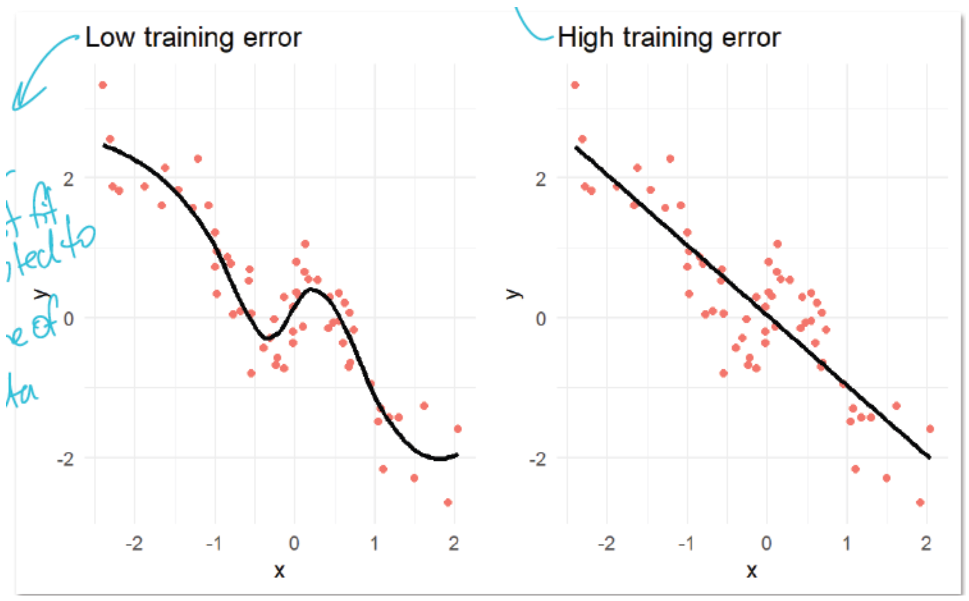
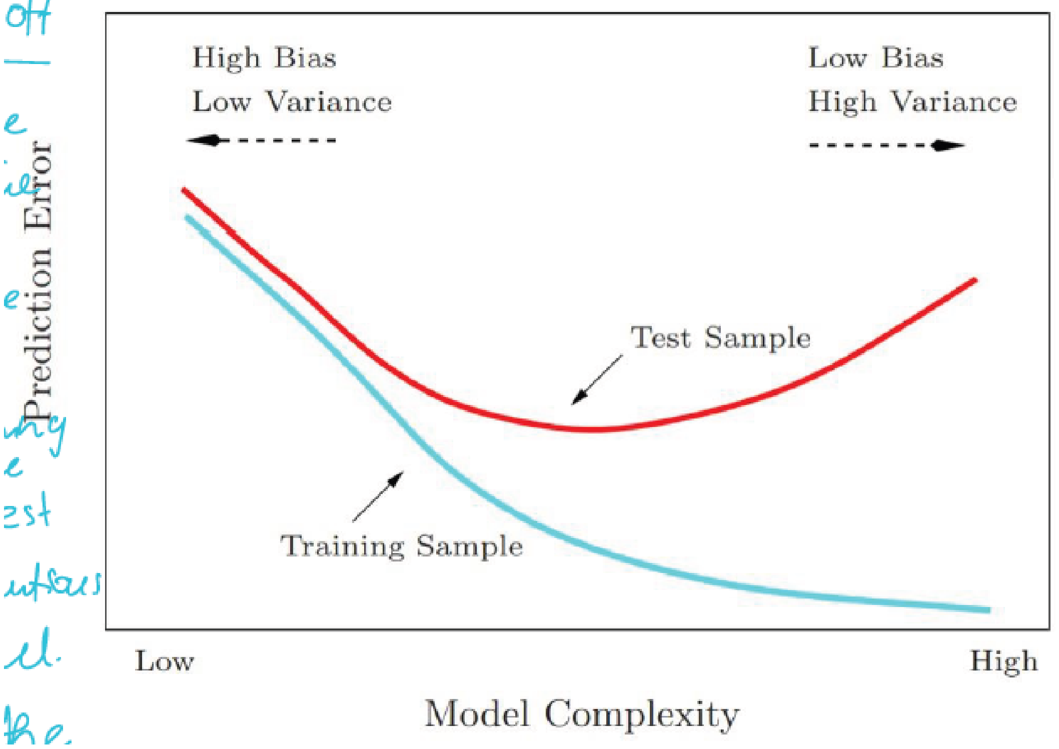
Week03 - Summary

Training vs Testing

- **Training error** is the performance metric applied to the observations used to train the model
- **Test error** is the average error when applying a model to predict the response on new (test) observations that were not used in the training of the model
    - Training error can underestimate the test error



off
e
ie
e
why
e
est
utilises
el.
the.



Bias-Variance Tradeoff

- We observe that the more complex we make our model, the better it seems to perform, i.e. the prediction error gets lower in the training sample
- However, in the test sample, we must be cautious not to overfit the model. Hence we must find the right balance between bias and variance

Common test set approach

- Randomly divide the available set of samples into two:
  1. Training set
  2. Test set
- The model:
  - Is fitted on the training set
  - Can be used to predict the responses in the test set

- Quality of the predictive performance uses the test set with:
    - Mean square error (MSE) for a quantitative response
    - Misclassification rate for a qualitative (categorical) response

Drawbacks of the test set approach

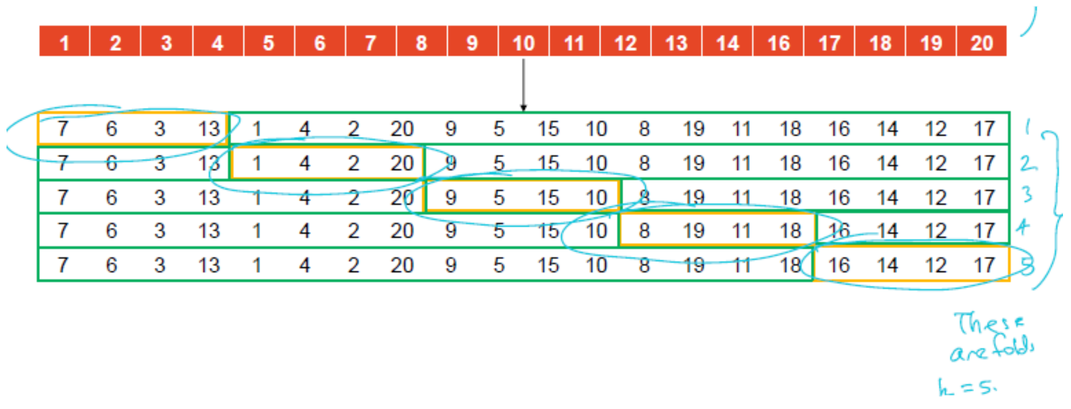
- The estimate of the test error can be highly variable
    - Depends on assignment of observations to the training and test sets
- Only a subset of the observations are used to:
    - Fit the model
    - Assess the model
- Test set error may overestimate the test error compared to a model fit on the entire data set

Cross-Validation

K-Fold Cross Validation

- Widely used approach for estimating test error
    - Estimates can be used to select best model and to give an idea of the test error of the final chosen model
- Idea: randomly divide the data into K equal-sized parts
    - Leave out part k, fit the model to the other $K-1$ parts (combined), and then obtain predictions for the left-out kth part
- This is done in term for each point $k=1,2,\ldots K$ and then the results are combined

Example: 5-Fold



Cross-Validation Process in Regression

- Define K folds as $C_1, C_2, \ldots, C_K$, where:
  - C_k denotes the indices of the observations in fold k ✓✓
  - There are n_k observations in fold k ✓✓
- Compute error metric for each fold
  - Mean square error $MSE_k = \sum_{i \in C_k} (y_i - \hat{y}_i)^2 / n_k$
  - $\hat{y}_i$ are the predictions on the test fold, k
- Aggregate over all folds
  - Overall error = $\sum_k MSE_k / K$

Cross-Validation for Classification Problems

- Same as regression but change the assessment metric
- Compute the accuracy for each fold by calculating:
  - A_k: the accuracy of the classifier in fold k

    A
    - $A_k = \frac{1}{n_k} \sum_{i \in C_k} \mathbb{1}_{\{\hat{y}_i = y_i\}}$
    - n: the total number of observations in the data set
    - n_k: the number of observations in the belonging to class k
- Aggregate over all folds
  - $\boxed{\text{Overall accuracy} = \frac{1}{K} \sum_k A_k}$

Repeated Cross-Validation Properties

Repeated CV provides a less-biased CV error estimate

- Repeated CV also gives you the variance of the CV error
- However, it comes with a computational cost
- Implemented in the `caret` package in R

Example of CV Procedure

- Consider a high-dimensional data set
  - All entirely numeric
  - Need dimension reduction to proceed
- You decide to use the following CV procedure:
  1. Compute correlation matrix, select the top 50 variables that have the highest correlation with the response
  2. Use these 50 variables as features and perform K-fold cross-validation

- Variable selection performed once using both the training and test data sets
- Information can leak from the test to the training set
- Hence, the CV error estimate is likely to be biased
- Ideally, refrain from using the test data in any way in the training step

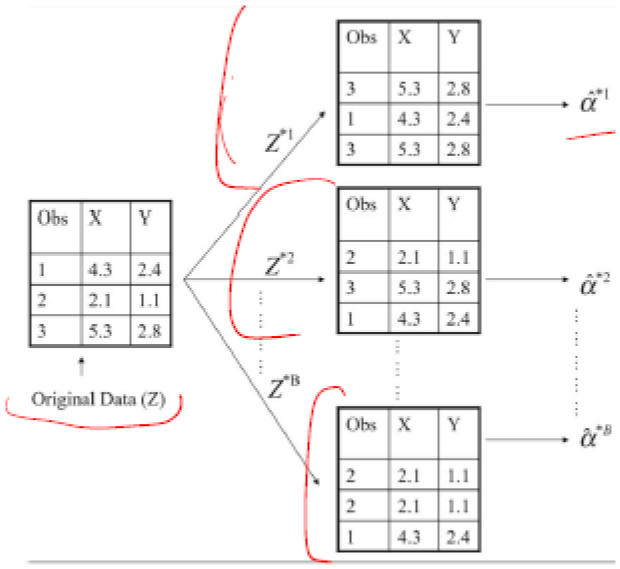
Corrected CV Procedure

- Split the data set into K folds
- For each $k = 1, 2, \ldots, K$
  - Determine the variables that correlate the best with the response using all the data except the data in fold k

- Train your model using the selected variables above
- Run your classification algorithm and record accuracy against the test set

Bootstrap

- Essentially sampling with replacement



Obs	X	Y
1	4.3	2.4
2	2.1	1.1
3	5.3	2.8

Obs	X	Y
3	5.3	2.8
1	4.3	2.4
3	5.3	2.8

Obs	X	Y
2	2.1	1.1
3	5.3	2.8
1	4.3	2.4

Obs	X	Y
2	2.1	1.1
2	2.1	1.1
1	4.3	2.4

Simple Investment Example

- Goal: invest a fixed sum of money in two financial assets
    - That yield returns of X and Y where X and Y are random quantities
- Problem: need to decide allocation in each asset
    - Invest fraction α of our wealth in X and $(1-\alpha)$ in Y
- Criterion: minimise the total risk of the investment
    - Mathematically involves minimising $Var(\alpha X + (1-\alpha)Y)$
    - The solution (via calculus) is:

$$\boxed{\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}}$$

Where $\sigma_X^2 = Var(X)$, $\sigma_Y^2 = Var(Y)$ and $\sigma_{XY} = Cov(X,Y)$

Proof:

$$Var(\alpha X + (1-\alpha)Y) = \alpha^2 Var(X) + (1-\alpha)^2 Var(Y) + 2\alpha(1-\alpha)Cov(X,Y)$$

To minimise this expression, we take the derivative with respect to α and set it to 0:

$$\frac{\partial}{\partial\alpha}(\alpha^2 Var(X) + (1-\alpha)^2 Var(Y) + 2\alpha(1-\alpha)cov(X,Y)] = 0$$

simplifying, we get:

$$2\alpha Var(X) - 2(1-\alpha)Var(Y) + 2cov(X,Y) = 0$$

solving for α, we get:

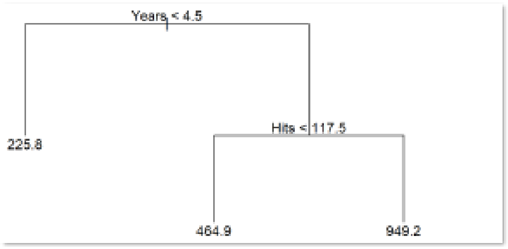
$$\alpha = \frac{Var(Y) - Cov(X,Y)}{Var(X) + Var(Y) - 2cov(X,Y)}$$

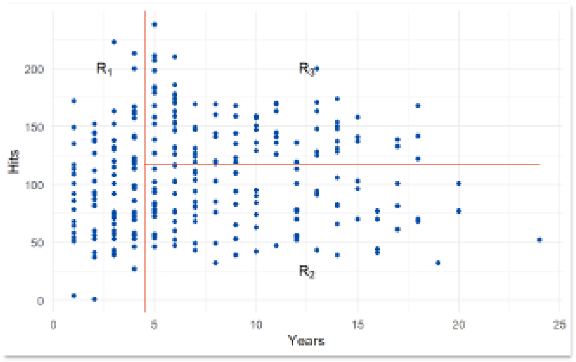
$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}$$, where $\sigma_X^2 = Var(X)$, $\sigma_Y^2 = Var(X)$, $\sigma_{XY} = Cov(X,Y)$

- Suppose that X and Y can be sampled from the population repeatedly
- Can estimate the standard deviation of α_hat
    - Paired observations (X, Y) can be repeatedly simulated, say 100 pairs to get a single estimate of α; repeat this process to get 1000 estimates for α

Regression Trees

- Segments the players into three regions.
  1. $R_1 = \{X | \text{Years} < 4.5\}$
  2. $R_2 = \{X | \text{Years} \geq 4.5, \text{Hits} < 117.5\}$
  3. $R_3 = \{X | \text{Years} \geq 4.5, \text{Hits} \geq 117.5\}$





Details on tree building process

The goal is to find boxes $R_1, \ldots, R_J$ that minimises the residual sum of squares (RSS), given by:

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

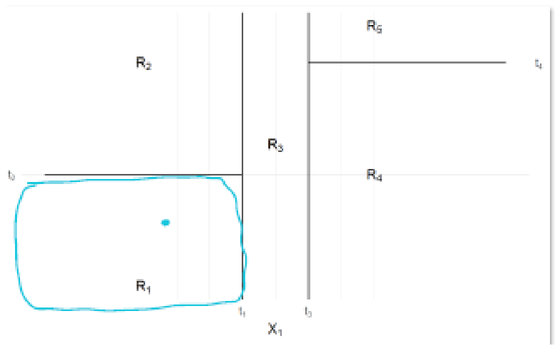
Where $\hat{y}_{R_j}$ is the mean response for the training observations within the j^{th} box

Tree building with recursive binary splitting

- Take a top-down, **greedy** approach that is known as **recursive binary splitting**
  - **Top-down:** begins at the top of the tree
    - Then successively splits the predictor space
    - Each split is indicated via two new branches further down on the tree
  - **Greedy:** at each step: the best split is made at that particular step
    - Rather than looking ahead and picking a split will lead to a better tree in some feature step

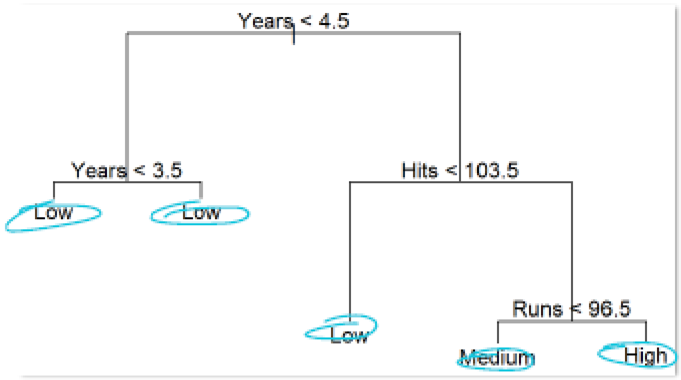
Using regression trees for prediction

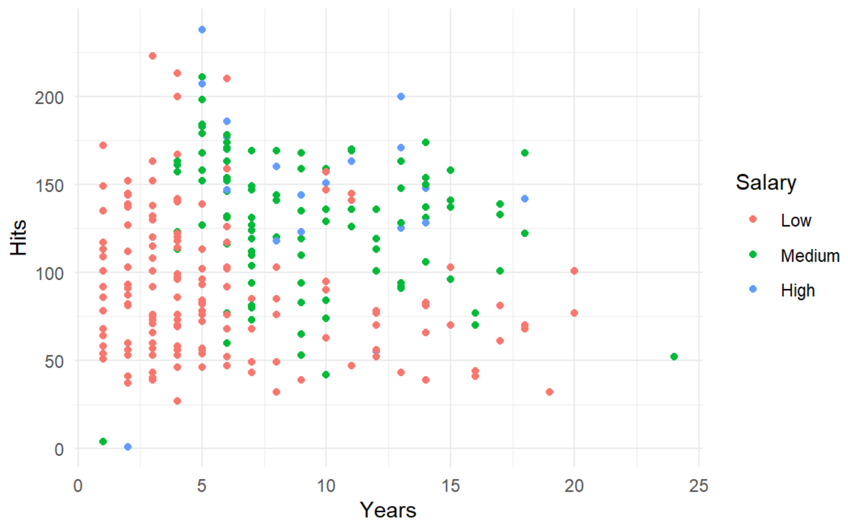
- Partition the predictor space
    - The set of possible values for $X_1, X_2, \ldots, X_p$ into J distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$
- Mean value with each region used for prediction
    - If observation in region R_J, mean of training observations R_J used for prediction



Decision Trees

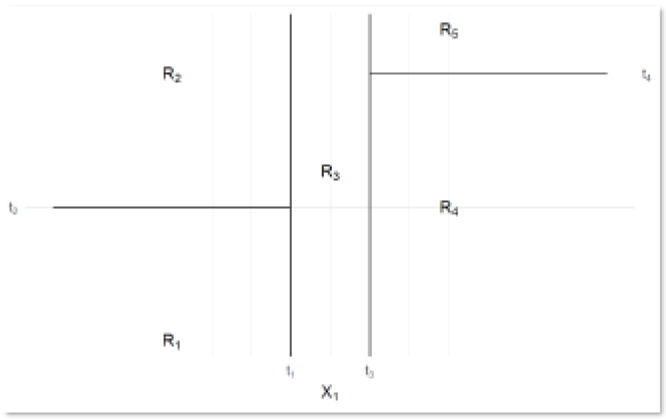
- What if baseball player salary was categorical (ordinal)? **Salary** is colour-coded





Decision Trees for Classification

- Very similar to a regression tree
    - Exception being the production of a qualitative response rather than a quantitative one
- For a classification tree
    - Inspect the region that the observation belongs and predict the most commonly occurring class in that region



Gini Index: Tree building with qualitative target

- Still recursive binary splitting to grow a classification tree
  - Can’t use RSS as criterion for binary splits
- Alternative measure such as Gini Index is used instead
- The Gini Index is defined by:

$$G = \sum_j \sum_{k=1}^{K} \hat{p}_{jk} \left(1 - \hat{p}_{jk}\right)$$

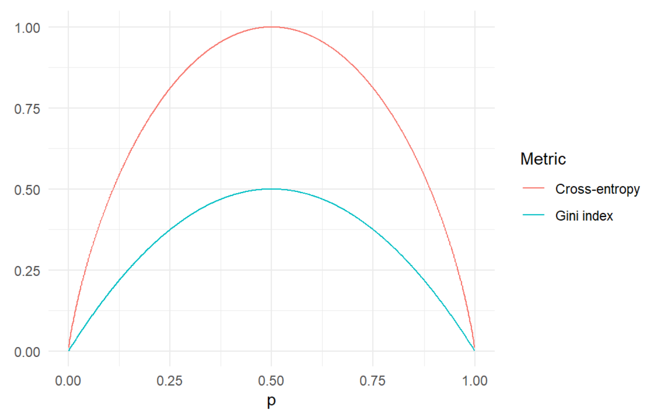
$\hat{p}_{jk}$ represents the proportion of training observations in the j^{th} region that are from the k^{th} class

Gini Index

- Is a measure of total variance across the K classes
- Gini index is referred to as a measure of node **purity**
  - A small value indicates that a node contains predominantly observations from a single class

Cross-Entropy: An alternative to Gini

- $D = -\sum_m \sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$
- It turns out that the Gini index and the cross-entropy are very similar numerically

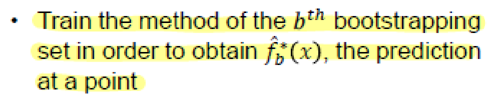


Ensemble Methods with Bootstrapping

Bagging (Bootstrap Aggregation)

- Bootstrap aggegation, or bagging
    - General purpose procedure for reducing the variance of a statistical learning method
    - Useful and frequently used in the context of decision trees
- Recall that given a set of n observations $Z_1, \ldots, Z_n$ each with a variance of σ^2
    - The variance of the mean Z_bar given by σ^2/n
    - In other words, averaging a set of observations reduces variance
- Usually not possible to have access to multiple training sets
    - Using **bootstrapping** instead to create multiple training sets

- Use bootstrapping; can generate B different bootstrapped training data sets

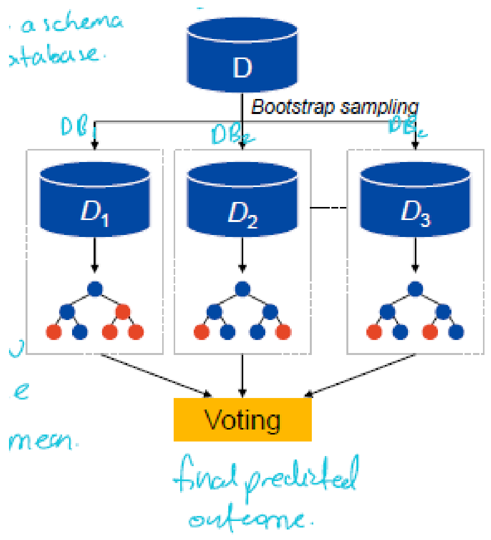


- Train the method of the b^{th} bootstrapping set in order to obtain $\hat{f}_b^*(x)$, the prediction at a point

- Average all the observations to obtain:

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

- This is called bagging



Out-of-Bag Error Estimation

- Test error of a bagged model is easy
- Each bagged trees using bootstrapped data
  - Only around two-thirds of the original observations are used
  - The remaining one-third of the observations not used
  - Referred to as the out-of-bag (OOB) observations
- Enables prediction of 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
- The estimate is essentially the leave one out (LOO) cross-validation error for bagging, if B is large

From Bagging to Random Forest

- Random forests (sometimes) provide an improvement over bagged trees
  - A small change decorrelates the trees
  - Reduces the variance when we average the trees
- Random forest approach
  - Create decision trees on bootstrapped training samples
  - When tree fitting, 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
  - A fresh selection of m predictors is taken at each split

The fundamental difference between bagging and random forest is that in Random forests, only a subset of features are selected at random out of the total and the best split feature from the subset is used to split each node in a tree, unlike in bagging where all features are considered for splitting a node.

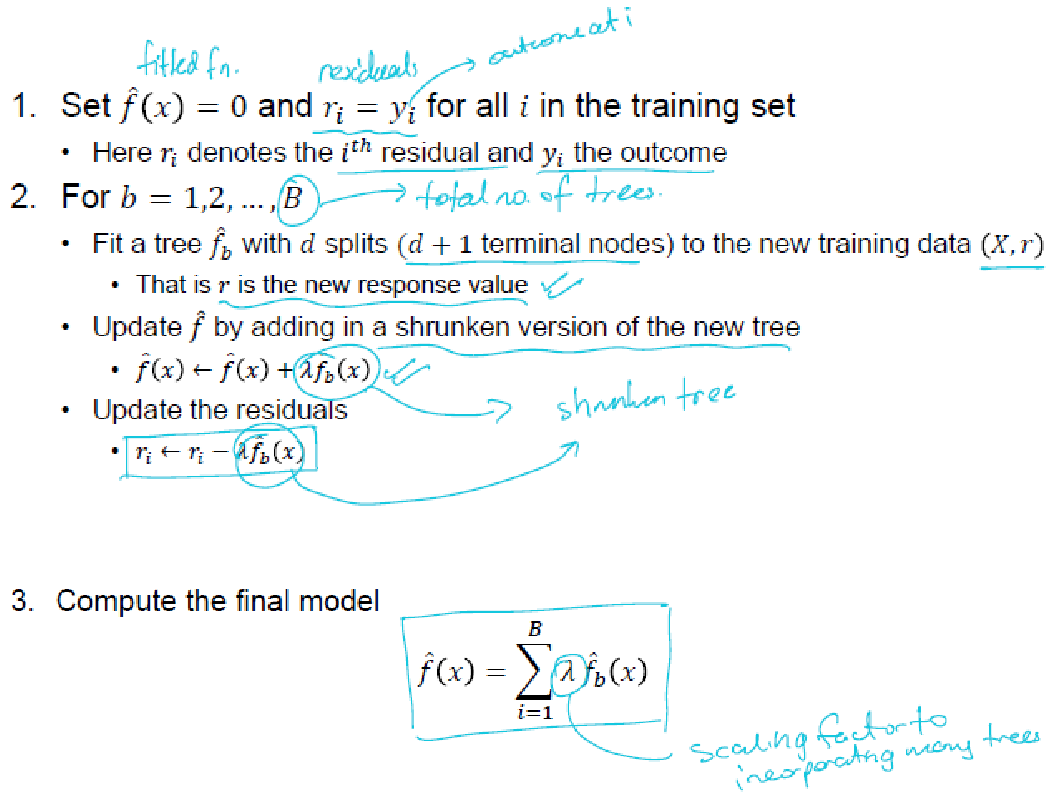
Ensemble Methods with Boosting

- Instead of computing a single tree once
- Learn from the mistakes of the current tree
- Take a sample of decision trees into account
- Make final prediction model using the aggregated result from an ensemble of trees

Boosting

- Bootstrap aggregation combines the results of separately grown trees
- Boosting similar except that the trees are grown *sequentially*

Boosting for Regression Trees



fitted fn. residuals → outcome at i

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set
   - Here r_i denotes the i^{th} residual and y_i the outcome
2. For $b = 1,2, \dots, B$ → total no. of trees.
   - Fit a tree $\hat{f}_b$ with d splits ($d + 1$ terminal nodes) to the new training data (X, r)
     - That is r is the new response value ✓
   - Update $\hat{f}$ by adding in a shrunken version of the new tree
     - $\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}_b(x)$ ✓ → shrunken tree
   - Update the residuals
     - $\boxed{r_i \leftarrow r_i - \lambda \hat{f}_b(x)}$

3. Compute the final model

$$\boxed{\hat{f}(x) = \sum_{i=1}^{B} \lambda \hat{f}_b(x)}$$

scaling factor to incorporating many trees

Parameters to tune in Boosting

- Number of trees B
- The shrinkage parameter λ a small positive number
  - Typical values are between 0.01 and 0.001
- Number of split d in each tree
  - Controls the complexity of the boosted ensemble
  - If $d = 1$, then the tree is just a stump; this actually usually works quite well

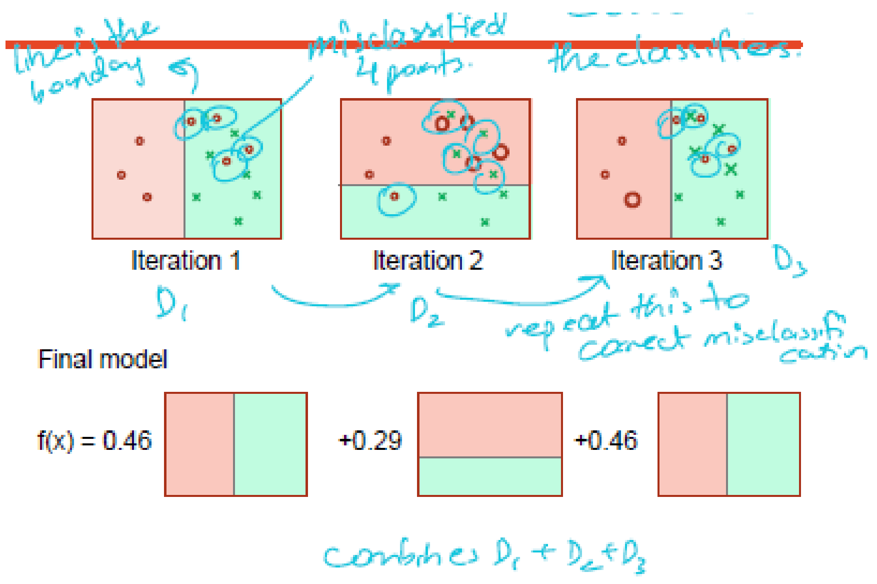
Bagging vs Boosting

- Both ensemble methods get N learners from 1 learner
  - Built independently for bagging
  - Built sequentially for boosting

- Trees built-in
    - Boosting are weak learners (sometimes just a stump)
    - Random forest have higher complexity
- Number of parameters to tune
    - Low in random forest
    - High in boosting (depending on which variation you are using)

Another Boosting Algorithm: Adaboost

- Basic idea
    - At each iteration, reweigh the data to place more weight in data points that the classifier got wrong
- Combine all the weak classifiers by taking a weighted combination
- Put more weight on the weak classifiers with the higher accuracies


line is the boundary
misclassified 4 points.
the classifiers.
Iteration 1
Iteration 2
Iteration 3
D1
D2
D3
repeat this to correct misclassification
Final model
f(x) = 0.46
+0.29
+0.46
combines D1 + D2 + D3