Regression Trees

September 23, 2019

Piecewise constant approximation to regression

- Let us consider a generic regression problem

$$Y = f(X) + \epsilon,$$

where X belongs to some set of possible values for the input.

- The goal is to estimate $f(X)$.
- In the regression tree approach, we fit $f(X)$ in the simplest possible way, namely, by functions that are constant on a partition of the domain of X.
- Formally, let $R_1, \dots, R_M$ be the partition of the domain of X, i.e. R_i's are disjoint and they cover the input space. We seek for a fit of the form

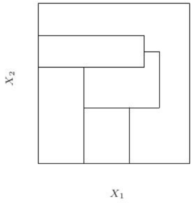
$$\widehat{f}(X) = \sum_{m=1}^{M} c_m \, \mathbb{I}_{R_m}(X),$$

where $\mathbb{I}_A$ is indicator function of a set A.

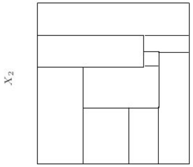
Binary partition

The choice of computationally convenient partitions is desirable, which can be done in the case of the X input space given by $\mathbb{R}^p$.

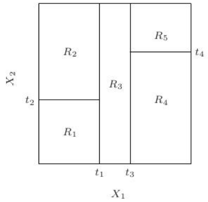
- One can consider simple partition



- One can consider rectangular partitions



- The recurrent split type partition – binary partition



- Split sets formal definition:

$$R_m = \bigcap_{k=1}^{K_1} \{x \in \mathbb{R}^p : x_{i_{1k}} > c_{1k}\} \cap \bigcap_{k=1}^{K_2} \{x \in \mathbb{R}^p : x_{i_{2k}} \leq c_{2k}\}$$

Mathematical formalism

The partition sets

$$R_m = \bigcap_{k=1}^{K_1} \{x \in \mathbb{R}^p : x_{i_{1k}} > c_{1k}\} \cap \bigcap_{k=1}^{K_2} \{x \in \mathbb{R}^p : x_{i_{2k}} \leq c_{2k}\}$$

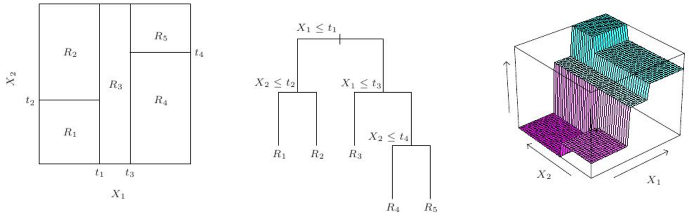
- K_1–number of times one goes to the right of a partition point.
- K_2–number of times one goes to the left of a partition point.
- i_{1k}-the coordinate of x based on which the kth split point requires to go up with this coordinate to end up in R_m.
- i_{2k}-the coordinate of x based on which the kth split point requires to go down with this coordinate to end up in R_m.
- c_{1k}'s–the split points which the i_{1k}th coordinate in the set R_m is above.
- c_{2k}'s–the split points which the i_{1k}th coordinate in the set R_m is below.

This is a bit messy, isn't it?
Is it the best we can do?

No, regression tree visualization is much better than the mathematical formalism.

Binary partition = binary tree

- A binary partition represents as a decision tree T – a sequence of decisions



- **How to choose the values over each partition?**
  Answer: **Minimize the least squares**

$$\sum_{i=1}^{N} |y_i - \hat{f}(x_i)|^2 = \sum_{i=1}^{N} \sum_{m=1}^{M} \sum_{x_i \in R_m} |y_i - \hat{c}_m|^2 = \sum_{m=1}^{M} N_m\, Q_m(T),$$

  where N_m is the number of the inputs in R_m and

$$N_m\, Q_m(T) = \sum_{\{i : x_i \in R_m\}} |y_i - \hat{c}_m|^2$$

- For a fixed partition (tree) T what is the best choice of $\hat{c}_m$?

The optimal solution – not so optimal

- The solution is

$$\hat{c}_m = \frac{\sum_{x_i \in R_m} y_i}{N_m}$$

- If we do not put any limitation on how large the final tree can be, then we get a **“perfect” fit** which is useless.
- Namely, the perfect fit is a partition such that in each partition cell there is only one data point and we put the observed value of Y for this cell.
- We have to limit the number of sets in the partition.
- We will also look for an efficient way of building the regression tree, so that the sum of squares is getting fast smaller.

Growing tree – a ‘greedy’ way

- We will **grow a regression tree** (finding splitting variables x_{j_m}’s and split points t_m’s).
- In general, it is difficult to find the optimal tree with limited number of partition sets.
- The idea is to grow a tree fast and then once a large tree is created to prune (trim) it by removing not so optimal branches
- It is done by removing some split points t_m’s.
- When growing the tree at each step we want to do so that the least squares are decreasing fast.
- Terminology: R_m are called terminal nodes or leaves of the tree, the split points are also called the internal nodes.

Details on tree growing

- Let j_r denote the **index of the split variable at the step r**, i.e. the coordinate X_{j_r} will be used for the rth split.
- Let t_r be the **split point to be used at the step r**.
- How to determine j_r and t_r efficiently?
- Let $R_{1,r-1}, \dots R_{m_{r-1},r-1}$ be the sets obtained for the $r-1$ partition (leaves at the step $r-1$).
- Fix j so that a potential split variable is x_j.
- A potential split point t of $R_{k,r-1}$ must be one of x_j's for those x's that are in $R_{k,r-1}$.
- For such x_j's let consider the split $R_1(x_j)$, $R_2(x_j)$ of $R_{k,r-1}$.
- Choose x_j's such that the reduction of the mean square error is the largest, i.e. choose x_j's maximizing

$$N_{k,r-1}\, Q_k(T_{r-1}) - \sum_{x_i \in R_1(x_j)} (y_i - \hat{c}_{1,x_j})^2 - \sum_{x_i \in R_2(x_j)} (y_i - \hat{c}_{2,x_j})^2,$$

$N_{k,r-1}$ – the number of x's in $R_{k,r-1}$, $\hat{c}_{1,x_j}$ and $\hat{c}_{2,x_j}$ – averages of the responses over the splits $R_1(x_j)$, $R_2(x_j)$, respectively.

Choice of the new split

- The described procedure should run over all:
  1. partitions of the previous step, i.e. over
  $$R_{1,r-1}, R_{2,r-1}, \dots, R_{m_{r-1},r-1}$$
  2. choices of j's, i.e. all choices of the split variable x_j
  3. all splitting points $x_{j,i}$ for the inputs x_i's in $R_{k,r-1}$.
- The choice of the splitting that results in the largest drop in the mean square error determines to which a partition the set $R_{k,r-1}$ will be split next, by which a splitting variable x_{j_r}, and by which a split value t_r.
- It should be noted that t_r can be chosen as one of the jth coordinate of the inputs x_i residing in $R_{k,r-1}$.
- We proceed with the tree growing until a prescribed number of the leaves is achieved (usually chosen generously, more rectangles then it is expected to be needed – greedy aspect of the approach). Alternatively, we proceed until there is no less than a prescribed number of observations per node (easier to control).

The obtained tree is too big

- Which branches need to be grown is difficult to decide during the growing process since some small drop in the mean square value can later lead to a larger drop in the same branch at the future steps of tree growing phase and other way around.
- The obtained tree is typically over-fitting the data (too many sets R_j's comparing to the number of the data points).
- We are seeking a reduction of the tree size.
- This can be achieved by cutting some of the branches of an overgrown tree.

Cost complexity (weakest link) pruning

- We choose the **cost complexity** function of a tree T with $|T|$ leaves:

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m \, Q_m(T) + \alpha|T|,$$

  where α is a pre-specified cost per leave – **penalty for the size is $\alpha|T|$**.

- A **pruned subtree** T_0 of T is a tree that can be obtained from the original tree by collapsing successively from the bottom its branches into smaller number of branches (one collapses at a higher located node so that all sub-nodes located on the collapsed branch below have to be collapsed into this common branch). This is done as follows
  - Collapse two branches at the bottom (leaves) into one, one at the time.
  - Check which of these produces the **smallest drop** in the complexity cost. The collapsing (pruning) that produces the minimal drop is kept to redefine the tree (all other collapsing producing higher complexity cost drop are not used at this step).
  - Repeat until we get the final **single leaf** (the single set partition).
  - From the obtained sequence of successively pruned subtrees we choose the one with T_0 for which $C_\alpha(T_0)$ is the smallest. [1]

[1] It can be shown that this is indeed the most cost efficient subtree of the original tree.

Pruning versus growing

It may appear that the pruning process is the reverse of growing a tree.

However this is not the case!

We collapse those that has the smallest drop in the complexity cost, not the highest.

- Trimming go through a different path than growing although the starting point one process is the terminal point of the other processes and vice verse.
- It is shown by a mathematical argument that so created sequence of trees will include the subtree with the **minimal value of complexity** among those that can be obtained from the originally grown tree.
- We do not discuss the details of such an argument. Can you provide such?

Choosing cost parameter – validation step

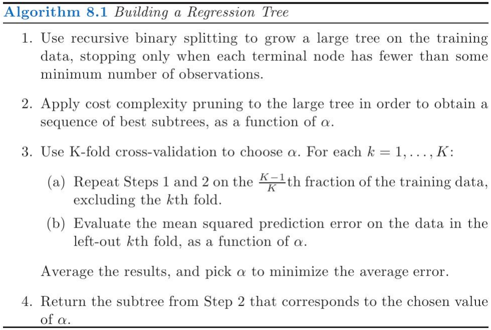
- The remaining unsolved issue is selection of α.
- A number of candidates is taken: $\alpha_1, \dots, \alpha_L$.
- On the **training set** optimal sub-trees T_{α_l} will be build using each α_l's.
- The one that is producing the smallest mean square error, i.e. $\hat{\alpha} = \alpha_{l_0}$ that minimizes means square error on the **validation set** of the data

$$\sum_{m=1}^{|T_{\hat{\alpha}}|} N_m \, Q_m(T_{\hat{\alpha}}) = \min_{l=1,\dots,L} \sum_{m=1}^{|T_{\alpha_l}|} N_m \, Q_m(T_{\alpha_l})$$

- On the **testing set**, the chosen method is tested to obtain a realistic error of the fit.
- There is no guarantee that the method produce anything sensible but usually it does – the nature data mining approach.
- Sometimes there is a reluctance to 'waste' data for validation so the cross-validation is used.

Choosing cost parameter – cross-validation

- Frequently, a cross-validation method is used instead of test-validation approach
- Popular approach:
  - the advantage is that it does not require extra validation data and it produces a method with optimal properties on the training data
  - the disadvantage that these reasonable properties still apply only to the training data and maybe false outside of it
- Proceed according to the algorithm

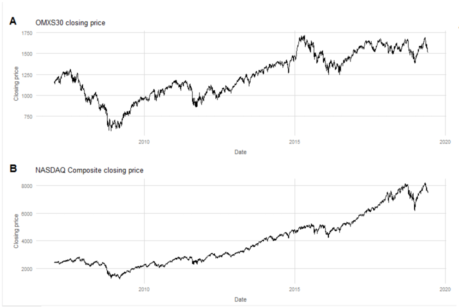


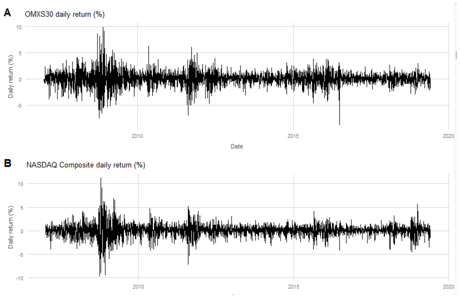
Algorithm 8.1 *Building a Regression Tree*

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold cross-validation to choose α. For each $k = 1, \dots, K$:
   (a) Repeat Steps 1 and 2 on the $\frac{K-1}{K}$th fraction of the training data, excluding the kth fold.
   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

   Average the results, and pick α to minimize the average error.
4. Return the subtree from Step 2 that corresponds to the chosen value of α.

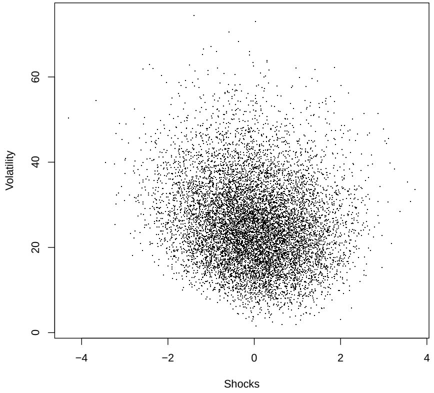
Financial volatility data





- What is volatility? The daily price shocks enter the return data multiplied by a random scale called volatility.
- GARCH and similar models allow to disentangle volatility from shocks.
- Studying their relation is very important for a financial analyst.

Data: 10000 volatility-shock pairs



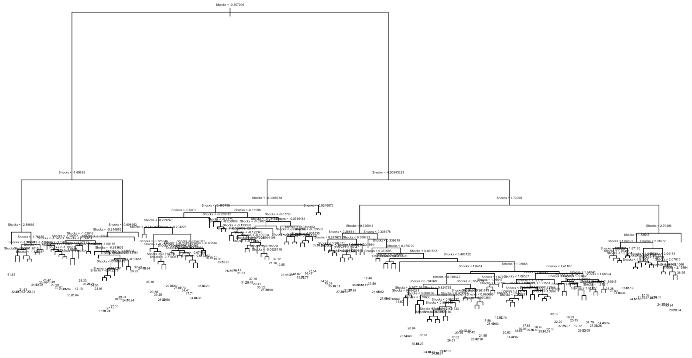
- Is there any visible relation between the two?
- Is the volatility larger when shocks are large or when they are small?
- When the volatility is larger: for positive or negative shocks?
- **Try to learn from the data.**

Regression tree: a data mining/machine learning approach

Let us choose three subsets from our data

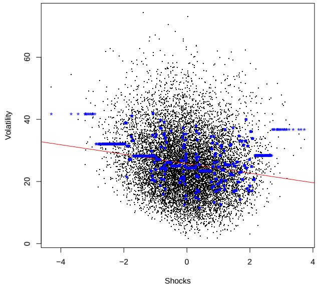
- **Training sample** - the one on which we will learn something: 50% of the data.
- **Validating sample** - the one on which the choice of the method will be validated: 25% of the data.
- **Testing sample** - the one on which the chosen method will be evaluated: 25% of the data.

Rich tree trained on the training sample

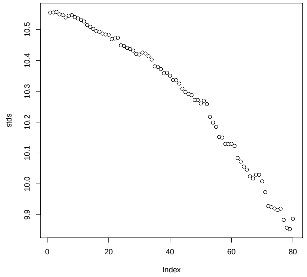


Trimming/prunning of the tree

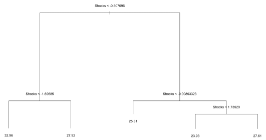
- What does our ‘rich’ tree tell us?



- It does overfit.
- We trim the original tree hoping to get the better fit (pruning and checking if the tree produces better ’fruits’). We utilize for this the validating data set.

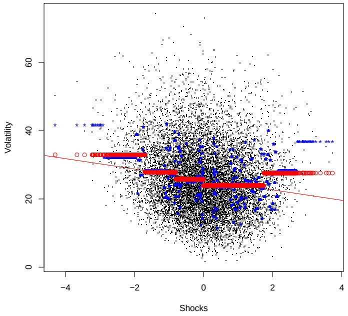


The smallest standard deviation (9.853157) with $k = k[80]$, use this pruned tree.

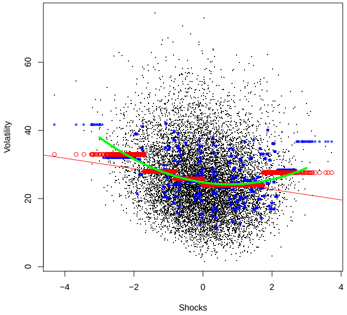


Final fit comparison – volatility smile

The illustration of the fits: 'volatility smile' with the negative news having larger volatility (risk).



One can try regress against the quadratic function.



Further details in the lab.

Two-classes classification problem

- The binary tree approach can be applied to a classification problem
- An object with features measurement $\mathbf{X}$: $p \times 1$ belongs to one of classes $\mathbf{G}_0$ and $\mathbf{G}_1$.
- Split the feature space into two parts $\mathcal{R}_0$ and $\mathcal{R}_1 = \mathcal{R}_0^c$.
    - If $\mathbf{x} \in \mathcal{R}_0$ classify to class $\mathbf{G}_0$.
    - If $\mathbf{x} \in \mathcal{R}_1$ classify to class $\mathbf{G}_1$.
- $Y = 0$ if the object at hand is in Class 0 ($\mathbf{G}_0$) and $Y = 1$ otherwise ($\mathbf{G}_1$).
- Classification as a prediction binary variable:

$$R(\mathbf{X}) = \begin{cases} 1; & \mathbf{X} \in \mathcal{R}_1 \\ 0; & \mathbf{X} \in \mathcal{R}_0 \end{cases}$$

- Let R_m, $m = 1, \ldots, M$ be a partition of the feature (input) space. Let $\hat{p}_m$ be the proportion of Class 1 objects that are in R_m. Define

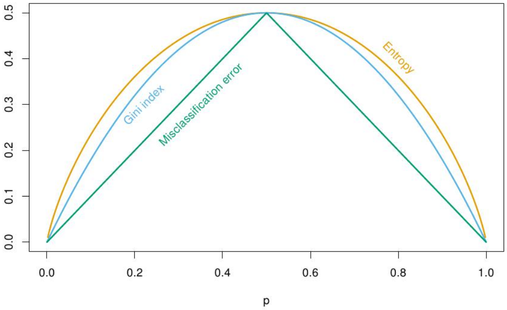
$$\mathcal{R}_1 = \bigcup_{\substack{m=1,\ldots,M \\ \hat{p}_m > 1-\hat{p}_m}} R_m$$

Growing binary tree – the split point rule

- We generally follow the same way as before but this time we need a different criterion for choosing splitting variables and split points.
- Three measures are common:
  - Misclassification error:
    $$Q_m(T) = \min(\hat{p}_m, 1 - \hat{p}_m)$$
  - Gini index:
    $$Q_m(T) = 2\hat{p}_m(1 - \hat{p}_m)$$
  - Cross-entropy or deviance:
    $$Q_m(T) = -\hat{p}_m \log \hat{p}_m - (1 - \hat{p}_m)\log(1 - \hat{p}_m)$$
- All three functions are **convex** so then they decrease with a refinement of a partition.
- As before, the selection in which branches of the tree are growing depends how much the function
  $$\sum_{m=1}^{|T|} N_m\, Q_m(T)$$
  is decreasing with a choice of additional node (split point) – the one that causes the **highest drop** is preferable.

Which criterion to choose?

- Misclassification error: $Q_m(T) = \min(\hat{p}_m, 1 - \hat{p}_m)$
- Gini index: $Q_m(T) = 2\hat{p}_m(1 - \hat{p}_m)$
- Cross-entropy or deviance: $Q_m(T) = -\hat{p}_m \log \hat{p}_m - (1 - \hat{p}_m) \log(1 - \hat{p}_m)$
- Graphs of the three criteria



- For growing a tree, the Gini index or deviance are preferred, while for cost-complexity pruning often the misclassification error is used.

Spam example

- Classification tree methodology for the spam example
  - the **deviance** to grow the tree

$$Q_m(T) = -\hat{p}_m \log \hat{p}_m - (1 - \hat{p}_m)\log(1 - \hat{p}_m)$$

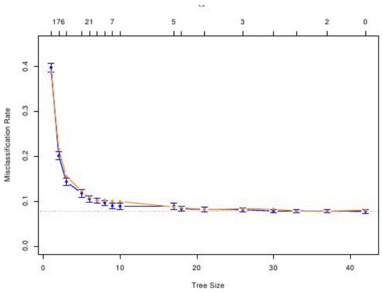
  - the **misclassification rate** to prune it

$$Q_m(T) = \min(\hat{p}_m, 1 - \hat{p}_m)$$

- 10-fold cross-validation was used to determine α – the cost of a nod
  - The tree was built ten times. One on each of the ten nine-tenth data subsets.
  - These ten trees were then trimmed using various preselected α's.
  - For each choice of α we have then ten values of the misclassification rates corresponding to the ten folds of the data
  - We choose the value of α that minimizes the sum of ten misclassification rates.
  - For the chosen α, the standard deviation of the misclassification rate (based on the ten values) is estimated.
  - The original tree is then trimmed on the complete data by using the obtained cost α.

Testing and visualization of the results

- After trimming the original tree, it is tested on the unused testing data set that was set aside
- The results are presented on the following graph (blue line shows averaged cross-validation misclassification rates together with their standard deviations, brown line shows the misclassification rates for the testing sample).



The cross-validation error rates are indexed by a sequence of values of α (not tree size) – top horizontal scale; for trees grown in different folds, a value of α might imply different sizes of the pruned subtrees. The sizes shown at the base of the plot refer to the sizes $|T_\alpha|$ of the pruned original tree.

Discussion and interpretation of the results

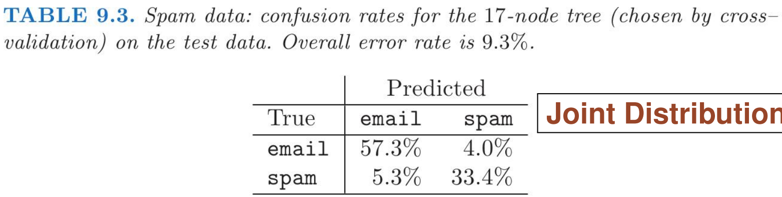


TABLE 9.3. *Spam data: confusion rates for the 17-node tree (chosen by cross–validation) on the test data. Overall error rate is 9.3%.*

	Predicted	
True	`email`	`spam`
`email`	57.3%	4.0%
`spam`	5.3%	33.4%

Joint Distribution

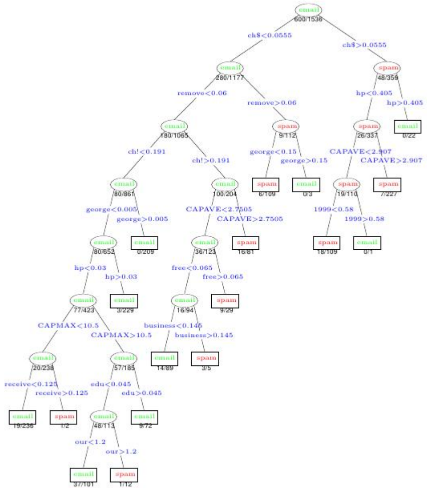
- Interpretation in terms of sensitivity and specificity:
  - Sensitivity: probability of predicting spam given true state is spam.
  - Specificity: probability of predicting e-mail given true state is e-mail.

Conditional probabilities:

$$\textit{Sensitivity} = \frac{33.4}{33.4 + 5.3} = 86.3\%$$

$$\textit{Specificity} = \frac{57.3}{57.3 + 4.0} = 93.4\%$$

Pruned tree and conclusions



- The error flattens out at around 17 terminal nodes
- The pruned tree is shown.
- Of the 13 distinct features chosen by the tree, 11 overlap with the 16 **significant features in the additive model**.
- **The overall error rate shown is about 50% higher than for the additive model!**
- The split variables are shown in blue on the branches, and the classification is shown in every node.
- The numbers under the terminal nodes indicate misclassification rates on the test data.
- The name of the node informs about the dominant class in the partition set.

How to introduce asymmetric cost of misclassification?

- The regression tree in our example was obtained by considering that the both type of mistakes are equally costly. Often they are not as in the spam problem.
- One way to introduce preferences is to duplicate or **multiplicate** the data that we consider undesirable to be misclassified.
- For example, the spam data can be replicate five times while the regular messages kept at their original values.
- It is equivalent to putting the **weight of 5/6** for the spam and **1/6** for the regular mail.
- **Weighting will lead to another tree than the original one!**

Working with the final tree to obtain asymmetry

- One can put a different **cost of misclassification** as discussed in the general approaches to classification.
- For the tree it is equivalent choosing a value of $\theta > 0$ in

$$\mathcal{R}_1(\theta) = \bigcup_{\substack{m=1,\dots,M \\ \frac{\hat{p}_m}{1-\hat{p}_m} > \theta}} R_m,$$

  where $\mathcal{R}_1$ is the part of feature space $\mathcal{X}$ that we go with the spam choice.
- The larger $\theta > 1$ the more costly is misclassification of e-mail as a spam (it is more difficult to be classified as a spam). Why?
- This approach can be used to **any method of classification** by setting the cost of misclassification proportional to θ.

Sensitivity vs. Specificity

- It is possible to imagine that one method works well for one cost of misclassification but works worse for another.
- How to compare two classification methods across whole range of possible misclassification costs?
- The error of misclassification not always tell the whole story because it does not distinguish between two type errors.
- Medical terms for testing of a disease:
  Sensitivity=chances of detecting a disease (**true positive**),
  Specificity=chances of not claiming disease for a healthy individual (**true negative**).
- Be careful in other context with proper interpretation of ‘disease’ and ‘test’.
- Both are probabilities and may depend on a common parameter $P_1(\theta)$ and $P_0(\theta)$. Their dependence can be visualized using different methods.
- How sensitivity and specificity for a given cost θ express using the general theory of classification?

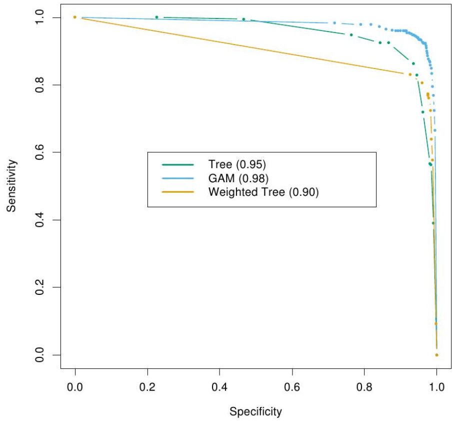
$$P(X \in \mathcal{R}_{0,\theta}|Y = 0) = 1 - P(X \in \mathcal{R}_{1,\theta}|Y = 1), \quad P(X \in \mathcal{R}_{1,\theta}|Y = 1),$$

where $R_{1,\theta}$ is the ‘detection’ region developed under the cost θ.

Comparisons of distributions - topic in visualization

- How different distributions can be graphically compared?
- **qq-plot**: quantile plot. This is a plot of the curve: $\{q_1(\alpha), q_2(\alpha); \alpha \in (0, 1)\}$, where $q_i(\alpha)$ are quantiles of two distributions that we want to compare.
  - If quantile line is 'well represented' by a straight line then the underlying distributions differ only by the location and scale. If the line is $q_1 = q_2$, then the two distribution are equal.
  - Note that the quantile function is an inverse of the cumlative distributions function $q(\alpha) = q$ if and only if $F(q) = \alpha$.
- **density plots (histograms)**: plotting two (estimated) density functions $f_1(x)$ and $f_2(x)$
- **cdf plots**: plotting two (estimated) cdf functions $F_1(x)$ and $F_2(x)$.
- **receiver operating curve (ROC)**: plotting $\{F_1(x), F_2(x); x \in \mathbb{R}\}$ – further the curve is from the diagonal of the square $[0, 1]^2$ more different distributions are
- ROC is often used to compare two different models for the probability $P_1(\theta)$ and $P_2(\theta)$ depending on the same parameter θ, for example, sensitivity and specificity.

ROC for the classfication rules

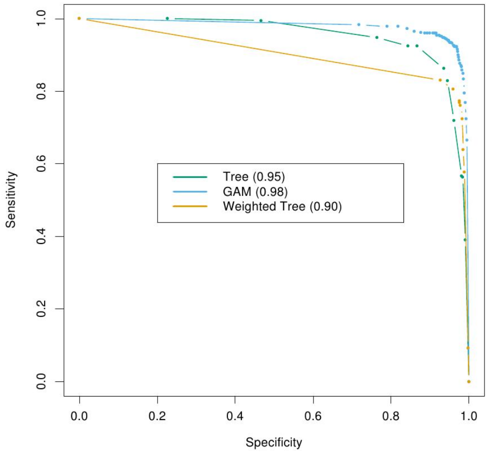


- One can put a different cost of misclassification by choosing a value of $\theta > 0$ in

$$\mathcal{R}_2(\theta) = \bigcup_{\substack{m=1,\dots,M \\ \frac{\hat{p}_m}{1-\hat{p}_m} > \theta}} R_m$$

- The larger $\theta > 1$ the more costly is misclassification of e-mail as a spam (it is more difficult to be classified as a spam).
- The following is ROC for the probability of misclassification as a spam vs. misclassification as e-mail (in function of θ)
- The weighting refers to the weight 5 for each e-mail classified as a spam (in computing the misclassification cost during the tree building)

Comparisons between methods



- ROC curves for the classification rules fit to the spam data.
- Curves that are closer to the northeast corner represent better classifiers.
- GAM classifier dominates the trees.
- The weighted tree achieves better sensitivity for higher specificity than the unweighted tree.
- The numbers in the parentheses in the legend represent the area under the curve.