SF2935 Modern Methods of Statistical Learning

Jimmy Olsson

Tree-based regression and classification

18 September 2018

Idag

Overview

Regression trees

Classification trees

Bagging, random forests

Today

Overview

Regression trees

Classification trees

Bagging, random forests

Setup

- Our data consists of p predictors (inputs) $X = (X_1, \ldots, X_p)$ and a response Y.
- We will consider continuous as well as discrete response variables, corresponding to regression and classification, respectively.
- Let $\mathbb{X} \subseteq \mathbb{R}^p$ be the *predictor space*, i.e., the set of all possible predictors.
- We are given data in terms of N observations $\{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$, where

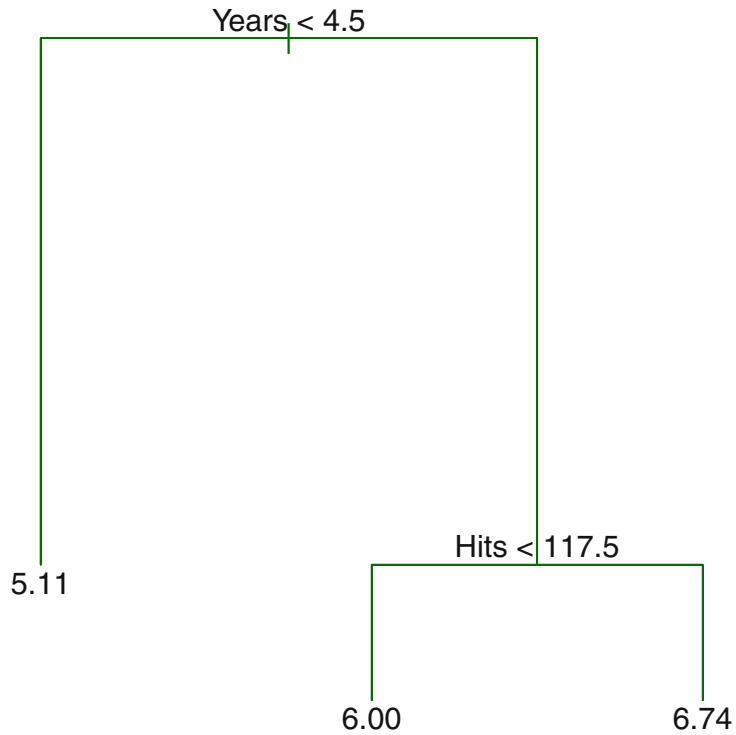
$$x^{(i)} = (x_1^{(i)}, \ldots, x_p^{(i)}).$$

Setup

- Given an input vector X, we wish to predict the response Y.
- We will do this by segmenting the predictor space $\mathbb{X}$ into simple, non-overlapping regions $\{R_j\}_{j=1}^{J}$ covering $\mathbb{X}$.
- For every input in R_j, we make the same prediction, which is simply the mean or the majority class of the response values for the training observations in R_j.
- The segmenting rules will be summarized by a *binary tree*, where the k^{th} node corresponds to a split of form

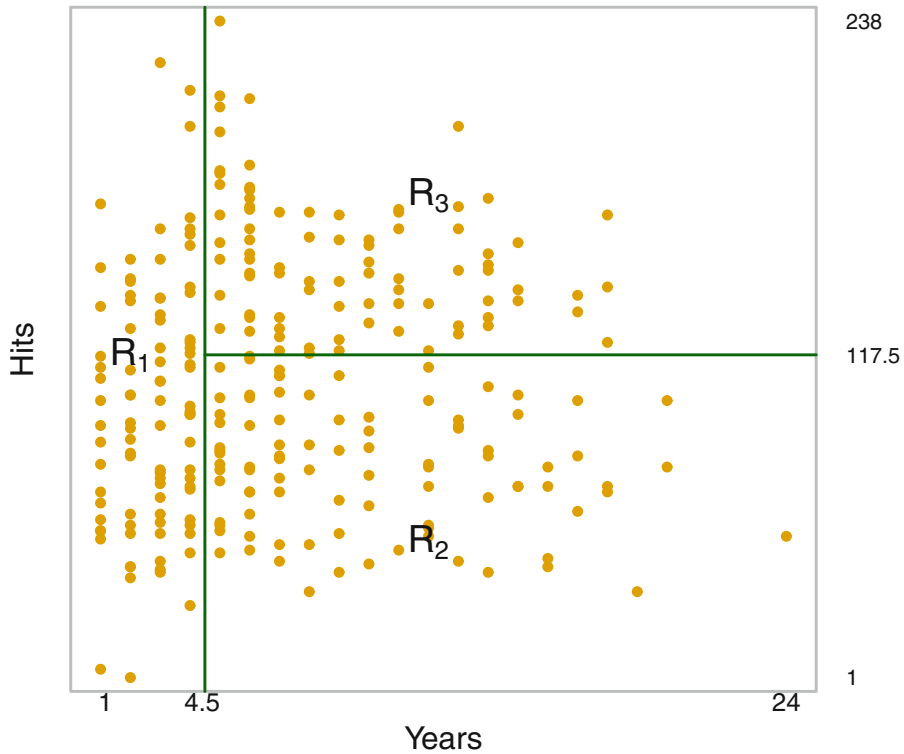
$$X_{j_k} < t_k \Rightarrow \text{left branch},$$
$$X_{j_k} \geq t_k \Rightarrow \text{right branch}.$$

Example: baseball player salaries



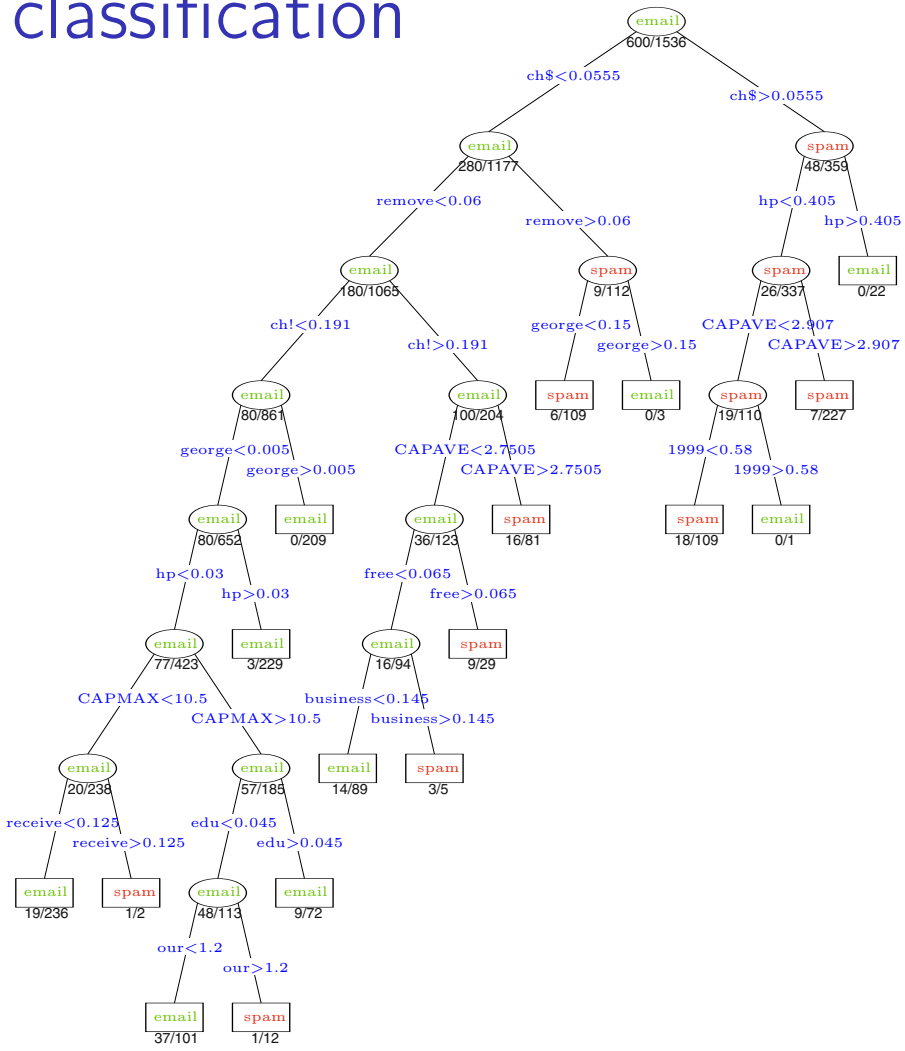
Figur: A regression tree for predicting the log salary (in $1,000$\$) of a baseball player, based on the number of years played and the number of hits made in the previous year.

Example: baseball player salaries (cont.)



Figur: The three-region partition provided by the regression tree in the previous figure.

Example: spam classification



Figur: Tree for spam classification. Split variables are shown in blue. `CAPMAX` and `CAPAVE` = maximal and average length of the longest uninterrupted sequences of capital letters, respectively.

Questions

- How do we construct such trees?
- How large should a tree be?
- What are the pros and cons of tree-based methods?

Today

Overview

Regression trees

Classification trees

Bagging, random forests

Regression trees vs. linear regression

- We generally want to approximate a relation $Y = f(X) + \varepsilon$.
- Linear regression involves linear regression functions

$$f(x) = \beta_0 + \sum_{j=1}^{p} \beta_j x_j.$$

- Today we will instead consider

$$f(x) = \sum_{j=1}^{J} c_j \mathbb{1}_{R_j}(x),$$

for a partition (segmentation) $\{R_j\}_{j=1}^{J}$ of $\mathbb{X}$.

Building a regression tree: cost function

- We will construct the tree by minimizing

$$\mathrm{RSS}(c_1, \ldots, c_J, R_1, \ldots, R_J) = \sum_{i=1}^{N} (y^{(i)} - f(x^{(i)}))^2$$

$$= \sum_{j=1}^{J} \sum_{i:x^{(i)} \in R_j} (y^{(i)} - f(x^{(i)}))^2 = \sum_{j=1}^{J} \sum_{i:x^{(i)} \in R_j} (y^{(i)} - c_j)^2.$$

- For each j, the c_j minimizing each sum of squares $\sum_{i:x^{(i)} \in R_j} (y^{(i)} - c_j)^2$ is always

$$\hat{c}_j = \bar{y}_{R_j} = \text{average over all } y^{(i)} \text{ such that } x^{(i)} \in R_j.$$

- To find the minimizing R_j is however computationally infeasible in general.

Building a regression tree: top-down minimization

- Thus, in order to find regions $\{R_j\}_{j=1}^J$ minimizing

$$\sum_{j=1}^{J} \sum_{i:x^{(i)} \in R_j} (y^{(i)} - \bar{y}_{R_j})^2 \overset{\text{not.}}{=} \sum_{j=1}^{J} Q_{R_j}$$

  we proceed with a *top-down*, *greedy* approach.

- In the first step, when splitting $\mathbb{X}$ into

$$R_1(j,s) = \{x \in \mathbb{X} : x_j < s\} \quad \text{and} \quad R_2(j,s) = \{x \in \mathbb{X} : x_j \geq s\},$$

  we minimize

$$Q_{R_1(j,s)} + Q_{R_2(j,s)}$$

  w.r.t. j and s.

Building a regression tree: top-down minimization (cont.)

- For a given j, the optimal s is quite easily found by scanning though the data.
- Repeating for all j yields an optimal pair (j, s).
- Having found the best split, we partition the data into the two resulting regions and repeat the splitting process on each of these. And so on, and so forth.
- We stop growing the tree when some stopping criterion is reached, e.g., when no region contains more than five training data.

Bias versus variance

- The tree size is a tuning parameter governing the model's complexity.
- A too large tree leads to overfitting (high variance on test sets), while a too small tree leads to high bias.
- The optimal tree size should be determined adaptively from the data.
- One possibility is to split a node only if the split implies an RSS reduction exceeding a certain threshold; this is however treacherous in top-down minimization, as an apparently meaningless split may imply a good split later on.

Cost complexity pruning

- Grow a large tree T_0 while storing the different Q_{R_j} associated with each split.
- We define a subtree $T \subseteq T_0$ obtained by *pruning* T_0 in a *bottom-up* fashion.
- Let $|T|$ denote the number of leaves $R_m(T)$ in T.
- Define, for some $\alpha \geq 0$, the cost function

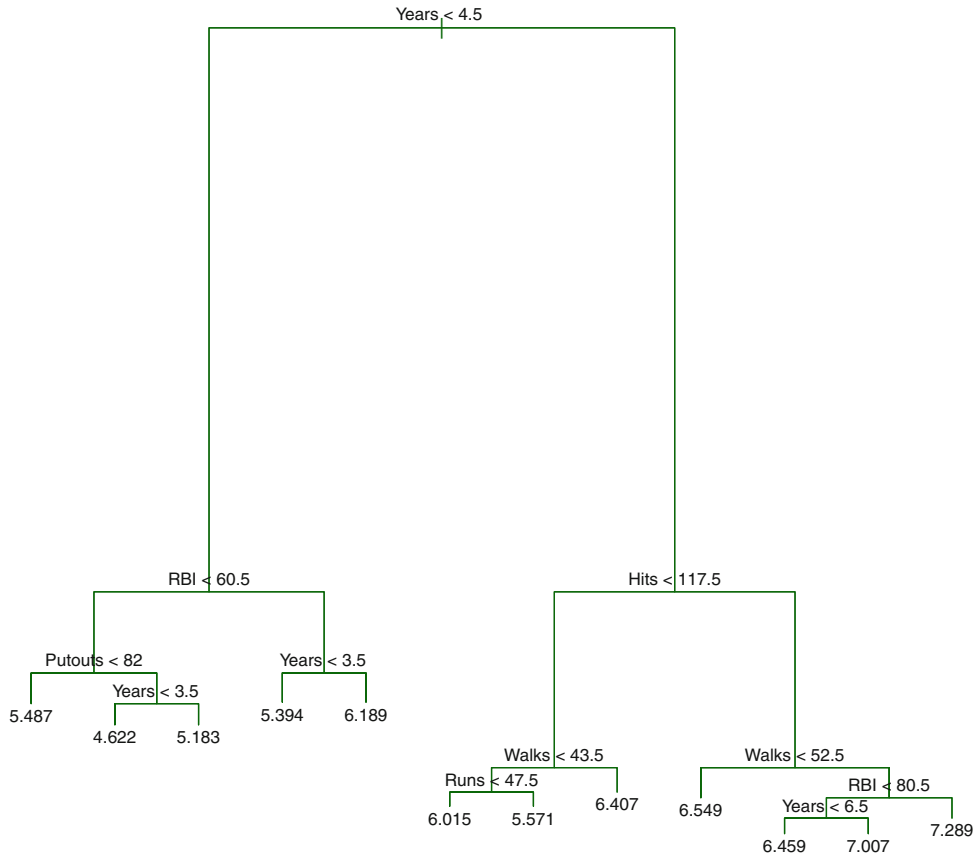
$$C_\alpha(T) = \sum_{m=1}^{|T|} Q_{R_m(T)} + \alpha|T|,$$

  where the $Q_{R_m(T)}$ are available from the first step.
- The idea is to find a subtree T_α minimizing $C_\alpha(T)$.

Cost complexity pruning (cont.)

- Large values of α leads to smaller trees (cf. lasso).
- For $\alpha = 0$, the solution is T_0.
- To find T_α, we create a sequence of subtrees by removing successively leaves whose elimination leads to the smallest increase of $\sum_{m=1}^{|T|} Q_{R_m(T)}$. This sequence must contain T_α.
- To find the optimal α, apply K-fold cross validation.

Example: baseball player salaries (cont.)



Figur: Unpruned tree for the log salary (in $1,000$) of a baseball player.

Today

Overview

Regression trees

Classification trees

Bagging, random forests

Node impurity

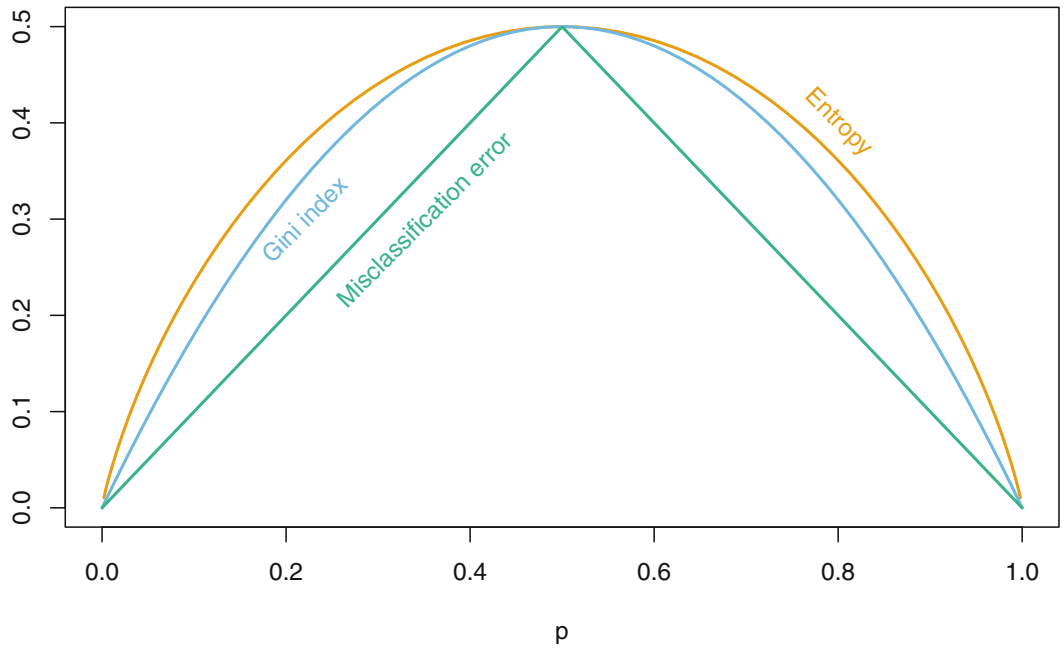
- The task of growing a classification tree is very similar to that of growing a regression tree. The only difference is that the RSS is not meaningful any longer.
- We thus need to find alternative measures $Q_m(T)$ of *node impurity*. For this purpose, define

$$\hat{p}_{m,k} = \frac{1}{|R_m|} \sum_{i:x^{(i)} \in R_m} \mathbb{1}_{\{y^{(i)}=k\}} \quad \text{and} \quad k(m) = \arg\max_k \hat{p}_{m,k}$$

and let, e.g.,

$$Q_m(T) = \begin{cases} 1 - \hat{p}_{m,k(m)} & \textit{misclassification error}, \\ \sum_{k=1}^{K} \hat{p}_{m,k}(1 - \hat{p}_{m,k}) & \textit{Gini index}, \\ -\sum_{k=1}^{K} \hat{p}_{m,k} \log \hat{p}_{m,k} & \textit{cross-entropy}. \end{cases}$$

Node impurity measures

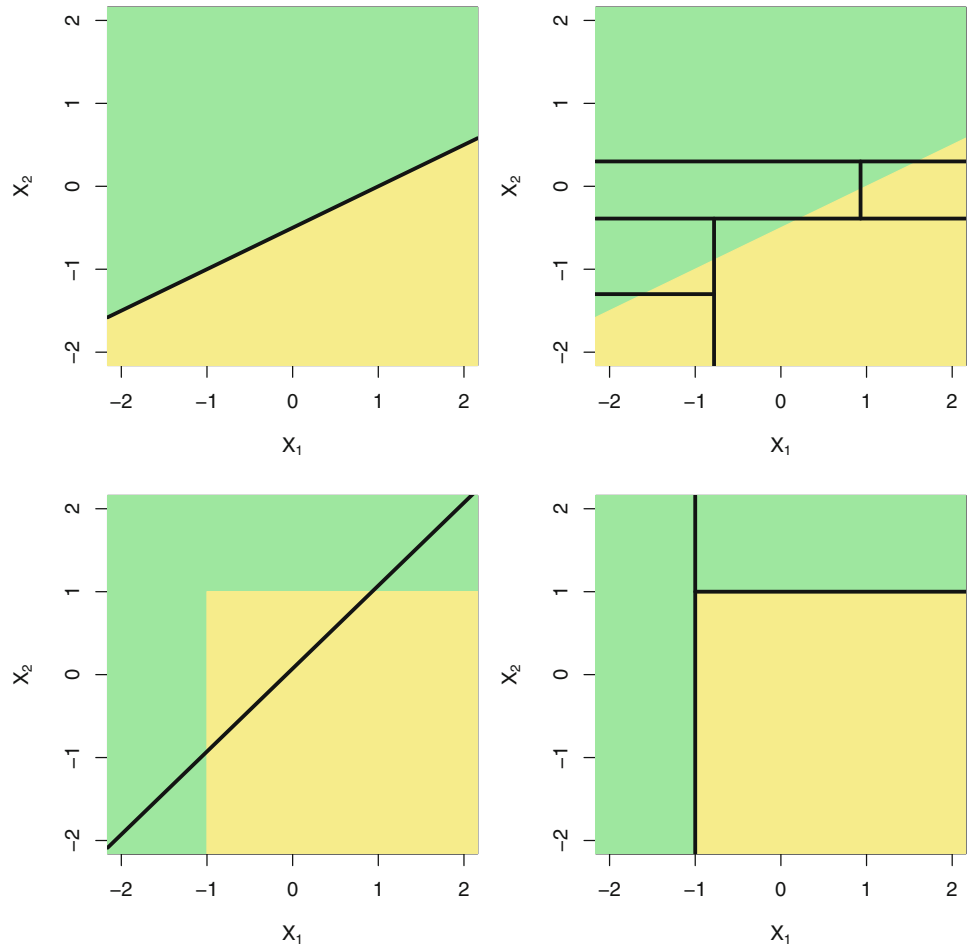


Figur: Node impurity measures for two-class classification, as a function of the proportion p in class 2. (Cross-entropy has been scaled.)

Pros and cons with tree-based methods

- Pros:
  - Easy to explain,
  - mirror more closely human decision-making?,
  - handle easily qualitative predictors without the need of creating dummy variables.
- Cons:
  - Do not, in their most basic form, have the same level of predictive accuracy as some other regression and classification approaches,
  - can be very non-robust (i.e., small changes in the data leads to completely different trees).
- The predictive accuracy can however be drastically improved using ideas as *bagging* and *random forests* discussed in the following.

Trees vs. linear models



Figur: True decision boundary is indicated by the shaded regions. Linear boundary vs. decision tree (boxes).

Today

Overview

Regression trees

Classification trees

Bagging, random forests

Prelude: the bootstrap

- Given an i.i.d. sample $\{x^{(i)}\}_{i=1}^N$ from a probability density p, the *bootstrap algorithm* is based on the approximation

$$\hat{p}(x) = \frac{1}{N} \sum_{i=1}^{N} \delta_{x^{(i)}}(x)$$

  of p.
- Given $\hat{p}$, a new sample $\{x^{*(i)}\}_{i=1}^N$ with approximately the same distribution as $\{x^{(i)}\}_{i=1}^N$ can be formed by sampling from $\hat{p}$, i.e., by drawing repeatedly N times among $\{x^{(i)}\}_{i=1}^N$ with replacement.
- In machine learning applications, this can be used for creating artificial replicates of the training set.

Bagging

- In the case of regression, we have approximated the regression function using

$$\hat{f}(x) = \sum_{j=1}^{J} \bar{y}_{R_j} \mathbb{1}_{R_j}(x),$$

  where the partition $\{R_j\}_{j=1}^{N}$ is formed using a decision tree.
- The decision trees discussed above suffer generally from *high variance*, i.e., building the tree using a different training set may lead to a quite different $\hat{f}$.
- Better would be to use B independent training sets of similar size, each yielding a tree $\hat{f}_b$, and use the model

$$\hat{f}_{\text{avg}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x).$$

Bagging (cont.)

- We use bootstrap sampling for creating artificially B such training sets, each yielding a tree $\hat{f}_b^*$, and use the model

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b^*(x).$$

- The trees $\hat{f}_b^*$ are grown deep without pruning, implying that each tree has high variance but low bias.
- In the classification case, we can record the class predicted by each of the B trees, and take the *most popular vote*.

Bagging (cont.)

- The number of trees B is not a critical parameter with bagging; using a very large value of B will not lead to overfitting.
- The reason is that the probability of making an error *converges* as $B \to \infty$.
- The probability of making an error depends also on the expected correlation between errors of different trees $\hat{f}_b(X)$ and $\hat{f}_{b'}(X)$ as (X, Y) varies.

Some theory

- Indeed, in the classification case, let the *margin function* be

$$m(x, y) = \frac{1}{B} \sum_{b=1}^{B} \mathbb{1}_{\{\hat{f}_b(x)=y\}} - \max_{k \neq y} \frac{1}{B} \sum_{b=1}^{B} \mathbb{1}_{\{\hat{f}_b(x)=k\}}$$

- Moreover, define the *generalization error* by

$$\text{PE} = \mathbb{P}(m(X, Y) < 0).$$

- Breiman (2001) shows that PE *converges* as $B \to \infty$. In addition, the PE can be bounded as

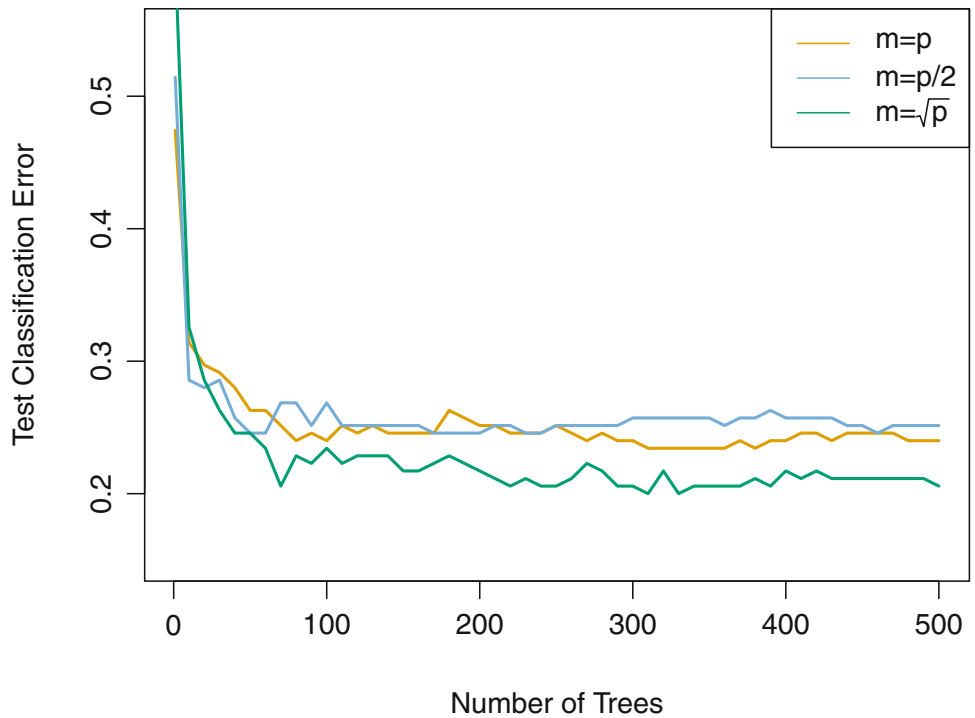
$$\text{PE} \leq (1 - s^2)\rho/s^2,$$

where, roughly, s is the expected limiting margin function—the *strength*—and ρ describes the expected *correlation* between the trees' errors as (X, Y) varies.

Random park $\Rightarrow$ random forest

- In order to *decorrelate* the trees, it is desirable to build them different.
- Say that there is one strong predictor X_m along with a number of other moderately strong predictors $\Rightarrow$ most or all of the trees will use this strong predictor in the top split $\Rightarrow$ all trees will look quite similar $\Rightarrow$ highly correlated predictions.
- Thus, at each split, consider only splitting of a *randomly chosen subset* of $m \leq p$ predictors.
- Therefore, on average $(p - m)/p$ of the splits will not even consider the strong predictor (why?), and so other predictors will have more of a chance.
- Typically, one uses $m \approx \sqrt{p}$; $m = p$ corresponds to standard bagging.

Example: cancer-type prediction



Figur: Here $p = 500$. The test error as a function of the number of trees. Each colored line corresponds to a different value of m. A single classification tree has an error rate of 45.7%.

References

Breiman, L. (2001). “Random forests”. I: *Machine Learning* 45, s. 5–32.