UNIVERSITY OF ARKANSAS

DASC 4113 Machine Learning

Lecture 8

Ukash Nakarmi

Tree Based Methods

UNIVERSITY OF ARKANSAS

Learning Objectives

In this class, we will learn:

- A different approach, based on decision trees, to do regression and classification.
- How decision trees are built using binary splitting.
- How to use ensemble techniques to improve tree-based techniques.

Decision Tree for Regression

Informal Example:

Linear Regression (Global)
Piecewise Linear Regression (Some sense of Locality, but estimate parameters for each region)
Main Idea in Regression Trees , Difference with Linear and Piecewise Linear()

UNIVERSITY OF ARKANSAS

Decision Tree for Regression

Informal Example:

UNIVERSITY OF ARKANSAS

Decision Tree for Regression

Informal Example:

UNIVERSITY OF ARKANSAS

Decision Tree for Regression

Informal Example:

UNIVERSITY OF ARKANSAS

Decision Tree for Regression

Informal Example:

Decision Tree: The Basics

- Split Predictors into different regions using some set of rules.
- Use tree to represent regions and rules.
- Use mean/mode of each region of training data to make prediction of samples that fall in that region.

Unlike Linear/Polynomial Regression which are global model, Tree based models divide(partition) predictors space into small regions.

Regression Trees

Example:

Hitters Dataset:

Data Set with some predictors such as X = Years, Hits and Y = Salary of baseball players.

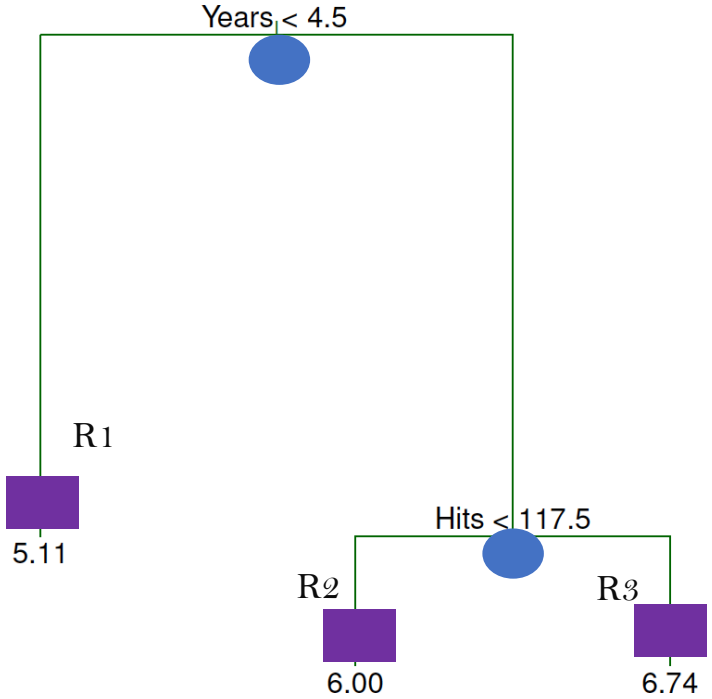


Fig 8.1, Regression Tree Example

Leaves

Internal Nodes

Rj jth Region

5.11
6.00
6.74

Salaries in log scale for players in each predictor region

Example: Log(salary) = 5.11

Regression Trees

Questions:

How do we build regression trees?

What are the rules?

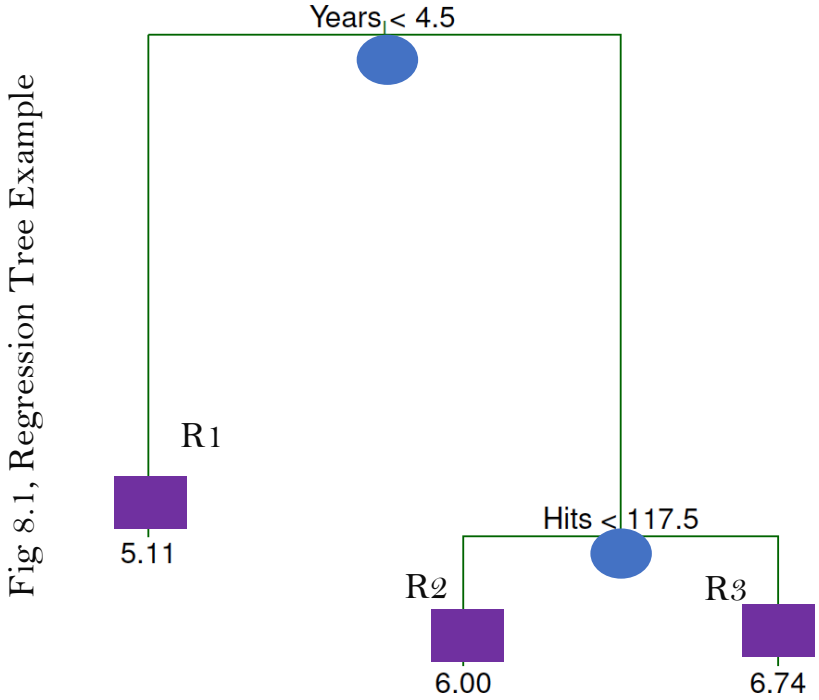


Fig 8.1, Regression Tree Example

Building Regression Trees

Building regression trees broadly involves two tasks:

1. Divide/partition predictor space

   Divide predictor space (range of input values) of each predictor in J distinct non-overlapping regions $R_1, R_2, \ldots R_J$

2. Predict

   For every observation $x_i \in R_j$, we make the same prediction (mean of responses in R_j)

Rules/Criterion: Build Regions such that following error term in minimized

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

j: Index for region

$\hat{y}_{R_j}$: mean response of the j^{th} region

y_i : True response for ith sample

Building Regression Trees

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

Q. How do we build regions that satisfy(minimizes) this rule?

We use:

1. Recursive Binary Tree

At every stage we partition predictor space into two regions using Greedy Approach

Greedy Approach

At each step:

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

1. Build Two regions using Greedy Approach

i.e., Among all predictors $X_1, X_2, \ldots X_p$ and among all cut points that divide the space into two regions find best cut point s such that error at that point is minimized.

$$\sum_{i:\, x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2,$$

Error Associated with Region 1

Error Associated with Region 2

2. Keep doing it until each terminal node has no more than certain numbers of samples

Regression Trees

Are we done?

We could stop here, but it has two issues:

1. The cost of building such tree could be large as numbers of predictors increases. (*We could still use recursion to build efficient trees.* not discussed in this lecture)
2. Such a tree could overfit leading much variance in the testing data.

To address this issue:

We do Pruning:

Add number of leaves as a penalty on our loss

Tradeoff between complexity(error) and numbers of leaves

Regression Trees

Are we done?

We could stop here, but :

1. Such a tree could overfit, leading to much variance in the testing data.

To address this issue:

We do Pruning:

Add number of leaves as a penalty on our loss

Tradeoff between complexity(error) and numbers of leaves

Cost Complexity Pruning/ Weakest Link Pruning

Once we have formed a tree with J Regions (J leaf nodes) using the greedy method, pruning allows us to choose a subtree that do not overfit.

To do so, we introduce a penalty term in the cost

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

|T| : Number of leaf nodes in the subtree

Note: Now for a tree with same number of leaves, cost increases as α increases.
So, when we try to minimize the cost, we are also looking for tree with not many leaves.

Cost Complexity Pruning/ Weakest Link Pruning

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

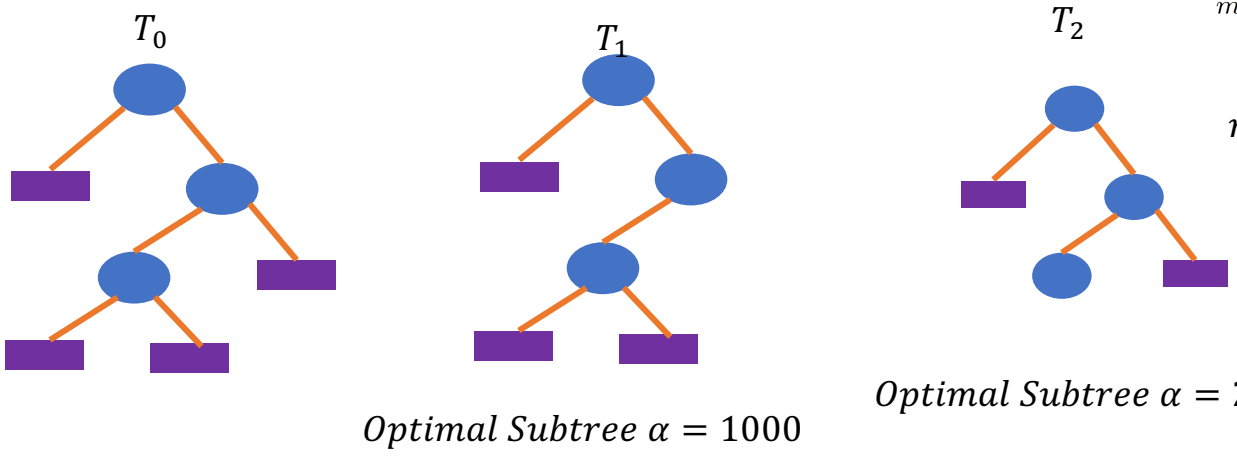
- Choose Subtree with minimum cost for different values of α. We call such subtree as optimal subtree.

Note: For each of different values of alpha, there exists a subtree which has minimum cost.

Cost Complexity Pruning/ Weakest Link Pruning

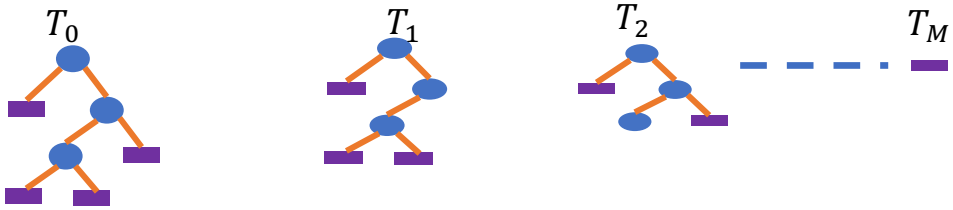
Example: Binary Tree with J = 4 (Regions) using Greedy method

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



- Optimal subtree is a subtree for which the cost is minimum for that particular value of alpha.
- As alpha increases, numbers of leaves decreases, eventually leading to a tree where all samples are in one region. (Case where we predict response for all samples as mean of observations.)

K-fold Cross Validation



- Let's say, we did for 10 different values of alpha giving us 10 optimal subtrees (models) one corresponding to each value of alpha.
- For each optimal subtree, do K-fold cross validation.

- Choose a model (optimal subtree) with minimum cross validation error.

Regression Tree: Summary

Algorithm 8.1 *Building a Regression Tree*

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. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:

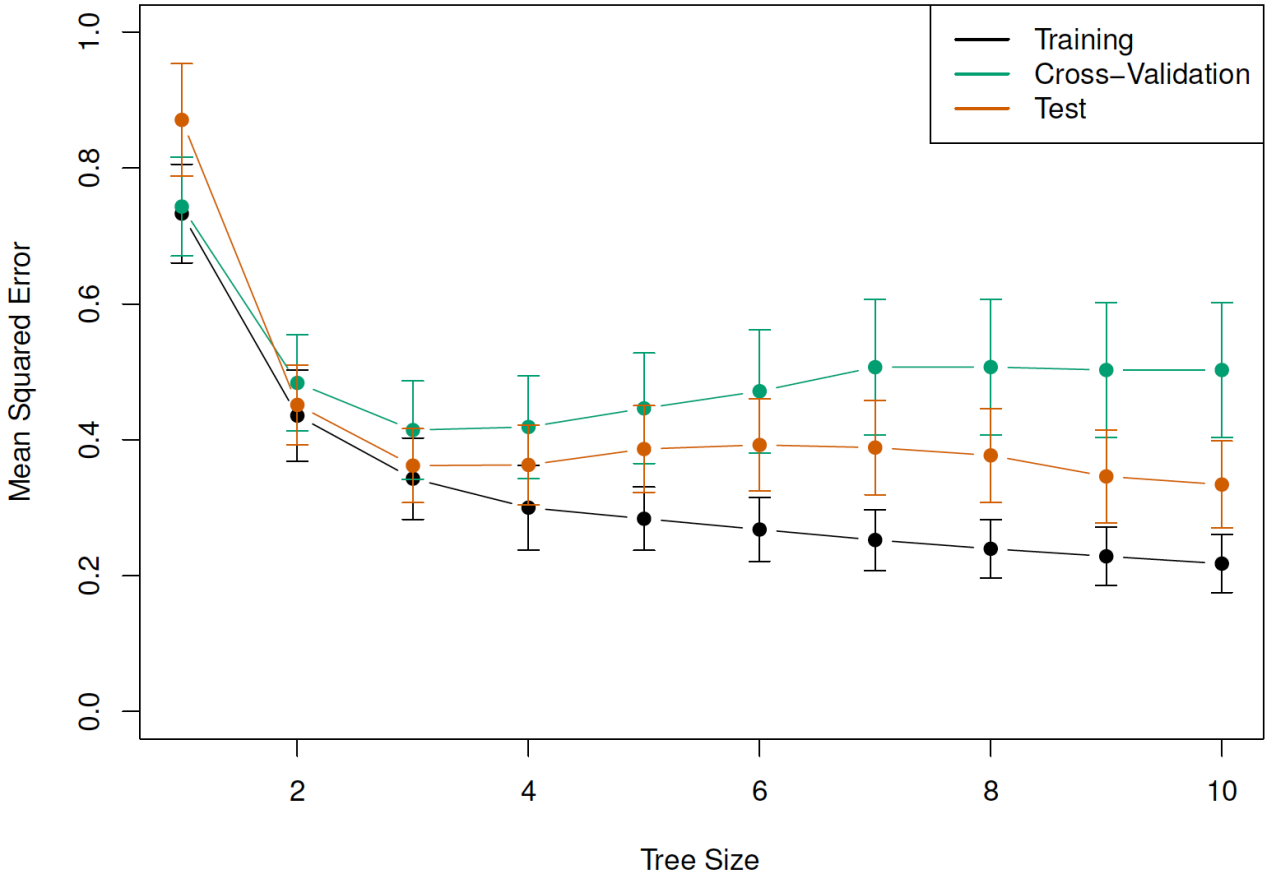
   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

   Average the results for each value of α, and pick α to minimize the average error.

4. Return the subtree from Step 2 that corresponds to the chosen value of α.

Regression Tree: Hitters Data Result



Classification Trees

All Steps are similar except:

Definition of Error is different (No Longer RSS)

Classification Trees

All Steps are similar except:

Definition of Error is different (No Longer RSS)

Classification Trees

Prediction:

- Instead of using mean of the region ($\hat{y}_j$) to predict the response for samples in region R_j, in Classification Trees, we predict that each observation belongs to the most commonly occurring class in that region.

Criterion: (Was RSS in Regression Trees)

- Classification Error Rate: Fraction of training observation in region R_j that do not belong most most commonly occurring class in that region.

$$E = 1 - \max_k(\hat{p}_{mk})$$

$\hat{p}_{mk}$: Proportion of training observations in the m^{th} region from class k.

- We assign observations as class k for which $\hat{p}_{mk}$ is maximum.
- Error = 1 - proportion of class with maximum samples.

Classification Trees

Modified Criterion:

Classification Error Rate is not very sensitive to growth of tree. Two other alternative criterion:

1. The Gini Index:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

2. The Entropy:

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

Note: -ve because $0 \leq \hat{p}_{mk} \leq 1$. $\hat{p}_{mk}$ is a proportion of numbers of samples in class k to total numbers of samples, $log\hat{p}_{mk}$ -> negative value. We want to represent entropy as a positive quantity.

Regression and Classification Trees

- Qualitative Predictors can be used directly.
- Do not need to create dummy variables.

GINI and Entropy Measures the Purity of the Node

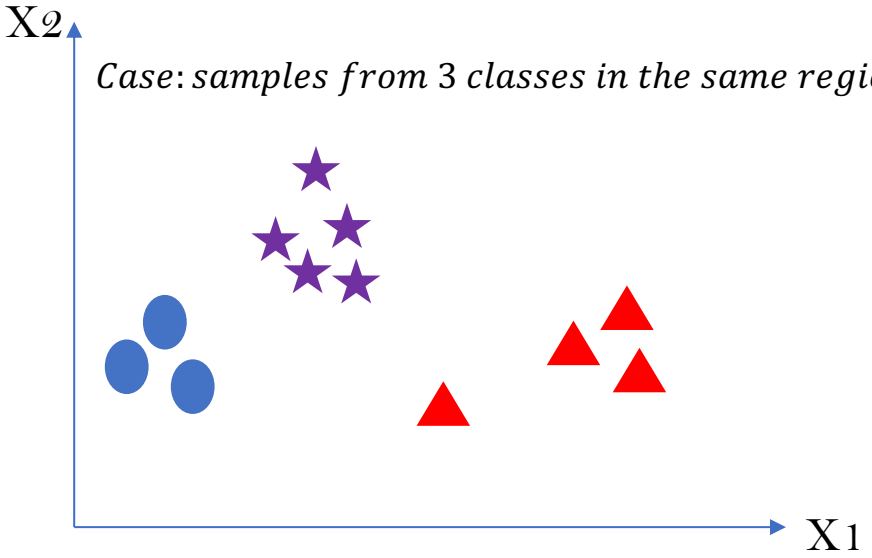
$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

$\hat{p}_{mk}$: Proportion of training observations in the m^{th} region from class k.

GINI and Entropy Measures the Purity of the Node

Example: Cancer, Early-Stage Cancer, Normal Classification

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$



Samples in Region m in a Decision Tree

X1, X2: Two Inputs/Features

Normal Samples k = 1

Early-stage Cancer samples, k = 2

Cancer samples, k = 3

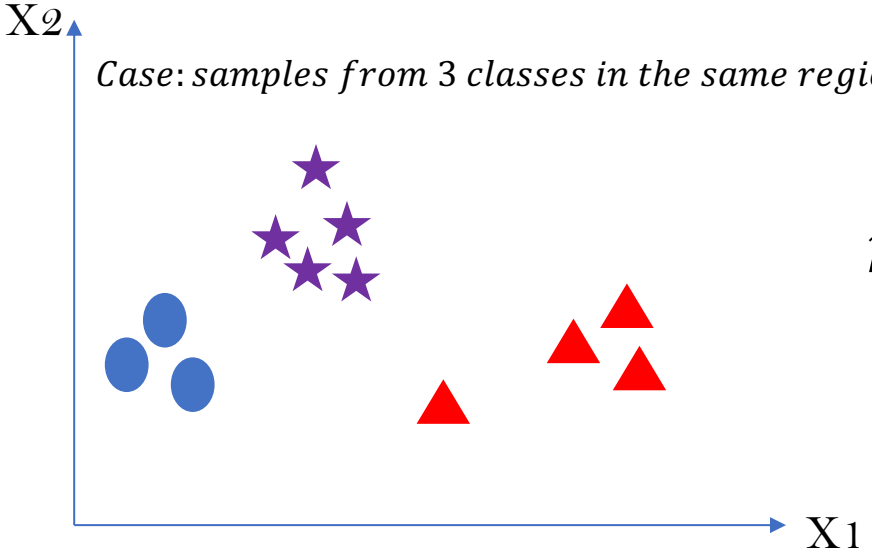
K = 3

Total Samples in the region = 12

GINI and Entropy Measures the Purity of the Node

Example: Cancer, Early-Stage Cancer, Normal Classification

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$



$$\hat{p}_{m1} = \frac{3}{12}, \quad \hat{p}_{m2} = \frac{5}{12}, \quad \hat{p}_{m3} = \frac{4}{12}$$

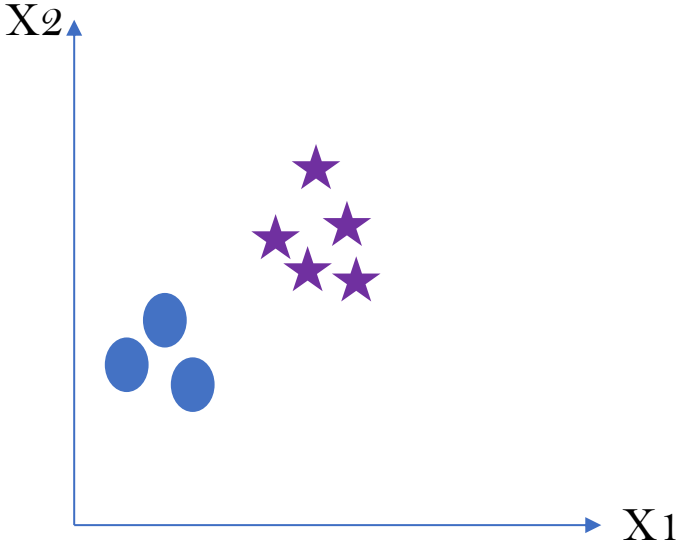
Samples in Region m in a Decision Tree

$$G1 = \frac{3}{12}\left(1 - \frac{3}{12}\right) + \frac{5}{12}\left(1 - \frac{5}{12}\right) + \frac{4}{12}\left(1 - \frac{4}{12}\right) = 0.6597$$

GINI and Entropy Measures the Purity of the Node

Example: Cancer, Early-Stage Cancer, Normal Classification

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$



K = 2

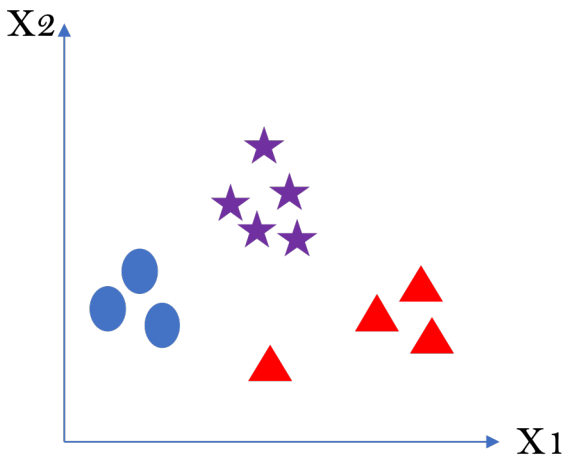
Total Samples in the region = 8

$$\hat{p}_{m1} = \frac{3}{8}, \quad \hat{p}_{m2} = \frac{5}{8}$$

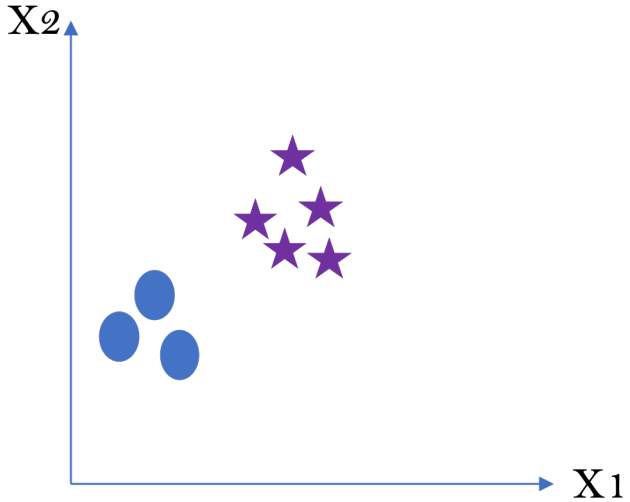
Samples in Region m in a Decision Tree

$$G2 = \frac{3}{7}\left(1 - \frac{3}{8}\right) + \frac{5}{8}\left(1 - \frac{5}{8}\right) = 0.3772$$

GINI and Entropy Measures the Purity of the Node



$G1 = 0.6597$



$G2 = 0.3772$

$G2 < G1$.

Region with Lower the GINI Index => has samples from fewer different classes. (Pure)

- Hence, smaller is the error.
- Minimizing GINI Index makes sense.

GINI and Entropy Measures the Purity of the Node

$Task$: $Calculate\ Entropy\ for\ two\ cases\ and\ comapre$:

Ensemble Methods

- Bagging -> Bootstrapping
- Random Forest -> Randomness: $m \sim \sqrt{P}$
- Boosting -> Residuals
- Bayesian Additive regression Trees -> Randomness and Residuals

Bagging (Bootstrap Aggregation)

- Is a general-purpose method to reduce variance of statistical models.

(Recall: By variance of a model, we mean how much the model would change if we have some change in the training data.)

- Uses Bootstrapping(resample with repetition) to create multiple data sets.
- Helps to minimize the variance of Decision Trees.

- General Idea:

Given a set of n – independent observations each with variance of σ^2:

The variance of the mean of the observations is given by σ^2/n. (Refer to Variance Sum law)

i.e. Averaging set of observation reduces variance.

Bagging (Bootstrap Aggregation)

- Generate B different training data set using Bootstrapping.
- For each data set, train a model (Decision Tree).

  i.e., For b^{th} bootstrapped training data set, get the trained model:

$$\hat{f}^{*b}(x)$$

- Then, Bagging Prediction is given by :

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

Out-of-Bag Error Estimation

- Recall: In Bagging, We build tree for each B bootstrapped dataset.

- In bootstrapping, each bagged tree make use of about $2/3^{rd}$ of the n samples (observations).
- The remaining $1/3$ of samples not used by the b^{th} bag are called Out of Bag Samples.
- Sample x_i is not used in $B/3$ Decision Trees.
- We use those $B/3$ Decision Trees that do not use ith sample to make prediction for x_i

  If It's a regression problem: We use average of $B/3$ predictions.

  If It's a Classification problem : We use majority vote of $B/3$ trees to predict output class.

  Eliminates Necessity of cross-validation

Out-of-Bag Error Estimation

- In bootstrapping, each bagged tree make use of about $2/3^{rd}$ of the n samples (observations).
- The remaining $1/3$ of samples not used by the b^{th} bag are called Out of Bag Samples.

Illustration:

Random Forest

- Similar to Bagging, uses bootstrapping to create B training datasets.

- But at split point while building tree, instead of using all p input variables:
- Randomly chooses m< p variables.
- Make a Split based in those m variables.
- Typically, $m \sim \sqrt{p}$

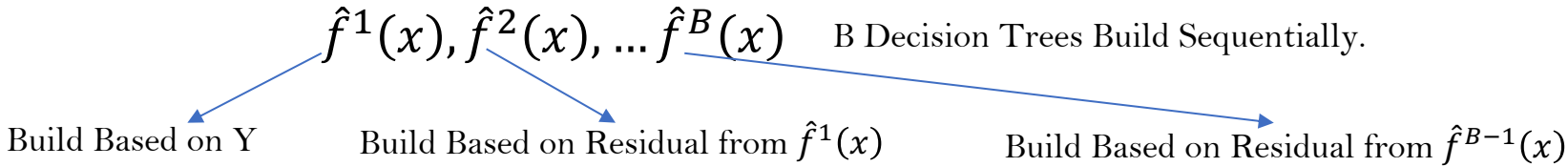
- Helps to decorrelate input variables.

Boosting

- Does not use bootstrapping.
- Unlike in Bagging, where we build B trees corresponding to B bootstrapped data, we build B decision trees sequentially.
- Each of these trees could be rather small (d numbers of leaves)
- Final Decision tree is sum of all these B decision trees.

- Creates a new tree that fits the residual from the current model.

i.e., Build a tree based on Residual (Error from the current model) not based on the original response Y.



Final Tree is Weighted Sum of all B Trees

Boosting

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,

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

Bayesian Additive Regression Trees (BART)

Note: All these methods Bagging, Random Forest, Boosting differs from each other in a way the Decision Trees are Generated.

BART uses different approach to generate trees, which is somehow mixture of both random forest (only m random inputs) and Boosting (Residual Approach).

Some Notations and Definitions:

K: Numbers of Regression Trees

B: Numbers of Iterations

$\hat{f}_k^b(x)$: Prediction for input x using k^{th} Regression Tree at b^{th} Iteration.

$\hat{f}^b(x) = \sum_{k=1}^{k} \hat{f}_k^b(x), b = 1, 2, \ldots B$: Sum of all K Trees at b^{th} Iteration.

Residual: In the bth Iteration residual for the kth Tree

Bayesian Additive Regression Trees (BART)

Initialization:

At First Iteration: All Trees only have a single node with:

$$\hat{f}_k^1(x) = \frac{1}{nK}\sum_{i=1}^{n} y_i$$ Mean of Responses/Number of Trees

And

$$\hat{f}^1(x) = \sum_{k=1}^{K} \hat{f}_k^1(x) = \frac{1}{n}\sum_{i=1}^{n} y_i$$

First Iteration Predicts output of any input to be the mean of response
(Not a good Prediction)
We improve this at each iteration.

Bayesian Additive Regression Trees (BART)

Partial Residual:

In the b^{th} Iteration, residual for the k^{th} Tree: Response – Prediction from all but k^{th} tree.

$$r_i = y_i - \sum_{k'<k} \hat{f}_{k'}^{b}(x_i) - \sum_{k'>k} \hat{f}_{k'}^{b-1}(x_i)$$

Instead of building a new tree, we randomly perturb a tree from previous iteration, fit all perturbations to the residuals. Choose perturbation that improves the fit. (Error is computed based on Residual, not based on original Response.)

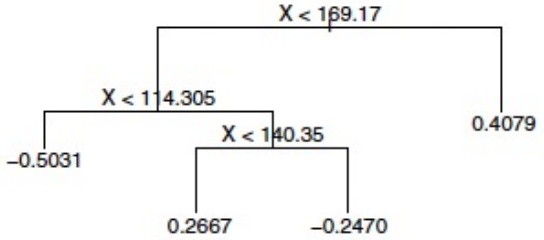
Very useful technique on Iteration based Methods. (You can look for Basis Pursuit Algorithm, which uses similar approach to learn the basis of representation.)

Bayesian Additive Regression Trees (BART)

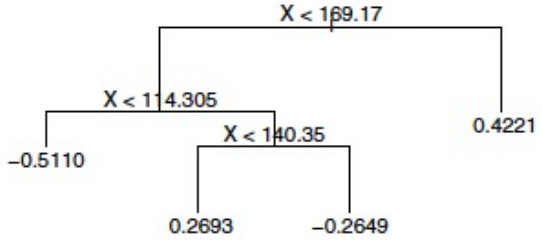
Perturbation: Example

At bth iteration, randomly choose a perturbation of a tree at b-1 iteration $\hat{f}_k^{b-1}(X)$

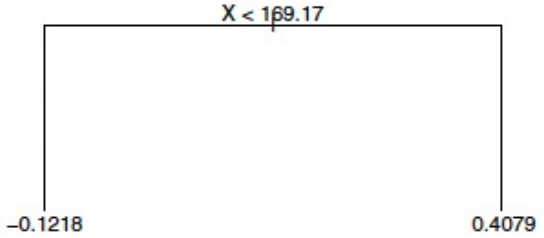
(a): $\hat{f}_k^{b-1}(X)$



(b): Possibility #1 for $\hat{f}_k^b(X)$



(c): Possibility #2 for $\hat{f}_k^b(X)$



(d): Possibility #3 for $\hat{f}_k^b(X)$

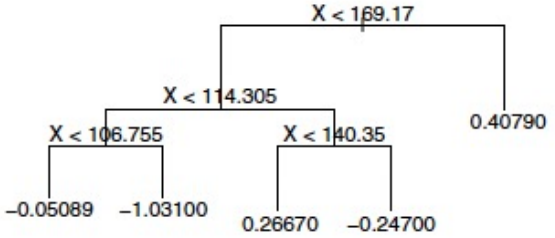


Figure 8.12

Bayesian Additive Regression Trees (BART)

Algorithm 8.3 *Bayesian Additive Regression Trees*

1. Let $\hat{f}_1^1(x) = \hat{f}_2^1(x) = \cdots = \hat{f}_K^1(x) = \frac{1}{nK}\sum_{i=1}^{n} y_i$.
2. Compute $\hat{f}^1(x) = \sum_{k=1}^{K} \hat{f}_k^1(x) = \frac{1}{n}\sum_{i=1}^{n} y_i$.
3. For $b = 2, \ldots, B$:
   (a) For $k = 1, 2, \ldots, K$:
      i. For $i = 1, \ldots, n$, compute the current partial residual

$$r_i = y_i - \sum_{k'<k} \hat{f}_{k'}^b(x_i) - \sum_{k'>k} \hat{f}_{k'}^{b-1}(x_i).$$

      ii. Fit a new tree, $\hat{f}_k^b(x)$, to r_i, by randomly perturbing the kth tree from the previous iteration, $\hat{f}_k^{b-1}(x)$. Perturbations that improve the fit are favored.

   (b) Compute $\hat{f}^b(x) = \sum_{k=1}^{K} \hat{f}_k^b(x)$.
4. Compute the mean after L burn-in samples,

$$\hat{f}(x) = \frac{1}{B-L}\sum_{b=L+1}^{B} \hat{f}^b(x).$$

Bayesian Additive Regression Trees (BART)

L – BURN IN SAMPLES

4. Compute the mean after L burn-in samples,

$$\hat{f}(x) = \frac{1}{B-L} \sum_{b=L+1}^{B} \hat{f}^b(x).$$

Q. Why calculate average only after L iterations not average over iteration 1 to B ?

Hint:
Concept of Weak Learners
Slide # 42, Initialization

Take Away

- Decision Trees are build using Recursive Binary Split and Greedy Approach.
- Decision Tree without pruning are likely to overfit the data.
- Pruning adds numbers of leaf nodes in the decision tree as a penalty in the cost.
- Pruning can be done by choosing appropriate alpha (weight parameter) and k-fold cross validation allows us to choose right model.

- Bagging, Random Forest, Boosting and BART are method of Ensembles.
- Create Many different trees, use all trees to decrease the variability of model.

- Bagging: Uses bootstrapping and average of K bootstrap models.
- Random Forest: Uses Bootstrap as well but, At every split, use only $m<p$ numbers of input variables.
- Boosting: Builds B trees sequentially using residuals.
- BART: Somehow mixture of Random Forest and Boosting: Trees are chosen from the Perturbation.

Final Prediction is made based on the average of predictions of B-L iterations.

Note: We do not Prune in Bagging, Random Forest, Boosting.