Machine Learning

Decision Tree

Lecturer: Duc Dung Nguyen, PhD.
Contact: nddung@hcmut.edu.vn

Faculty of Computer Science and Engineering
Hochiminh city University of Technology

Contents

Decision-Tree Learning

Introduction

- Decision Trees
- TDIDT: Top-Down Induction of Decision Trees

ID3

- Attribute selection
- Entropy, Information, Information Gain
- Gain Ratio
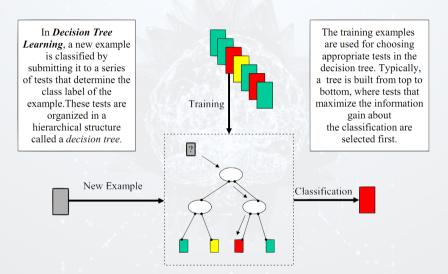
C4.5

- Numeric Values
- Missing Values
- Pruning

Regression and Model Trees

Decision-Trees

A decision tree consists of

- **Nodes**: test for the value of a certain attribute
- **Edges**: correspond to the outcome of a test and connect to the next node or leaf
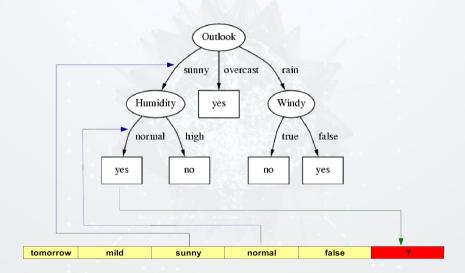- **Leaves**: terminal nodes that predict the outcome

Decision-Trees

In **Decision Tree Learning**, a new example is classified by submitting it to a series of tests that determine the class label of the example. These tests are organized in a hierarchical structure called a *decision tree*.

The training examples are used for choosing appropriate tests in the decision tree. Typically, a tree is built from top to bottom, where tests that maximize the information gain about the classification are selected first.

Training

New Example

Classification

Decision-Trees

Day	Temperature	Outlook	Humidity	Windy	Play Golf?
07-05	hot	sunny	high	false	no
07-06	hot	sunny	high	true	no
07-07	hot	overcast	high	false	yes
07-09	cool	rain	normal	false	yes
07-10	cool	overcast	normal	true	yes
07-12	mild	sunny	high	false	no
07-14	cool	sunny	normal	false	yes
07-15	mild	rain	normal	false	yes
07-20	mild	sunny	normal	true	yes
07-21	mild	overcast	high	true	yes
07-22	hot	overcast	normal	false	yes
07-23	mild	rain	high	true	no
07-26	cool	rain	normal	true	no
07-30	mild	rain	high	false	yes

today	cool	sunny	normal	false	?
tomorrow	mild	sunny	normal	false	?

Family of decision tree learning algorithms: TDIDT: Top-Down Induction of Decision Trees.
Learn trees in a Top-Down fashion:

- Divide the problem in subproblems.
- Solve each problem.

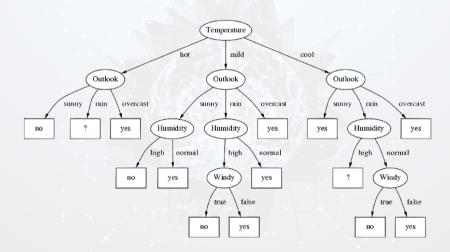## Decision-Trees: ID3 Algorithm

Function ID3

- **Input:** Example set S
- **Output:** Decision Tree DT

If all examples in S belong to the same class c, return a new leaf and label it with c. Else:

- Select an attribute A according to some heuristic function.
- Generate a new node DT with A as test.
- For each Value v_i of A, let S_i = all examples in S with $A = v_i$. Use ID3 to construct a decision tree DT_i for example set S_i. Generate an edge that connects DT and DT_i

A good attribute prefers attributes that split the data so that each successor node is as pure as possible.

In other words, we want a measure that prefers attributes that have a high degree of "order":

- Maximum order: All examples are of the same class
- Minimum order: All classes are equally likely

$\rightarrow$ **Entropy** is a measure for (un-)orderedness.
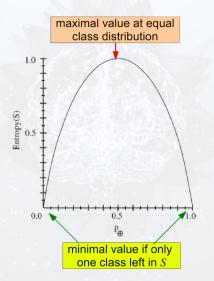
- S is a set of examples
- $p_\oplus$ is the proportion of examples in class $\oplus$
- $p_\ominus = 1 - p_\oplus$ is the proportion of examples in class $\ominus$

Entropy:

$$E(S) = -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus \tag{1}$$

Entropy can be easily generalized for $n > 2$ classes:

$$E(S) = -p_1 \log p_1 - p_2 \log p_2 ... - p_n \log p_n = -\sum_{i=1}^{n} p_i \log p_i \qquad (2)$$

p_i is the proportion of examples in S that belong to the i-th class.

Problem: Entropy only computes the quality of a single (sub-)set of examples.

Solution: Compute the weighted average over all sets resulting from the split weighted by their size.

$$I(S, A) = \sum_i \frac{|S_i|}{|S|} E(S_i) \tag{3}$$

When an attribute A splits the set S into subsets S_i, we then compute the average entropy and compare the sum to the entropy of the original set S.

Information Gain for Attribute A:

$$Gain(S, A) = E(S) - I(S, A) = E(S) - \sum_i \frac{|S_i|}{|S|} E(S_i) \qquad (4)$$

The attribute that maximizes the difference is selected.

Entropy is the only function that satisfies all of the following three properties:

- When node is pure, measure should be zero.
- When impurity is maximal (i.e. all classes equally likely), measure should be maximal.
- Measure should obey multistage property.

Problematic: attributes with a large number of values.
Subsets are more likely to be pure if there is a large number of different attribute values.
Information gain is biased towards choosing attributes with a large number of values.
This may cause several problems:

- Overfitting: selection of an attribute that is non-optimal for prediction
- Fragmentation: data are fragmented into (too) many small sets.

Intrinsic information of a split:

$$IntI(S, A) = -\sum_i \frac{|S_i|}{|S|} \log \frac{|S_i|}{|S|} \tag{5}$$

Modification of the information gain that reduces its bias towards multi-valued attributes. Takes number and size of branches into account when choosing an attribute. Corrects the information gain by taking the intrinsic information of a split into account.
Definition of Gain Ratio:

$$GR(S, A) = \frac{Gain(S, A)}{IntI(S, A)} \tag{6}$$

There are many alternative measures to Information Gain. Most popular altermative is Gini index.

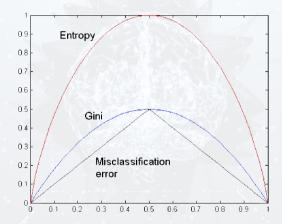Impurity measure (instead of entropy):

$$Gini(S) = 1 - \sum_i p_i^2 \tag{7}$$

Average Gini index (instead of average entropy / information):

$$Gini(S, A) = \sum_i \frac{|S_i|}{|S|}.Gini(S_i) \tag{8}$$

Gini Gain could be defined analogously to information gain but typically avg. Gini index is minimized instead of maximizing Gini gain.

For an algorithm to be useful in a wide range of real-world applications it must:

- Permit numeric attributes
- Allow missing values
- Be robust in the presence of noise
- Be able to approximate arbitrary concept descriptions (at least in principle)

Standard method: binary splits
Unlike nominal attributes, every attribute has many possible split points and computationally more demanding.
Solution is straightforward extension:

- Evaluate info gain (or other measure) for every possible split point of attribute
- Choose "best" split point
- Info gain for best split point is info gain for attribute

Efficient computation needs only one scan through the values!

- Linearly scan the sorted values, each time updating the count matrix and computing the evaluation measure
- Choose the split position that has the best value

Cheat		No		No		No		Yes		Yes		Yes		No		No		No		No	
								Taxable Income													
Sorted Values →		60		70		75		85		90		95		100		120		125		220	
Split Positions →		55		65		72		80		87		92		97		110		122		172		230
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		_0.300_		0.343		0.375		0.400		0.420	

- Splitting (multi-way) on a nominal attribute exhausts all information in that attribute.
- Not so for binary splits on numeric attributes! Numeric attribute may be tested several times along a path in the tree.
- Disadvantage: tree is hard to read
- Remedy: pre-discretize numeric attributes, or use multi-way splits instead of binary ones.

If an attribute with a missing value needs to be tested:

- split the instance into fractional instances (pieces)
- one piece for each outgoing branch of the node
- a piece going down a branch receives a weight proportional to the popularity of the branch
- weights sum to 1

Info gain or gain ratio work with fractional instances, use sums of weights instead of counts. During classification, split the instance in the same way. Merge probability distribution using weights of fractional instances

The smaller the complexity of a concept, the less danger that it overfits the data. Thus, learning algorithms try to keep the learned concepts simple.

Based on statistical significance test. Stop growing the tree when there is no statistically significant association between any attribute and the class at a particular node.
Most popular test: chi-squared test. Only statistically significant attributes were allowed to be selected by information gain procedure.
Pre-pruning may stop the growth process prematurely: early stopping.

Decision-Trees: Post-Pruning

Learn a complete and consistent decision tree that classifies all examples in the training set correctly .

As long as the performance increases

- Try simplification operators on the tree

- Evaluate the resulting trees

- Make the replacement the results in the best estimated performance
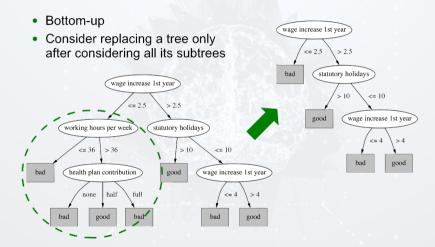
then return the resulting decision tree.

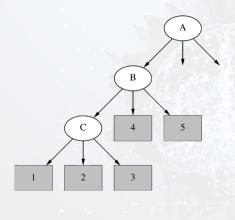Two subtree simplification operators

- Subtree replacement
- Subtree raising

- Bottom-up
- Consider replacing a tree only after considering all its subtrees
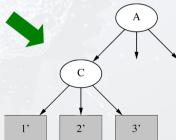
- Delete node B
- Redistribute instances of leaves 4 and 5 into C

Prune only if it does not increase the estimated error.
Reduced Error Pruning:

- Use hold-out set for pruning
- Essentially the same as in rule learning

- Split training data into a growing and a pruning set
- Learn a complete and consistent decision tree that classifies all examples in the growing set correctly
- As long as the error on the pruning set does not increase, try to replace each node by a leaf, evaluate the resulting (sub-)tree on the pruning set then make the replacement the results in the maximum error reduction.
- Return the resulting decision tree.

Decision Lists

- An ordered list of rules
- The first rule that fires makes the prediction
- can be learned with a covering approach

Decision Graphs

- Similar to decision trees, but nodes may have multiple predecessors

Each decision tree can be converted into a rule set. A decision tree can be viewed as a set of non-overlapping rules and typically learned via divide-and-conquer algorithms (recursive partitioning) Transformation of rule sets / decision lists into trees is less trivial

- Many concepts have a shorter description as a rule set
- Low complexity decision lists are more expressive than low complexity decision trees
- Exceptions: if one or more attributes are relevant for the classification of all examples

Regression Task: the target variable is numerical instead of discrete.
Two principal approaches

- Discretize the numerical target variable
- Adapt the classification algorithm to regression data

Differences to Decision Trees (Classification Trees)

- Leaf Nodes: Predict the average value of all instances in this leaf
- Splitting criterion: Minimize the variance of the values in each subset
- Termination criteria: Lower bound on standard deviation in a node and lower bound on number of examples in a node
- Pruning criterion: Numeric error measures, e.g. Mean-Squared Error