UPPSALA UNIVERSITET

- Decision trees
- Ensemble methods
  - Bagging
  - Random forests
  - (Gradient) Boosting

Machine learning – Block 2

Måns Magnusson
Department of Statistics, Uppsala University

Autumn 2025

This week's lecture

- Trees
- Bagging
- Random Forest
- Boosting (Trees)

Section 1

Decision trees

Decision trees: basic idea

- Have you ever played the game "20 questions"?

Decision trees: basic idea

- Have you ever played the game "20 questions"?
- Decision trees is more or less that game!

Decision trees: basic idea

- Have you ever played the game "20 questions"?
- Decision trees is more or less that game!
- In the case of classification (prediction), the idea is to classify the new observation by
  1. Asking a questions
  2. Based on the previous answer, ask new question
  3. Questions are asked until a conclusion is reached

Decision trees: basic idea

Name	Body temp	Gives birth	Legs	Class
Human	warm-blooded	yes	yes	mammal
Whale	warm-blooded	yes	no	mammal
Cat	warm-blooded	yes	yes	mammal
Cow	warm-blooded	yes	yes	mammal
Python	cold-blooded	no	no	reptile
Komodo dragon	cold-blooded	no	yes	reptile
Turtle	cold-blooded	no	yes	reptile
Salmon	cold-blooded	no	no	fish
Eel	cold-blooded	no	no	fish
Pigeon	warm-blooded	no	yes	bird
Penguin	warm-blooded	no	yes	bird

1. Does it give live birth?

Decision trees: basic idea

Name	Body temp	Gives birth	Legs	Class
Human	warm-blooded	yes	yes	mammal
Whale	warm-blooded	yes	no	mammal
Cat	warm-blooded	yes	yes	mammal
Cow	warm-blooded	yes	yes	mammal
Python	cold-blooded	no	no	reptile
Komodo dragon	cold-blooded	no	yes	reptile
Turtle	cold-blooded	no	yes	reptile
Salmon	cold-blooded	no	no	fish
Eel	cold-blooded	no	no	fish
Pigeon	warm-blooded	no	yes	bird
Penguin	warm-blooded	no	yes	bird

1. Does it give live birth? (No!)
2. Is it warm-blooded?

Decision trees: basic idea

Name	Body temp	Gives birth	Legs	Class
Human	warm-blooded	yes	yes	mammal
Whale	warm-blooded	yes	no	mammal
Cat	warm-blooded	yes	yes	mammal
Cow	warm-blooded	yes	yes	mammal
Python	cold-blooded	no	no	reptile
Komodo dragon	cold-blooded	no	yes	reptile
Turtle	cold-blooded	no	yes	reptile
Salmon	cold-blooded	no	no	fish
Eel	cold-blooded	no	no	fish
Pigeon	warm-blooded	no	yes	bird
Penguin	warm-blooded	no	yes	bird

1. Does it give live birth? (No!)
2. Is it warm-blooded? (No!)
3. Does it have legs?

Decision trees: basic idea

Name	Body temp	Gives birth	Legs	Class
Human	warm-blooded	yes	yes	mammal
Whale	warm-blooded	yes	no	mammal
Cat	warm-blooded	yes	yes	mammal
Cow	warm-blooded	yes	yes	mammal
Python	cold-blooded	no	no	reptile
Komodo dragon	cold-blooded	no	yes	reptile
Turtle	cold-blooded	no	yes	reptile
Salmon	cold-blooded	no	no	fish
Eel	cold-blooded	no	no	fish
Pigeon	warm-blooded	no	yes	bird
Penguin	warm-blooded	no	yes	bird

1. Does it give live birth? (No!)
2. Is it warm-blooded? (No!)
3. Does it have legs? (Yes!) → Reptile

Komodo dragon!

Regression trees: The regions of a tree

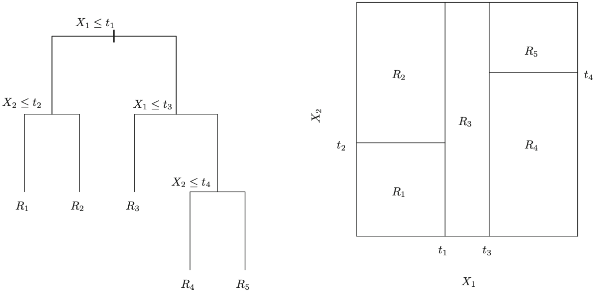

Figure: Regions of a tree (Garreth et al, 2013, Fig. 8.3)

Regression Trees

$$T(x) = \sum_{m=1}^{M} \gamma_m I(x \in R_m)\,,$$

where M is the total number of regions and $I(x \in R_m)$ is an indicator variable if x_i belongs to region R_m and γ_m is the prediction for region m.

Regression Trees

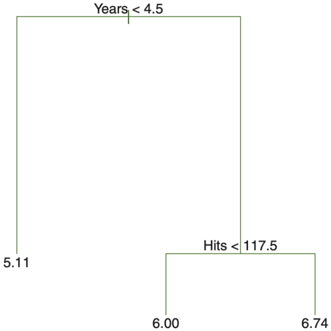

Figure: Regression Tree (Garreth et al, 2013, Fig. 8.1.)

- The `Hitters dataset`: log Salaries of Baseball players.
- The end of the tree contain the observations.

Regression Trees

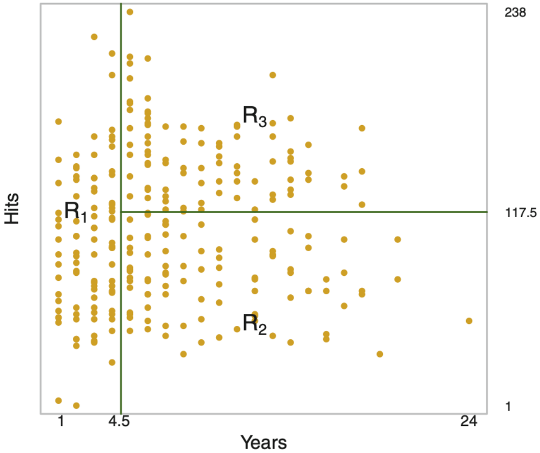

Figure: Hitters data and regression tree regions (Garreth et al, 2013, Fig. 8.2.)

Estimating a Decision Tree

1. A tree has two groups of parameters $\Theta = (\gamma, R)$ that we need to learn.

Estimating a Decision Tree

1. A tree has two groups of parameters $\Theta = (\gamma, R)$ that we need to learn.
2. We want a tree that minimize $L(\theta) = \sum_i^N (y_i - T_\Theta(x_i))^2$, here the squared loss

Estimating a Decision Tree

1. A tree has two groups of parameters $\Theta = (\gamma, R)$ that we need to learn.
2. We want a tree that minimize $L(\theta) = \sum_i^N (y_i - T_\Theta(x_i))^2$, here the squared loss
3. Usually we estimate γ_m as the mean of y_i in the region as:

$$\hat{\gamma}_m = \frac{1}{N_m} \sum_{x_i \in R_m}^{N_m} y_i \, ,$$

where N_m is the number of observations in region R_m.

Estimating a Decision Tree

1. A tree has two groups of parameters $\Theta = (\gamma, R)$ that we need to learn.
2. We want a tree that minimize $L(\theta) = \sum_i^N (y_i - T_\Theta(x_i))^2$, here the squared loss
3. Usually we estimate γ_m as the mean of y_i in the region as:

$$\hat{\gamma}_m = \frac{1}{N_m} \sum_{x_i \in R_m}^{N_m} y_i \, ,$$

   where N_m is the number of observations in region R_m.
4. Learning R_m exact is generally computationally infeasable so we use a greedy heuristic.

Growing a Decision Tree: Greedy Algorithm

Let $\mathcal{S}$ be the set of all observations $\{1, ..., N\}$ and `S[[m]]` be the set of observation indecies in R_m and `l` is the minimal number of leafs per node.

Input: $\mathcal{S}$, `X`, `y`, `l`

```
1. S[[1]] = $\mathcal{S}$,M = 1, m = 1
2. while m <= M then do:
    2.1 if(size(S[[m]]) >= 2*l)
        2.1.1 S[[M+1]], S[[M+2]], j[m], s[m] =
              split_tree(X[S[[m]],], y[S[[m]],],l)
        2.1.2 M = M + 2
    2.2 else
        2.2.1 compute $\hat{\gamma}$ for S[[m]]
    2.3 m = m + 1
```

Output: `j`, `s`, γ

Example of a tree: `j = {Years, Hits}`, `s = {4.5, 117.5}`, $\hat{\gamma} = \{122, 317, 245\}$

How to do a split?

Here we try to compute Eq. (9.12-9.14) in ESL:

$$R_1(j,s) = \{X | X_j \leq s\} \text{ and } R_2(j,s) = \{X | X_j > s\}$$

$$\min_{j,s} \left(\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right)$$

Inner minimization is solved by:

$$\hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m}^{N_m} y_i \,,$$

How to do a split? Pseudo-code

Input: $\mathbf{X}, \mathbf{y}, l$

```
1. $SS = \text{Inf}$ # Store sum of squares in matrix of dim $P \times N_s$
2. $S = \text{Inf}$ # Store split point in matrix of dim $P \times N_s$
3. for $j \in \{1, ..., P\}$ # all features
    3.1 for $k \in \{1, ..., N_s\}$ # all observations in set $s$
        3.1.1 $s = x_{k,j}$ # Split point (use the value of x)
        3.1.2 if $(|R_1(s,j)| < l$ or $|R_2(s,j)| < l)$ next # Dont create too few leaves
        3.1.3 $\hat{c}_1 = \frac{1}{|R_1(s,j)|} \sum_{x_i \in R_1(s,j)} y_i$
        3.1.4 $\hat{c}_2 = \frac{1}{|R_2(s,j)|} \sum_{x_i \in R_2(s,j)} y_i$
        3.1.5 $SS_{k,j} = \sum_{x_i \in R_1(s,j)} (y_i - c_1)^2 + \sum_{x_i \in R_2(s,j)} (y_i - c_2)^2$ # Compute Sum of Squares
        3.1.6 $S_{k,j} = \text{s}$
4. $k_{final}, j_{final} = \min_{k,j} SS$
5. $s_{final} = S_{k_{final}, j_{final}}$
6. return $R_1(s_{final}, j_{final}), R_2(s_{final}, j_{final}), s_{final}, j_{final}$
```

Decision trees: Classification Trees

- How do we do if we have a classification tree?

Decision trees: Classification Trees

- How do we do if we have a classification tree?
- We just change the loss function $L(\theta)$.

Decision trees: Classification Trees

- How do we do if we have a classification tree?
- We just change the loss function $L(\theta)$.
- Let $p(j|t)$ be the fraction of observations in class j at the node t and let J be the number of classes.

Decision trees: Classification Trees

- How do we do if we have a classification tree?
- We just change the loss function $L(\theta)$.
- Let $p(j|t)$ be the fraction of observations in class j at the node t and let J be the number of classes.
- The Gini for node t is defined as

$$Gini(t) = 1 - \sum_{j=1}^{J} p(j|t)^2$$

Decision trees: Classification Trees

- How do we do if we have a classification tree?
- We just change the loss function $L(\theta)$.
- Let $p(j|t)$ be the fraction of observations in class j at the node t and let J be the number of classes.
- The Gini for node t is defined as

$$Gini(t) = 1 - \sum_{j=1}^{J} p(j|t)^2$$

- Gini is a measure of "impurity". If all observations belong to the same class, then

$$Gini(t) = 1 - 1^2 - 0 - \ldots - 0 = 0.$$

Decision trees: Classification Trees

- How do we do if we have a classification tree?
- We just change the loss function $L(\theta)$.
- Let $p(j|t)$ be the fraction of observations in class j at the node t and let J be the number of classes.
- The Gini for node t is defined as

$$Gini(t) = 1 - \sum_{j=1}^{J} p(j|t)^2$$

- Gini is a measure of "impurity". If all observations belong to the same class, then

$$Gini(t) = 1 - 1^2 - 0 - \ldots - 0 = 0.$$

- The Gini is maximized when all classes have the same number of observations at t.

Decision trees: Classification Trees

- How do we do if we have a classification tree?
- We just change the loss function $L(\theta)$.
- Let $p(j|t)$ be the fraction of observations in class j at the node t and let J be the number of classes.
- The Gini for node t is defined as

$$Gini(t) = 1 - \sum_{j=1}^{J} p(j|t)^2$$

- Gini is a measure of "impurity". If all observations belong to the same class, then

$$Gini(t) = 1 - 1^2 - 0 - \ldots - 0 = 0.$$

- The Gini is maximized when all classes have the same number of observations at t.
- One criterion for splitting could be to minimize the Gini in the next level of the tree. That way we will get "purer" nodes.

CART

- The algorithm is a simplified "teaching" version of the Classification and Regression Trees algorithm

CART

- The algorithm is a simplified "teaching" version of the Classification and Regression Trees algorithm
- CART:
  - Uses the same greedy splitting idea.

CART

- The algorithm is a simplified ”teaching” version of the Classification and Regression Trees algorithm
- CART:
  - Uses the same greedy splitting idea.
  - Splits are chosen using impurity/loss reduction (e.g., Gini, entropy, SSE).

CART

- The algorithm is a simplified ”teaching” version of the Classification and Regression Trees algorithm
- CART:
  - Uses the same greedy splitting idea.
  - Splits are chosen using impurity/loss reduction (e.g., Gini, entropy, SSE).
  - Tree growth is usually depth-first.

CART

- The algorithm is a simplified "teaching" version of the Classification and Regression Trees algorithm
- CART:
  - Uses the same greedy splitting idea.
  - Splits are chosen using impurity/loss reduction (e.g., Gini, entropy, SSE).
  - Tree growth is usually depth-first.
  - Output is a recursive binary tree object rather than the 'j, s, y' lists.

CART

- The algorithm is a simplified "teaching" version of the Classification and Regression Trees algorithm
- CART:
  - Uses the same greedy splitting idea.
  - Splits are chosen using impurity/loss reduction (e.g., Gini, entropy, SSE).
  - Tree growth is usually depth-first.
  - Output is a recursive binary tree object rather than the 'j, s, y' lists.
  - CART includes cost-complexity pruning

Cost-Complexity Pruning

Goal: Reduce overfitting by shrinking a large tree into a simpler subtree.

For any subtree T, define:

- Training error:

$$R(T) = \sum_{m=1}^{|T|} \sum_{i \in R_m} (y_i - \hat{\gamma}_m)^2$$

- Tree size: $|T|$ = number of terminal/leaf nodes/regions
- Full loss:

$$R_\alpha(T) = R(T) + \alpha |T|$$

$\alpha \geq 0$ controls the penalty:

- $\alpha = 0$: prefers a large tree (overfitting)
- large α: prefers a small tree (underfitting)

Weakest-Link Pruning Algorithm

1. Grow a large tree T_0.
2. For each internal node, compute the increase in $R_\alpha(T)$ caused by pruning its children.
3. Prune the branch that increases $R_\alpha(T)$ the least.
4. Repeat to produce a sequence:

$$T_0 \supset T_1 \supset T_2 \supset \cdots \supset T_K$$

5. Use cross-validation to choose the optimal α and corresponding subtree T.

Important concepts in trees

- Tree depth: the length of the longest path from the root to a leaf (i.e. greatest number of questions that the tree can ask).

Important concepts in trees

- Tree depth: the length of the longest path from the root to a leaf (i.e. greatest number of questions that the tree can ask).
- Leaf size: the number of observations in a leaf.

Important concepts in trees

- Tree depth: the length of the longest path from the root to a leaf (i.e. greatest number of questions that the tree can ask).
- Leaf size: the number of observations in a leaf.

Decision trees can become quite large, which may lead to:

- Overfitting (high variance)
- Difficulties interpreting the tree

Important concepts in trees

- Tree depth: the length of the longest path from the root to a leaf (i.e. greatest number of questions that the tree can ask).
- Leaf size: the number of observations in a leaf.

Decision trees can become quite large, which may lead to:

- Overfitting (high variance)
- Difficulties interpreting the tree

The solution to this is

- Pruning: forcing the tree to be smaller by adding a stopping condition, e.g. a maximum depth or minimal leaf size.

Important concepts in trees

- Tree depth: the length of the longest path from the root to a leaf (i.e. greatest number of questions that the tree can ask).
- Leaf size: the number of observations in a leaf.

Decision trees can become quite large, which may lead to:

- Overfitting (high variance)
- Difficulties interpreting the tree

The solution to this is

- Pruning: forcing the tree to be smaller by adding a stopping condition, e.g. a maximum depth or minimal leaf size.
- But decision trees are quite bad predition models...

Section 2

Ensemble methods

Wisdom of the crowd

Simulated example (Prize academy, see ESL):

1. 50 members vote in 10 categories, each with 4 nominations.

Wisdom of the crowd

Simulated example (Prize academy, see ESL):

1. 50 members vote in 10 categories, each with 4 nominations.
2. For any category, only 15 voters have some knowledge ($p > 0.25$)

Wisdom of the crowd

Simulated example (Prize academy, see ESL):

1. 50 members vote in 10 categories, each with 4 nominations.
2. For any category, only 15 voters have some knowledge ($p > 0.25$)
3. For each category, the 15 experts are chosen at random from the 50.

'Individual' = a single weak learner/member

'Consensus' = the aggregated majority vote of many learners/members

Wisdom of the crowd

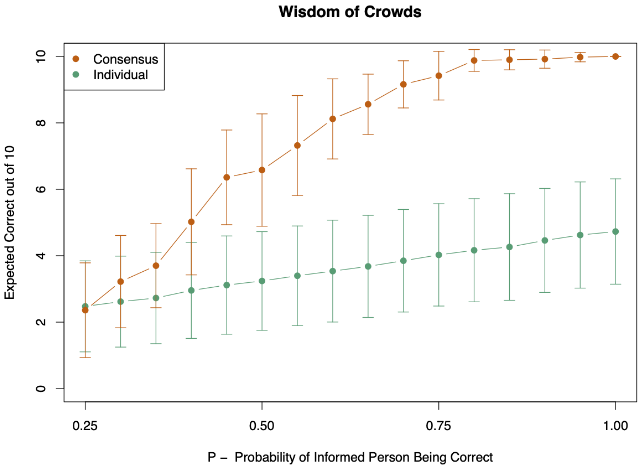

Figure: Simulated Award Voting, Fig. 8.11 (ESL)

General idea of ensembles

The idea of an ensemble is simple: If it difficult to find one really good model perhaps we can find several weaker models and combine their predictions.

General idea of ensembles

The idea of an ensemble is simple: If it difficult to find one really good model perhaps we can find several weaker models and combine their predictions.

A simple example: Say you have one outcome Y and 4 covariates X_1, X_2, X_3, X_4. The goal is to predict Y. A possible ensemble would be to fit

$$\begin{aligned} y &= \alpha_1 + \beta_1 X_1 + \epsilon_1 \\ y &= \alpha_2 + \beta_2 X_2 + \epsilon_2 \\ y &= \alpha_3 + \beta_3 X_3 + \epsilon_3 \\ y &= \alpha_4 + \beta_4 X_4 + \epsilon_4 \end{aligned}$$

and then use the mean of their predictions

$$\hat{y}_{ensemble} = \frac{1}{4}\sum_{i=1}^{4}\hat{y} = \frac{1}{4}\sum_{i=1}^{4}(\hat{\alpha}_i + \hat{\beta}_i X_i) \tag{1}$$

Two key parts of an ensemble

1. The prediction models (sometimes called 'learners')
   - A single model in an ensemble can be a simple or a complex model
   - Often the ensemble contains many simple models.

Two key parts of an ensemble

1. The prediction models (sometimes called ‘learners’)
   - A single model in an ensemble can be a simple or a complex model
   - Often the ensemble contains many simple models.
2. The weighting of each prediction in the final ensemble prediction
   - Many different algorithms for weighting together the predictions from many models
   - Models/learners with better predictive power can be given larger weights in the final prediction

Ensambles of decision trees

- A common type of ensembles is ensembles of decision trees.
- We will focus on this case, but note that any type of model can be included in an ensemble in principle.

Bagging and Boosting

Remember, the error of a prediction/classification can be decomposed as

$$error = bias^2 + variance + bayeserror. \tag{2}$$

Bagging and Boosting

Remember, the error of a prediction/classification can be decomposed as

$$error = bias^2 + variance + bayeserror. \tag{2}$$

- Complex models/strong learners (with many parameters) tend to have small bias and large variance (tend to be overfitted)

Bagging and Boosting

Remember, the error of a prediction/classification can be decomposed as

$$error = bias^2 + variance + bayeserror. \qquad (2)$$

- Complex models/strong learners (with many parameters) tend to have small bias and large variance (tend to be overfitted)
- Shallow models/weak learners (with few parameters) tend to have small variance and large bias

Bagging and Boosting

Remember, the error of a prediction/classification can be decomposed as

$$\mathit{error} = \mathit{bias}^2 + \mathit{variance} + \mathit{bayeserror}. \qquad (2)$$

- Complex models/strong learners (with many parameters) tend to have small bias and large variance (tend to be overfitted)
- Shallow models/weak learners (with few parameters) tend to have small variance and large bias

Bagging: Ensemble methods that aim to decrease the variance of complex/strong learners with low bias and large variance

Boosting: Ensemble methods that aim to decrease the bias of shallow/weak learners with low variance and large bias

Subsection 1

Bagging

Bagging (Bootstrap AGGregating)

- Train several deep trees and combine their results
- Use bootstrap to train different trees

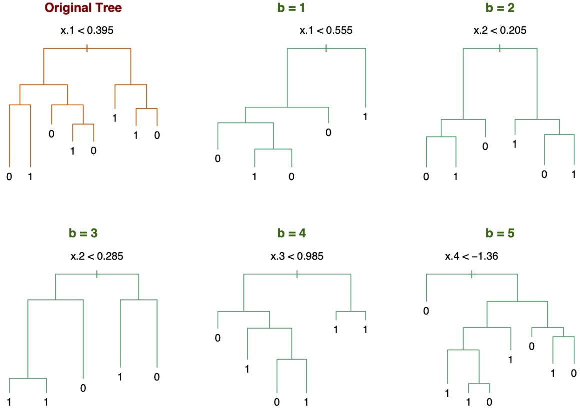

Figure: Bagging trees, Fig. 8.9 (ESL)

Bagging (Bootstrap AGGregating)

1. Draw, *with replacement*, a random sample of N units from the original sample
2. Fit a prediction model (e.g., a deep decision tree)
3. Repeat steps 1-2 B times
4. Weight together the predictions from the B models into a final ensemble prediction as

$$\hat{f}_{\text{bag}}(x_i) = \frac{1}{B}\sum_b \hat{f}^b(x_i) = \frac{1}{B}\sum_b \hat{T}(x_i|\Theta_b)$$

Subsection 2

Random forests

Random Forest

- A bagging ensemble method, but...

Random Forest

- A bagging ensemble method, but...
- Sample of N observations and K covariates/features

Random Forest

- A bagging ensemble method, but...
- Sample of N observations and K covariates/features
- It is common to use $k = \sqrt{K}$ (rounded down) for classification and $k = K/3$ for regression. But these are only rules of thumb: k is a *tuning parameter*.

Random Forest: Algorithm

1. Draw, *with replacement*, a random sample of N units from the original sample
2. **Draw, without replacement, a random subset of k covariates/features**
3. Fit a prediction model (e.g., a decision tree)
4. Repeat step 1-3 B times
5. Weight together the predictions from the B models into a final ensemble prediction

$$\hat{f}_{\text{rf}}(x_i) = \frac{1}{B}\sum_{b} \hat{T}(x_i|\Theta_b)$$

Random Forest variance reduction

- Consider each tree to be an i.i.d. random variable with variance σ^2.
- The average of these trees then have variance

$$\mathbb{V}(\hat{f}(x) = \frac{1}{B}\sigma^2.$$

Random Forest variance reduction

- Consider each tree to be an i.i.d. random variable with variance σ^2.
- The average of these trees then have variance

$$\mathbb{V}(\hat{f}(x) = \frac{1}{B}\sigma^2.$$

- Trees constructed from the same set of covariates will be correlated and therefore not independent.
- The variance of the average of these correlated trees then becomes

$$\mathbb{V}(\hat{f}(x)) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2.$$

UPPSALA
UNIVERSITET

Random Forest variance reduction

- The variance of the average of correlated trees:

$$\mathbb{V}(\hat{f}(x)) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2.$$

- The second term will vanish with increasing B leaving just the first term left: a function of the correlation between the trees

Random Forest variance reduction

- The variance of the average of correlated trees:

$$\mathbb{V}(\hat{f}(x)) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2.$$

- The second term will vanish with increasing B leaving just the first term left: a function of the correlation between the trees
- The remaining part of the variance is minimized by only consider a subset of the covariates when constructing trees - reducing the correlation between trees.

Bagging vs. Random forest

- In bagging, the trees are often highly correlated
  - If some covariates are strong predictors of the outcome (in the training data), many trees in the ‘bag’ will us the same covariates in their decisions

Bagging vs. Random forest

- In bagging, the trees are often highly correlated
  - If some covariates are strong predictors of the outcome (in the training data), many trees in the 'bag' will us the same covariates in their decisions
- In a random forest, the trees are less similar/correlated since all covariates are not available when each tree is constructed.

Bagging vs. Random forest

- In bagging, the trees are often highly correlated
  - If some covariates are strong predictors of the outcome (in the training data), many trees in the 'bag' will us the same covariates in their decisions
- In a random forest, the trees are less similar/correlated since all covariates are not available when each tree is constructed.
- Intuition: A random forest (with many trees) uses the predictive ability of all covariates rather than just a few → improved out of sample performance.

Subsection 3

(Gradient) Boosting

Boosting

1. In boosting, models/trees are trained sequentially

Boosting

1. In boosting, models/trees are trained sequentially
2. Each new model tries to target weak spots of the previous models

Boosting (AdaBoost)

1. Initialize weight for all observations as $w_i = n^{-1}$
2. Repeat B times
   - 2.1 Fit a simple classifier $h_b(x)$ using w_is
   - 2.2 Compute the weighted error (between 0-1) as

$$e_b = \frac{\sum w_i I(y_i \neq h_b(x_i))}{\sum w_i}$$

   - 2.3 Compute the importance of the classifier as

$$\alpha_b = \log((1 - e_b)/e_b) = \text{logit}(1 - e_b)$$

   - 2.4 Add the classifier to the ensemble

$$\hat{f}_b(x) = \hat{f}_{b-1}(x) + \alpha_b h_b(x)$$

   - 2.5 Update weights so that badly classified observations is weighted more

$$w_i \leftarrow w_i \exp(\alpha_b I(y_i \neq h_b(x_i)))$$

3. Output the final ensemble $\hat{f}_B(x)$

Boosting example

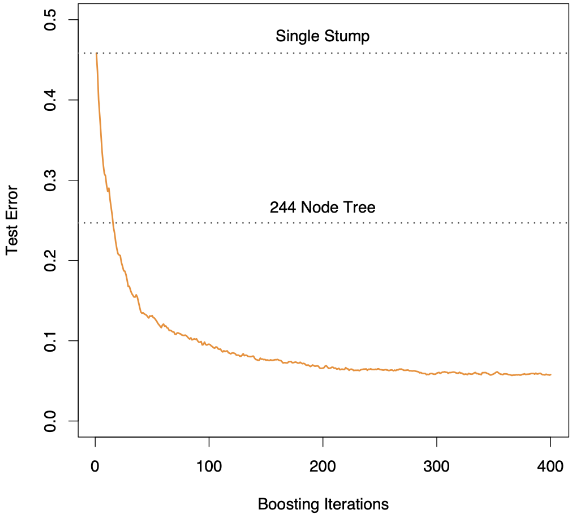

Figure: Boosting example (using stumps as h_b), Fig. 10.2 (ESL)

Boosting trees

- A more general approach is (gradient) boosting trees

Boosting trees

- A more general approach is (gradient) boosting trees
- Let

$$f_{\text{gb}}(x) = \sum_{b}^{B} \hat{T}(x|\Theta_b)$$

Boosting trees

- A more general approach is (gradient) boosting trees
- Let

$$f_{\text{gb}}(x) = \sum_{b}^{B} \hat{T}(x|\Theta_b)$$

- We want to minimize the loss

$$L(y, f_{\text{gb}}(x))$$

Boosting trees

- A more general approach is (gradient) boosting trees
- Let

$$f_{\text{gb}}(x) = \sum_{b}^{B} \hat{T}(x|\Theta_b)$$

- We want to minimize the loss

$$L(y, f_{\text{gb}}(x))$$

- This means finding

$$\Theta_b = \arg\min \sum_{i}^{N} L(y_i, f_{b-1}(x_i) + \hat{T}(x|\Theta_b))$$

Gradient Boosting Trees

1. Initialize $f_0(x)$
2. Repeat B times
   - 2.1 For $i = 1, 2, ..., N$ compute

     $$r_{ib} = -\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}$$

     (its the residuls using a squared loss function)
   - 2.2 Compute a regression tree $\hat{T}(x|\Theta_b)$ on r_{bi}s
   - 2.3 Add the classifier to the ensemble

     $$\hat{f}_b(x) = \hat{f}_{b-1}(x) + \hat{T}(x|\Theta_b)$$
3. Output the final ensemble $\hat{f}_B(x)$

Comparisons

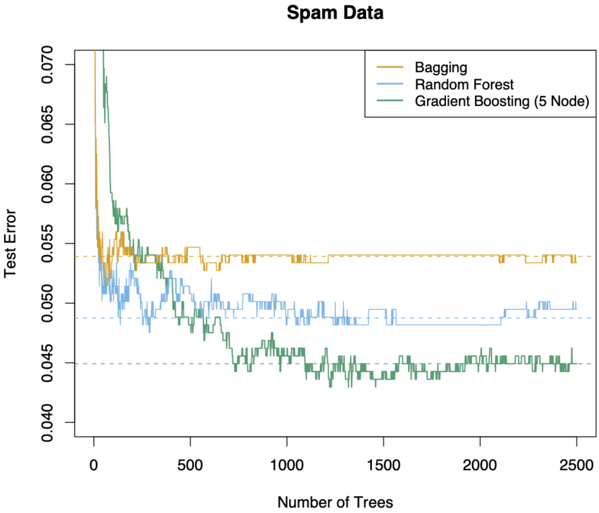

Figure: Comparing bagging, random forests and boosting, Fig. 15.1 (ESL)

XGBoost

- State of the Art method

XGBoost

- State of the Art method
- Use gradient boosting trees with regularization

$$L(y, f_{\text{boost}}(x)) + \sum_b \Omega(\hat{T}(x_i|\Theta_b))$$

XGBoost

- State of the Art method
- Use gradient boosting trees with regularization

$$L(y, f_{\text{boost}}(x)) + \sum_b \Omega(\hat{T}(x_i|\Theta_b))$$

where

$$\Omega(\hat{T}(x_i|\Theta_b)) = \lambda_1 Q_b + \lambda_2 ||\gamma_b||_2^2$$

where Q_b is the number of leafs in tree b and $||\cdot||_2$ is the Euclidian/L^2 norm.

XGBoost

- State of the Art method
- Use gradient boosting trees with regularization

$$L(y, f_{\text{boost}}(x)) + \sum_b \Omega(\hat{T}(x_i|\Theta_b))$$

where

$$\Omega(\hat{T}(x_i|\Theta_b)) = \lambda_1 Q_b + \lambda_2 ||\gamma_b||_2^2$$

where Q_b is the number of leafs in tree b and $||\cdot||_2$ is the Euclidian/L^2 norm.

- Is scalable to very large data