Advances in Decision Tree Construction

Johannes Gehrke
Cornell University
johannes@cs.cornell.edu
http://www.cs.cornell.edu/johannes

Wei-Yin Loh
University of Wisconsin-Madison
loh@stat.wisc.edu
http://www.stat.wisc.edu/~loh

Tutorial Overview

- Part I: Classification Trees
  - Introduction
  - Classification tree construction schema
  - Split selection
  - Pruning
  - Data access
  - Missing values
  - Evaluation
  - Bias in split selection

(Short Break)

- Part II: Regression Trees

Tutorial Overview

- Part I: Classification Trees
  - Introduction
  - Classification tree construction schema
  - Split selection
  - Pruning
  - Data access
  - Missing values
  - Evaluation
  - Bias in split selection

(Short Break)

- Part II: Regression Trees

Classification

Goal: Learn a function that assigns a record to one of several predefined classes.

Classification Example

- Example training database
  - Two predictor attributes: Age and Car-type (**S**port, **M**inivan and **T**ruck)
  - Age is ordered, Car-type is categorical attribute
  - Class label indicates whether person bought product
  - Dependent attribute is *categorical*

Age	Car	Class
20	M	Yes
30	M	Yes
25	T	No
30	S	Yes
40	S	Yes
20	T	No
30	M	Yes
25	M	Yes
40	M	Yes
20	S	No

Types of Variables

- *Numerical*: Domain is ordered and can be represented on the real line (e.g., age, income)
- *Nominal* or *categorical*: Domain is a finite set without any natural ordering (e.g., occupation, marital status, race)
- *Ordinal*: Domain is ordered, but absolute differences between values is unknown (e.g., preference scale, severity of an injury)

Definitions

- Random variables $X_1, \ldots, X_k$ (*predictor variables*) and Y (*dependent variable*)
- X_i has domain dom(X_i), Y has domain dom(Y)
- P is a probability distribution on dom(X_1) x … x dom(X_k) x dom(Y)
  Training database D is a random sample from P
- A *predictor* d is a function
  d: dom(X_1) … dom(X_k) → dom(Y)

Classification Problem

- C is called the *class label*, d is called a *classifier.*
- Take r be record randomly drawn from P.
  Define the *misclassification rate* of d:
  RT(d,P) = P(d(r.X_1, …, r.X_k) != r.C)

Problem definition: Given dataset D that is a random sample from probability distribution P, find classifier d such that RT(d,P) is minimized.

(More on regression problems in the second part of the tutorial.)

Goals and Requirements

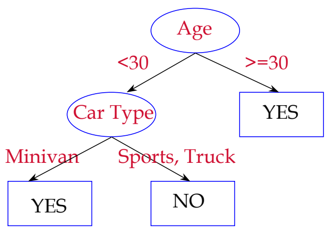
Goals:

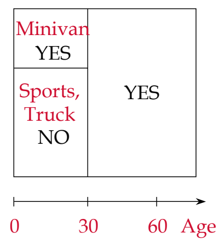
- To produce an accurate classifier/regression function
- To understand the structure of the problem

Requirements on the model:

- High accuracy
- Understandable by humans, interpretable
- Fast construction for very large training databases

What are Decision Trees?





Decision Trees

- A *decision tree* T encodes d (a classifier or regression function) in form of a tree.
- A node t in T without children is called a *leaf node*. Otherwise t is called an *internal node*.
- Each internal node has an associated *splitting predicate*. Most common are binary predicates. Example splitting predicates:
  - Age <= 20
  - Profession in {student, teacher}
  - 5000*Age + 3*Salary – 10000 > 0

Internal and Leaf Nodes

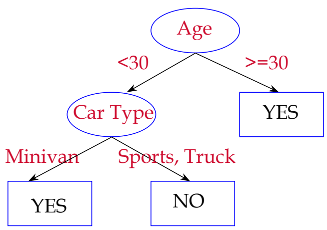
Internal nodes:

- Binary Univariate splits:
  - Numerical or ordered X: X <= c, c in dom(X)
  - Categorical X: X in A, A subset dom(X)
- Binary Multivariate splits:
  - Linear combination split on numerical variables: $\Sigma\, a_i X_i <= c$
- k-ary (k>2) splits analogous

Leaf nodes:

- Node t is labeled with one class label c in dom(C)

Example



Encoded classifier:

If (age<30 and
carType=Minivan)
Then YES

If (age <30 and
(carType=Sports or
carType=Truck))
Then NO

If (age >= 30)
Then NO

Evaluation of Misclassification Error

Problem:
- In order to quantify the quality of a classifier d, we need to know its misclassification rate RT(d,P).
- But unless we know P, RT(d,P) is unknown.
- Thus we need to estimate RT(d,P) as good as possible.

Approaches:
- Resubstitution estimate
- Test sample estimate
- V-fold Cross Validation

Resubstitution Estimate

The *Resubstitution estimate* R(d,D) estimates RT(d,P) of a classifier d using D:
- Let D be the training database with N records.
- $R(d,D) = 1/N\ \Sigma\ I(d(r.X) \mathrel{!=} r.C))$
- Intuition: R(d,D) is the proportion of training records that is misclassified by d
- Problem with resubstitution estimate:
Overly optimistic; classifiers that overfit the training dataset will have very low resubstitution error.

Test Sample Estimate

- Divide D into D_1 and D_2
- Use D_1 to construct the classifier d
- Then use resubstitution estimate $R(d,D_2)$ to calculate the estimated misclassification error of d
- Unbiased and efficient, but removes D_2 from training dataset D

V-fold Cross Validation

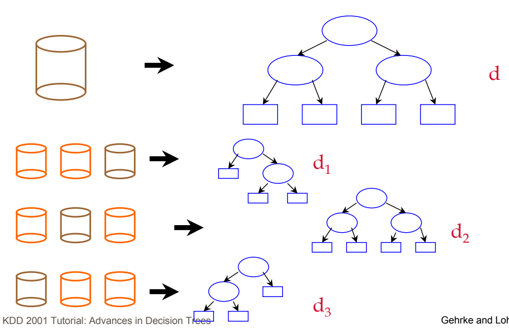
Procedure:

- Construct classifier d from D
- Partition D into V datasets $D_1, \ldots, D_V$
- Construct classifier d_i using $D \setminus D_i$
- Calculate the estimated misclassification error $R(d_i,D_i)$ of d_i using test sample D_i

Final misclassification estimate:

- Weighted combination of individual misclassification errors: $R(d,D) = 1/V\ \Sigma\ R(d_i,D_i)$

Cross-Validation: Example



Cross-Validation

- Misclassification estimate obtained through cross-validation is usually nearly unbiased
- Costly computation (we need to compute d, and $d_1, \ldots, d_V$); computation of d_i is nearly as expensive as computation of d
- Preferred method to estimate quality of learning algorithms in the machine learning literature

Tutorial Overview

- Part I: Classification Trees
  - Introduction
  - Classification tree construction schema
  - Split selection
  - Pruning
  - Data access
  - Missing values
  - Evaluation
  - Bias in split selection

(Short Break)

- Part II: Regression Trees

Decision Tree Construction

- Top-down tree construction schema:
  - Examine training database and find best splitting predicate for the root node
  - Partition training database
  - Recurse on each child node

BuildTree(Node *t*, Training database *D*, Split Selection Method *S*)
(1) Apply *S* to *D* to find splitting criterion
(2) **if** (*t* is not a leaf node)
(3) Create children nodes of *t*
(4) Partition *D* into children partitions
(5) Recurse on each partition
(6) **endif**

Decision Tree Construction (Contd.)

- Three algorithmic components:
  - Split selection (CART, C4.5, QUEST, CHAID, CRUISE, …)
  - Pruning (direct stopping rule, test dataset pruning, cost-complexity pruning, statistical tests, bootstrapping)
  - Data access (CLOUDS, SLIQ, SPRINT, RainForest, BOAT, UnPivot operator)

Split Selection Methods

- Multitude of split selection methods in the literature
- In this tutorial:
  - Impurity-based split selection: CART (most common in today's data mining tools)
  - Model-based split selection: QUEST

Split Selection Methods: CART

- Classification And Regression Trees (Breiman, Friedman, Ohlson, Stone, 1984; considered "the" reference on decision tree construction)
- Commercial version sold by Salford Systems (www.salford-systems.com)
- Many other, slightly modified implementations exist (e.g., IBM Intelligent Miner implements the CART split selection method)

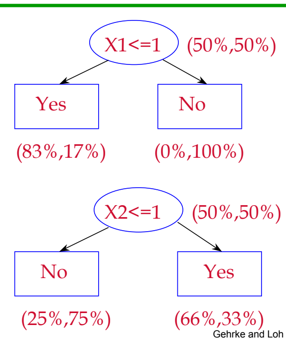
CART Split Selection Method

Motivation: We need a way to choose quantitatively between different splitting predicates

- Idea: Quantify the *impurity* of a node
- Method: Select splitting predicate that generates children nodes with minimum impurity from a space of possible splitting predicates

Intuition: Impurity Function

X1	X2	Class
1	1	Yes
1	2	Yes
1	2	Yes
1	2	Yes
1	2	Yes
1	1	No
2	1	No
2	1	No
2	2	No
2	2	No



Impurity Function

Let p(j|t) be the proportion of class j training records at node t. Then the node impurity measure at node t:
i(t) = phi(p(1|t), …, p(J|t))

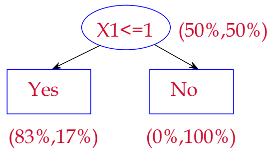
<u>Properties:</u>

- phi is symmetric
- Maximum value at arguments (J^{-1}, …, J^{-1})
- phi(1,0,…,0) = … =phi(0,…,0,1) = 0

The *reduction in impurity* through splitting predicate s (t splits into children nodes t_L with impurity phi(t_L) and t_R with impurity phi(t_R)) is:
$\Delta_{phi}(s,t)$ = phi(t) – p_L phi(t_L) – p_R phi(t_R)

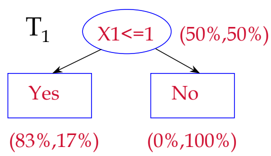
Example

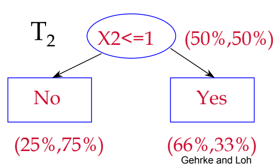
Root node t: p(1|t)=0.5; p(2|t)=0.5
Left child node t:
P(1|t)=0.83; p(2|t)=-.17
Impurity of root node: phi(0.5,0.5)
Impurity of left child node:
phi(0.83,0.17)
Impurity of right child node:
phi(0.0,1.0)
Impurity of whole tree:
0.6* phi(0.83,0.17) + 0.4 * phi(0,1)
Impurity reduction:
phi(0.5,0.5) - 0.6*phi(0.83,0.17) - 0.4*phi(0,1)



Error Reduction as Impurity Function

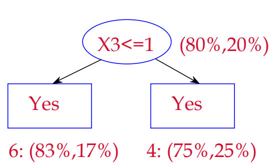
- Possible impurity function: Resubstitution error R(T,D).
- Example:
R(no tree, D) = 0.5
$R(T_1,D)$ = 0.6*0.17
$R(T_2,D)$ =
0.4*0.25 + 0.6*0.33





Problems with Resubstitution Error

- Obvious problem: There are situations where no split can decrease impurity
- Example:
R(no tree, D) = 0.2
$R(T_1,D)$
=0.6*0.17+0.4*0.25
=0.2
- More subtle problems exist



Remedy: Concavity

Concave Impurity Functions

Use impurity functions that are concave: phi″ < 0

Example concave impurity functions

- Entropy: phi(t) = - Σ p(j|t) log(p(j|t))
- Gini index: phi(t) = Σ $p(j|t)^2$

Nonnegative Decrease in Impurity

Theorem: Let phi(p_1, …, p_J) be a strictly concave function on j=1, …, J, $\Sigma_j p_j = 1$.

Then for any split s: $\Delta_{phi}(s,t) >= 0$

With equality if and only if: $p(j|t_L) = p(j|t_R) = p(j|t)$, j = 1, …, J

CART Univariate Split Selection

- Use gini-index as impurity function
- For each numerical or ordered attribute X, consider all binary splits s of the form
  X <= x
  where x in dom(X)
- For each categorical attribute X, consider all binary splits s of the form
  X in A, where A subset dom(X)
- At a node t, select split s* such that $\Delta_{phi}(s^*,t)$ is maximal over all s considered

CART: Shortcut for Categorical Splits

Computational shortcut if |Y|=2.

- Theorem: Let X be a categorical attribute with dom(X) = {b_1, …, b_k}, |Y|=2, phi be a concave function, and let
  $p(X=b_1) <= \ldots <= p(X=b_k)$.
  Then the best split is of the form:
  X in {b_1, b_2, …, b_l} for some l < k
- Benefit: We need only to check k-1 subsets of dom(X) instead of $2^{(k-1)}$-1 subsets

Problems with CART Split Selection

- Biased towards variables with more splits (M-category variable has 2^{M-1}-1) possible splits, an M-valued ordered variable has (M-1) possible splits
(Explanation and remedy later)
- Computationally expensive for categorical variables with large domains

QUEST: Model-based split selection

"The purpose of models is not to fit the data but to sharpen the questions."

Karlin, Samuel (1923 -)
(11th R A Fisher Memorial Lecture, Royal Society 20, April 1983.)

Split Selection Methods: QUEST

- Quick, Unbiased, Efficient, Statistical Tree (Loh and Shih, Statistica Sinica, 1997)
Freeware, available at www.stat.wisc.edu/~loh
Also implemented in SPSS.
- Main new ideas:
  - Separate splitting predicate selection into variable selection and split point selection
  - Use statistical significance tests instead of impurity function

QUEST Variable Selection

Let $X_1, \ldots, X_l$ be numerical predictor variables, and let $X_{l+1}, \ldots, X_k$ be categorical predictor variables.

1. Find p-value from ANOVA F-test for each numerical variable.
2. Find p-value for each X^2-test for each categorical variable.
3. Choose variable $X_{k'}$ with overall smallest p-value $p_{k'}$

(Actual algorithm is more complicated.)

QUEST Split Point Selection

CRIMCOORD transformation of categorical variables into numerical variables:

1. Take categorical variable X with domain $\text{dom}(X)=\{x_1, \ldots, x_l\}$
2. For each record in the training database, create vector $(v_1, \ldots, v_l)$ where $v_i = I(X=x_i)$
3. Find principal components of set of vectors V
4. Project the dimensionality-reduced data onto the largest discriminant coordinate dx_i
5. Replace X with numeral dx_i in the rest of the algorithm

CRIMCOORDs: Examples

- $\text{Values}(X|Y=1) = \{4c_1, c_2, 5c_3\}$, $\text{values}(X|Y=2) = \{2c_1, 2c_2, 6c_3\}$
  ➔ $dx_1 = 1$, $dx_2 = -1$, $dx_3 = -0.3$
- $\text{Values}(X|Y=1) = \{5c_1, 5c_3\}$, $\text{values}(X|Y=2) = \{5c_1, 5c_3\}$
  ➔ $dx_1 = 1$, $dx_2 = 0$, $dx_3 = 1$
- $\text{Values}(X|Y=1) = \{5c_1, 5c_3\}$, $\text{values}(X|Y=2) = \{5c_1, c_2, 5c_3\}$
  ➔ $dx_1 = 1$, $dx_2 = -1$, $dx_3 = 1$

Advantages

- Avoid exponential subset search from CART
- Each dx_i has the form $\Sigma\, b_i\, I(X=x_i)$ for some $b_1, \ldots, b_l$, thus there is a 1-1 correspondence between subsets of X and a dx_i

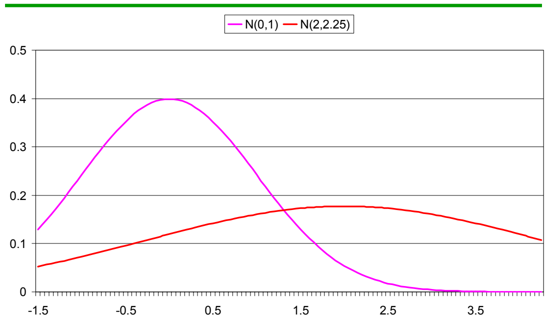
QUEST Split Point Selection

- Assume X is the selected variable (either numerical, or categorical transformed to CRIMCOORDS)
- Group J>2 classes into two superclasses
- Now problem is reduced to one-dimensional two-class problem
  - Use exhaustive search for the best split point (like in CART)
  - Use quadratic discriminant analysis (QDA, next few bullets)

QUEST Split Point Selection: QDA

- Let x_1, x_2 and s_1^2, s_2^2 the means and variances for the two superclasses
- Make normal distribution assumption, and find intersections of the two normal distributions $N(x_1,s_1^2)$ and $N(x_2,s_2^2)$
- QDA splits the X-axis into three intervals
- Select as split point the root that is closer to the sample means

Illustration: QDA Splits



QUEST Linear Combination Splits

- Transform all categorical variables to CRIMCOORDS
- Apply PCA to the correlation matrix of the data
- Drop the smallest principal components, and project the remaining components onto the largest CRIMCOORD
- Group J>2 classes into two superclasses
- Find split on largest CRIMCOORD using ES or QDA

Key Differences CART/QUEST

Feature	QUEST	CART
Variable selection	Statistical tests	ES
Split point selection	QDA or ES	ES
Categorical variables	CRIMCOORDS	ES
Monotone transformations for numerical variables	Not invariant	Invariant
Ordinal Variables	No	Yes
Variables selection bias	No	Yes (No)

Tutorial Overview

- Part I: Classification Trees
  - Introduction
  - Classification tree construction schema
  - Split selection
  - Pruning
  - Data access
  - Missing values
  - Evaluation
  - Bias in split selection

(Short Break)

- Part II: Regression Trees

Pruning Methods

- Test dataset pruning
- Direct stopping rule
- Cost-complexity pruning (not covered)
- MDL pruning
- Pruning by randomization testing

Stopping Policies

A stopping policy indicates when further growth of the tree at a node t is counterproductive.

- All records are of the same class
- The attribute values of all records are identical
- All records have missing values
- At most one class has a number of records larger than a user-specified number
- All records go to the same child node if t is split (only possible with some split selection methods)

Test Dataset Pruning

- Use an independent test sample D' to estimate the misclassification cost using the resubstitution estimate R(T,D') at each node
- Select the subtree T' of T with the smallest expected cost

Reduced Error Pruning

(Quinlan, C4.5, 1993)

- Assume observed misclassification rate at a node is p
- Replace p (pessimistically) with the upper 75% confidence bound p', assuming a binomial distribution
- Then use p' to estimate error rate of the node

Pruning Using the MDL Principle

(Mehta, Rissanen, Agrawal, KDD 1996)
Also used before by Fayyad, Quinlan, and others.

- MDL: Minimum Description Length Principle
- Idea: Think of the decision tree as encoding the class labels of the records in the training database
- MDL Principle: The best tree is the tree that encodes the records using the fewest bits

How To Encode a Node

Given a node t, we need to encode the following:

- Nodetype: One bit to encode the type of each node (leaf or internal node)

For an internal node:

- Cost(P(t)): The cost of encoding the splitting predicate P(t) at node t

For a leaf node:

- n*E(t): The cost of encoding the records in leaf node t with n records from the training database (E(t) is the entropy of t)

How To Encode a Tree

Recursive definition of the minimal cost of a node:

- Node t is a leaf node:

$$cost(t)= n*E(t)$$

- Node t is an internal node with children nodes t_1 and t_2. Choice: Either make t a leaf node, or take the best subtrees, whatever is cheaper:

$$cost(t) = \min(n*E(t), 1+cost(P(t))+cost(t_1)+cost(t_2))$$

How to Prune

1. Construct decision tree to its maximum size
2. Compute the MDL cost for each node of the tree bottom-up
3. Prune the tree bottom-up:
   If cost(t)=n*E(t), make t a leaf node.
   Resulting tree is the final tree output by the pruning algorithm.

Performance Improvements: PUBLIC

(Shim and Rastogi, VLDB 1998)

- MDL bottom-up pruning requires construction of a complete tree before the bottom-up pruning can start
- Idea: Prune the tree during (not after) the tree construction phase
- Why is this possible?
  - Calculate a lower bound on cost(t) and compare it with n*E(t)

PUBLIC Lower Bound Theorem

- Theorem: Consider a classification problem with k predictor attributes and J classes. Let T_t be a subtree with s internal nodes, rooted at node t, let n_i be the number of records with class label i. Then

  $$\text{cost}(T_t) >= 2*s+1+s*\log k + \Sigma n_i$$
- Lower bound on $\text{cost}(T_t)$ is thus the minimum of:
  - n*E+1 (t becomes a leaf node)
  - $2*s+1+s*\log k + \Sigma n_i$ (subtree at t remains)

Large Datasets Lead to Large Trees

- Oates and Jensen (KDD 1998)
- Problem: Constant probability distribution P, datasets $D_1, D_2, \ldots, D_k$ with
  $|D_1| < |D_2| < \ldots < |D_k|$
  $|D_k| = c\,|D_{k-1}| = \ldots = c^k\,|D_1|$
- Observation: Trees grow
  $|T_1| < |T_2| < \ldots < |T_k|$
  $|T_k| = c'\,|T_{k-1}| = \ldots = c'^k\,|T_1|$
- But: No gain in accuracy due to larger trees
  $R(T_1,D_1) \sim R(T_2,D_2) \sim \ldots \sim R(T_k, D_k)$

Pruning By Randomization Testing

- Reduce pruning decision at each node to a hypothesis test
- Generate empirical distribution of the hypothesis under the null hypothesis for a node

Node n with subtree T(n) and pruning statistic S(n)
For (i=0; i<K; i++)
1. Randomize class labels of the data at n
2. Build and prune a tree rooted at n
3. Calculate pruning statistic $S_i(n)$

Compare S(n) to empirical distribution of $S_i(n)$ to estimate significance of S(n)
If S(n) is not significant enough compared to a significance level alpha, then prune T(n) to n

Tutorial Overview

- Part I: Classification Trees
  - Introduction
  - Classification tree construction schema
  - Split selection
  - Pruning
  - Data access
  - Missing values
  - Evaluation
  - Bias in split selection

(Short Break)

- Part II: Regression Trees

SLIQ

Shafer, Agrawal, Mehta (EDBT 1996)

- Motivation:
  - Scalable data access method for CART
  - To find the best split we need to evaluate the impurity function at all possible split points for each numerical attribute, at each node of the tree
  - Idea: Avoids re-sorting at each node of the three through pre-sorting and maintenance of sort orders
- Ideas:
  - Uses vertical partitioning to avoid re-sorting
  - Main-memory resident data structure with schema (class label, leaf node index)
    Very likely to fit in-memory for nearly all training databases

SLIQ: Pre-Sorting

Age	Car	Class
20	M	Yes
30	M	Yes
25	T	No
30	S	Yes
40	S	Yes
20	T	No
30	M	Yes
25	M	Yes
40	M	Yes
20	S	No

Age	Ind
20	1
20	6
20	10
25	3
25	8
30	2
30	4
30	7
40	5
40	9

Ind	Class	Leaf
1	Yes	1
2	Yes	1
3	No	1
4	Yes	1
5	Yes	1
6	No	1
7	Yes	1
8	Yes	1
9	Yes	1
10	No	1

SLIQ: Evaluation of Splits

Age	Ind
20	1
20	6
20	10
25	3
25	8
30	2
30	4
30	7
40	5
40	9

Ind	Class	Leaf
1	Yes	2
2	Yes	2
3	No	2
4	Yes	3
5	Yes	3
6	No	2
7	Yes	2
8	Yes	2
9	Yes	2
10	No	3

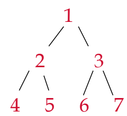
Node2	Yes	No
Left	2	0
Right	3	2

Node3	Yes	No
Left	0	1
Right	2	0

SLIQ: Splitting of a Node

Age	Ind
20	1
20	6
20	10
25	3
25	8
30	2
30	4
30	7
40	5
40	9

Ind	Class	Leaf
1	Yes	4
2	Yes	5
3	No	5
4	Yes	7
5	Yes	7
6	No	4
7	Yes	7
8	Yes	7
9	Yes	7
10	No	6



SLIQ: Summary

- Uses vertical partitioning to avoid re-sorting
- Main-memory resident data structure with schema (class label, leaf node index)
  Very likely to fit in-memory for nearly all training databases

SPRINT

Shafer, Agrawal, Mehta (VLDB 1996)

- Motivation:
  - Scalable data access method for CART
  - Improvement over SLIQ to avoid main-memory data structure
- Ideas:
  - Create vertical partitions called attribute lists for each attribute
  - Pre-sort the attribute lists

Recursive tree construction:

1. Scan all attribute lists at node t to find the best split
2. Partition current attribute lists over children nodes while maintaining sort orders
3. Recurse

SPRINT Attribute Lists

Age	Car	Class
20	M	Yes
30	M	Yes
25	T	No
30	S	Yes
40	S	Yes
20	T	No
30	M	Yes
25	M	Yes
40	M	Yes
20	S	No

Age	Class	Ind
20	Yes	1
20	No	6
20	No	10
25	No	3
25	Yes	8
30	Yes	2
30	Yes	4
30	Yes	7
40	Yes	5
40	Yes	9

Car	Class	Ind
M	Yes	1
M	Yes	2
T	No	3
S	Yes	4
S	Yes	5
T	No	6
M	Yes	7
M	Yes	8
M	Yes	9
S	No	10

SPRINT: Evaluation of Splits

Age	Class	Ind
20	Yes	1
20	No	6
20	No	10
25	No	3
25	Yes	8
30	Yes	2
30	Yes	4
30	Yes	7
40	Yes	5
40	Yes	9

Node1	Yes	No
Left	1	2
Right	6	1

SPRINT: Splitting of a Node

1. Scan all attribute lists to find the best split
2. Partition the attribute list of the splitting attribute X
3. For each attribute X_i != X

Perform the partitioning step of a hash-join between the attribute list of X and the attribute list of X_i

SPRINT: Hash-Join Partitioning

Age	Class	Ind	
20	Yes	1	
20	No	6	Right Child
20	No	10	Right Child
25	No	3	R
25	Yes	8	
30	Yes	2	
30	Yes	4	R
30	Yes	7	
40	Yes	5	R
40	Yes	9	

Car	Class	Ind
M	Yes	1
M	Yes	2
M	Yes	7
M	Yes	8
M	Yes	9

SPRINT: Summary

- Scalable data access method for CART split selection method
- Completely scalable, can be (and has been) implemented "inside" a database system
- Hash-join partitioning step expensive (each attribute, at each node of the tree)

RainForest
(Gehrke, Ramakrishnan, Ganti, VLDB 1998)

Training Database

Age	Car	Class
20	M	Yes
30	M	Yes
25	T	No
30	S	Yes
40	S	Yes
20	T	No
30	M	Yes
25	M	Yes
40	M	Yes
20	S	No

AVC-Sets

Age	Yes	No
20	1	2
25	1	1
30	3	0
40	2	0

Car	Yes	No
Sport	2	1
Truck	0	2
Minivan	5	0

Refined RainForest Top-Down Schema

BuildTree(Node *n*, Training database *D*,
Split Selection Method ***S***)

[(1) Apply ***S*** to *D* to find splitting criterion]
(1a) **for** each predictor attribute *X*
(1b) Call ***S***.findSplit(AVC-set of *X*)
(1c) **endfor**
(1d) ***S***.chooseBest();
(2) **if** (*n* is not a leaf node) ...

S: C4.5, CART, CHAID, FACT, ID3, GID3, QUEST, etc.

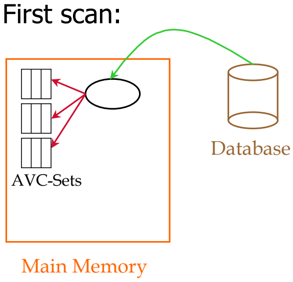
RainForest Data Access Method

Assume datapartition at a node is D. Then the following steps are carried out:

1. Construct AVC-group of the node
2. Choose splitting attribute and splitting predicate
3. Partition D across the children

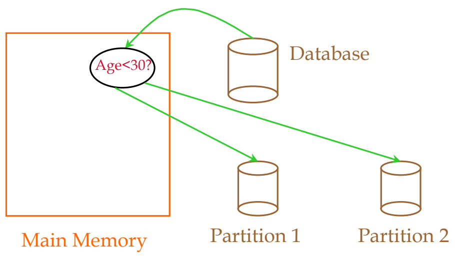
RainForest Algorithms: RF-Write

First scan:



RainForest Algorithms: RF-Write

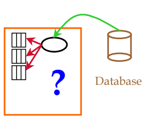
Second Scan:

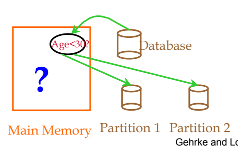


RainForest Algorithms: RF-Write

Analysis:

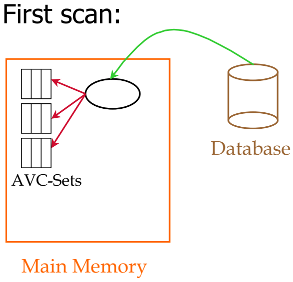
- Assumes that the AVC-group of the root node fits into main memory
- Two database scans per level of the tree
- Usually more main memory available than one single AVC-group needs





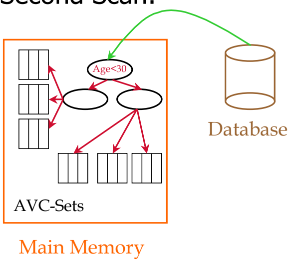
RainForest Algorithms: RF-Read

First scan:



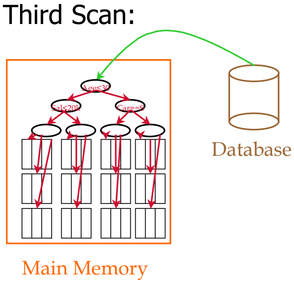
RainForest Algorithms: RF-Read

Second Scan:



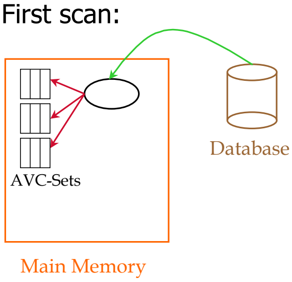
RainForest Algorithms: RF-Read

Third Scan:



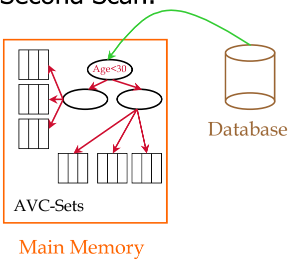
RainForest Algorithms: RF-Hybrid

First scan:



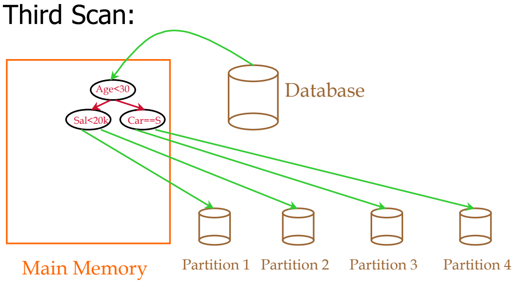
RainForest Algorithms: RF-Hybrid

Second Scan:



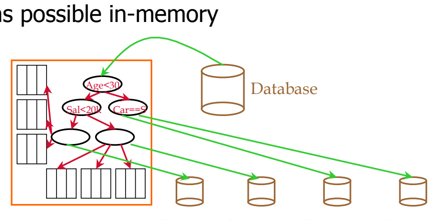
RainForest Algorithms: RF-Hybrid

Third Scan:



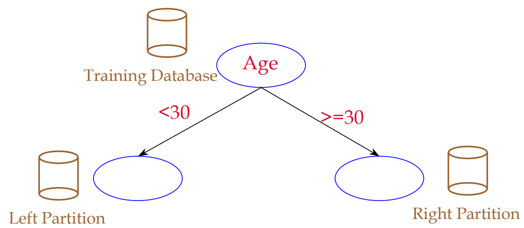
RainForest Algorithms: RF-Hybrid

Further optimization: While writing partitions, concurrently build AVC-groups of as many nodes as possible in-memory

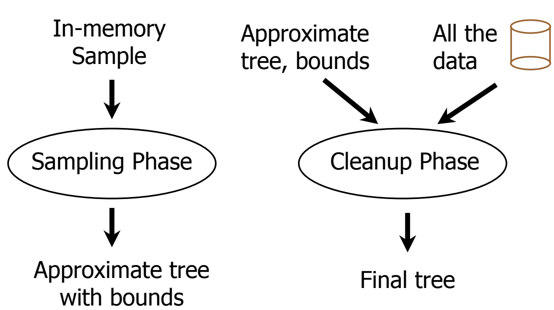


BOAT

(Gehrke, Ganti, Ramakrishnan, Loh; SIGMOD 1999)



BOAT: Algorithm Overview



Tutorial Overview

- Part I: Classification Trees
  - Introduction
  - Classification tree construction schema
  - Split selection
  - Pruning
  - Data access
  - Missing Values
  - Evaluation
  - Bias in split selection

(Short Break)

- Part II: Regression Trees

Missing Values

- What is the problem?
  - During computation of the splitting predicate, we can selectively ignore records with missing values (note that this has some problems)
  - But if a record r misses the value of the variable in the splitting attribute, r can not participate further in tree construction

Algorithms for missing values address this problem.

Mean and Mode Imputation

Assume record r has missing value r.X, and splitting variable is X.

- Simplest algorithm:
  - If X is numerical (categorical), impute the overall mean (mode)
- Improved algorithm:
  - If X is numerical (categorical), impute the mean(X|t.C) (the mode(X|t.C))

Surrogate Splits (CART)

Assume record r has missing value r.X, and splitting predicate is P_X.

- Idea: Find splitting predicate $Q_{X'}$ involving another variable X' != X that is most similar to P_X.
  - Similarity sim(Q,P|D) between splits Q and P:
    Sim(Q,P|D) = |{r in D: P(r) and Q(r)}|/|D|
  - 0 <= sim(Q,P|D) <= 1
  - Sim(P,P) = 1

Surrogate Splits: Example

Consider splitting predicate X1 <= 1.

Sim((X1 <= 1), (X2 <= 1)|D) = (3+4)/10

Sim((X1 <= 1), (X2 <= 2)|D) = (6+3)/10

(X2 <= 2) is the preferred surrogate split.

X1	X2	Class
1	1	Yes
1	1	Yes
1	1	Yes
1	2	Yes
1	2	Yes
1	2	No
2	2	No
2	3	No
2	3	No
2	3	No

Tutorial Overview

- Part I: Classification Trees
  - Introduction
  - Classification tree construction schema
  - Split selection
  - Pruning
  - Data access
  - Missing Values
  - Evaluation
  - Bias in split selection

(Short Break)

- Part II: Regression Trees

Choice of Classification Algorithm?

- Example study: (Lim, Loh, and Shih, Machine Learning 2000)
  - 33 classification algorithms
  - 16 (small) data sets (UC Irvine ML Repository)
  - Each algorithm applied to each data set
- Experimental measurements:
  - Classification accuracy
  - Computational speed
  - Classifier complexity

Experimental Setup

Algorithms:

- Tree-structure classifiers (IND, S-Plus Trees, C4.5, FACT, QUEST, CART, OC1, LMDT, CAL5, T1)
- Statistical methods (LDA, QDA, NN, LOG, FDA, PDA, MDA, POL)
- Neural networks (LVQ, RBF)

Setup:

- 16 primary data sets, created 16 more data sets by adding noise
- Converted categorical predictor variables to 0-1 dummy variables if necessary
- Error rates for 6 data sets estimated from supplied test sets, 10-fold cross-validation used for the other data sets

Results

Rank	Algorithm	Mean Error	Time
1	Polyclass	0.195	3 hours
2	Quest Multivariate	0.202	4 min
3	Logistic Regression	0.204	4 min
6	LDA	0.208	10 s
8	IND CART	0.215	47 s
12	C4.5 Rules	0.220	20 s
16	Quest Univariate	0.221	40 s
...			

- Number of leaves for tree-based classifiers varied widely (median number of leaves between 5 and 32 (removing some outliers))
- Mean misclassification rates for top 26 algorithms are not statistically significantly different, bottom 7 algorithms have significantly lower error rates

Tutorial Overview

- Part I: Classification Trees
  - Introduction
  - Classification tree construction schema
  - Split selection
  - Pruning
  - Data access
  - Missing Values
  - Evaluation
  - Bias in split selection

(Short Break)

- Part II: Regression Trees

Bias in Split Selection for ES

Assume: No correlation with the class label.

- Question: Should we choose Age or Car?
- Answer: We should choose both of them equally likely!

Age	Yes	No
20	15	15
25	15	15
30	15	15
40	15	15

Car	Yes	No
Sport	20	20
Truck	20	20
Minivan	20	20

Formal Definition of the Bias

- Bias: "Odds of choosing X_1 and X_2 as split variable when neither X_1 nor X_2 is correlated with the class label"
- Formally:
  $Bias(X_1,X_2) = Log_{10}(P(X_1,X_2)/(1-P(X_1,X_2))$,
  $P(X_1,X_2)$: probability of choosing variable X_1 over X_2

We would like: $Bias(X_1,X_2) = 0$ in the Null Case

Formal Definition of the Bias (Contd.)

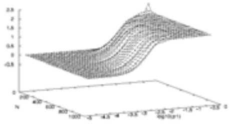
- Example: Synthetic data with two categorical predictor variables
  - X_1: 10 categories
  - X_2: 2 categories
- For each category: Same probability of choosing "Yes" (no correlation)

Car	Yes	No
Car1		
Car2		
Car3		
…		
Car10		

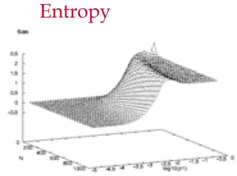
State	Yes	No
CA		
NY		

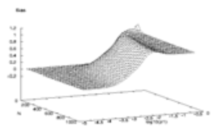
Evidence of the Bias

Gini



Entropy





Gain Ratio

One Explanation

Theorem: (Expected Value of the Gini Gain)

Assume:

- Two classlabels
- n: number of categories
- N: number of records
- p1: probability of having classlabel "Yes"

Then: E(ginigain) = 2p(1-p)*(n-1)/N

Expected ginigain increases linearly with number of categories!

Bias Correction: Intuition

- Value of the splitting criteria is biased under the Null Hypothesis.
- Idea: Use p-value of the criterion: Probability that the value of the criterion under the Null Case is as extreme as the observed value

Method:

1. Compute criterion (gini, entropy, etc.)
2. Compute p-value
3. Choose splitting variable

Correction Through P-Value

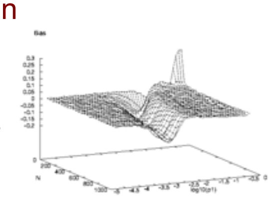
- New p-value criterion:
  - Maintains "good" properties of your favorite splitting criterion
  - <u>Theorem</u>: The correction through the p-value is nearly unbiased.

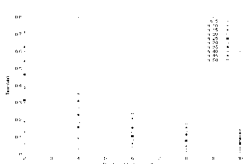
Computation:

1. Exact (randomization statistic; very expensive to compute)
2. Bootstrapping (Monte Carlo simulations; computationally expensive; works only for small p-values)
3. Asymptotic approximations (G^2 for entropy, Chi^2 distribution for Chi^2 test; don't work well in boundary conditions)
4. Tight approximations (cheap, often work well in practice)

Tight Approximation

- Experimental evidence shows that Gamma distribution approximates gini-gain very well.
- We can calculate:
  - Expected gain:
    E(gain) = 2p(1-p)*(n-1)/N
  - Variance of gain:
    Var(gain) = $4p(1-p)/N^2[(1-6p-6p^2) * (\text{sum } 1/N_i - (2n-1)/N) + 2(n-1)p(1-p)]$





Problem: ES and Missing Value

Consider a training database with the following schema: $(X_1, \ldots, X_k, C)$

- Assume the projection onto (X_1, C) is the following:

  $\{(1, Class1), (2, Class2), (NULL, Class_{13}), \ldots, (NULL, Class_{1N})\}$
  (X_1 has missing values except for the first two records)

- Exhaustive search will very likely split on X_1!

Concluding Remarks Part I

- There are many algorithms available for:
  - Split selection
  - Pruning
  - Data access
  - Handling missing values
- Challenges: Performance, getting the "right" model, data streams, new applications