Statistical Methods for Bioinformatics

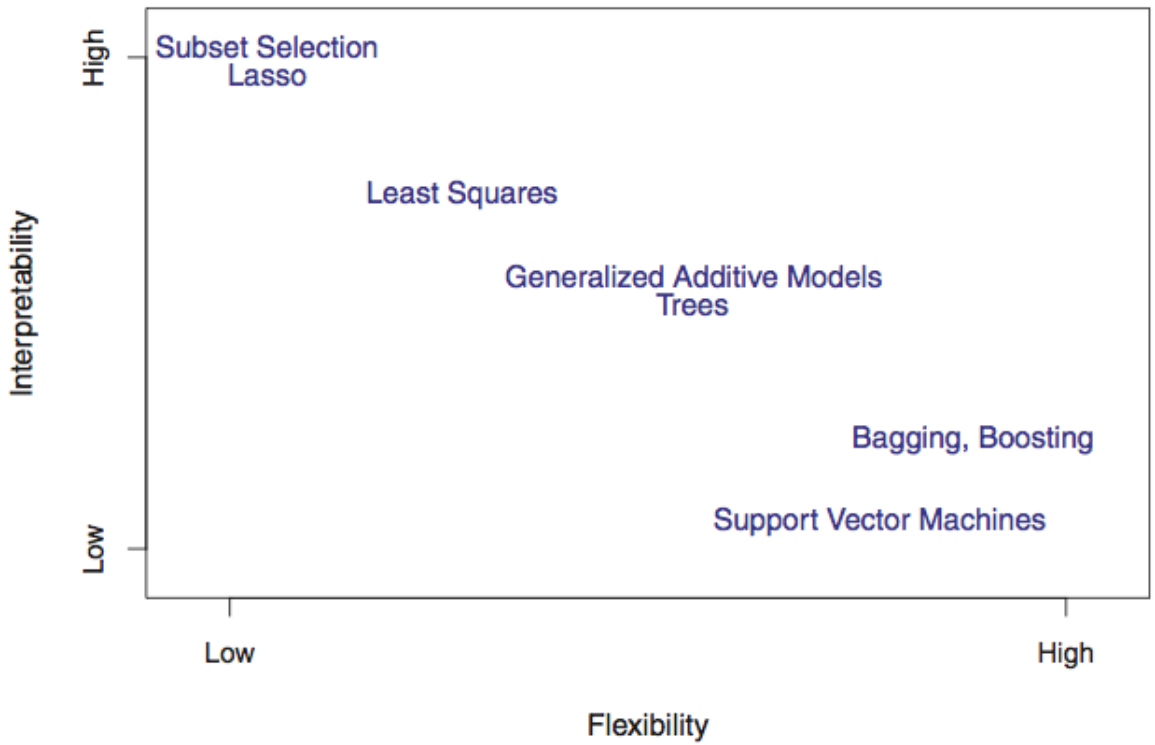
II-5: Trees, Bagging and Boosting

Today

1. Regression tree
2. Classification tree
3. Ensemble methods:
    1. Bagging
    2. Random Forests
    3. Boosting

Even more flexible models

- A default GAM does not inherently incorporate interactions between variables, though they can be included.
- Another form of flexibility is to focus on interactions between variables.
- One can consider decision trees, Random Forests and SVM (etc)

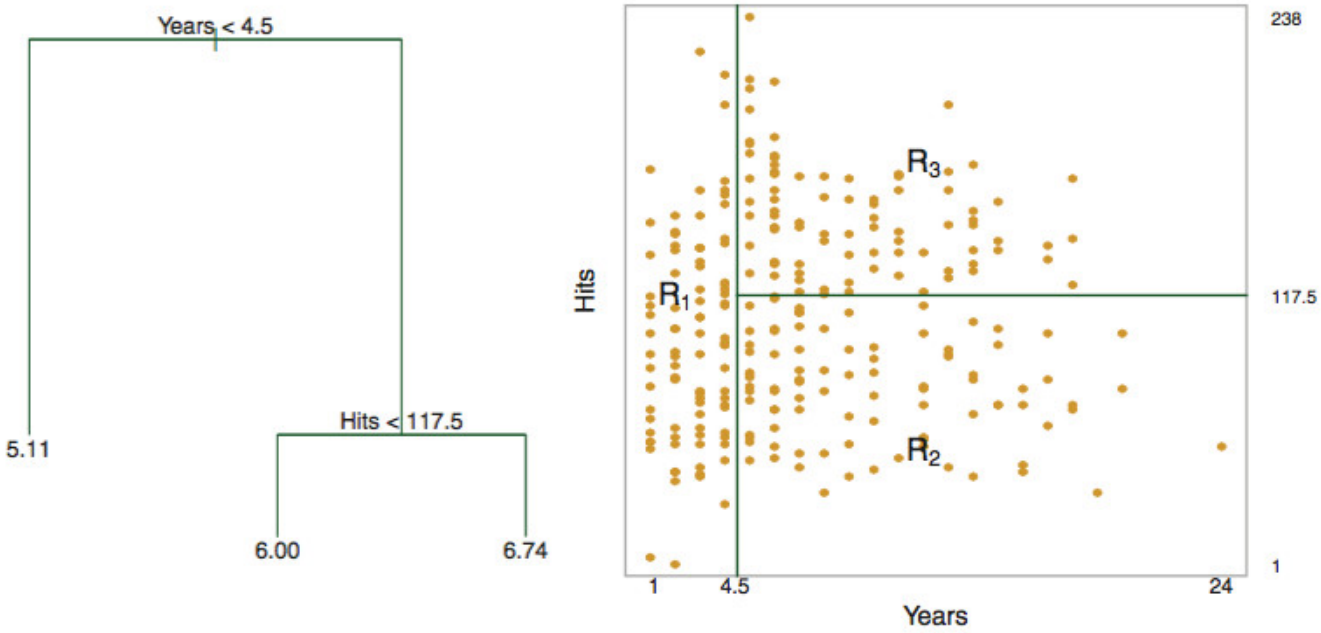


Trees are very broadly used

- Systematically structuring knowledge (Gene Ontology)
- Phylogenetic tree
- Many data structures
  - e.g. directory structure
- Decision trees as a procedure: e.g. in clinical practice

Tree-Based Methods

- Basic tree approaches are simple, and useful for interpretation
- Progressively stratifying or segmenting predictor space into regions.
- Readily exploit interactions between variables.



Building a tree

1. We divide the predictor space — that is, the set of possible values for X1,X2,...,Xp — into J distinct and non-overlapping regions, R1, R2, . . . , RJ .
2. For every observation that falls into the region Rj, we make the same prediction, which is simply the mean of the response values for the training observations in Rj.
3. The regions are high-dimensional rectangles/boxes
4. The goal is minimize RSS: $\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$

Building a tree

Procedure

1. Split the predictor space so that the biggest drop in RSS is achieved.
2. Then split one of two new spaces with the same criterion
3. Continue till some criterion is reached.

- This process can overfit the data if divisions continue till data scarcity
- Smaller trees tend to have less variance for a bit more bias.
- A strong limit on growth of the tree is often sub-optimal however
  - Stopping early may prevent finding v. good fits deeper in the tree.

Pruning a Tree

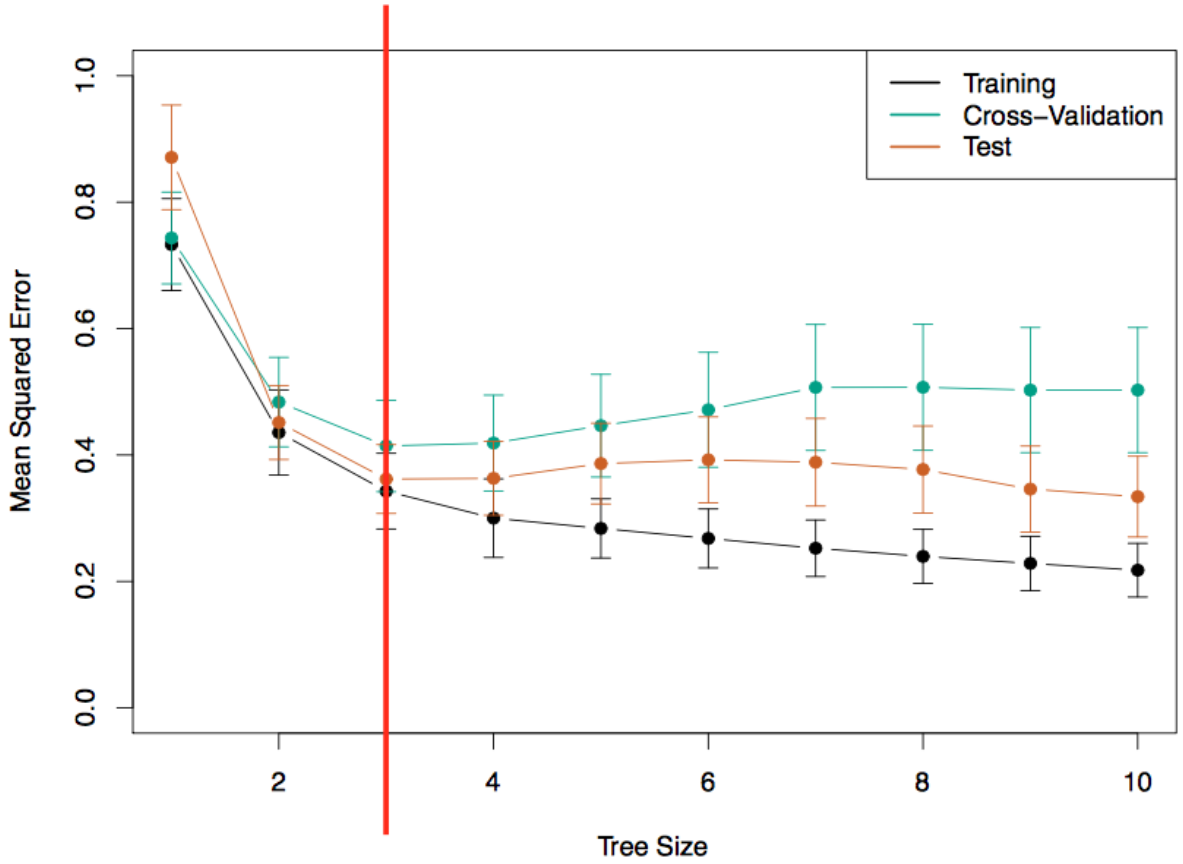
- **Strategy of choice** is to grow a tree and then prune it back.
- The branches that give the smallest drop in RSS for their number of splits are removed first. This is formalized as minimizing:

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

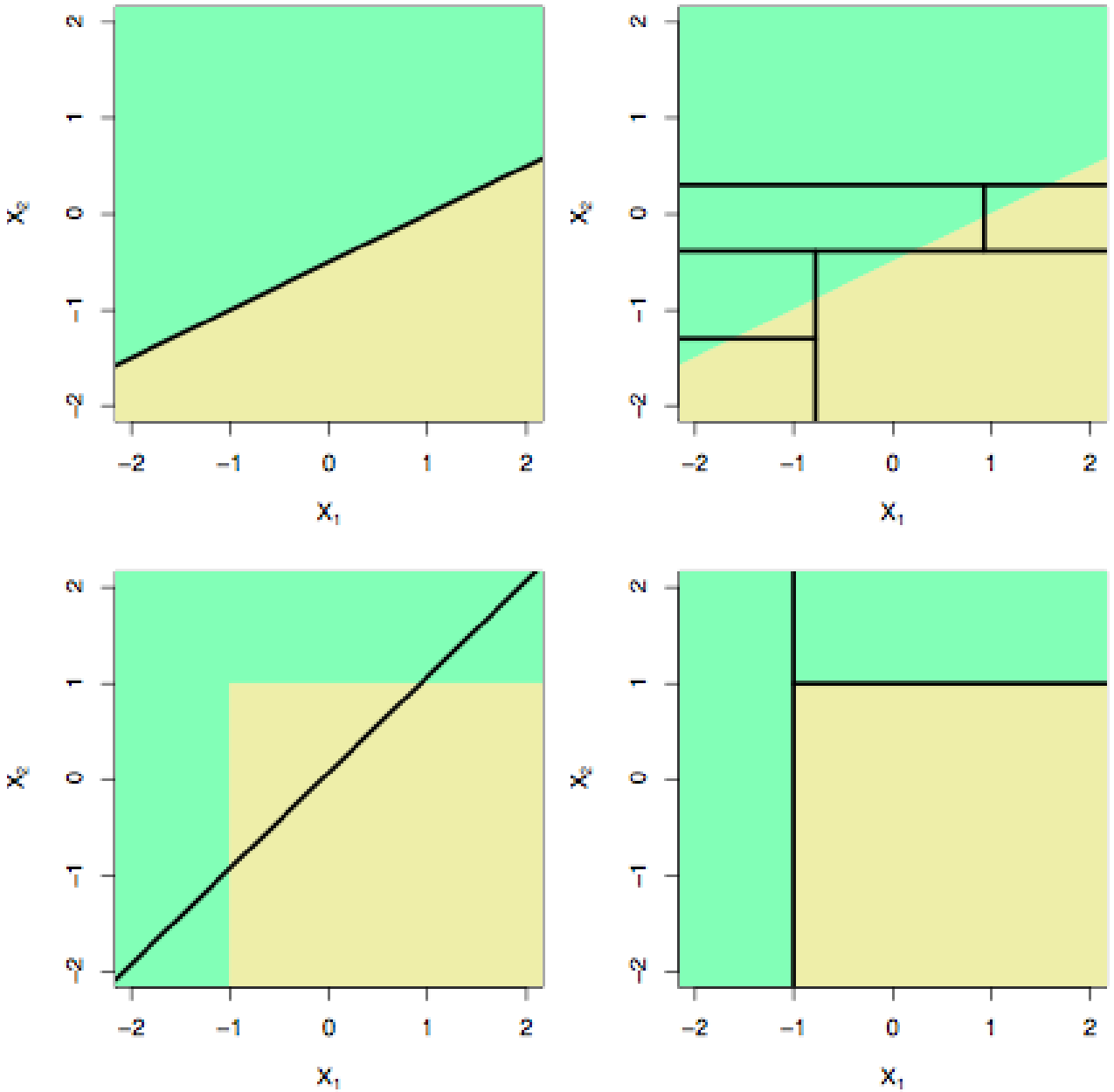
- $|T|$ represents the terminal node count
- α is a non-negative tuning parameter chosen with cross-validation

Example: Baseball Players' Salaries

The minimum cross validation error occurs at a tree size of 3



Trees vs Linear Model: classification example



Classification tree

- Same principle as regression tree
- Intuitive optimization function is to take for every box the most common class and take all examples not of this class as errors: $E = 1 - max_k(\hat{p}_{mk})$ with $\hat{p}_{mk}$ the proportion of observations in the m-th box of the k-th class.
- Above classification error is not very sensitive (many models have very similar scores) so we need something else
- Different cost function that measures purity of the nodes
  - Gini index $G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$
  - cross-entropy $D = -\sum_{k=1}^{K} p_{mk} log(p_{mk})$

Tree recap

- Transparent and easy to understand method
  - Plotting and interpretation is easy
- Naturally incorporates interactions between variables
- Naturally incorporates qualitative predictors
- But they tend to not perform very well for most datasets

Ensemble methods

From weak to strong

Can we combine multiple "weak" learning models to make one "strong" learning model?

Definition

Ensemble methods combine multiple instances of learning algorithms for predictions.

Goal

Improve predictive performance over any of the constituent instances

- Especially useful with high model variance and overlearning of individual models
- Averaging stabilizes prediction performance for variable models

Ensemble methods

In this class: 2 methods to represent the class

1. Bagging, with Random Forests as a variant
2. Boosting

- General-purpose procedures for reducing the variance of a statistical learning method
- Both particularly useful in the context of decision trees.

Ensemble methods

In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms.

- e.g. Bagging: a general-purpose procedure for reducing the variance of a statistical learning method; we introduce it here because it is particularly useful and frequently used in the context of decision trees.
- Important player was Leo Breiman (who proposed a.o. Random forests), a very creative man to advanced age.

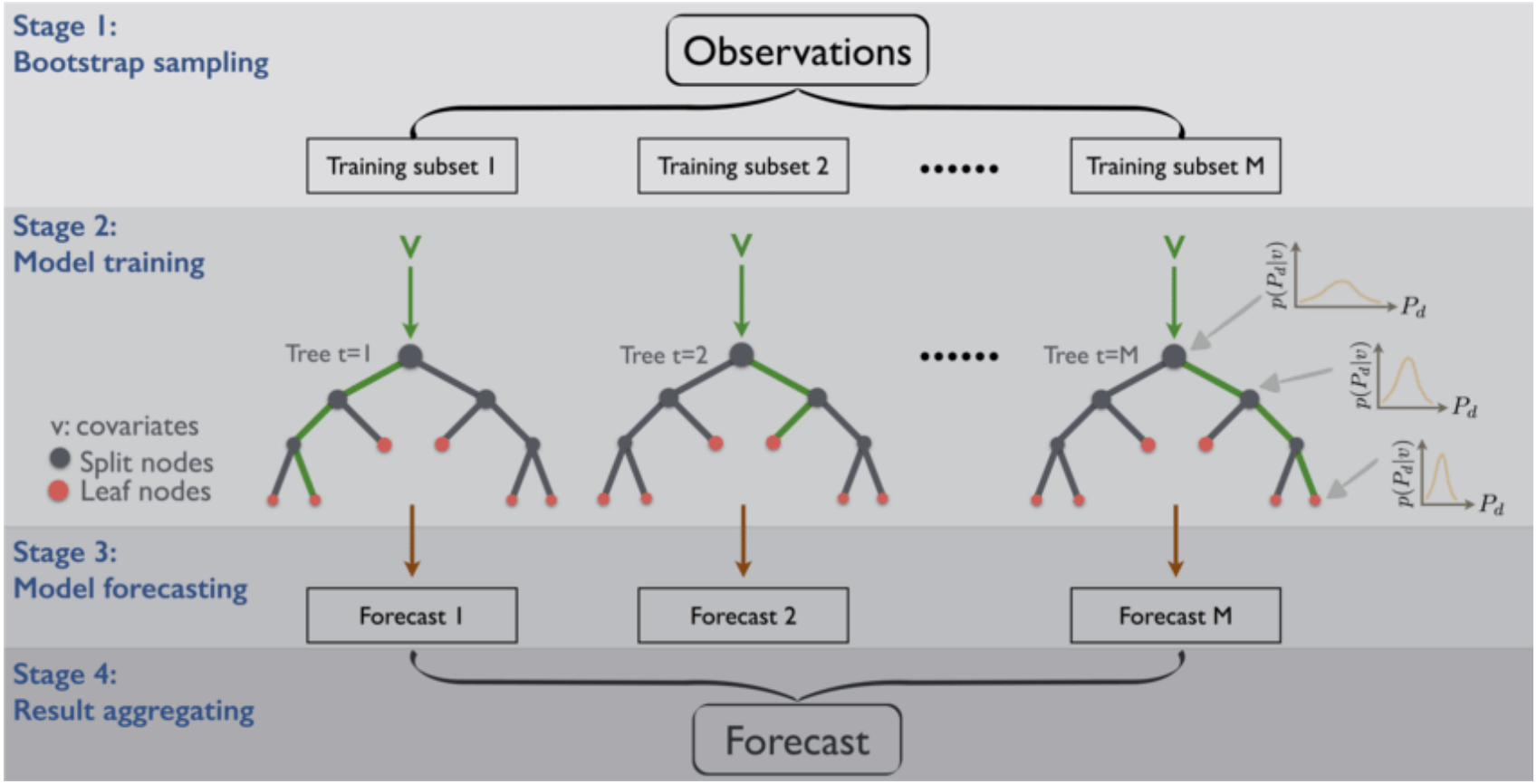
Bagging

- Bagging stands for "Bootstrap aggregating"

Procedure

1. Produce several identical sized training datasets by sampling with replacement (bootstrap)
2. Train models for all training sets using the same technique
3. Combine the individual model to come to a single predictor
   - average the predictions
   - majority vote for classification

Bagging



from He, Chaney, Schleiss, Sheffield. (2016). Spatial downscaling of precipitation using adaptable random forest

Bagging Performance Measurement

- Cross-Validation! or...

Out-of-Bag Error Estimation

- On average, each bagged tree uses about two-thirds of the observations
- The remaining one-third can be used to evaluate performance (the out-of-bag (OOB) observations)
- The response for an observation can be estimated with the trees for which it was not selected for learning.
- Average the predictions, or take majority vote, then estimate RSS or classification error.

Random Forests

- As in bagging, we build several decision trees on bootstrapped training samples
- Random forests improve over bagged trees by decorrelating the trees
  - Makes trees differ, exploring variables beyond strongest predicting ones
  - Averaging highly correlated quantities reduces variance less than averaging many uncorrelated quantities
- Each time a split in a tree is considered, a random selection of **m** predictors is chosen as split candidates from the full set of **p** predictors. The split is allowed to use only one of those **m** predictors
- A new selection of **m** predictors is taken at each split, and typically we choose $m \approx \sqrt{p}$ — the number of predictors considered at each split is approximately equal to the square root of the total number of predictors (4 out of the 13 for the Heart data)

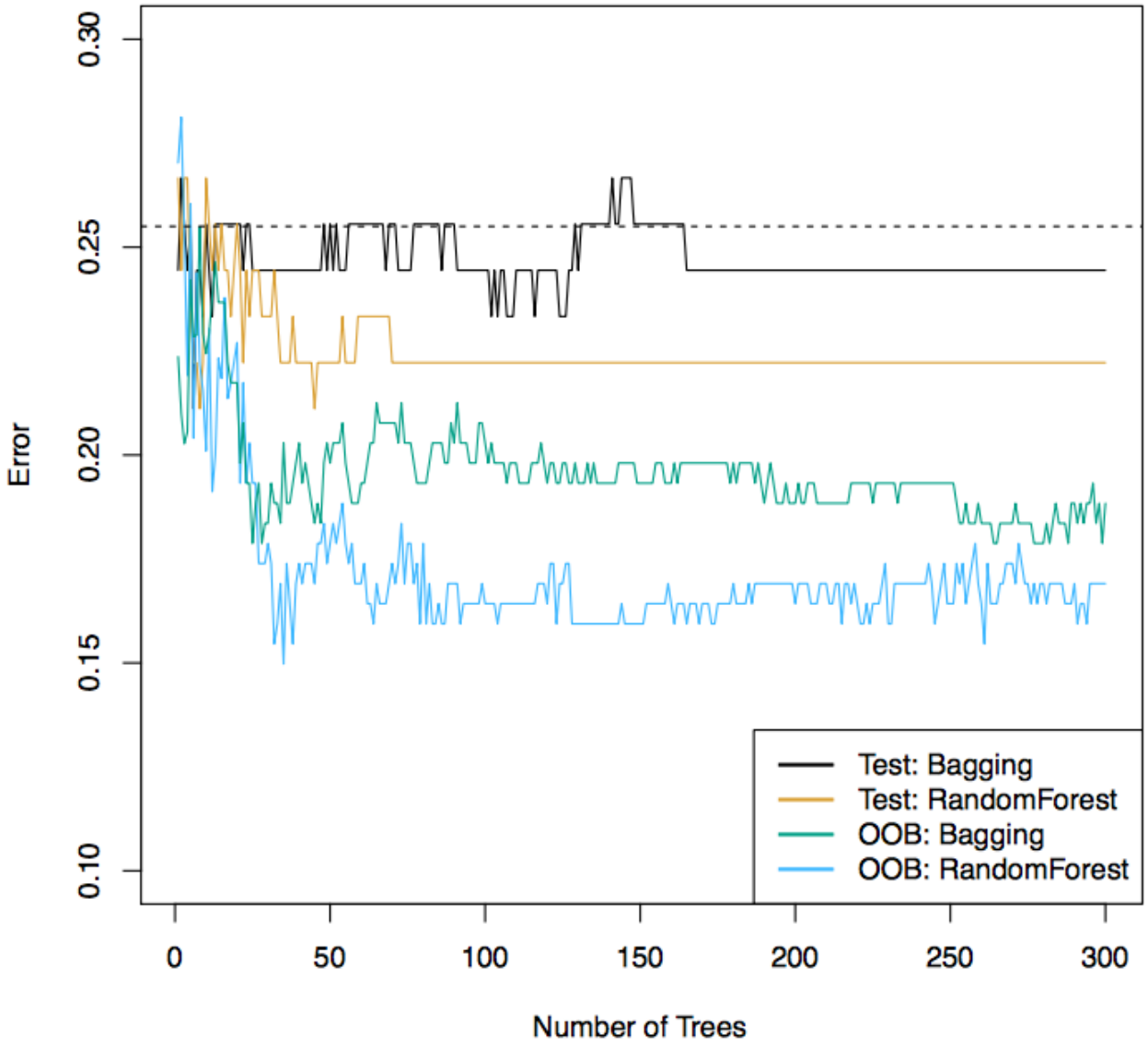
Random Forests in the context of Bioinformatics

- A good and popular predictor.
- Works well with multiple correlated variables
- Suitable for high dimensional datasets.
- It **can** yield an increase in predictive power at the cost of transparency

Note

Bagging and Random Forests don't overlearn with more trained trees! But the effect of stabilization is normally quickly achieved, leaving hardly any benefit for adding trees beyond a certain level.

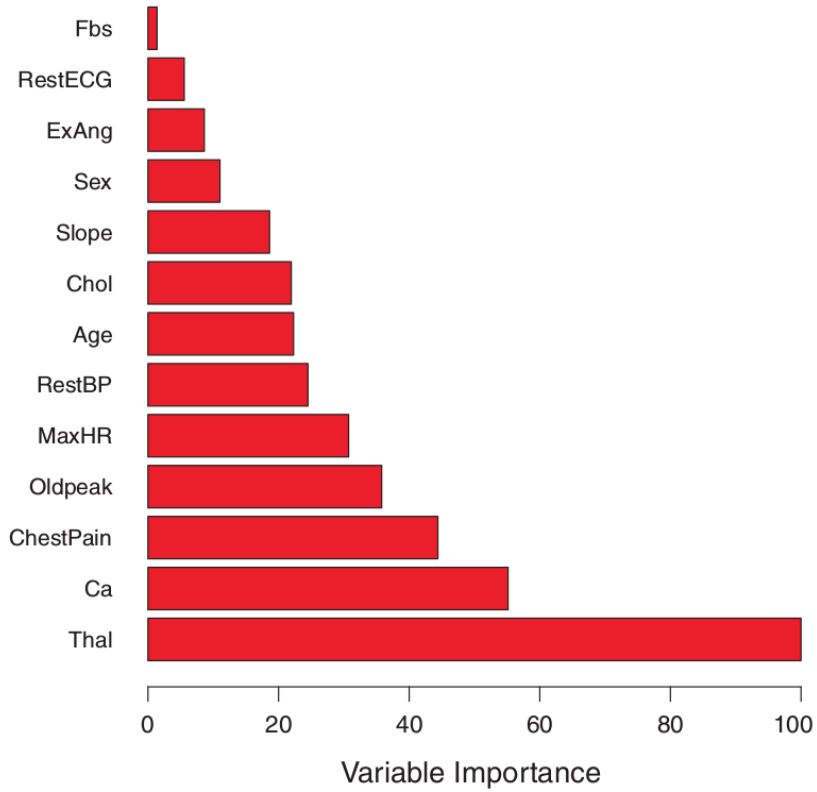
The heart data



Dotted line shows test error for single tree

Variable Importance to interpret Tree Ensembles

- Variable Importance measures the performance measure drop over the variable's tree splits:
  - Defined per tree, averaged over the ensemble
  - Regression Trees: RSS drop
  - Classification Trees: Gini index/cross-entropy drop



Boosting

- A popular ensemble method
- Progressive (or slow) learning
  - Successively learn and then combine multiple "weak" learning models to make one "strong" learning model
  - later models focus on unexplained variation by weighting the data
- Again a very general **meta**-procedure that works beyond just trees

Algorithm 8.2 *Boosting for Regression Trees*

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.

2. For $b = 1, 2, \ldots, B$, repeat:

   (a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).

   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:

   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{8.10}$$

   (c) Update the residuals,

   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$

3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

Boosting

Tuning features:

- Number of Trees in the ensemble (select with CV, overlearning can occur)
- Shrinkage parameter λ (speed of learning, value interacts with required number of trees)
- Depth of the individual trees (often a depth of 1 or 2)

AdaBoost: the adaptive booster

- There exist many variants and flavors of boosting. AdaBoost is a popular choice.
- Published by Freund and Schapire in 1997, Gödel prize 2003.
- Algorithm for *classification*. Slightly different from general "Boosting" as above.

Algorithm tweaks

- Use of a weighted error function:
  - Weights are given to datapoints to train a weak classifier with weights according to D_t for every iteration t.
  - $D_{t,i}$ is proportional to the error for sample i, at the current boosting iteration, high weight corresponds to high error.
- Instead of $\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$, λ is replaced by adaptive weights α_b, which are inversely proportional to the error rate.

AdaBoost

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$.
- Aim: select h_t with low weighted error:

$$\varepsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i].$$

- Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right)$.
- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

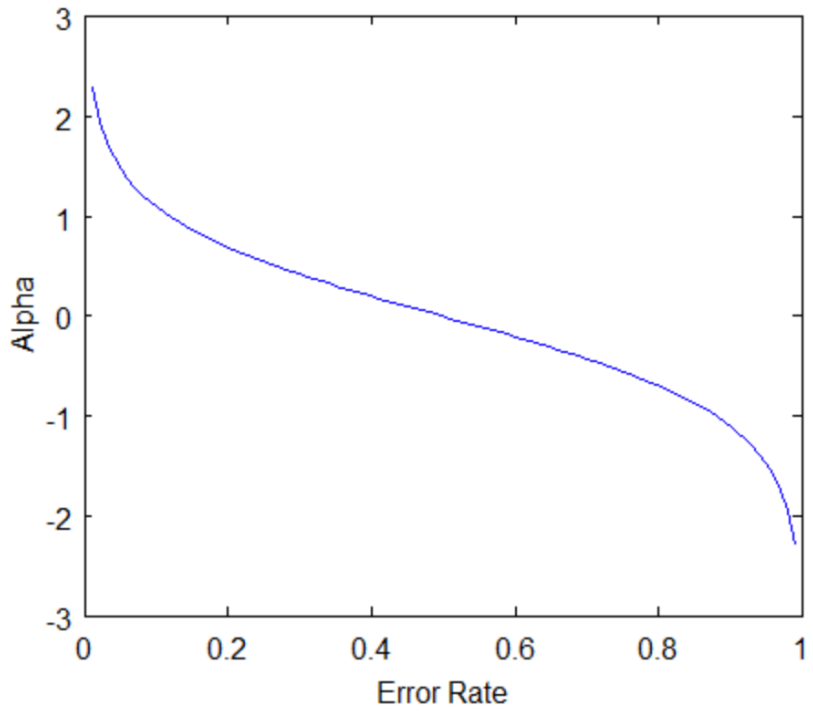
where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \operatorname{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right).$$

Fig. 1 The boosting algorithm AdaBoost.

α_t given by negative logit function

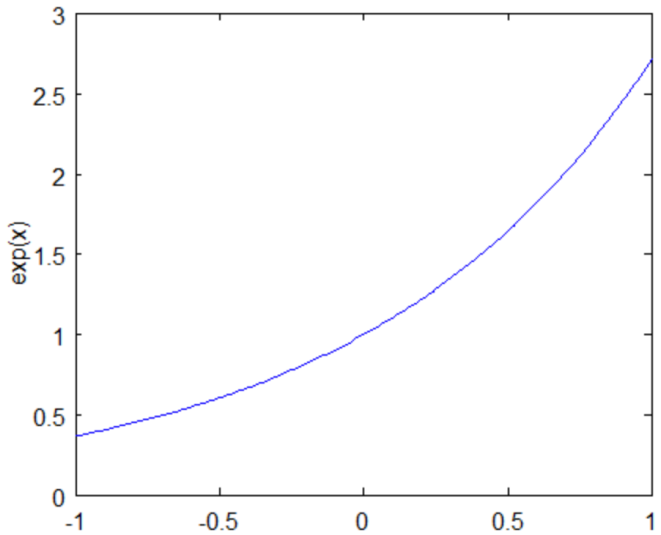


1. Weight is 0 when error rate is 0.5
2. Exponential increase/decrease apporaching bounds 0 (strong predictor) and 1 (inverse predictor)

$D_t(i)$ scales the impact of a point during learning

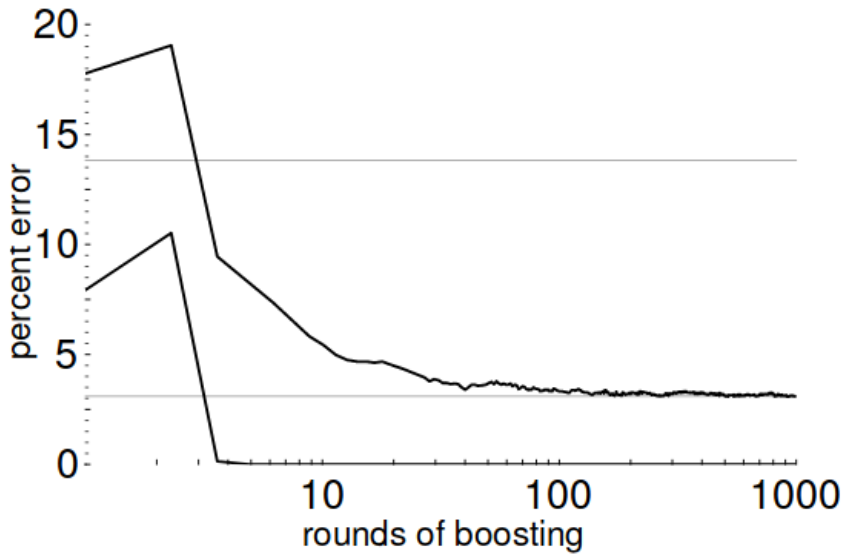
$$D_t(i+1) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

- Z scales the score to probabilities (range 0-1, summing to 1)
- $y_i h_t(x_i)$ is 1 when correctly classified, -1 when missclassified
- See below how e^x scales the weight of misclassified samples to more than 1, less than 1 for correct answers.



AdaBoost tends to resist over-learning

- Often a well performing classifier. All individual models can be poor/weak, but when better than random, the final model will converge to a "strong" learner.
- Can be sensitive to noisy data and outliers.
- In particular cases can resist over-learning.



The training and test percent error rates obtained using boosting on an OCR datasetwith C4.5 as the base learner. The top and bottom curves are test and training error, respectively. From Explaining AdaBoost by RE. Schapire

What you should learn from this chapter

- Basic principles of Regression and Classification Trees
  - learning and pruning
  - performance measures
- Bagging (incl. definitions, rationale)
  - Random Forests
- Boosting (incl. definitions, rationale)
- Variable Importance for trees

To do:

- Labs of chapter 8
- For the vd Vijver dataset of class 3: Can you improve predictive performance with trees?
  - Evaluate performance for a classification tree, a bagging of classification trees, a random forest, and classification trees with boosting
  - Compare the variable importance plots for the simple Bagging and for Random Forests