 1$; normal: $y = -1$.

Goal: find function $f(\mathbf{x}) \approx y$ which predicts y from **x**.

How to estimate error of $f(\mathbf{x})$ and avoid over-fitting the small dataset D?

Use cross-validation, i.e., test predictor $f(\mathbf{x})$ in an unexamined part of sample set D.

Decision trees

For biological sample, feature vector $\mathbf{x} = (x_1, \ldots, x_d)$ consists of *features* (or *biomarkers* or *attributes*) $x_i = A_i$ describing the biological sample from which **x** is obtained.

The decision tree approach

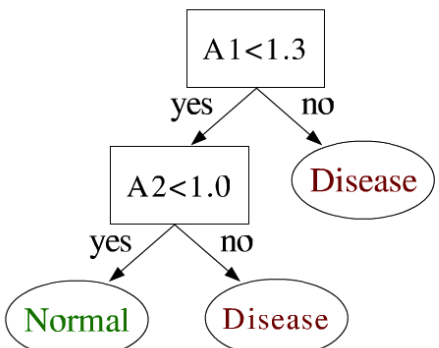
Decision tree approach to finding predictor $f(\mathbf{x}) = y$ from data set D:

- $\oplus$ form a tree whose nodes are features (attributes) $x_i = A_i$ in **x**
- $\oplus$ decide which features A_i to consider first in predicting y from **x**

  i.e., find features A_i with highest information gain - place these at top of tree

The decision tree approach

⊕ then use recursion - form sub-trees based on attributes not used in the higher nodes:



Advantages: interpretable, easy to use, scalable, robust

Decision tree example

Example 1 (Moore): UCI data repository (http://www.ics.uci.edu/~mlearn/MLRepository.html)

MPG (miles per gallon) ratings of cars:

Goal: predict MPG rating of a car from a set of features/attributes A_i

Decision tree example

Examples (each row is feature set for a sample car):

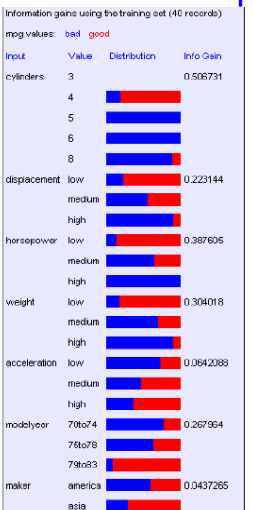
mpg	cylinders	displacement	horsepower	weight	acceleration	model year	maker
good	4	low	low	low	high	00to03	asia
bad	6	medium	medium	medium	medium	95to99	america
bad	4	medium	medium	medium	low	00to03	europe
bad	8	high	high	high	low	95to99	america
bad	6	medium	medium	medium	medium	95to99	america
bad	4	low	medium	low	medium	95to99	asia
bad	4	low	medium	low	low	00to03	asia
bad	6	high	high	high	low	00to03	america
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
bad	8	high	high	high	low	95to99	america
good	8	high	medium	high	high	04to08	america
bad	8	high	high	high	low	00to03	america
good	4	low	low	low	low	04to08	america
bad	6	medium	medium	medium	high	00to03	america
good	4	medium	low	low	low	04to08	america
good	4	low	low	medium	high	04to08	america
bad	8	high	high	high	low	95to99	america
good	4	low	medium	low	medium	00to03	europe
bad	5	medium	medium	medium	medium	00to03	europe

R. Quinlan

Decision tree example

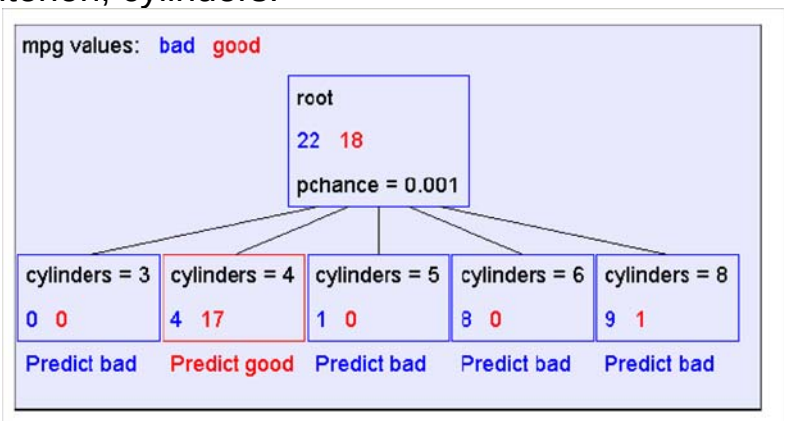
Simple assessment of information gain: how much does a particular feature A_i help to classify a car with respect to MPG?

Decision tree example



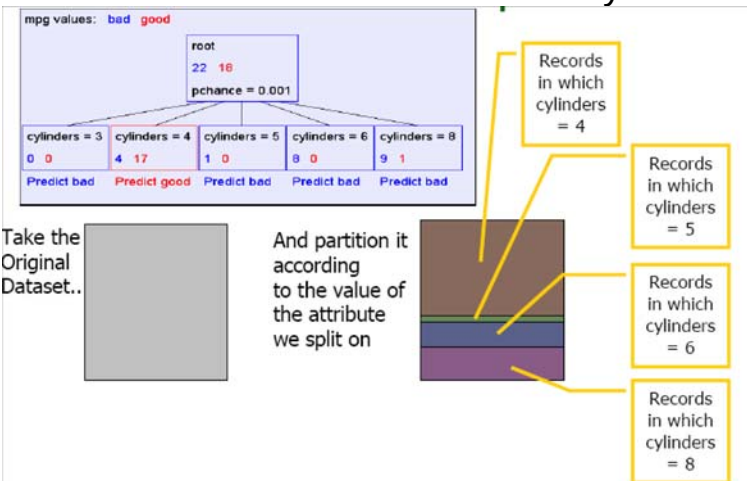
Decision tree example

Begin the decision tree: start with most informative criterion, cylinders:



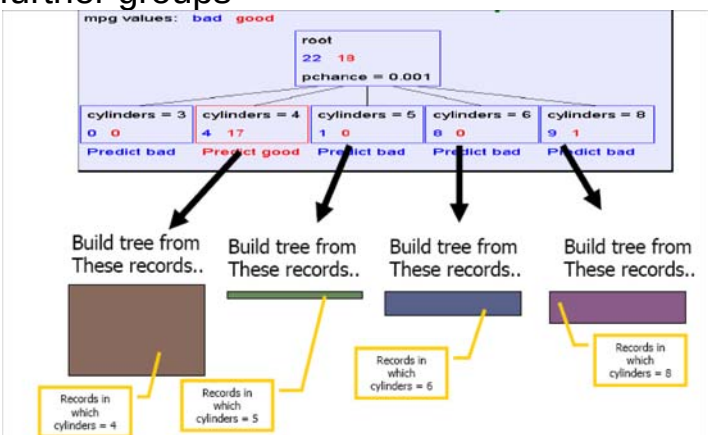
Decision tree example

Recursion: build next level of tree. Initially have:



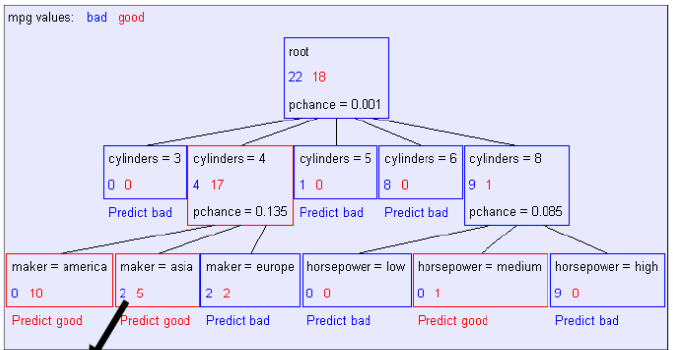
Decision tree example

Now build sub-trees: split each set of cylinder numbers into further groups-



Resulting next level:

Decision tree example

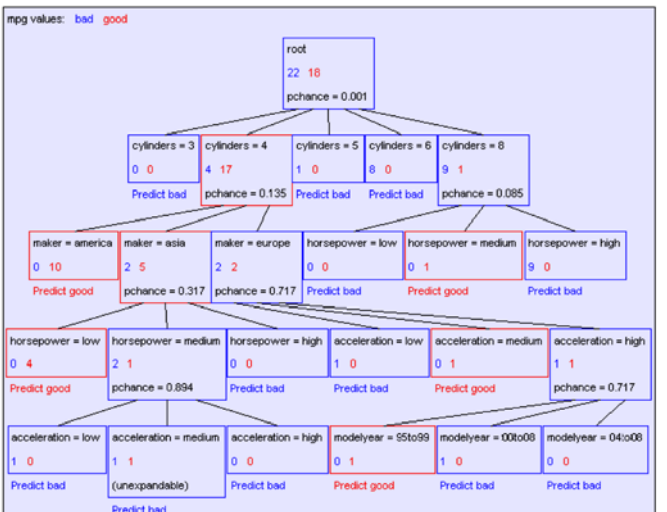


Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

(Similar recursion in the other cases)

Decision tree example

Final tree:



Decision tree example

Points:

- Don't split node if all records have same value (e.g. cylinders $= 6$)
- Don't split node if can't have more than 1 child (e.g. acceleration $=$ medium)

Pseudocode:

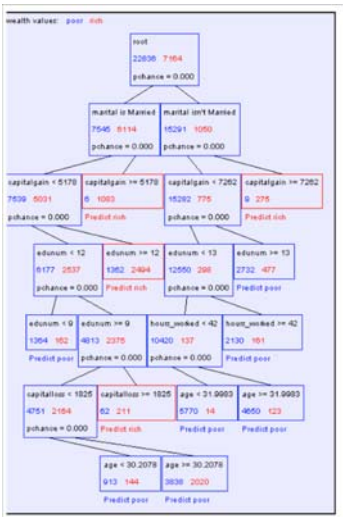
Program Tree(Input, Output)

If all output values are the same,
then return leaf (terminal) node which predicts the unique output
If input values are balanced in a leaf node (e.g. 1 good, 1 bad in acceleration)
then return leaf predicting majority of outputs on same level (e.g. bad in this case)
Else find attribute A_i with highest information gain

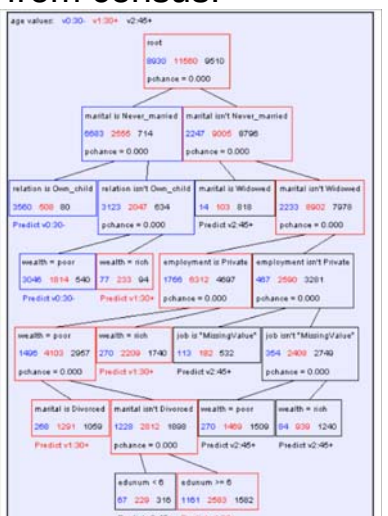
If attribute A_i at current node has m values

then Return internal (non-leaf) node with m children
Build child i by calling Tree(NewIn,
NewOut), where
NewIn $=$ values in
dataset consistent with value A_i and all
values above this node

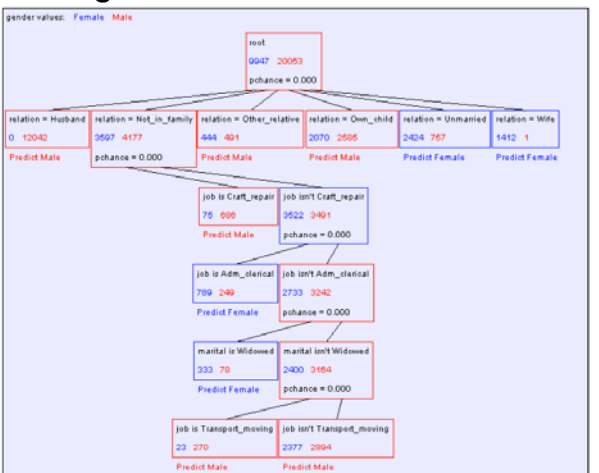
Another decision tree: prediction of wealth from census data (Moore):



Prediction of age from census:



Prediction of gender from census:



A. Moore

2. Important point: always cross-validate

It is important to test your model on *new* data (test data) which are different from the data used to train the model (training data).

This is *cross-validation.*

Cross-validation error – 2% is good; 40% is poor.

3. Background: mass spectroscopy

What does a mass spectrometer do?

1. It measures masses of molecules better than any other technique.

2. It can give information about chemical (in particular protein) compositions of tissue samples.

Mass spectroscopy

How does it work?

1. Takes unknown molecule M, adds i protons to it giving it charge $+i$ (forming MH_i^+)

2. Accelerates ion MH_i^+ in *known* electric field E.

3. Measures time of flight along a *known* distance D.

4. Time T of flight is inversely proportional to electric charge i and proportional to mass m of ion.

Mass spectroscopy

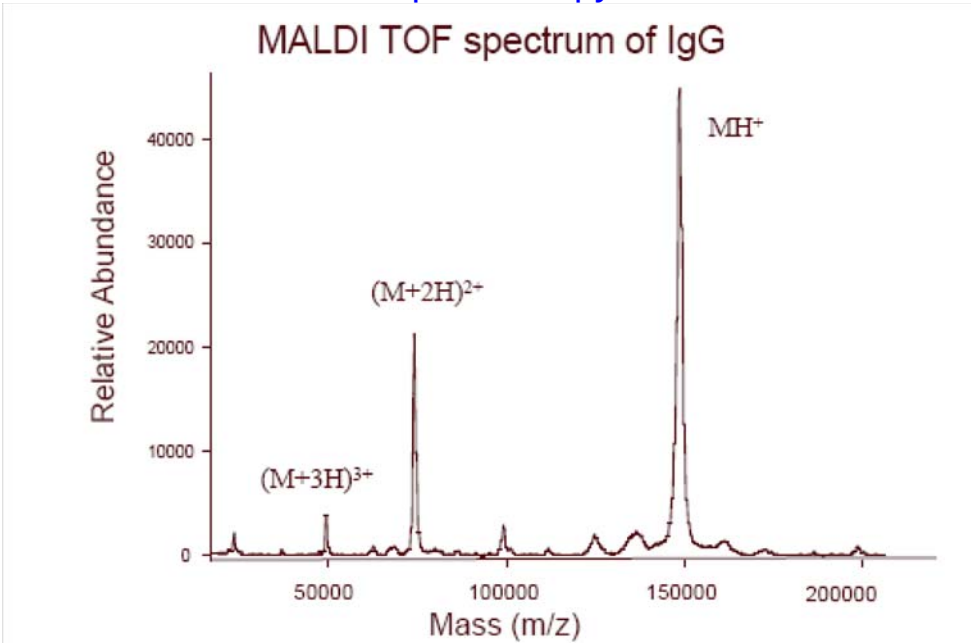
Thus

$$T \propto m/i$$

So mass spectrometer measures ratio of mass m and charge i (also known as z), i.e., $m/i = m/z$.

With a large number of molecules in a biosample, this gives a spectrum of z/m values, which allows identification of molecules in sample (below IgG $=$ immunoglobin G)

Mass spectroscopy

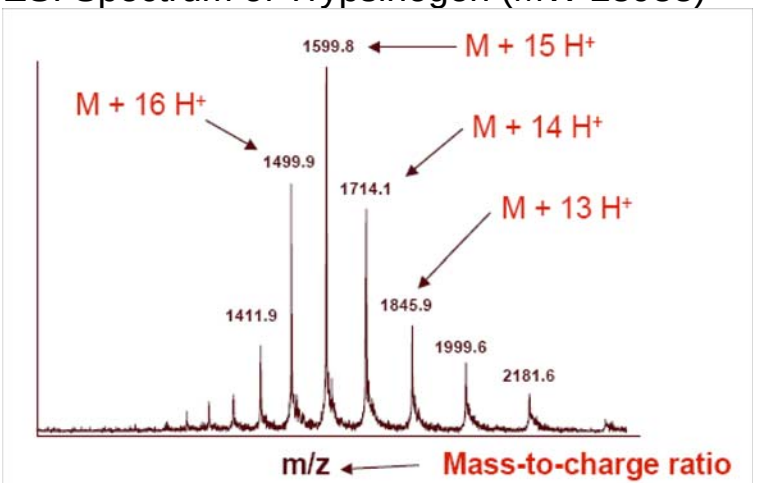


Mass spectroscopy

Mass spectrum shows the results:

Mass spectroscopy

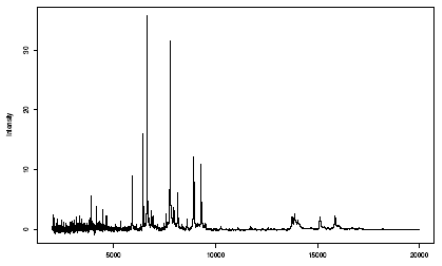
ESI Spectrum of Trypsinogen (MW 23983)

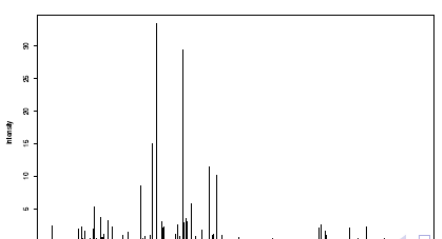


4. Dimensional reduction (G. Izmirlian):

Sometimes we perform a *dimension reduction* by reducing mass spectrum information of (human) subject i to store only peaks:

Mass spectroscopy





Mass spectroscopy

Then have (compressed) peak information in feature vector

$$\mathbf{x} = (x_1, \ldots, x_d),$$

with $x_k =$ location of k^{th} mass spectrum peak (above a fixed threshold).

Compressed or not, outcome value to feature vector $\mathbf{x}_i$ for subject i is $y_i = \pm 1$.

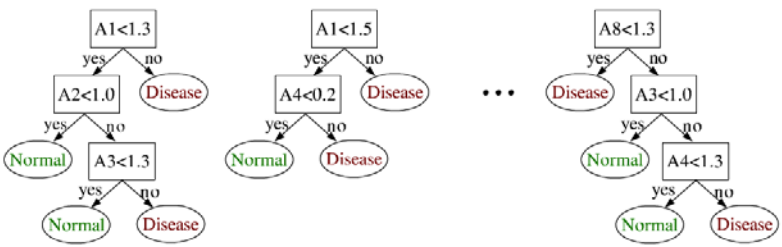
5. Random forest example

Example (Guerts, et al.):

Normal/sick dichotomy for RA and for IBD based on blood sample protein markers (above - Geurts, et al.):

We now build a forest of decision trees based on differing attributes in the nodes:

Random forest application



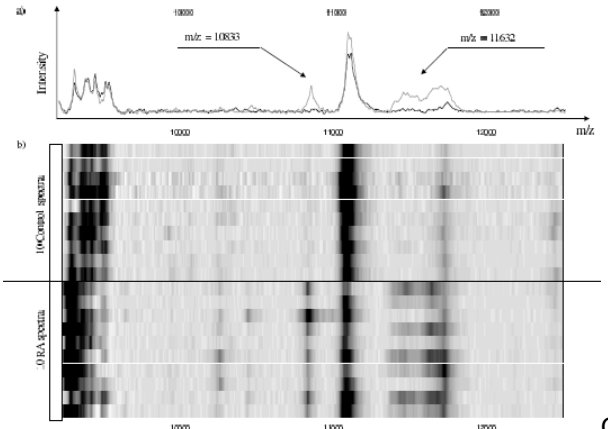
Note: different trees have access to a *different* random sub-collection of the feature set $\{A_i\}_{i=1}^{n}$, or to a *different* random subcollection of the data.

Random forest application

For example: Could use mass spectroscopy data as above to determine disease state

Mass Spec segregates proteins through spectrum of m/z ratios (again $m =$ mass; $z =$ charge).

Random forest application



Geurts, et al.

Random Forests:

Advantages: accurate, easy to use (Breiman software), fast, robust

Disadvantages: difficult to interpret

More generally: How to combine results of different predictors (e.g. decision trees)?

Random forests are examples of *ensemble methods*, which combine predictions of weak classifiers $p_i(\mathbf{x})$.

Ensemble methods: observations

1. **Boosting:** As seen earlier, take linear combination of predictions $p_i(\mathbf{x})$ by classifiers i (assume these are decision trees)

$$f(\mathbf{x}) = \sum_i a_i p_i(\mathbf{x}), \tag{1}$$

where $p_i(\mathbf{x}) = \begin{cases} 1 & \text{if } i^{th} \text{ tree predicts illness} \\ -1 & \text{otherwise} \end{cases}$,

and predict $y = 1$ if $f(\mathbf{x}) \geq 0$ and $y = -1$ if $f(\mathbf{x}) < 0$.

Ensemble methods: observations

2. **Bagging:** Take a vote: majority rules (equivalent in this case to setting $a_i = 1$ for all i in (1) above).

Example of a **Bagging** algorithm is *random forest*, where a forest of decision trees takes a vote.

General features of a random forest:

If original feature vector **x** $\in \mathbb{R}^d$ has d features $A_1, \ldots, A_d$,

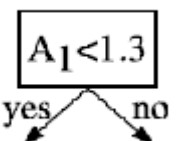
- Each tree uses a random selection of $m \approx \sqrt{d}$ features $\{A_{i_j}\}_{j=1}^{m}$ chosen from *all* features A_1, $A_2, \ldots,$ A_d; the associated feature space is different (but fixed) for each tree and denoted by F_k, $1 \leq k \leq K = \#$ trees.

  (Often $K = \#$ trees is large; e.g., $K = 500$).

- For each split in a tree node based on a given variable choose the variable A_i to be used from its information content.

Information content in a node

To compute information content of a node:



Information content in a node

Assume input set to node is S: then information content of node N is

$$I(N) = |S|\ H(S) - |S_L|\ H(S_L) - |S_R|\ H(S_R),$$

where

$|S|$ = input sample size; $|S_{L,R}|$ = size of left, right subclasses of S

$H(S)$ = Shannon entropy of S = $-\sum_{i=\pm 1} p_i \log_2 p_i$

Information content in a node

with

$$p_i = \widehat{P}(C_i|S) = \text{proportion of class } C_i \text{ in sample } S.$$

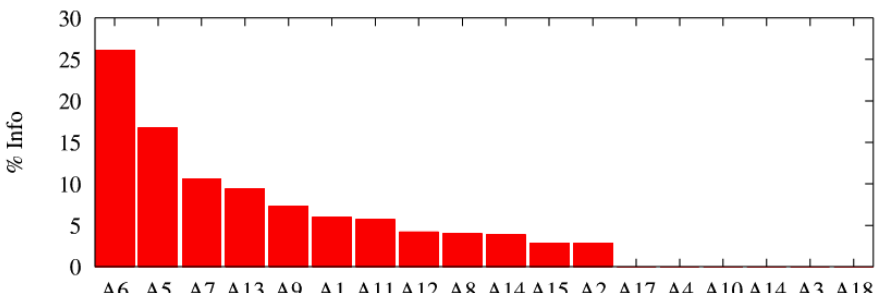
[later we will use *Gini index*, another criterion]

Thus $H(S) =$ "variablity" or "lack of full information" in the probabilities p_i forming sample S input into current node N.

$$I(N) = \text{"information from node } N\text{"}.$$

Information content in a node

For each variable A_i, average over all nodes N in all trees involving this variable to find average information content $H_{\mathsf{av}}(A_i)$ of A_i.



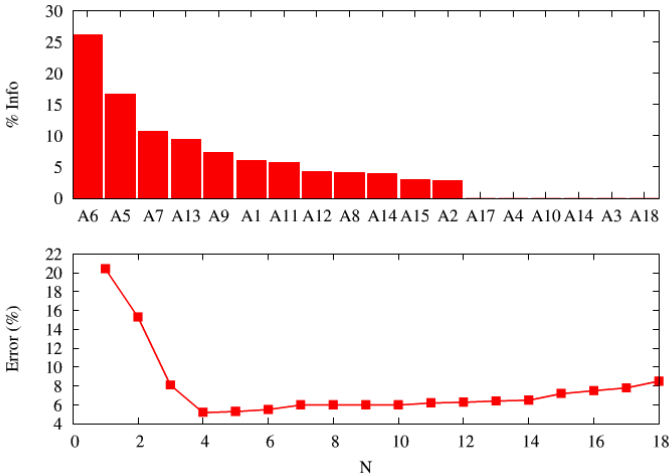
Information content in a node

(a) Rank all variables A_i according to information content

(b) For each fixed $n_1 < n$ rebuild the Random Forest using only the first n_1 variables.

Select n_1 which minimizes prediction error.

Information content in a node



Geurts, et al.

Random forests: application

Application to:

- early diagnosis of Rehumatoid arthritis
- rapid diagnosis of inflammatory bowel diseases (IBD)

Random forests: application

3 patient groups (University Hospital of Liege):

	RA	IBD
Disease patients	34	60
Negative controls	29	30
Inflammatory controls	40	30
Total	103	120

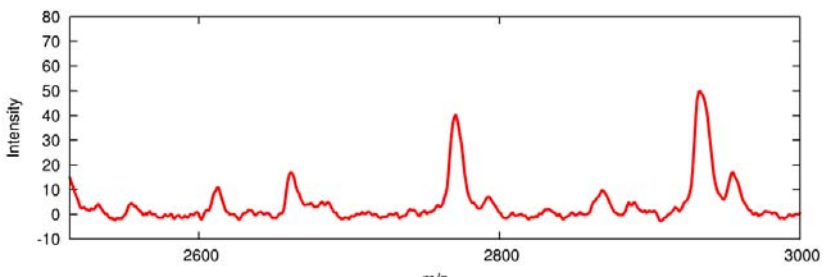
Mass spectra obtained by SELDI-TOF mass spectrometry on protein chip array proteins:

Random forests: application

- Hydrophobic (H4) chips
- weak cation-exchange (CM10) chips
- strong cation-exchange (Q10) chips

Random forests: application

Feature vectors for tissue classification: $\mathbf{x} \in F$ consists of about 15,000 Mass Spectometry values in each case.

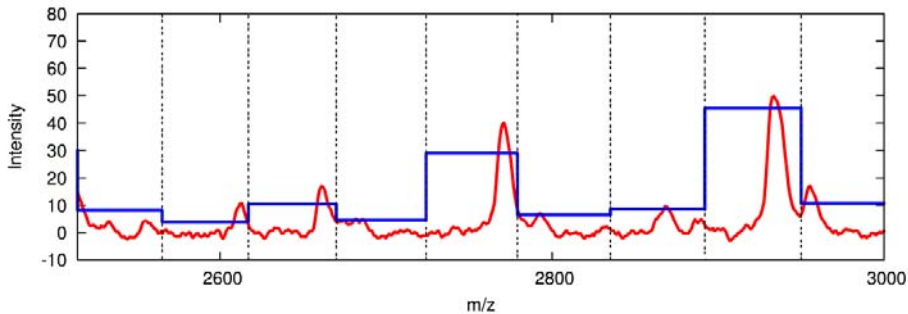


Guerts, et al.

Random forests: application

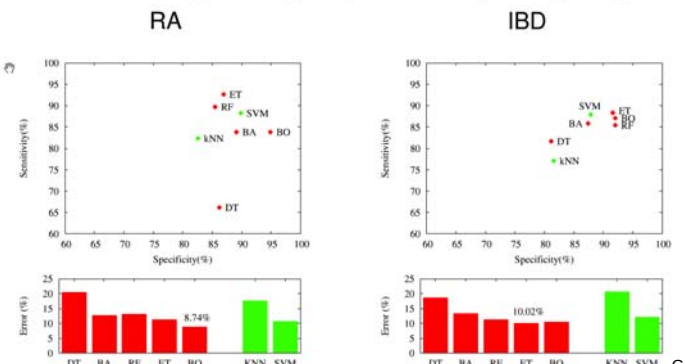
Effective dimension reduction method: Discretize horizontally and vertically to go from 15,000 to 300 variables

Random forests: application



Random forests: application

Sensitivity and specificity:



Guerts, et al.

Random forests: application

Above: accuracy measures for
DT=Decision tree;
RF=random forest;
kNN = k-nearest neighbors;
BA = bagging (bootstrapped resampled tree ensembles);
BO = Boosting;
ET = Extra trees (variation on RF)

Random forests: application

Note on sensitivity and specificity: use confusion matrix

Test outcome

	Actual Condition: True	Actual Condition: False
Positive	TP	FP
Negative	FN	TN

$$\text{Sensitivity} = \frac{TP}{TP + FN} = \frac{TP}{\text{Total positives}}$$

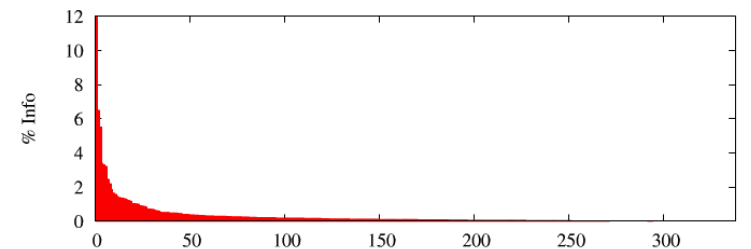
Random forests: application

$$\text{Specificity} = \frac{TN}{TN + FP} = \frac{TN}{\text{Total negatives}}$$

$$\text{Positive predictive value} = \frac{TP}{TP+FP} = \frac{TP}{\text{Total predicted positives}}$$

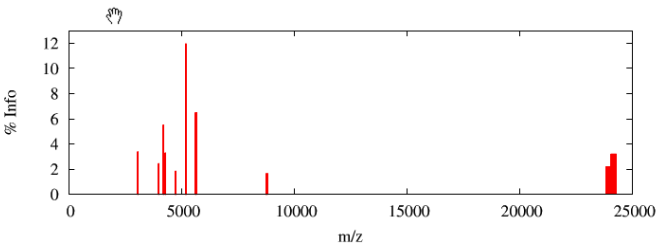
Random forests: application

Variable ranking on the IBD dataset:



10 most important variables in spectrum:

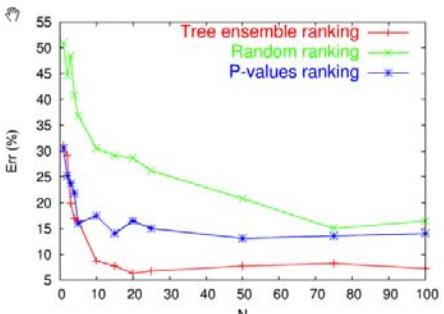
Random forests: application



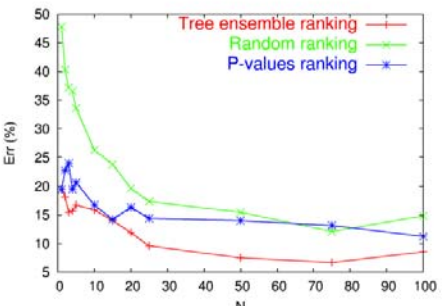
RF-based (tree ensemble) - based variable ranking
vs. variable ranking by individual variable p values:

Random forests: application

RA



IBD



6. RF software:

Spider:
http://www.kyb.tuebingen.mpg.de/bs/people/spider/whatisit.html

Leo Breiman:
http://www.stat.berkeley.edu/~breiman/RandomForests/cc_software.htm

WEKA machine learning software
http://www.cs.waikato.ac.nz/ml/weka/
http://en.wikipedia.org/wiki/Weka_(machine_learning)