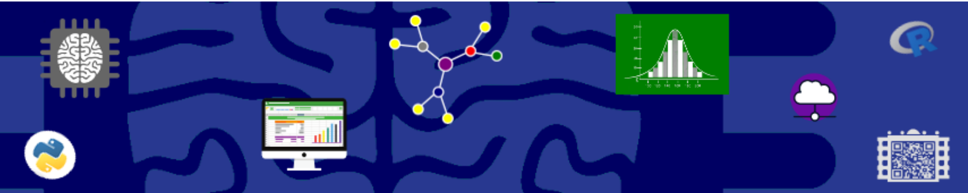
Introduction to Data Science

Winter Semester 2023/24

Oliver Ernst

TU Chemnitz, Fakultät für Mathematik, Professur Numerische Mathematik

Lecture Slides



Contents I

Contents II

Contents III

Contents

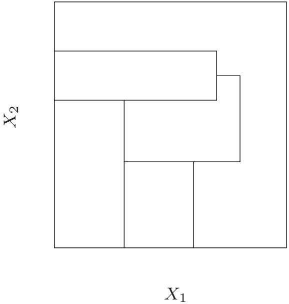
- In previous chapter, considered *piecewise* approximation for univariate models (piecewise constant, splines, etc.).
- Here: piecewise constant *multivariate approximation*.
- Much greater variety of possible domain partitions.
- **Recursive binary partitioning**: efficient representation using **binary trees**.
- Can be used for regression and classification.
- Refinements: **bagging**, **random forests**, **boosting**.
- Developed in 1980s by Leo Breiman and Jerry Friedman, popular algorithm known as **Classification and Regression Tree (CART)**.

Contents

- Consider bivariate model

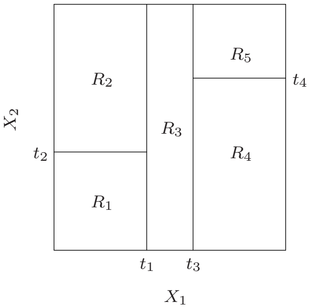
$$Y = f(X_1, X_2), \quad X_i \in [0, 1].$$

- Divide feature space into axis-aligned rectangles.
- Within each rectangle, predict Y as the mean of the observations it contains.



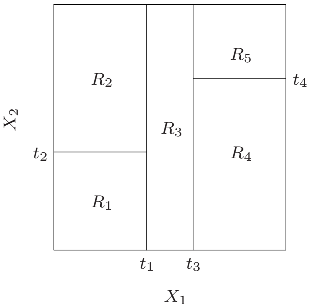
- Simpler structure: construct rectangles by **recursive binary partitioning**.
- Predicting $Y = \hat{y}_{R_m}$ in region R_m yields piecewise constant model

$$Y = \hat{f}(X) = \sum_{m=1}^{5} \hat{y}_{R_m} \mathbb{1}_{\{X \in R_m\}}$$



- Simpler structure: construct rectangles by **recursive binary partitioning**.
- Predicting $Y = \hat{y}_{R_m}$ in region R_m yields piecewise constant model

$$Y = \hat{f}(X) = \sum_{m=1}^{5} \hat{y}_{R_m} \, \mathbb{1}_{\{X \in R_m\}}$$

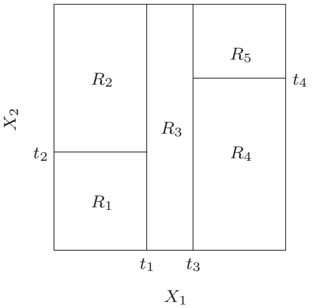


- Partitioning sequence:
  - First split at $X_1 = t_1$.

- Simpler structure: construct rectangles by **recursive binary partitioning**.
- Predicting $Y = \hat{y}_{R_m}$ in region R_m yields piecewise constant model

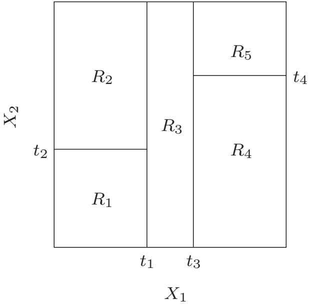
$$Y = \hat{f}(X) = \sum_{m=1}^{5} \hat{y}_{R_m} \mathbb{1}_{\{X \in R_m\}}$$

- Partitioning sequence:
  - First split at $X_1 = t_1$.
  - Next, split region $X_1 \le t_1$ at $X_2 = t_2$ and region $X_1 > t_1$ at $X_1 = t_3$.

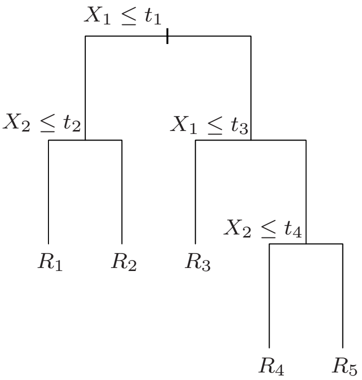


- Simpler structure: construct rectangles by **recursive binary partitioning**.
- Predicting $Y = \hat{y}_{R_m}$ in region R_m yields piecewise constant model

$$Y = \hat{f}(X) = \sum_{m=1}^{5} \hat{y}_{R_m} \, \mathbb{1}_{\{X \in R_m\}}$$



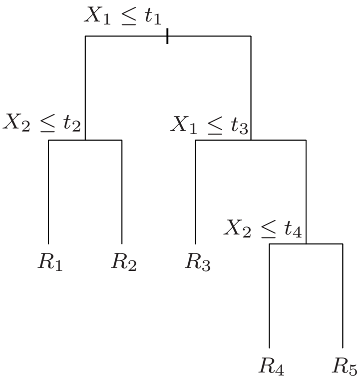
- Partitioning sequence:
  - First split at $X_1 = t_1$.
  - Next, split region $X_1 \le t_1$ at $X_2 = t_2$ and region $X_1 > t_1$ at $X_1 = t_3$.
  - Finally: region $X_1 > t_3$ is split at $X_2 = t_4$.

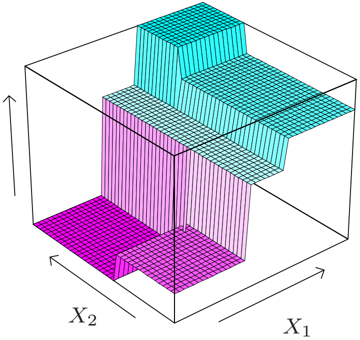


Recursive binary partition more conveniently represented by a binary tree; regions appear as **leaves**, internal nodes are the splits.

Tree-Based Methods

Basic idea





Recursive binary partition more conveniently represented by a binary tree; regions appear as **leaves**, internal nodes are the splits.

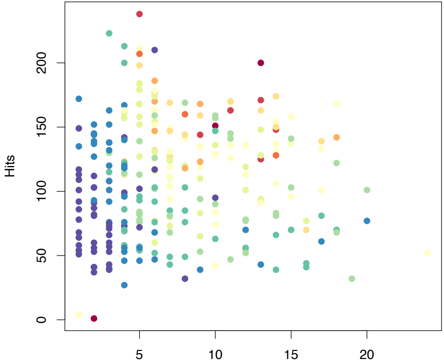
Equivalent representation as piecewise constant function.

Tree-Based Methods

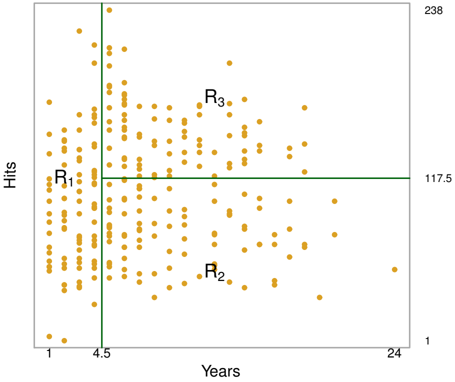
Hitters example

`Hitters` data set: predict baseball players' `Salary` based on

`Years`: # years played in major leagues
`Hits` : # hits made in previous year

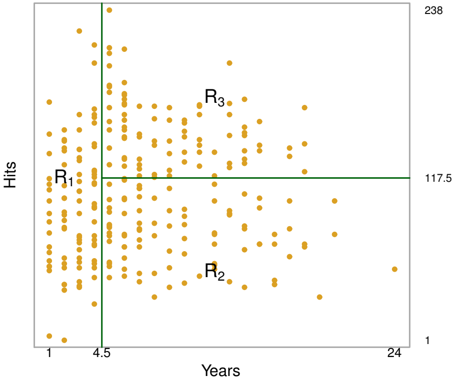


- `Salary` values color-coded from low (blue, green) to high (red).
- First remove observations missing `Salary` values.
- Log-transform `Salary` values [k$] to make distribution more bell-shaped.



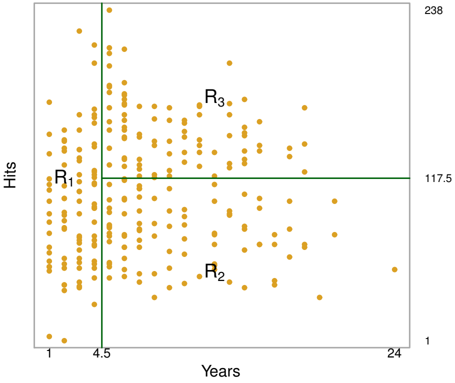
Observations of `Years` and `Hits` with partitioning arising from two splits.

- First split yields
  $R_1 = \{X : \texttt{Years} < 4.5\}$.



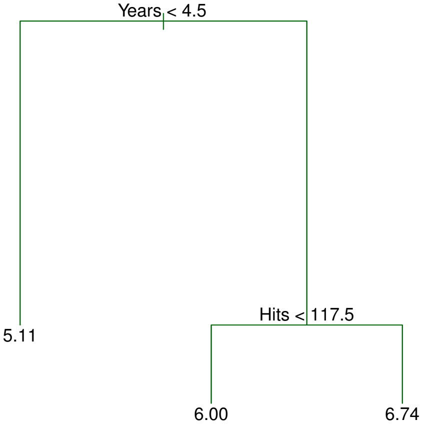
Observations of `Years` and `Hits` with partitioning arising from two splits.

- First split yields
  $R_1 = \{X : \texttt{Years} < 4.5\}$.
- Second split at $\texttt{Hits} = 117.5$ yields
  $R_2 = \{X : \texttt{Years} \geq 4.5, \texttt{Hits} < 117.5\}$
  and
  $R_3 = \{X : \texttt{Years} \geq 4.5, \texttt{Hits} \geq 117.5\}$



Observations of `Years` and `Hits` with partitioning arising from two splits.

- First split yields
  $R_1 = \{X : \texttt{Years} < 4.5\}$.
- Second split at `Hits` $= 117.5$ yields
  $R_2 = \{X : \texttt{Years} \geq 4.5, \texttt{Hits} < 117.5\}$
  and
  $R_3 = \{X : \texttt{Years} \geq 4.5, \texttt{Hits} \geq 117.5\}$
- Predicted `Salary` in these regions:
  $R_1 : \$1000 \times \mathrm{e}^{5.107} = \$165,174,$
  $R_2 : \$1000 \times \mathrm{e}^{5.999} = \$402,838,$
  $R_3 : \$1000 \times \mathrm{e}^{6.740} = \$845,346.$



Regression tree resulting from these splits.

- Left **branch** contains R_1, right branch R_2 and R_3.

Regression tree resulting from these splits.

- Left **branch** contains R_1, right branch R_2 and R_3.
- Tree has two **internal nodes**, three **terminal nodes** (leaves).

Regression tree resulting from these splits.

- Left **branch** contains R_1, right branch R_2 and R_3.
- Tree has two **internal nodes**, three **terminal nodes** (leaves).
- Number in each leaf gives mean value of $\log(\texttt{Salary})$ for corresponding region.

Tree-Based Methods

Hitters example

Regression tree resulting from these splits.

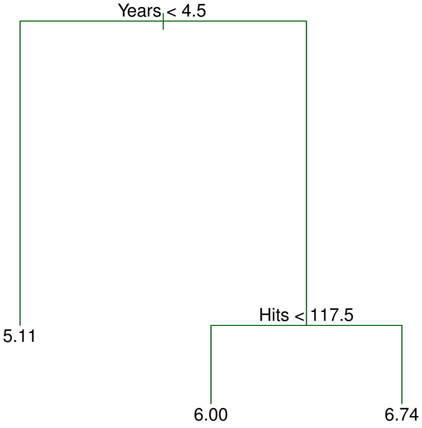
- Left **branch** contains R_1, right branch R_2 and R_3.
- Tree has two **internal nodes**, three **terminal nodes** (leaves).
- Number in each leaf gives mean value of $\log($`Salary`$)$ for corresponding region.
- Length of vertical lines indicate reduction in training error split achieves.
- Interpretation: `Years` is most important factor in determining `Salary` (less experienced players earn less); `Hits` important `Salary`-relevant feature only among experienced players.

Tree-Based Methods

Hitters example

Regression tree resulting from these splits.

- Left **branch** contains R_1, right branch R_2 and R_3.
- Tree has two **internal nodes**, three **terminal nodes** (leaves).
- Number in each leaf gives mean value of $\log(\texttt{Salary})$ for corresponding region.
- Length of vertical lines indicate reduction in training error split achieves.
- Interpretation: `Years` is most important factor in determining `Salary` (less experienced players earn less); `Hits` important `Salary`-relevant feature only among experienced players.
- Advantages of tree: easily interpretable, nice graphical representation.



Regression tree resulting from these splits.

- Left **branch** contains R_1, right branch R_2 and R_3.
- Tree has two **internal nodes**, three **terminal nodes** (leaves).
- Number in each leaf gives mean value of $\log(\texttt{Salary})$ for corresponding region.
- Length of vertical lines indicate reduction in training error split achieves.
- Interpretation: `Years` is most important factor in determining `Salary` (less experienced players earn less); `Hits` important `Salary`-relevant feature only among experienced players.
- Advantages of tree: easily interpretable, nice graphical representation.

- **Goal:** Partition feature space into high-dimensional rectangles $\{R_m\}_{m=1}^M$ in such a way that
$$\mathrm{RSS} = \sum_{m=1}^{M} \sum_{i \in R_m} (y_i - \hat{y}_{R_m})^2,$$
is minimized. ($\hat{y}_{R_m}$: mean of the response observations contained in R_m.)

- **Goal:** Partition feature space into high-dimensional rectangles $\{R_m\}_{m=1}^M$ in such a way that

$$\text{RSS} = \sum_{m=1}^{M} \sum_{i \in R_m} (y_i - \hat{y}_{R_m})^2,$$

  is minimized. ($\hat{y}_{R_m}$: mean of the response observations contained in R_m.)
- Comparing all possible partitions computationally infeasible, hence use **top-down**, **greedy** approach.

- **Goal:** Partition feature space into high-dimensional rectangles $\{R_m\}_{m=1}^M$ in such a way that

$$\mathrm{RSS} = \sum_{m=1}^{M} \sum_{i \in R_m} (y_i - \hat{y}_{R_m})^2,$$

  is minimized. ($\hat{y}_{R_m}$: mean of the response observations contained in R_m.)
- Comparing all possible partitions computationally infeasible, hence use **top-down**, **greedy** approach.
- Top-down refers to starting with the entire feature space and recursively splitting regions (**recursive binary splitting**).

- **Goal:** Partition feature space into high-dimensional rectangles $\{R_m\}_{m=1}^M$ in such a way that

$$\text{RSS} = \sum_{m=1}^{M} \sum_{i \in R_m} (y_i - \hat{y}_{R_m})^2,$$

  is minimized. ($\hat{y}_{R_m}$: mean of the response observations contained in R_m.)
- Comparing all possible partitions computationally infeasible, hence use **top-down**, **greedy** approach.
- Top-down refers to starting with the entire feature space and recursively splitting regions (**recursive binary splitting**).
- **Greedy** approach refers to determining the locally best split without looking ahead and possibly choosing a split leading to a better tree in some future step.

- To construct first split, consider splitting along variable X_j at splitting point $X_j = s$ and define half-spaces

$$R_1(j,s) := \{X : X_j \le s\}, \qquad R_2(j,s) := \{X : X_j > s\}. \tag{8.1}$$

- To construct first split, consider splitting along variable X_j at splitting point $X_j = s$ and define half-spaces

$$R_1(j,s) := \{X : X_j \le s\}, \qquad R_2(j,s) := \{X : X_j > s\}. \tag{8.1}$$

- Seek splitting variable index j and splitting point s which minimize

$$\min_{\hat{y}_{R_1}} \sum_{x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \min_{\hat{y}_{R_2}} \sum_{x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2. \tag{8.2}$$

- To construct first split, consider splitting along variable X_j at splitting point $X_j = s$ and define half-spaces

$$R_1(j,s) := \{X : X_j \le s\}, \qquad R_2(j,s) := \{X : X_j > s\}. \tag{8.1}$$

- Seek splitting variable index j and splitting point s which minimize

$$\min_{\hat{y}_{R_1}} \sum_{x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \min_{\hat{y}_{R_2}} \sum_{x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2. \tag{8.2}$$

- For fixed j, s, the two minimizing values of $\hat{y}_{R_1}$ and $\hat{y}_{R_2}$ are clearly the sample means of the response observations in R_1 and R_2, respectively.

- To construct first split, consider splitting along variable X_j at splitting point $X_j = s$ and define half-spaces

$$R_1(j,s) := \{X : X_j \le s\}, \qquad R_2(j,s) := \{X : X_j > s\}. \tag{8.1}$$

- Seek splitting variable index j and splitting point s which minimize

$$\min_{\hat{y}_{R_1}} \sum_{x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \min_{\hat{y}_{R_2}} \sum_{x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2. \tag{8.2}$$

- For fixed j, s, the two minimizing values of $\hat{y}_{R_1}$ and $\hat{y}_{R_2}$ are clearly the sample means of the response observations in R_1 and R_2, respectively.
- For each j, the optimal splitting point s can be found very quickly; with best split (j,s), partition data into the resulting two subregions and continue splitting recursively.

- Can continue recursive binary splitting until, e.g., cardinality of all leaves fall below given value.

- Can continue recursive binary splitting until, e.g., cardinality of all leaves fall below given value.
- If tree too complex, danger of overfitting: smaller tree (fewer splits) will have lower variance. Hence, alternative stopping criterion could be minimal reduction of RSS for each split.
  However, split with small RSS reduction may enable larger reduction in subsequent splits.

- Can continue recursive binary splitting until, e.g., cardinality of all leaves fall below given value.
- If tree too complex, danger of overfitting: smaller tree (fewer splits) will have lower variance. Hence, alternative stopping criterion could be minimal reduction of RSS for each split.
  However, split with small RSS reduction may enable larger reduction in subsequent splits.
- Better strategy: grow very large tree T_0, then **prune** it back to obtain a **subtree**.

- Can continue recursive binary splitting until, e.g., cardinality of all leaves fall below given value.
- If tree too complex, danger of overfitting: smaller tree (fewer splits) will have lower variance. Hence, alternative stopping criterion could be minimal reduction of RSS for each split.
  However, split with small RSS reduction may enable larger reduction in subsequent splits.
- Better strategy: grow very large tree T_0, then **prune** it back to obtain a **subtree**.
- Can compare different subtrees using **cross-validation**, but comparing all possible subtrees is infeasible.

- **Cost complexity pruning** (a.k.a. *weakest link pruning*): consider sequence of trees indexed by tuning parameter $\alpha \geq 0$.

- **Cost complexity pruning** (a.k.a. *weakest link pruning*): consider sequence of trees indexed by tuning parameter $\alpha \geq 0$.
- To each $\alpha \geq 0$ there corresponds a subtree $T \subset T_0$ which minimizes

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|, \tag{8.3}$$

where $|T|$ denotes the number of leaves of tree T. Tuning parameter α controls trade-off between fit to training data and tree complexity.

- **Cost complexity pruning** (a.k.a. *weakest link pruning*): consider sequence of trees indexed by tuning parameter $\alpha \geq 0$.
- To each $\alpha \geq 0$ there corresponds a subtree $T \subset T_0$ which minimizes

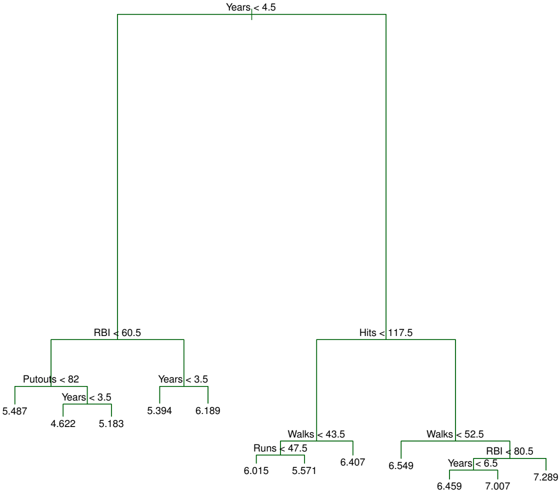
$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|, \tag{8.3}$$

  where $|T|$ denotes the number of leaves of tree T. Tuning parameter α controls trade-off between fit to training data and tree complexity.
- $\alpha = 0$ corresponds to T_0. For $\alpha > 0$ (8.3) minimized by smaller tree T_α (can show this is unique).
- To find T_α use **weakest link pruning**: successively collapse internal node producing smallest per-node increase in $\sum_{m,i}(y_i - \hat{y}_{R_m})^2$, continue until single-node tree reached. Can show: this tree sequence must contain T_α.
- Select α using validation set or cross-validation.

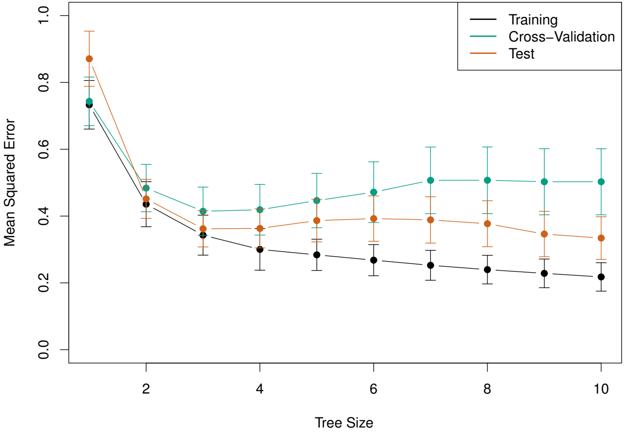
Algorithm 4: Regression tree.

1. Use recursive binary splitting to grow tree T_0 on the training data, stopping when each leaf contains fewer than some minimum number of observations.
2. Apply cost complexity pruning to T_0 to obtain sequence of best subtrees, as a function of α.
3. Use K-fold cross-validation to choose α: divide training observations into K folds. For each $k = 1, \dots, K$:
   1. Repeat steps 1 and 2 on all but k-th fold of training data.
   2. Evaluate test MSE on left out k-th fold, as function of α.

   Average MSE for each value of α, choode α minimizing average MSE.
4. Return subtree from Step 2 corresponding to minimizing α.



- `Hitters` data set using nine features.
- Randomly divide data set into 132 training and 131 test observations.
- Grow tree on training data.
- Vary α to obtain subtrees T_α with different numbers of leaves.
- Perform 6−fold CV to estimate MSE of T_α as function of α.
- Unpruned tree shown on left.



Training, CV and test MSEs for regression tree of `Hitters` data set as a function of α with bands indicating ± 1 standard error. CV MSE somewhat pessimistic, but reasonable estimate of test MSE.

- Tree-based piecewise constant prediction model for qualitative response.

- Tree-based piecewise constant prediction model for qualitative response.
- In place of mean value, predict in R_m the *most commonly occurring* response observation there (**majority vote**).

- Tree-based piecewise constant prediction model for qualitative response.
- In place of mean value, predict in R_m the *most commonly occurring* response observation there (**majority vote**).
- Grow classification tree using recursive binary splitting.
- In place of RSS, can use **classification error rate** E to determine optimal splits. This is simply the fraction of training observations not belonging to the most commonly occurring class, i.e.

$$E = 1 - \max_k \hat{p}_{m,k},$$

$\hat{p}_{m,k}$: proportion of training observations in R_m from k-th class.

- Tree-based piecewise constant prediction model for qualitative response.
- In place of mean value, predict in R_m the *most commonly occurring* response observation there (**majority vote**).
- Grow classification tree using recursive binary splitting.
- In place of RSS, can use **classification error rate** E to determine optimal splits. This is simply the fraction of training observations not belonging to the most commonly occurring class, i.e.

$$E = 1 - \max_k \hat{p}_{m,k},$$

  $\hat{p}_{m,k}$: proportion of training observations in R_m from k-th class.
- Classification error rate E not sensitive enough for tree-growing. Two other popular measures preferable:

- The **Gini index** is defined by

$$G = \sum_{k=1}^{K} \hat{p}_{m,k}(1 - \hat{p}_{m,k}) \tag{8.4}$$

and represents a measure of total variance across the K classes. Small if all $\hat{p}_{m,k}$ close to zero or one; indication of node *purity*, i.e., small value indicates node contains predominantly observations from a single class.

- The **Gini index** is defined by

$$G = \sum_{k=1}^{K} \hat{p}_{m,k}(1 - \hat{p}_{m,k}) \tag{8.4}$$

and represents a measure of total variance across the K classes. Small if all $\hat{p}_{m,k}$ close to zero or one; indication of node *purity*, i.e., small value indicates node contains predominantly observations from a single class.

- **Entropy** (or **deviance**) is defined by

$$D = -\sum_{k=1}^{K} \hat{p}_{m,k} \log \hat{p}_{m,k}. \tag{8.5}$$

Note $\hat{p}_{m,k} \log \hat{p}_{m,k} \leq 0$ since $\hat{p}_{m,k} \in [0, 1]$.
As for G, D small if $\hat{p}_{m,k}$ close to zero or one for all k.

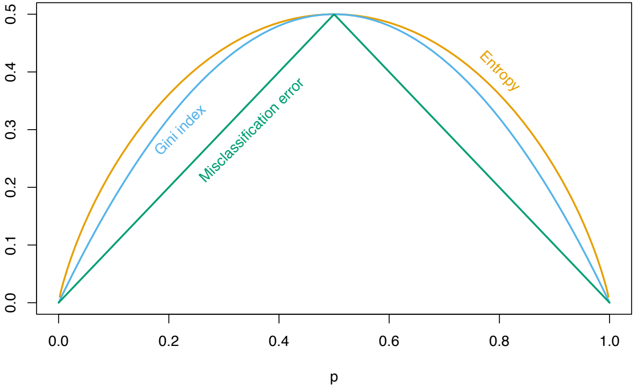
- The **Gini index** is defined by

$$G = \sum_{k=1}^{K} \hat{p}_{m,k}(1 - \hat{p}_{m,k}) \tag{8.4}$$

  and represents a measure of total variance across the K classes. Small if all $\hat{p}_{m,k}$ close to zero or one; indication of node *purity*, i.e., small value indicates node contains predominantly observations from a single class.
- **Entropy** (or **deviance**) is defined by

$$D = -\sum_{k=1}^{K} \hat{p}_{m,k} \log \hat{p}_{m,k}. \tag{8.5}$$

  Note $\hat{p}_{m,k} \log \hat{p}_{m,k} \leq 0$ since $\hat{p}_{m,k} \in [0, 1]$.
  As for G, D small if $\hat{p}_{m,k}$ close to zero or one for all k.
- Any of E, G or D can be used to build the tree, but pruning should be done using E to maximize prediction accuracy of final pruned tree.

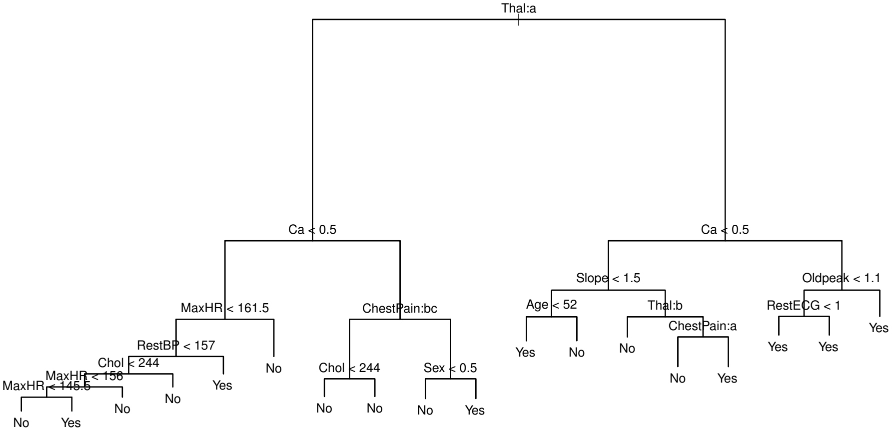


Node purity measures for two-class classification as a function of proportion p in class 2. Entropy has been scaled to pass through $(0.5, 0.5)$. For two classes, if p denotes the proportion in class 2, the three measures are $1 - \max(p, 1-p)$, $2p(1-p)$ and $-p \log p - (1-p)\log(1-p)$.

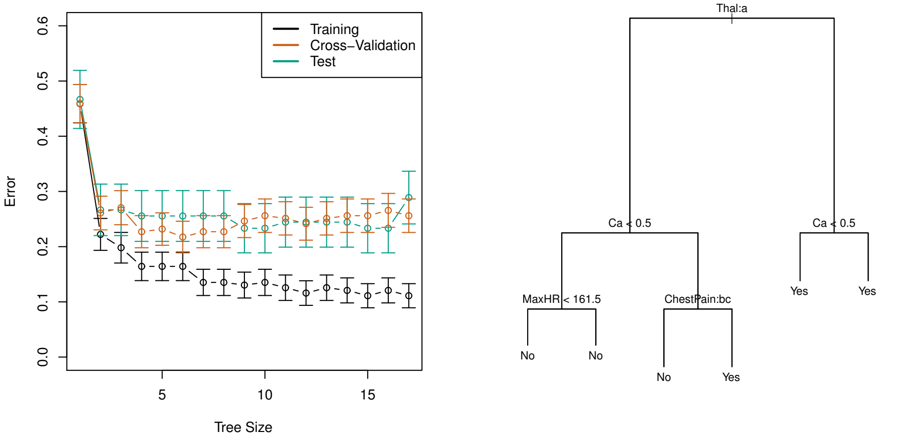
Tree-Based Methods

Heart example

- `Heart` data set: binary response `HD` for 303 patients who presented with chest pain.
- Response `Yes` indicates presence of heart disease (based on angiographic test), `No` indicates absence of heart disease.
- 13 predictors including `Age`, `Sex`, `Chol` (cholesterol measurement), and further heart and lung function measurements.
- Cross-validation results in tree with six leaves.



Heart data set: unpruned tree.



Heart data set. Left: training, CV and test MSE for different sizes of pruned tree. Right: pruned tree corresponding to minimal CV MSE.

- `Heart` data set contains a number of *qualitative* predictor variables such as `Sex`, `Thal` (Thallium stress test) and `ChestPain`.
- Splitting along one of these variables: assign some of the qualitative values to one branch, remaining values to other branch.
- In previous image: some internal nodes split quantitative variables.
- Top internal node splits `Thal`. Text `Thal:a` indicates left branch consists of observations with first value of `Thal` (normal), right consists of remaining values (fixed or reversible defects).
- Text `ChestPain:bc` on third split on left indicates left branch contains observations with second and third values of `ChestPain` variable (whose possible values are *typical angina*, *atypical angina*, *non-anginal pain* and *asymptomatic*).

Tree-Based Methods

Heart example, leaves with identical values

- Some leaves in `Heart` classification tree have the same predicion values.
- Example: split `RestECG` $<$ 1 near bottom right of unpruned tree, both subregions predict response value `Yes`. Why perform split in the first place?

Tree-Based Methods

Heart example, leaves with identical values

- Some leaves in `Heart` classification tree have the same predicion values.
- Example: split `RestECG` $<$ 1 near bottom right of unpruned tree, both subregions predict response value `Yes`. Why perform split in the first place?
- Split made to increase *node purity*.
- All 9 observations in right branch have leaf response value `Yes`. In left branch, 7/11 have response value `Yes`.
- Importance of node purity: given test observation belonging to region on right branch, then response certaionly `Yes`. For test observation on left branch, pobably `Yes`, but with much less certainty.
- Even though `RestECG` < 1 does not reduce classification error, it improves the Gini index and entropy, which are more sensitive to node purity.

- Prediction model of linear regression vs. regression tree

$$f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p, \qquad f(X) = \sum_{m=1}^{M} \hat{\boldsymbol{y}}_{R_m} \mathbb{1}_{\{X \in R_m\}}$$

with regression coefficients $\{\beta_j\}_{j=0}^{p}$ and partition of feature space into rectangular regions R_m.

- Prediction model of linear regression vs. regression tree

$$f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p, \qquad f(X) = \sum_{m=1}^{M} \hat{\boldsymbol{y}}_{R_m} \mathbb{1}_{\{X \in R_m\}}$$

  with regression coefficients $\{\beta_j\}_{j=0}^{p}$ and partition of feature space into rectangular regions R_m.
- If feature-response relation close to linear, linear regression model likely superior.
  Otherwise, tree-based models may outperform linear regression.

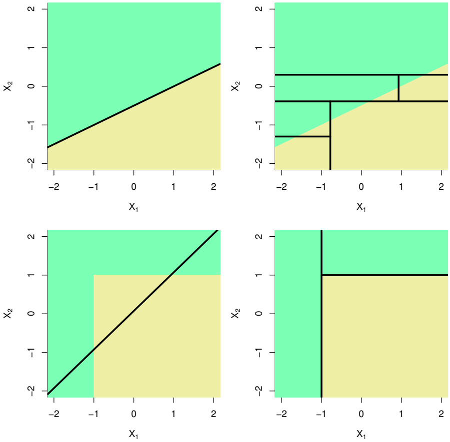
- Prediction model of linear regression vs. regression tree

$$f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p, \qquad f(X) = \sum_{m=1}^{M} \hat{y}_{R_m} \mathbb{1}_{\{X \in R_m\}}$$

  with regression coefficients $\{\beta_j\}_{j=0}^{p}$ and partition of feature space into rectangular regions R_m.
- If feature-response relation close to linear, linear regression model likely superior.
  Otherwise, tree-based models may outperform linear regression.
- Relative performances can be assessed by estimating test MSE via CV or validation set approach.
- Other considerations may also be relevant in comparison, such as interpretability or visualization.

Tree-Based Methods

Trees vs. linear models: example



Top row: 2D classification example with linear decision boundary (shaded regions), linear regression model superior. Bottom row: nonlinear)(piecewise constant) decision boundary captured perfectly by tree-based method,

+ Easy to explain (more so than linear regression).
+ Some argue decision trees more closely mimick human decision-making than linear regression/classification techniques (also widely used outside of statistical learning).
+ Trees, particularly small ones, easily displayed graphically, easily interpreted by non-experts.
+ Can handle qualitative predictors without introducing dummy variables.
- Prediction accuracy generally not as good as classical regression and classification techniques.
- Robustness (stability w.r.t. small data changes) often lacking.

Tree-Based Methods

Advantages and shortcomings of trees

- \+ Easy to explain (more so than linear regression).
- \+ Some argue decision trees more closely mimick human decision-making than linear regression/classification techniques (also widely used outside of statistical learning).
- \+ Trees, particularly small ones, easily displayed graphically, easily interpreted by non-experts.
- \+ Can handle qualitative predictors without introducing dummy variables.
- \- Prediction accuracy generally not as good as classical regression and classification techniques.
- \- Robustness (stability w.r.t. small data changes) often lacking.

Some of these disadvantages addressed by bagging, random forests, boosting.

Contents

- Recall the **bootstrap** approach introduced in Chapter 5 for randomly generating subsamples of a set of observations for estimating statistical quantities without collecting additional data.

- Recall the **bootstrap** approach introduced in Chapter 5 for randomly generating subsamples of a set of observations for estimating statistical quantities without collecting additional data.
- Here we revisit the bootstrap to show how it can be used as a **variance-reduction technique** for any statistical learning method.
- This is particularly relevant for decision trees, which tend to possess high variance.
- **Bagging**: **b**ootstrap **agg**regation.
- Variance can be reduced by averaging observations: for $\{X_k\}_{k=1}^N$ i.i.d. RV with variance σ^2, variance of $\overline{X} = (X_1 + \cdots + X_N)/N$ is σ^2/N.
- **Idea:** Collect N training sets, construct prediction model $\hat{f}_k$ for each, and average these to aggregate model

$$\hat{f}_{\text{avg}}(x) = \frac{1}{N} \sum_{k=1}^{N} \hat{f}_k(x).$$

- Collecting N data sets generally infeasible.

- Collecting N data sets generally infeasible.
- Bootstrap: randomly select (with replacement) N_b sets of samples from (single) original data set.

- Collecting N data sets generally infeasible.
- Bootstrap: randomly select (with replacement) N_b sets of samples from (single) original data set.
  For each resampled data set, construct prediction model $\hat{f}_k^*$, $k = 1, \dots, N_b$.

- Collecting N data sets generally infeasible.
- Bootstrap: randomly select (with replacement) N_b sets of samples from (single) original data set.
  For each resampled data set, construct prediction model $\hat{f}_k^*$, $k = 1, \dots, N_b$.
  Form bootstrap aggregate model

$$\hat{f}_{\text{bag}}(x) = \frac{1}{N_b} \sum_{k=1}^{N_b} \hat{f}_k^*(x).$$

  This is called **bagging**.

- Collecting N data sets generally infeasible.
- Bootstrap: randomly select (with replacement) N_b sets of samples from (single) original data set.
  For each resampled data set, construct prediction model $\hat{f}_k^*$, $k = 1, \dots, N_b$.
  Form bootstrap aggregate model

$$\hat{f}_{\text{bag}}(x) = \frac{1}{N_b} \sum_{k=1}^{N_b} \hat{f}_k^*(x).$$

  This is called **bagging**.
- For (regression) decision trees: grow (unpruned) tree for each resampled data set and average them.

- Collecting N data sets generally infeasible.
- Bootstrap: randomly select (with replacement) N_b sets of samples from (single) original data set.
  For each resampled data set, construct prediction model $\hat{f}_k^*$, $k = 1, \dots, N_b$.
  Form bootstrap aggregate model

$$\hat{f}_{\text{bag}}(x) = \frac{1}{N_b} \sum_{k=1}^{N_b} \hat{f}_k^*(x).$$

  This is called **bagging**.
- For (regression) decision trees: grow (unpruned) tree for each resampled data set and average them.
- For classification trees: replace average with **majority vote**, i.e., for each new predictor observation, have aggregate model predict that class occurring most commonly across all decision trees $\hat{f}_k^*$.

Out-of-bag error estimation

- Error of bagged model can be estimated without CV or validation sets by exploiting that we are using bootstrapped subsets of fixed observation set.

Tree-Based Methods

Out-of-bag error estimation

- Error of bagged model can be estimated without CV or validation sets by exploiting that we are using bootstrapped subsets of fixed observation set.
- Can show: on average, each bagged tree uses of 2/3 of the observations.

- Error of bagged model can be estimated without CV or validation sets by exploiting that we are using bootstrapped subsets of fixed observation set.
- Can show: on average, each bagged tree uses of 2/3 of the observations.
- Remaining third: "**out-of-bag**" (OOB) observations.
- For i-th observation: predict response using all trees for which it was OOB. Take average (regression) or majority vote (classification) to obtain aggregated prediction for all n observations, compare with response observation, i.e., MSE (regression) or classification error (classification), to obtain error estimate.

- Error of bagged model can be estimated without CV or validation sets by exploiting that we are using bootstrapped subsets of fixed observation set.
- Can show: on average, each bagged tree uses of 2/3 of the observations.
- Remaining third: "**out-of-bag**" (OOB) observations.
- For i-th observation: predict response using all trees for which it was OOB. Take average (regression) or majority vote (classification) to obtain aggregated prediction for all n observations, compare with response observation, i.e., MSE (regression) or classification error (classification), to obtain error estimate.
- Can show: for N_b sufficiently large, OOB error estimate virtually equivalent to LOOCV error estimate.
  This is a great benefit when performing bagging on large data sets, where CV would be computationally burdensome.

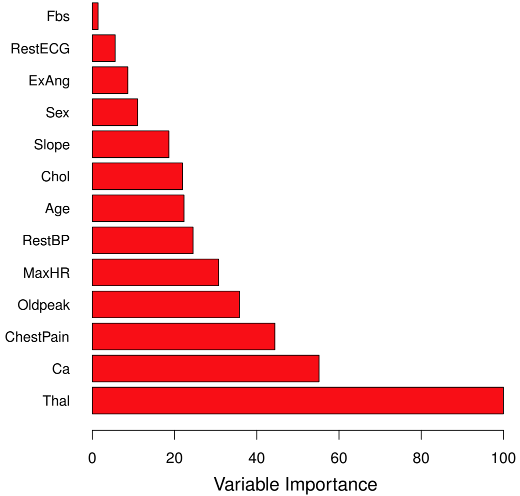
- Drawback of bagging: no single decision tree for interpretation.
  Increased prediction accuracy at the expense of interpretability.

- Drawback of bagging: no single decision tree for interpretation.
  Increased prediction accuracy at the expense of interpretability.
- Crucial interpretation element: which predictor variables most important?
- For regression trees, overall summary of importance of each predictor can be obtained by recording total amount RSS decreases when split performed along this variable, then averaging over all trees.

- Drawback of bagging: no single decision tree for interpretation. Increased prediction accuracy at the expense of interpretability.
- Crucial interpretation element: which predictor variables most important?
- For regression trees, overall summary of importance of each predictor can be obtained by recording total amount RSS decreases when split performed along this variable, then averaging over all trees.
- For classification trees: record total Gini index reduction due to splits along each predictor per tree, then average over all trees.
- Large value indicates important predictor variable.

Tree-Based Methods

Measuring variable importance: Heart data set



`Heart` data set. Variable importance (relative to maximum) in terms of mean decrease in Gini index.

- Improve over bagging by "**decorrelating**" trees.
- Build decision trees on bootstrapped samples as in bagging.
- When choosing next predictor variable to split, restrict selection to $m < p$ randomly chosen variables instead of full set of p predictors.
  New set of m splitting candidates chosen at each splitting step.
- Common choice: $m \approx \sqrt{p}$. Smaller m called for in case of many correlated predictors.
  $m = p$ recovers bagging.

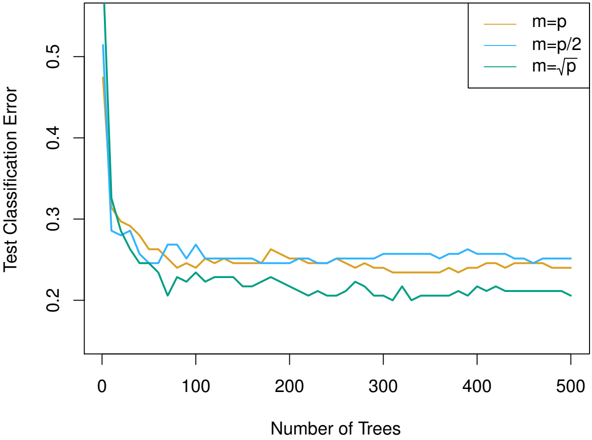
- Improve over bagging by "**decorrelating**" trees.
- Build decision trees on bootstrapped samples as in bagging.
- When choosing next predictor variable to split, restrict selection to $m < p$ randomly chosen variables instead of full set of p predictors.
  New set of m splitting candidates chosen at each splitting step.
- Common choice: $m \approx \sqrt{p}$. Smaller m called for in case of many correlated predictors.
  $m = p$ recovers bagging.
- **Rationale:** strongly dominant variables will be used in splitting for majority of bagged trees, leading to similarity among these trees, and thereby strong correlations.
  This limits the variance reduction from averaging.

- Improve over bagging by "**decorrelating**" trees.
- Build decision trees on bootstrapped samples as in bagging.
- When choosing next predictor variable to split, restrict selection to $m < p$ randomly chosen variables instead of full set of p predictors.
  New set of m splitting candidates chosen at each splitting step.
- Common choice: $m \approx \sqrt{p}$. Smaller m called for in case of many correlated predictors.
  $m = p$ recovers bagging.
- **Rationale:** strongly dominant variables will be used in splitting for majority of bagged trees, leading to similarity among these trees, and thereby strong correlations.
  This limits the variance reduction from averaging.
- On average, $(p-m)/p$ splits will not even consider a given strong predictor.
- This mechanism results in a decorrelation of the trees, making their average less variable, hence more reliable.

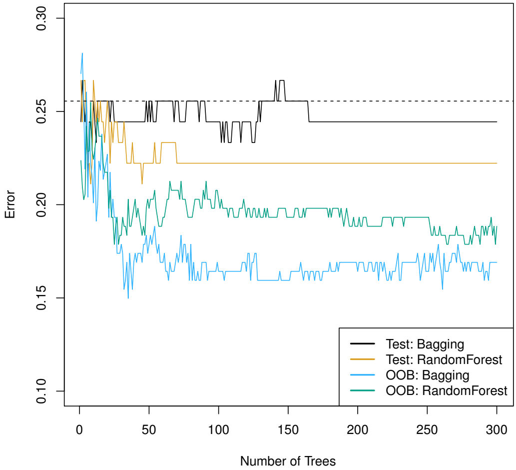
Tree-Based Methods

Random forests: gene expression example

- High-dimensional biological data set: expression measurements for 4,718 genes on tissue samples from 349 patients.
- Human genome contains $\approx 20,000$ genes.
- Individual genes have varying levels of **expression** (activity) in different body cells, tissue or biological conditions.
- Here: each patient sample assigned to one of 15 classes (normal or one of 14 cancer types).
- Goal: predict cancer type using random forests based on 500 genes with largest variance in training set.
- Random division into training and test set.
- Random forests applied for 3 different values of m.



Random forests for 15-class gene expression with $p = 500$: test error against # trees. Single tree has classification error rate of 45.7%. Null rate (always assign to dominant class) is 75.4%. As for bagging, no danger of overfitting as # trees increases.



Test error against number N_b of bootstrapped data sets. Random forests used $m = \sqrt{p}$. Dashed line: error of single classification tree. Solid green/blue: OOB errors considerably lower.

- **Boosting:** general approach for improving predictions of statistical learning methods, here in context of decision trees.
- Basic approach as in bagging, but trees grown **sequentially** using information from previously generated trees.
- No bootstrap sampling; instead, each tree fit to a modified version of original data set.

- **Boosting:** general approach for improving predictions of statistical learning methods, here in context of decision trees.
- Basic approach as in bagging, but trees grown **sequentially** using information from previously generated trees.
- No bootstrap sampling; instead, each tree fit to a modified version of original data set.
- Procedure: begin with tree fit to original data.
- Fit next tree to **residuals** of first model in place of observation responses. Then add this tree to the first, as a **model correction**.
- Each tree can be small (few leaves) determined by parameter d in the algorithm.
  By fitting small trees to residuals, $\hat{f}$ is slowly improved in areas where it previously didn't perform well.
- Boosting for classification trees slightly more complicated.

1. # trees N_b. Overfitting is possible with boosting, although it sets in slowly. Selection using CV.

1. \# trees N_b. Overfitting is possible with boosting, although it sets in slowly. Selection using CV.
2. Shrinkage parameter $\lambda > 0$ (small) determining learning rate.
   Typical values 10^{-2} or 10^{-3}.
   Very small λ can require very large N_b for good prediction.
3. \# splits d per tree, controls complexity of boosted ensemble.
   Can also use $d = 1$ ("stump") with single split, leads to an **additive model**.
   Since d splits can involve at most d variables, it controls the **interaction order** of the boosted model.

Algorithm 5: Boosting for regression trees.

1. Set $\hat{f}(x) \equiv 0$ and $r_i = y_i$, $i = 1, \dots, n$ (entire training set).
2. **for** $k = 1$ **to** N_b **do**

   Fit a tree $\hat{f}_k$ with d splits $(d+1)$ leaves to training data (X, r)

   Update $\hat{f}_k$ by adding damped version of new tree:

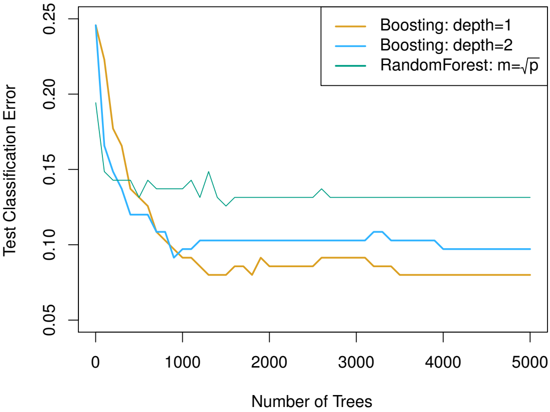
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}_k(x)$$

   Update residuals

   $$r_i \leftarrow r_i - \lambda \hat{f}_k(x_i), \quad i = 1, \dots, n.$$

3. Output boosted model

$$\hat{f}(x) = \sum_{k=1}^{N_b} \lambda \hat{f}_k(x).$$



Boosting and random forests for gene expression example: test error against # trees using $\lambda = 0.01$ for boosted models. Depth-1 trees slightly outperform depth-2 trees, both outperform random forest, but difference is within standard error. Single tree has error rate 24%.

Contents

Boosting

Setting

- Recall main idea: combine outputs of many **weak** models (slightly better than random guessing) to produce a poweful "committee".
- Pioneering algorithm: **AdaBoost.M1** [Fraund & Schapire, 1997]
- Consider classification problem:

$Y \in \{-1, 1\}$	response
X	predictor variable(s)
$G : X \mapsto G(X) \in \{-1, 1\}$	classifier

- Error on training samplke

$$\overline{\text{err}} := \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{y_i \neq G(x_i)\}}$$

- Expected error rate

$$\mathbf{E}_{XY} \left[\mathbb{1}_{\{Y \neq G(X)\}} \right]$$

Boosting

Setting

- Sequentially apply weak classifier to repeatedly modified versions of the data, yields sequence $\{G_m\}_{m=1,2,\dots}$ of classifiers,
- Combine these to

$$G(x) := \operatorname{sign}\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right).$$

- Weights $\alpha_1, \dots, \alpha_M$ computed by algorithm.
- Goal: give higher influence to more accureate classifiers.
- Modifications: apply **weights** $w_1, \dots, w_n$ to training observations $\{(x_i, y_i)\}_{i=1}^{n}$. Initialize to $w_i = 1/n \; \forall i$.
- At step m: increase weights of observations misclassified by G_{m-1}, decrease those of correctly classified observations
- Observations difficult to classify receive more weight. Force later classifiers to focus on those observations missed by previous ones.

Algorithm 6: AdaBoost.M1

1. Initialize obervation weights $w_i \leftarrow \frac{1}{n}$, $i = 1, \ldots, n$.
2. **for** $m = 1$ **to** M **do**

   Fit classifier G_m to training data using weights w_i.

   Compute
   $$\mathrm{err}_m \leftarrow \frac{\sum_{i=1}^n w_i \, \mathbb{1}_{\{y_i \neq G(x_i)\}}}{\sum_{i=1}^n w_i}$$

   Compute
   $$\alpha_m \leftarrow \log((1 - \mathrm{err}_m)/\mathrm{err}_m).$$

   Set
   $$w_i \leftarrow w_i \cdot \exp[\alpha_m \mathbb{1}\{y_i \neq G(x_i)\}], \qquad i = 1, \ldots, n.$$
3. Output $G(x) = \mathrm{sign}\left[\sum_{m=1}^M \alpha_m G_m(x)\right]$.

Algorithm 7: Forward Stagewise Additive Modeling

1. Initialize $f_0 \equiv 0$.
2. **for** $m = 1$ **to** M **do**

   Compute

$$(\beta_m, \gamma_m) \leftarrow \underset{\beta,\gamma}{\arg\min} \sum_{i=1}^{n} L\left(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)\right)$$

   Set

$$f_m(x) \leftarrow f_{m-1}(x) + \beta_m b(x; \gamma_m).$$