Classification and Regression Trees

FA590 Statistical Learning in Finance

Dr. Zonghao Yang

Stevens Institute of Technology

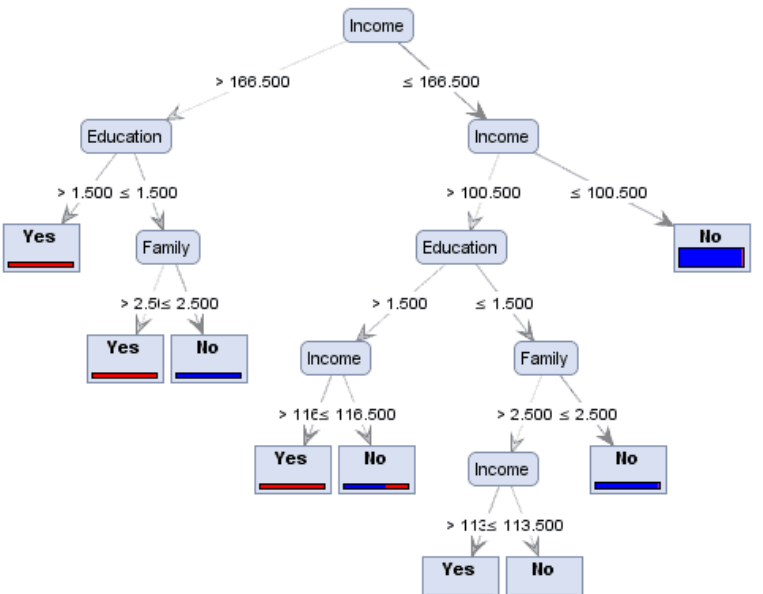
Previous Lecture

- Understand the mathematical foundations of support vector machines (SVMs), including linear classification and separating hyperplanes
- Formulate SVMs as optimization problems for both separable and non-separable cases
- Learn how to apply kernel transformations to extend SVMs for non-linear decision boundaries
- Compare SVMs with other classification methods, such as logistic regression, and understand the strengths and weaknesses of each approach

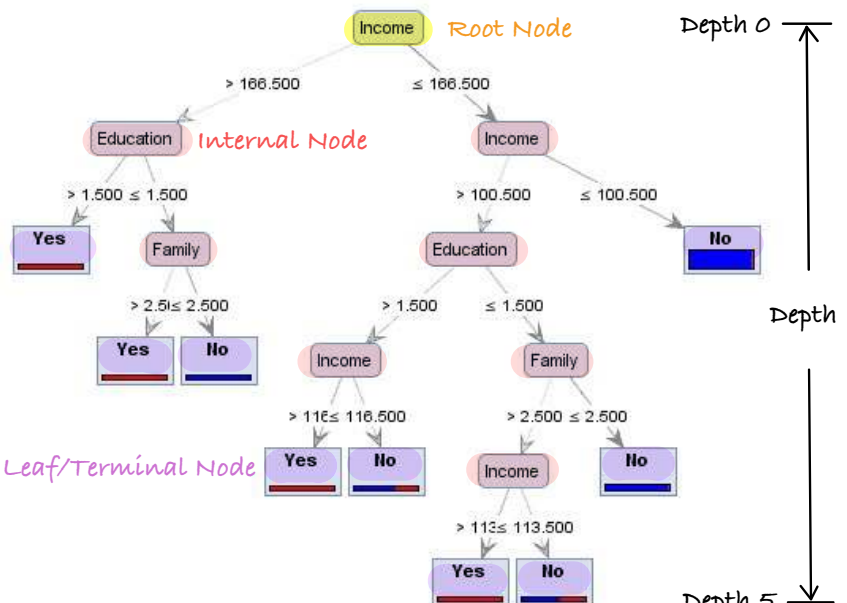
Part I

Introduction to Trees

Motivation Example: Loan Default



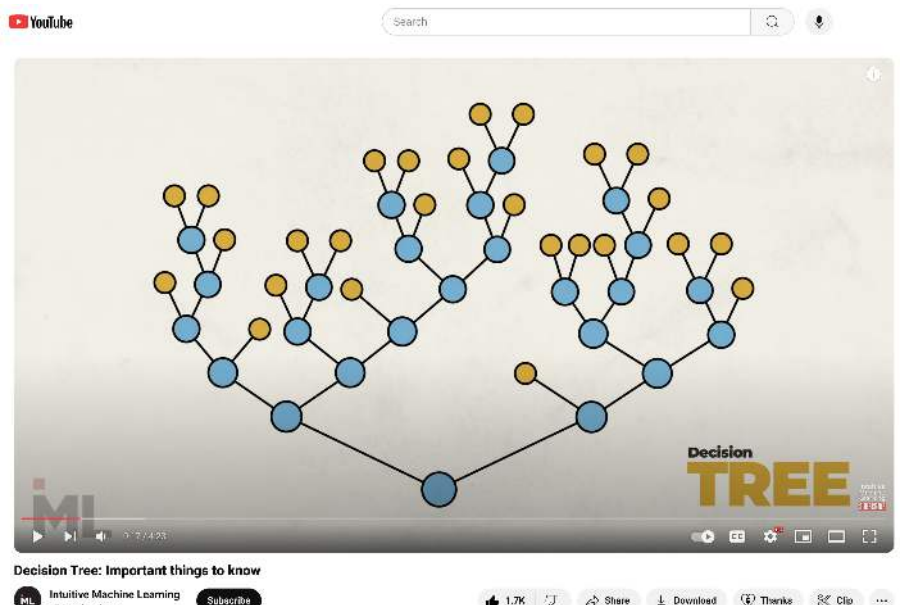
TERMINOLOGIES



Observations

- The decision trees accept both numerical and categorical features
  - In Python implementation, many pre-defined packages and decision tree classifiers rely on computational methods that require the features to be numerical
- The numerical threshold can be different for the same data
- The final classifications can be repeated
- **Binary tree** Each node has either 2 children or 0 children

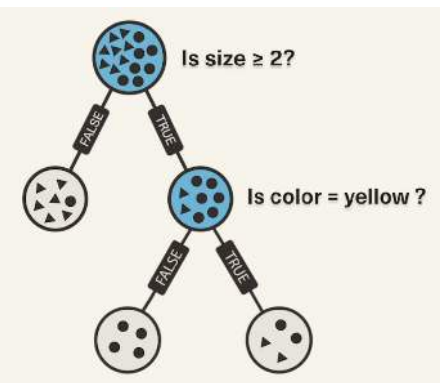
Animation



YouTube: Decision Tree, Intuitive Machine Learning

Video Example: Decision Tree

- A binary classification problem
  - Decision trees can be used for both regression and classification
- **Decision Trees** A **nonparametric supervised** algorithm that uses a tree-like model of decisions and their possible consequences
  - Decision tree is the first nonparametric algorithm we see in this class
- Predicting given a new sample



Video Example: Training a Decision Tree

- Three steps to train a decision tree
  1. **Feature Selection** Which feature do I use as the node?
  2. **Grow the Tree by Splitting the Node** Where should I split the feature?
  3. **Pruning** Does removing the subtree improve the predictive performance?
- Steps 1 and 2 form the best decision rule , which is used to split a node into two subnodes
- Steps 1 and 2 are repeated recursively to further split each of the two subnodes until there are no decision rules left or the group impurity is 0

Video Example: Information Gain

- The criterion to choose the decision rule is the **information gain** based on **Gini impurity**

$$\text{Gini Impurity} = \sum_{i=1}^{n} p(i) \times (1 - p(i))$$

- There are other criteria to measure the information gain
  - Classification: Misclassification error and cross entropy
  - Regression: Mean squared error

Video Example: Overfitting

- A decision tree can grow so deep that it overfits the training data
- Two general approaches to avoid overfitting
  - Early stopping: max depth of the tree, min samples in a leaf
  - Pruning, i.e., reduce model complexity by removing subtrees from the decision tree

VIDEO EXAMPLE: IMPROVE DECISION TREE

- Random Forest: Ensemble a “forest" of decision trees
- In our next lecture, we will introduce random forest, and
  - Adaboost
  - Gradient-boosted trees
  - XGBoost
  - ...

Learning Objectives

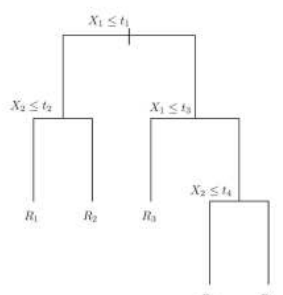
- Understand the fundamental components and structure of decision trees, including nodes, branches, and terminal nodes
- Learn the processes of training a decision tree, including feature selection, splitting, and pruning to prevent overfitting
- Explore metrics like Gini impurity and cross-entropy to evaluate and select optimal splits in classification tasks
- Recognize the differences between decision trees and other models, particularly in handling missing data, nonlinear relationships, and categorical features

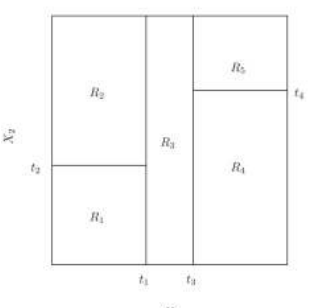
Part II

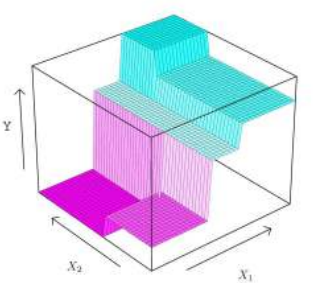
Regression Trees

Binary Decision Tree on $\mathbb{R}^2$

Consider a binary tree on $\{(X_1, X_2)|X_1, X_2 \in \mathbb{R}\}$







Fitting a Regression Tree

- The decision tree gives the partition of $\mathcal{X}$ into regions:

$$\{R_1, \ldots, R_M\}$$

- Recall that a partition is a disjoint union , that is:

$$\mathcal{X} = R_1 \cup R_2 \cup \cdots \cup R_M \text{ and } R_i \cap R_j = \emptyset \quad \forall i \neq j$$

- Given the partition $\{R_1, \ldots, R_M\}$, final prediction is

$$f(x) = \sum_{m=1}^{M} c_m \mathbb{I}(x \in R_m)$$

where $\hat{c}_m = \bar{y}_m$

Complexity of a Tree

- Let $|T| = M$ denote the number of leaf nodes in T
  - $|T|$ measures the complexity of a tree
- Finding the optimal binary tree of a given complexity is computationally intractable
- We proceed with a **greedy algorithm**
  - Build the tree one node at a time, without any planning ahead

Determining the Root Node

- Let $\mathbf{x} = (x_1, \ldots, x_p) \in \mathbb{R}^p$; and assume all are continuous variables
- Given a **splitting variable** $j \in \{1, \ldots, p\}$ and a **split point** $s \in \mathbb{R}$, the partition follows

$$R_1(j, s) = \{\mathbf{x} \mid x_j \leq s\}$$
$$R_2(j, s) = \{\mathbf{x} \mid x_j > s\}$$

- For each splitting variable j and split point s,

$$\hat{c}_1(j, s) = \text{ave}(y_i \mid x_i \in R_1(j, s))$$

$$\hat{c}_2(j, s) = \text{ave}(y_i \mid x_i \in R_2(j, s))$$

- The objective is to find j, s that minimize the loss function

$$L(j, s) = \sum_{i: x_i \in R_1(j,s)} (y_i - \hat{c}_1(j, s))^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{c}_2(j, s))^2$$

The Set of Split Points

- Consider splitting on the j-th feature x_j
- If $x_{j(1)}, \ldots, x_{j(n)}$ are the sorted values of the j'th feature,
  - Only need to check split points between adjacent values
  - Often take split points halfway between adjacent values

$$s_j \in \left\{ \frac{1}{2}\left(x_{j(r)} + x_{j(r+1)}\right) \mid r = 1, \ldots, n-1 \right\}$$

- So only need to check performance of $n - 1$ splits

- Together there are $p \times (n-1)$ pairs of (j, s) (decision rules): $\hat{j}, \hat{s} = \arg\min_{j,s} L(j, s)$

Then Proceed Recursively

- Steps to proceed
  1. We have determined R_1 and R_2
  2. Find best split for points in R_1
  3. Find best split for points in R_2
  4. Continue...

- Stopping conditions
  - Limit max depth of tree
  - Minimum node size (e.g., 5)

PRUNING

- **Pruning** The approach of CART (Breiman et al., 1984)[1]
- Pruning is an important step of training a decision tree to avoid overfitting
- Consider an internal node *n*, pruning the subtree rooted at *n* means
  - eliminate all descendants of *n* AND
  - *n* becomes a terminal node

[1] Breiman, L., Friedman, J., Olshen, R.A., & Stone, C.J. (1984). Classification and Regression Trees (1st ed.). Chapman and Hall/CRC.

PRUNING: VISUALIZATION

- Full tree T_0
- Subtree after pruning $T \subset T_0$

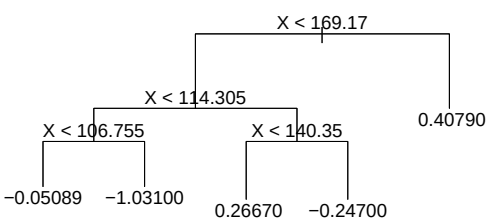


Figure. Full Tree

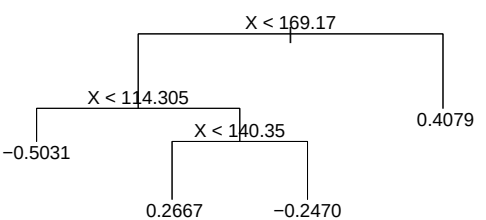


Figure. Subtree

Pruning in CART

- Suppose we want to prune a big tree T_0

- Let $L(T)$ be the loss of the subtree T, i.e., MSE on training
- Clearly, for any subtree $T \subset T_0$, $L(T) \geq L(T_0)$
- We prune one pair of leaf nodes at a time
- Find a proper subtree $T_1 \subset T_0$ that minimizes $L(T_1) - L(T_0)$.
  - Can get T_1 by removing a single pair of leaf nodes, and their internal node parent becomes a leaf node
  - This T_1 will have one fewer internal node than T_0 and one fewer leaf node

Pruning in CART (Cont.)

- Then find proper subtree $T_2 \subset T_1$ that minimizes $L(T_2) - L(T_1)$
- Repeat until we have removed all internal nodes and are left with just a single node (a leaf node)
- If N_{Int} is the number of internal nodes of T_0, then we end up with a nested sequence of trees:

$$\mathcal{T} = \{T_0 \supset T_1 \supset T_2 \supset \cdots \supset T_{|N_{\text{Int}}|}\}$$

- **Choose the best subtree as the final decision tree based on the validation results**

Part III

Classification Trees

Classification Trees

- Training a classification tree is the same as the regression tree, except that we need to modify the criteria for splitting nodes

- Consider a classification problem with K classes: $y = \{1, 2, \ldots, K\}$
- Let node m represent region R_m, with N_m observations
- Denote proportion of observations in R_m with class k by

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{\{i: x_i \in R_m\}} \mathbb{I}(y_i = k)$$

- Predicted class probability distribution for node m is $(\hat{p}_{m1}, \ldots, \hat{p}_{mK})$
- Predicted classification for node m is

$$k(m) = \arg\max_k \hat{p}_{mk}$$

Misclassification Error

- Consider node m representing region R_m with N_m observations
- Suppose we predict

$$k(m) = \arg\max_k \hat{p}_{mk}$$

  as the class for all inputs in region R_m
- **Misclassification Error** What is the misclassification rate on the training data?

$$1 - \hat{p}_{mk(m)}$$

- Should we use misclassification error as the criteria for node splitting? Maybe not

Splitting Example

- Two class problem: 400 observations in each class
  - Split 1: (300,100) and (100,300) [each region has 300 of one class and 100 of other]
  - Split 2: (200,400) and (200,0) [one region has 200 of one class and 400 of other, other region pure]
- Misclassification rate for the two splits are same
- In split 1, we will need to further split both sub nodes
- In split 2, we are already done with the node (200,0)

- **Takeaway** Instead of misclassification error, we want the splitting criteria to be some impurity measure

Node Impurity Measures

- Consider leaf node m representing region R_m, with N_m observations
- Three measures $Q_m(T)$ of node impurity for leaf node m
  - **Misclassification Error**

$$1 - \hat{p}_{mk(m)}$$

  - **Gini Impurity**

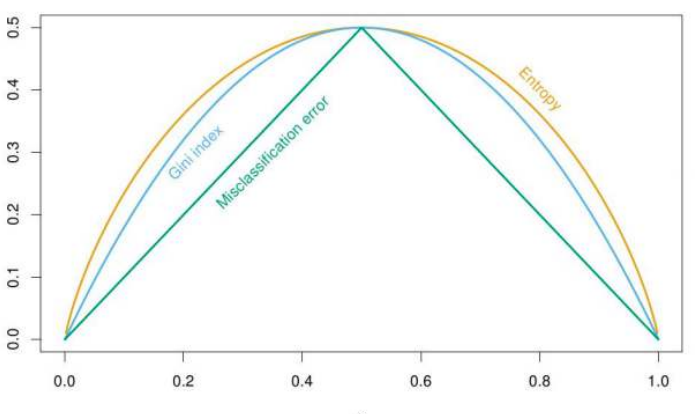
$$\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

  - **Cross Entropy**

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

Two-Class Node Impurity Measure

- Misclassification error: $1 - \max(p, 1-p)$
- Gini impurity: $2p(1-p)$
- Cross entropy: $-p\log p - (1-p)\log(1-p)$



SPLITTING NODES

- Let R_L and R_R be regions corresponding to a potential node split
- Suppose we have N_L points in R_L and N_R points in R_R
- Let $Q(R_L)$ and $Q(R_R)$ be the node impurity measures
- Choose the split that minimizes the weighted average of node impurities

$$N_L Q(R_L) + N_R Q(R_R)$$

Comments on Node Impurity Measures

- For building the tree, Gini and Entropy seem to be more effective
  - They push for more pure nodes, not just misclassification rate
  - Differentiable
  - A good split may not change misclassification rate at all!

Part IV

More on Trees

Categorical Features

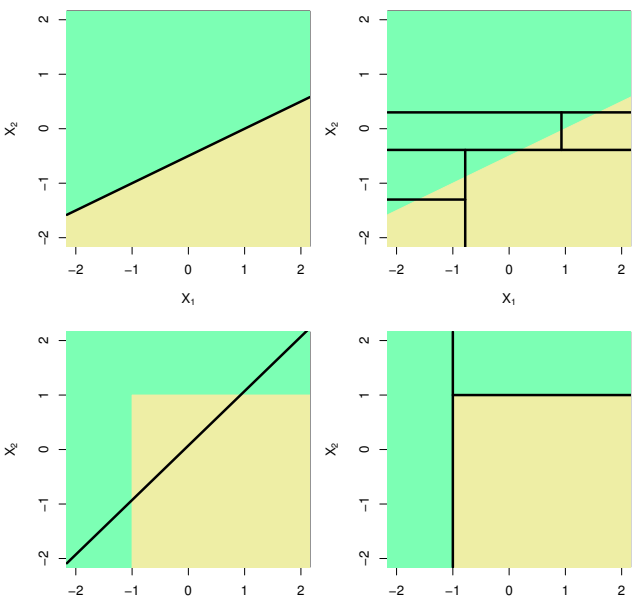
- So far we assumed all features are continuous
- Suppose we have a categorical feature with q possible values (unordered), and we want to find the best split into 2 groups
- There are $2^{q-1} - 1$ distinct splits, which can easily become intractable when q is large
  - For binary classification $y = \{0, 1\}$, there is an efficient algorithm

Categorical Features in Binary Classification

- Assign each category a number
- Then find optimal split as though it were a numeric feature
- For binary classification, this is equivalent to searching over all splits
- However, this trick doesn't work for multiclass, so numerical approximations are necessary

TREES VS LINEAR MODEL

Trees are good at capturing nonlinear relations, but not linear relations



Trees for Nonlinear Feature Discovery

- Suppose tree T gives partition $R_1, \ldots, R_m$
- Predictions are

$$f(x) = \sum_{m=1}^{M} c_m \mathbb{I}(x \in R_m)$$

- Each region R_m can be treated as a unique feature, represented by the function $x \mapsto \mathbb{I}(x \in R_m)$
- Can use these nonlinear features as engineered features in parametric algorithms, e.g., linear regression
  - Create a set of dummy variables to indicate which region does the sample belong

Comments about Trees

- Decision tree is a nonparametric method
  - No fixed number of parameters or a fixed functional form
  - Model complexity grows with data
- Trees are easy to visualize and interpret
- Trees make no use of **geometry**
  - No inner products or distances
  - A “nonmetric” method
  - Feature scale irrelevant
- Prediction functions are not continuous
  - Not so bad for classification
  - May not be desirable for regression