PQHS 471
Lecture 9: Tree-based Methods
Bagging, Random Forests, Boosting

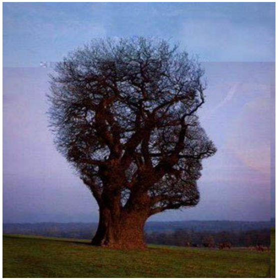
Tree-based Methods

- Here we describe tree-based methods for regression and classification.
- These involve stratifying or segmenting the predictor space into a number of simple regions.
- Since the set of splitting rules used to segment the predictor space can be summarized in a tree, these types of approaches are known as decision-tree methods.

Pros and Cons

- Tree-based methods are simple and useful for interpretation.
- However they typically are not competitive with the best supervised learning approaches in terms of prediction accuracy.
- Hence we also discuss bagging, random forests, and boosting. These methods grow multiple trees which are then combined to yield a single consensus prediction.
- Combining a large number of trees can often result in dramatic improvements in prediction accuracy, at the expense of some loss interpretation.

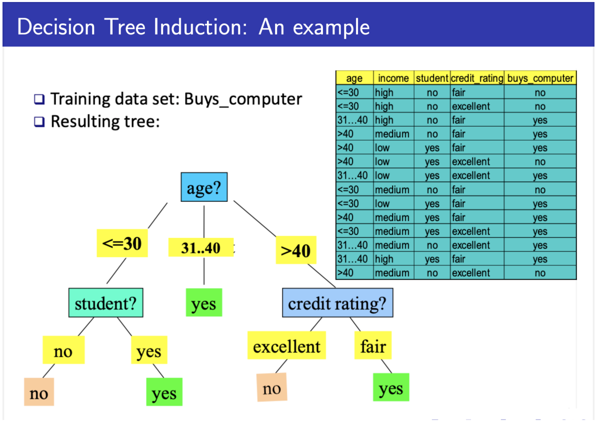
The Basics of Decision Trees



- Decision trees can be applied to both regression and classification problems.

Decision Trees classification revisit

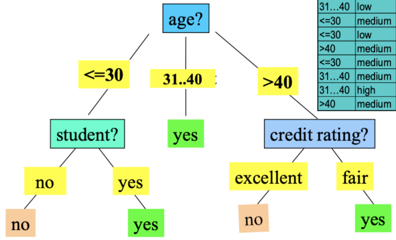
Decision trees in classification (Lecture 6).



Decision Tree Induction: An example

- ❑ Training data set: Buys_computer
- ❑ Resulting tree:

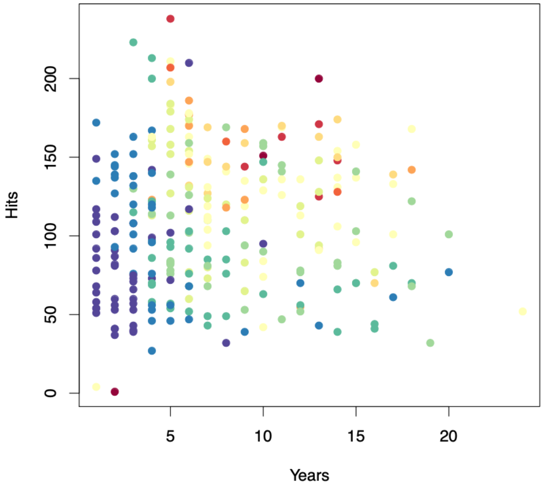
age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no



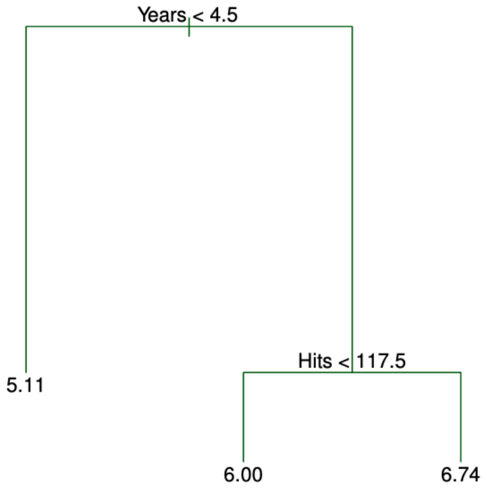
Regression Trees

Decision tree in regression: example

Baseball salary data: how would you stratify it?
Salary is color-coded from low (blue, green) to high (yellow,red)



Decision tree in regression: example



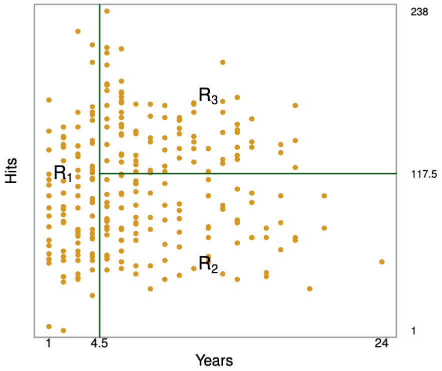
Example details

- For the **Hitters** data, a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year.
- At a given internal node, the label (of the form $X_j < t_k$) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to $X_j \geq t_k$. For instance, the split at the top of the tree results in two large branches. The left-hand branch corresponds to $Years < 4.5$, and the right-hand branch corresponds to $Years \geq 4.5$.
- The tree has two internal nodes and three terminal nodes, or leaves. The number in each leaf is the mean of the response for the observations that fall there.

Example results

Overall, the tree stratifies or segments the players into 3 regions of predictor space:

$R_1 = \{X|Years < 4.5\}$
$R_2 = \{X|Years \geq 4.5, Hits < 117.5\}$
$R_3 = \{X|Years \geq 4.5, Hits \geq 117.5\}$



Decision tree terminology revisit

- In keeping with the tree analogy, the regions R1, R2, and R3 are known as terminal nodes.
- Decision trees are typically drawn upside down, in the sense that the leaves are at the bottom of the tree.
- The points along the tree where the predictor space is split are referred to as internal nodes.
- In the **Hitters** tree, the two internal nodes are indicated by $Years < 4.5$ and $Hits < 117.5$.

Example tree interpretation

- Years is the most important factor in determining Salary, and players with less experience earn lower salaries than more experienced players.
- Given that a player is less experienced, the number of Hits that he made in the previous year seems to play little role in his Salary.
- But among players who have been in the major leagues for five or more years, the number of Hits made in the previous year does affect Salary, and players who made more Hits last year tend to have higher salaries.
- Surely an over-simplification, but compared to a regression model, it is easy to display, interpret and explain.

Tree building philosophy

- We divide the predictor space — that is, the set of possible values for $X_1, X_2, ..., X_p$ — into J distinct and non-overlapping regions, $R_1, R_2, ..., R_J$.
- For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j.

Tree building details

- In theory, the regions could have any shape. However, we choose to divide the predictor space into high-dimensional rectangles, or boxes, for simplicity and for ease of interpretation of the resulting predictive model.
- The goal is to find boxes $R_1, R_2, ..., R_J$ that minimize the RSS, given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

  where where $\hat{y}_{R_j}$ is the mean response for the training observations within the jth box.

How to build a tree

- Unfortunately, it is computationally infeasible to consider every possible partition of the feature space into J boxes.
- For this reason, we take a top-down, greedy approach that is known as recursive binary splitting.
- The approach is top-down because it begins at the top of the tree and then successively splits the predictor space; each split is indicated via two new branches further down on the tree.
- It is greedy because at each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

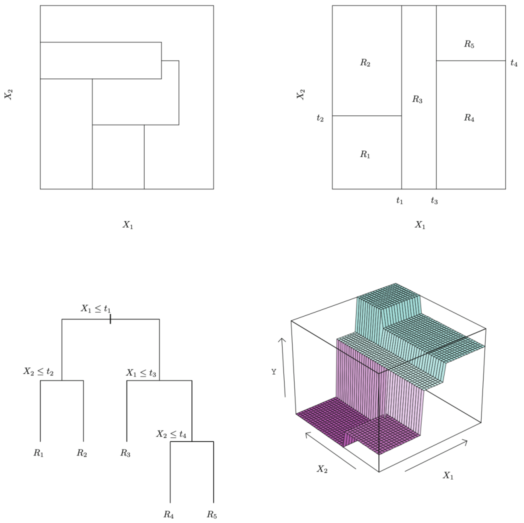
How to build a tree

- We first select the predictor X_j and the cutpoint s such that splitting the predictor space into the regions $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ leads to the greatest possible reduction in RSS.
- Next, we repeat the process, looking for the best predictor and best cutpoint in order to split the data further so as to minimize the RSS within each of the resulting regions.
- However, this time, instead of splitting the entire predictor space, we split one of the two previously identified regions. We now have three regions.
- Again, we look to split one of these three regions further, so as to minimize the RSS. The process continues until a stopping criterion is reached; for instance, we may continue until no region contains more than five observations.

Prediction using a tree

- Give a new observation, follow the tree to decide its belonged region("box")
- Use the mean of the training observations in that region as the prediction.

A five-region example



Pruning a tree

- The process described above may produce good predictions on the training set, but is likely to overfit the data, leading to poor test set performance. Why?
- A smaller tree with fewer splits (that is, fewer regions $R_1, R_2, ..., R_J$) might lead to lower variance and better interpretation at the cost of a little bias.
- One possible alternative to the process described above is to grow the tree only so long as the decrease in the RSS due to each split exceeds some (high) threshold.
- This strategy will result in smaller trees, but is too short-sighted: a seemingly worthless split early on in the tree might be followed by a very good split — that is, a split that leads to a large reduction in RSS later on.

Pruning a tree

- A better strategy is to grow a very large tree T_0, and then prune it back in order to obtain a subtree.
- Cost complexity pruning – also known as weakest link pruning – is used to do this.
- We consider a sequence of trees indexed by a nonnegative tuning parameter α. For each value of α there corresponds a subtree $T \subset T_0$ such that

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

  is minimized.

$|T|$: the number of terminal nodes of the tree T.
R_m: the rectangle (i.e. the subset of predictor space) corresponding to the mth terminal node.
$\hat{y}_{R_m}$: the mean of the training observations in R_m.

Choosing the best subtree

- The tuning parameter α controls a trade-off between the subtree's complexity and its fit to the training data.
- We select an optimal value $\hat{\alpha}$ using cross-validation.
- We then return to the full data set and obtain the subtree corresponding to $\hat{\alpha}$.

Summary: tree algorithm

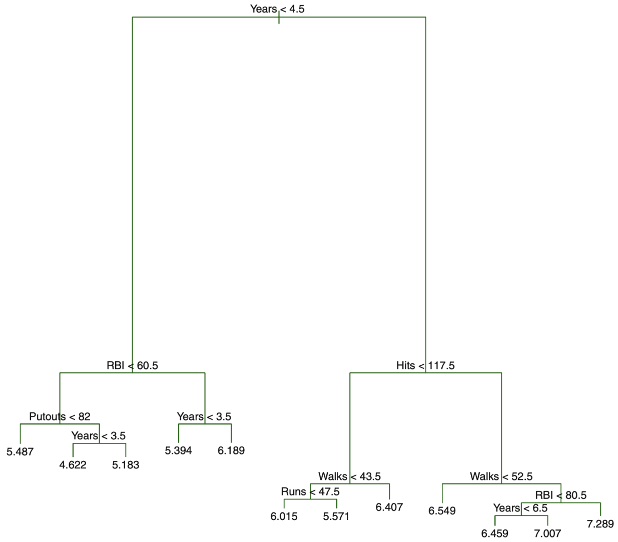
1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold cross-validation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

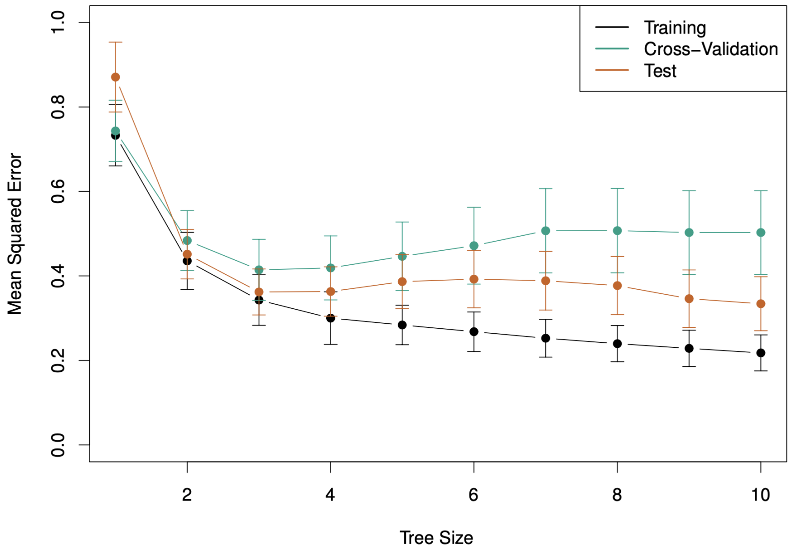
Baseball example continued

- First, we randomly divided the data set in half, yielding 132 observations in the training set and 131 observations in the test set.
- We then built a large regression tree on the training data and varied α in order to create subtrees with different numbers of terminal nodes.
- Finally, we performed six-fold cross-validation in order to estimate the cross-validated MSE of the trees as a function of α.

Baseball example continued



Baseball example continued



Classification Trees

Classification Trees

- Very similar to a regression tree, except that it is used to predict a qualitative response rather than a quantitative one.
- For a classification tree, we predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs.

Build a classification trees

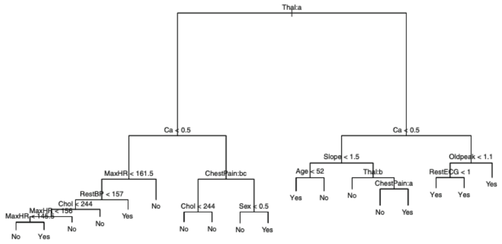
Recall contents in Lecture 6.

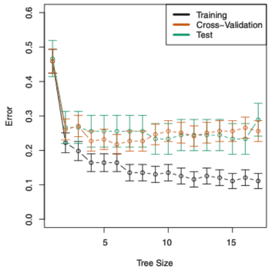
- Just as in the regression setting, we use recursive binary splitting to grow a classification tree.
  — Recall (DTI: Decision Tree Induction)
- RSS cannot be used as a criterion for making the binary splits.
- Recall: Information Gain, Gain Ratio, Gini Index.

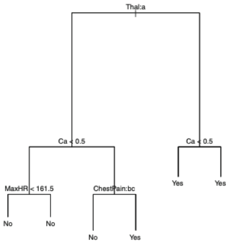
Classification trees example

- These data contain a binary outcome HD for 303 patients who presented with chest pain.
- An outcome value of Yes indicates the presence of heart disease based on an angiographic test, while No means no heart disease.
- There are 13 predictors including Age, Sex, Chol (a cholesterol measurement), Thal (Thallium stress test), and other heart and lung function measurements.
- Cross-validation yields a tree with 6 terminal nodes. See next figure.

Classification trees example







Tree vs. Linear models

- Linear regression:

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

- Regression trees:

$$f(X) = \sum_{m=1}^{M} c_m \cdot I_{(X \in R_m)}$$

where R_m represent partitions of feature space.

Tree vs. Linear models

- Linear regression:

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

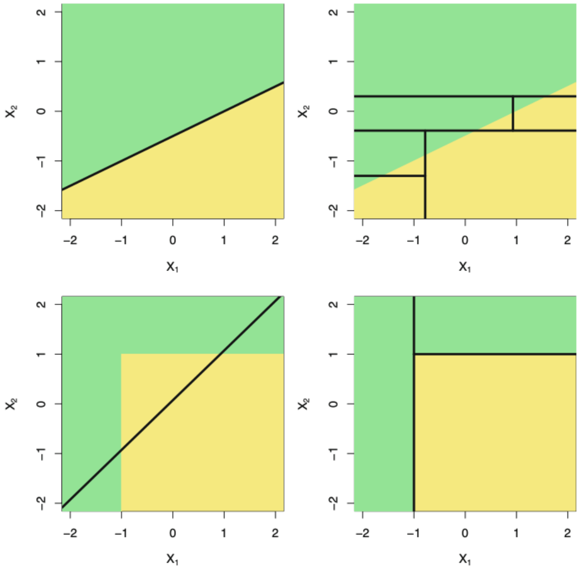
- Regression trees:

$$f(X) = \sum_{m=1}^{M} c_m \cdot I_{(X \in R_m)}$$

  where R_m represent partitions of feature space.

Which model is better? Depends on the problem at hand.

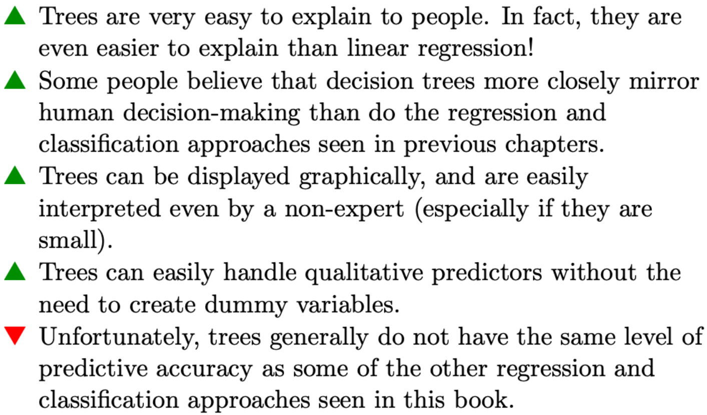
Tree vs. Linear models



Advantages and Disadvantages of Trees

- ▲ Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!
- ▲ Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous chapters.
- ▲ Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- ▲ Trees can easily handle qualitative predictors without the need to create dummy variables.
- ▼ Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches seen in this book.

Advantages and Disadvantages of Trees



▲ Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!

▲ Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous chapters.

▲ Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).

▲ Trees can easily handle qualitative predictors without the need to create dummy variables.

▼ Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches seen in this book.

However, by **aggregating** many decision trees, the predictive performance of trees can be substantially improved. We introduce these concepts next.

Bagging, Random Forest

Bagging

- **B**ootstrap **agg**regat**ing**, or **bagging**, is 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.
- Recall that given a set of n independent observations $Z_1, ..., Z_n$, each with variance σ^2, the variance of the mean $\bar{Z}$ of the observations is given by σ^2/n.
- In other words, averaging a set of observations reduces variance. Of course, this is not practical because we generally do not have access to multiple training sets.

Bagging

- Instead, we can bootstrap, by taking repeated samples from the (single) training data set.
- In this approach we generate B different bootstrapped training data sets. We then train our method on the bth bootstrapped training set in order to get $\hat{f}^{*b}(x)$, the prediction at a point x. We then average all the predictions to obtain

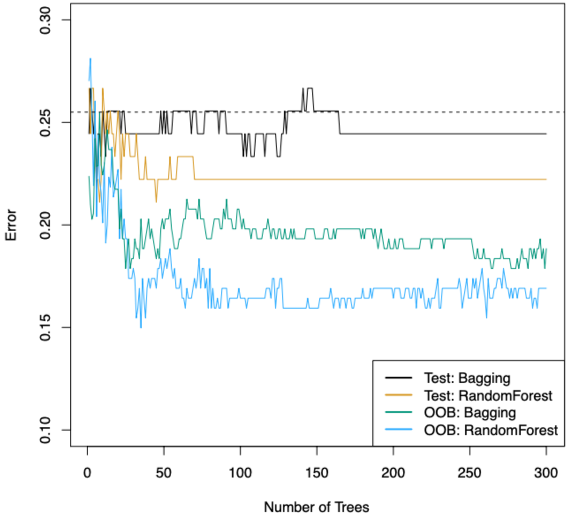
$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x)$$

This is called bagging.

Bagging classification trees

- The above procedure applied to regression trees.
- For classification trees: for each test observation, we record the class predicted by each of the B trees, and take a majority vote: the overall prediction is the most commonly occurring class among the B predictions.

Example: bagging the heart data



dashed line: single tree

Out-of-Bag error

- It turns out that there is a very straightforward way to estimate the test error of a bagged model.
- Recall that the key to bagging is that trees are repeatedly fit to bootstrapped subsets of the observations. **One can show that on average, each bagged tree makes use of around two-thirds of the observations.**
- The remaining one-third of the observations not used to fit a given bagged tree are referred to as the out-of-bag (OOB) observations.
- We can predict the response for the ith observation using each of the trees in which that observation was OOB. This will yield around $B/3$ predictions for the ith observation, which we average.
- This estimate is essentially the LOO cross-validation error for bagging, if B is large.

Random Forests

A random selection of predictors.

Picture taken from: ▸ World Heritage Centre

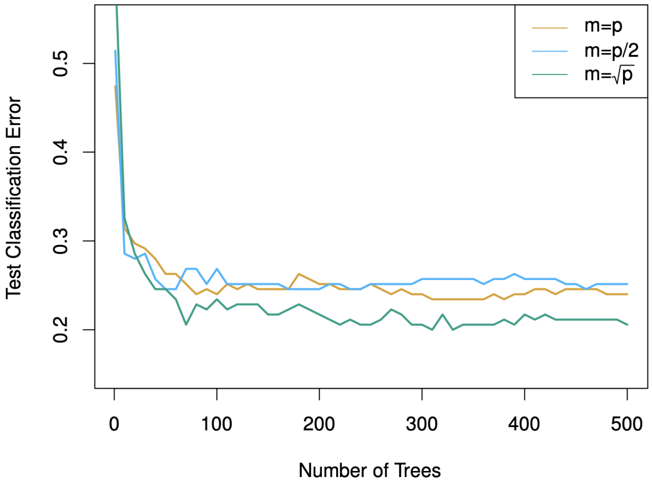
Random Forests

- Random forests provide an improvement over bagged trees by way of a small tweak that decorrelates the trees. This reduces the variance when we average the trees.
- As in bagging, we build a number of decision trees on bootstrapped training samples.
- But when building these decision trees, 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 fresh selection of m predictors is taken at each split, and typically we choose $m \approx \sqrt{p}$ — that is, 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 Forest example: gene expression data

- We applied random forest to a high-dimensional biological data set consisting of expression measurements of 4,718 genes measured on tissue samples from 349 patients.
- There are around 20,000 genes in humans, and individual genes have different levels of activity, or expression, in particular cells, tissues, and biological conditions.
- Each of the patient samples has a qualitative label with 15 different levels: either normal or one of 14 different types of cancer.
- We use random forests to predict cancer type based on the 500 genes that have the largest variance in the training set.
- We randomly divided the observations into a training and a test set, and applied random forests to the training set for three different values of the number of splitting variables m.

Random Forest example: gene expression data



Boosting

Boosting

- Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification. We only discuss boosting for decision trees.
- Recall that bagging involves creating multiple copies of the original training data set using the bootstrap, fitting a separate decision tree to each copy, and then combining all of the trees in order to create a single predictive model.
- Notably, each tree is built on a bootstrap data set, independent of the other trees.
- Boosting works in a similar way, except that the trees are grown **sequentially**: each tree is grown using information from previously grown trees.

Boosting algorithm 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:
   - 2.1 Fit a tree $\hat{f}^b$ with d splits ($d + 1$ terminal nodes) to the training data (X, r).
   - 2.2 Update $\hat{f}$ by adding in a shrunken version of the new tree:

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

   - 2.3 Update the residuals,

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

3. Output the boosted model,

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

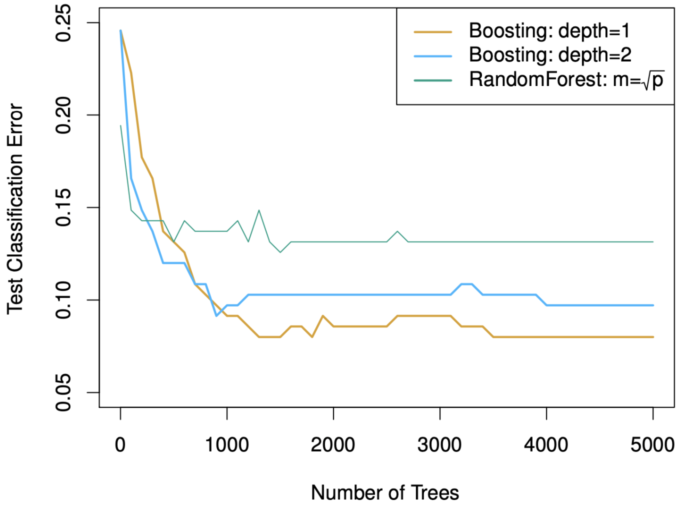
The idea behind the procedure

- Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead learns slowly.
- Given the current model, we fit a decision tree to the residuals from the model. We then add this new decision tree into the fitted function in order to update the residuals.
- Each of these trees can be rather small, with just a few terminal nodes, determined by the parameter d in the algorithm.
- By fitting small trees to the residuals, we slowly improve $\hat{f}$ in areas where it does not perform well. The shrinkage parameter λ slows the process down even further, allowing more and different shaped trees to attack the residuals.

Boosting for classification

- Boosting for classification is similar in spirit to boosting for regression, but is a bit more complex. We will not go into detail here, nor do we in the text book.
- Students can learn about the details in Elements of Statistical Learning, chapter 10.
- The R package **gbm** (gradient boosted models) handles a variety of regression and classification problems.

Gene expression example extension



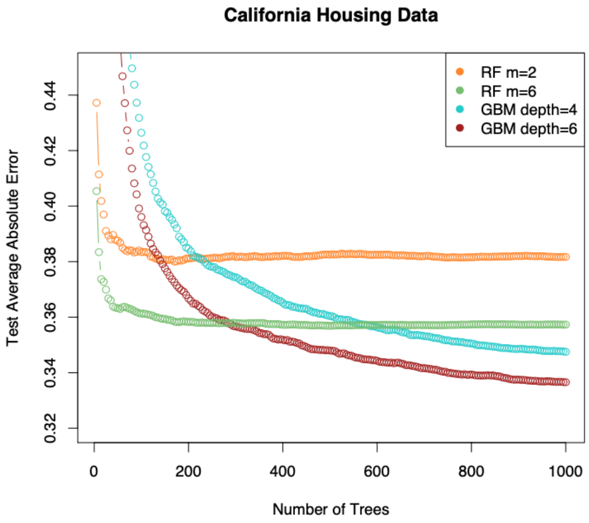
Gene expression example figure details

- Results from performing boosting and random forests on the fifteen-class gene expression data set in order to predict cancer versus normal.
- The test error is displayed as a function of the number of trees. For the two boosted models, $\lambda = 0.01$. Depth-1 trees slightly outperform depth-2 trees, and both outperform the random forest, although the standard errors are around 0.02, making none of these differences significant.
- The test error rate for a single tree is 24%.

Tuning parameters for boosting

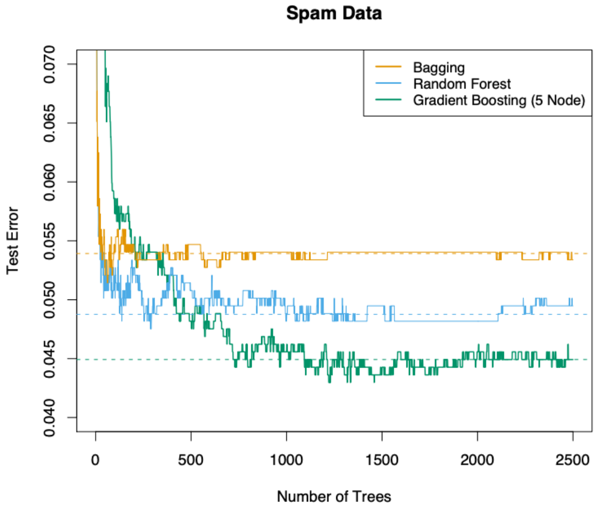
- The number of trees B. Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select B.
- The shrinkage parameter λ, a small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem. Very small λ can require using a very large value of B in order to achieve good performance.
- The number of splits d in each tree, which controls the complexity of the boosted ensemble. Often $d = 1$ works well, in which case each tree is a stump, consisting of a single split and resulting in an additive model. More generally d is the interaction depth, and controls the interaction order of the boosted model, since d splits can involve at most d variables.

Boosting example: CA housing data



from Elements of Statistical Learning, chapter 15.

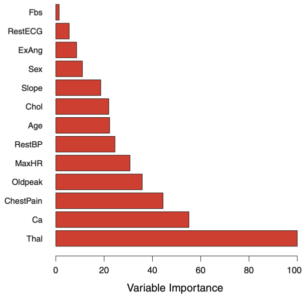
Boosting example: Spam data



from Elements of Statistical Learning, chapter 15.

Variable importance measure

- For bagged/RF regression trees, we record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees. A large value indicates an important predictor.
- Similarly, for bagged/RF classification trees, we add up the total amount that the Gini index/Entropy is decreased by splits over a given predictor, averaged over all B trees.



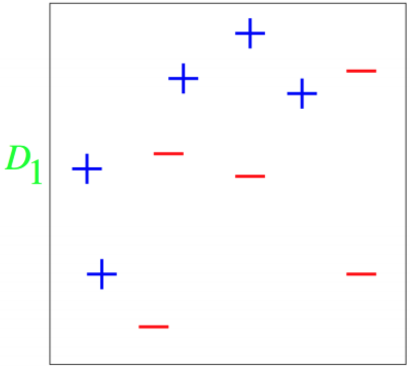
Variable importance plot for the Heart data

AdaBoost

AdaBoost

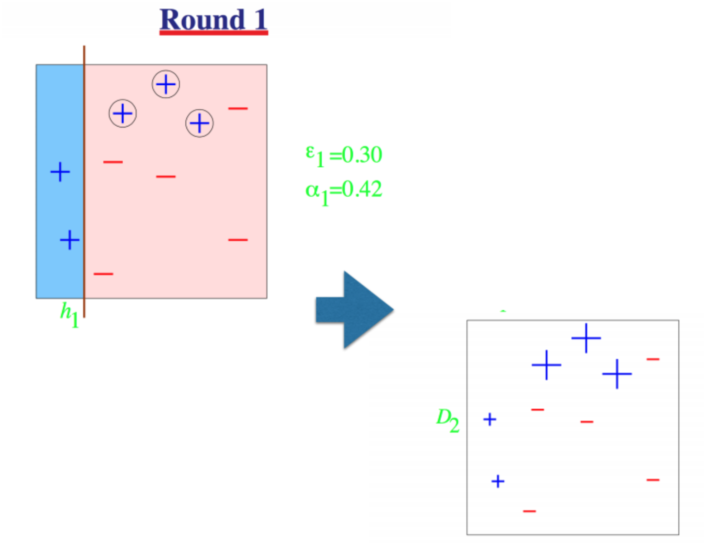
- **Ada**ptive **Boost**ing, a.k.a AdaBoost, construct a sequence of weak classifiers, and combine them into a strong classifier.
- "weak": better than random coin-tossing.
- Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy.
- Learn a series of classifiers are iteratively learned.
- Introduce weight to observations.
- Use a mechnism to **pay more attention to the training observations that were misclassified** in the previous classifier.

AdaBoost

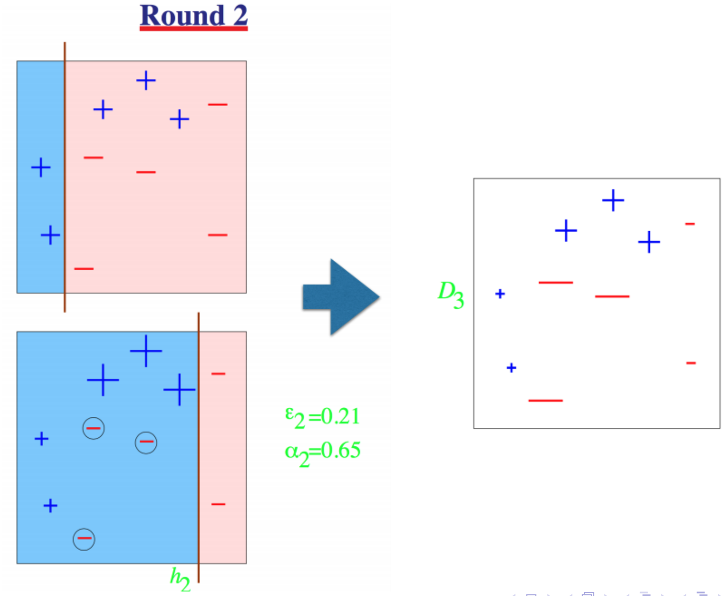


Figures from Yoav Freund and Rob Schapire

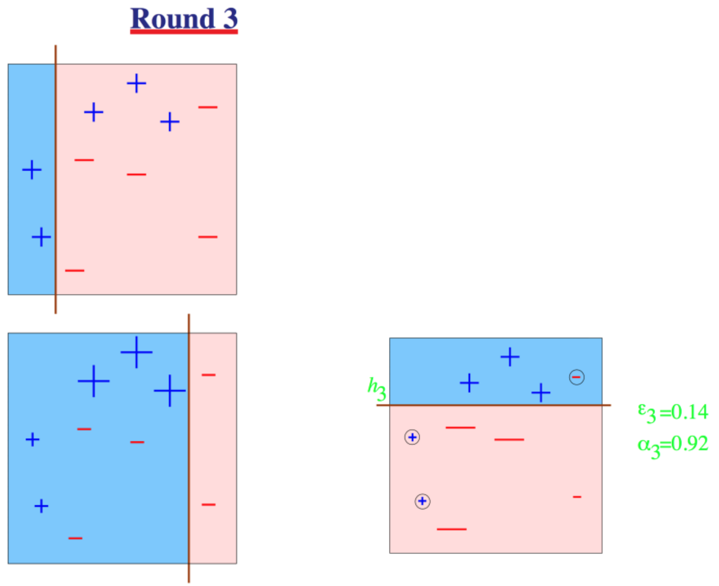
AdaBoost



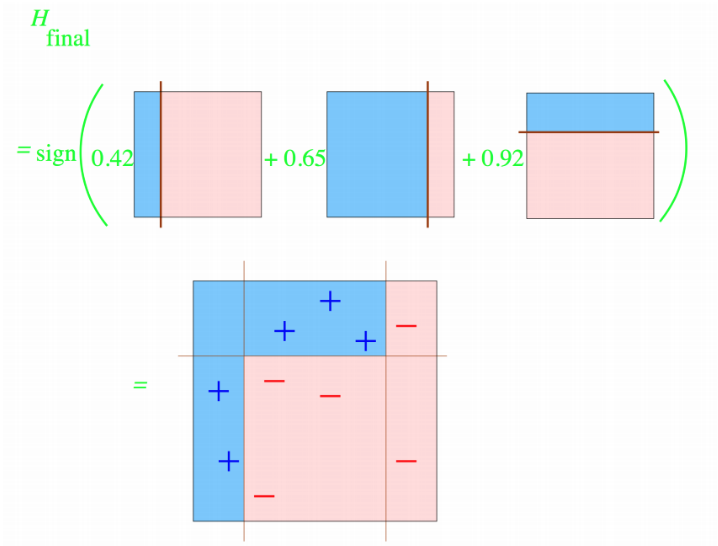
AdaBoost



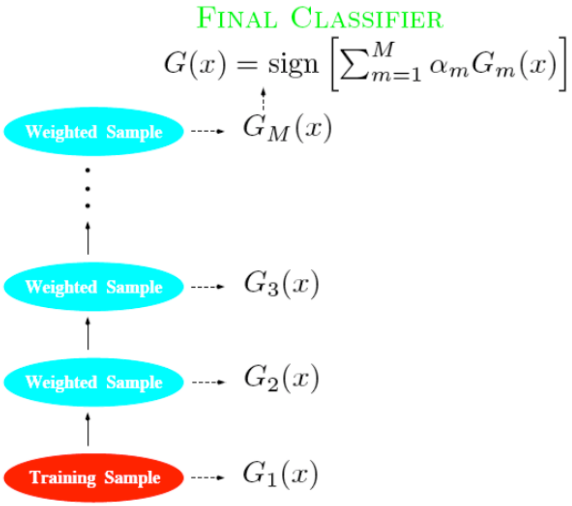
AdaBoost



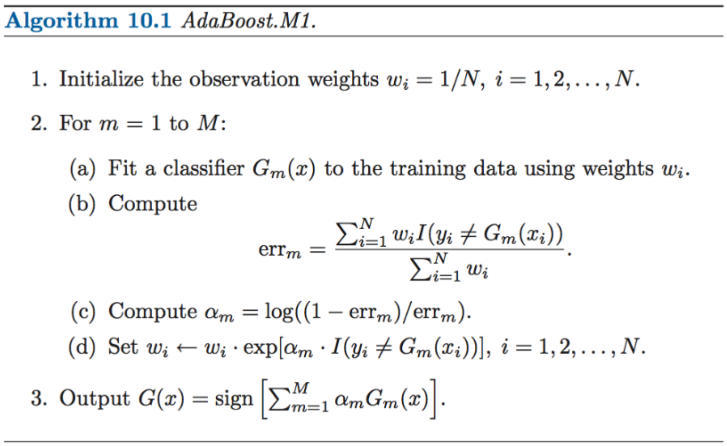
AdaBoost



AdaBoost


FINAL CLASSIFIER
$G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$
Weighted Sample
$G_M(x)$
Weighted Sample
$G_3(x)$
Weighted Sample
$G_2(x)$
Training Sample
$G_1(x)$

AdaBoost



Algorithm 10.1 *AdaBoost.M1.*

```
1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to $M$:
    (a) Fit a classifier $G_m(x)$ to the training data using weights $w_i$.
    (b) Compute
        $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
    (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
    (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.
```

(c): is the weight of the current weak classifier in the final model
(d): is the weight for individual obs. If an obs. is correctly classified at this step, its weight doesn't change. If incorrectly classified, its weight increases.

AdaBoost

- Flexible.
- Able to select features.
- Good generalization.
- Could fit noise (i.e. overfit), sensitive to noisy data and outliers.
- Adaptive in the sense that subsequent weak classifiers are tweaked in favor of those obs. misclassified by previous classifiers.

Ensemble methods

Create multiple models and then combine them to produce improved results.

- Bagging
  Parallel training with different training sets.
- Random Forest
  Decorrelated bagged trees.
- Boosting
  Sequential training, re-weight training examples to focus on misclassified observations.

The “mixture of experts” approach.

Summary

- Decision trees are simple and interpretable models for regression and classification.
- However they are often not competitive with other methods in terms of prediction accuracy.
- Bagging, random forests and boosting are good methods for improving the prediction accuracy of trees. They work by growing many trees on the training data and then combining the predictions of the resulting ensemble of trees.
- The latter two methods— random forests and boosting— are among the state-of-the-art methods for supervised learning. However their results can be difficult to interpret.

Textbook chapters

- ISLR: chapter 8: 8.1 - 8.2
- Optional readings: ESL: 8.7, 10.1-10.9, chpt 15.