Decision trees

Special Course in Computer and Information Science II

Adam Gyenge

Helsinki University of Technology

6.2.2008

Introduction

Outline:

- Definition of decision trees
- ID3
- Pruning methods

Bibliography:

- J. Ross Quinlan: Induction of Decision Trees. Machine Learning 1(1): 81-106 (1986)
- J. Ross Quinlan: Simplifying Decision Trees. International Journal of Man-Machine Studies 27(3): 221-234 (1987)
- T. Mitchell: Machine Learning. McGraw Hill (1997)

Decision trees

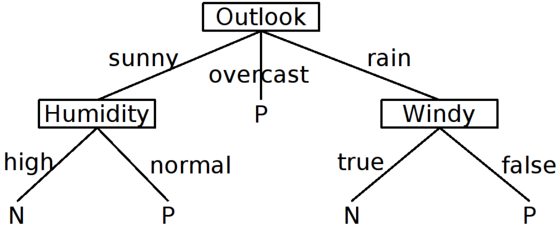
- There are objects in the world, they have many attributes.
- The attributes can take their values from a given set (type: category, integer or real).
- A decision tree, using the attributes, gives a category for the object (*classification*).
- Each non-leaf node of the tree is a test on an attribute, having a child for each result of the test.
- For an incoming object, we shall first do the test in the root, then follow the route down to the leaves.
- Each leaf is a category, if we reach one of them, then the category of the object is the category belonging to that leaf.

A simple example

Whether to play tennis or not. Attributes:

- outlook $\in$ {sunny, overcast, rain}
- humidity $\in$ {high, normal}
- windy $\in$ {true, false}

A possible tree:



Decision rules

A path from the root to a leaf: a decision rule.

$$\textit{if} \;\; T_1 \wedge T_2 \wedge \cdots \wedge T_n \;\; \textit{then} \;\; Y = y_i$$

Where T_j is a test on the j-th attribute (it may be 'always true') and y_i is the i-th category.
The decision tree represents a consistent set of decision rules.

Learning a decision tree 1

- Input: a training set of objects, whose class is known.
- Output: a decision tree, that can categorize **any** object $\longrightarrow$ *induction*
- Now, for simplicity, there will be only two classes (*Positive* and *Negative*), and all the attributes are category type

Learning a decision tree 2

- If there are two objects with identical attributes but different categories, no correct tree exists
- There can be several trees that classify the entire training set correctly
- Aim: construct the simplest tree
- We expect, that this classifies correctly more objects outside the training set

ID3

Recursive algorithm:

1. If all the training examples are from the same class, then let this be a category node and RETURN. Otherwise:
2. If there are no more attributes left, then let this be a category node using majority voting and RETURN. Otherwise:
3. Choose the 'best' attribute for the current node
4. For each value of the attribute create a child node
5. Split the training set: assign each example to the corresponding child node
6. For each node:
   6.1 If the node is empty, then let it be a category node using majority voting in the examples of it's parents
   6.2 If the node is not empty, then call this algorithm on it

Choosing the best attribute 1

Which is the best attribute?

- We want the classes of examples in the child-nodes to be homogeneous
- We assume, that the training set represents well the original population:

$$Pr(C = Positive) = \frac{p}{p+n}$$

$$Pr(C = Negative) = \frac{n}{p+n}$$

where

- C is the class attribute
- p is the number of samples with positive class
- n is the number of samples with negative class

Choosing the best attribute 2

Entropy:

- The expected number of bits, required to encode the class labels as messages, drawn randomly from the samples (S)

$$H(S) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

- The less entropy means less uncertainty $\rightarrow$ the more homogeneous the sample is

Choosing the best attribute 3

- Assume attribute A taking V different values
- $\{S_1, \dots, S_v\}$ are the sample sets belonging to each value
- The average entropy of the sorted sample set is:

$$E(S, A) = \sum_{i=1}^{v} \frac{|S_i|}{|S|} H(S_i)$$

- The information gained by choosing A as branching attribute is:

$$gain(A) = H(S) - E(S, A)$$

- The best attribute: for which $gain(A)$ is maximum, or (since $H(S)$ is the same for each attribute) $E(S, A)$ is minimum.

A simple example

No	Outlook	Temperature	Humidity	Windy	Class
1	sunny	hot	high	false	N
2	sunny	hot	high	true	N
3	overcoast	hot	high	false	P
4	rain	mild	high	false	P
5	rain	cool	normal	false	P
6	rain	cool	normal	true	N
7	overcast	cool	normal	true	P
8	sunny	mild	high	false	N
9	sunny	cool	normal	false	P
10	rain	mild	normal	false	P
11	sunny	mild	normal	true	P
12	overcast	mild	high	true	P
13	overcast	hot	normal	false	P
14	rain	mild	high	true	N

Calculations

There are 9 positive and 5 negative examples $(9 + 5 = 14)$.

$$H(S) = -\frac{9}{14}\log\frac{9}{14} - \frac{5}{14}\log\frac{5}{14} = 0.94$$

For the attribute 'Outlook':

- sunny: 2 positive, 3 negative (2+3=5), $H(S_{sunny}) = 0.971$
- overcast: 4 positive, 0 negative (4+0=4), $H(S_{overcast}) = 0$
- rain: 3 positive, 2 negative (3+2=5), $H(S_{rain}) = 0.971$

$E(S, Outlook) =$

$$\frac{5}{14}H(S_{sunny}) + \frac{4}{14}H(S_{overcast}) + \frac{5}{14}H(S_{rain}) = 0.694$$

Gain of attributes

The information gain of the 'Outlook' attribute:

$$gain(outlook) = 0.940 - 0.694 = 0.246$$

Similarly:

$$gain(temperature) = 0.029$$

$$gain(humidity) = 0.151$$

$$gain(windy) = 0.048$$

Result: 'Outlook' is the best attribute for the root node

Problem of irrelevant attributes

- There may be some attribute, that are irrelevant for the decision
- In these cases, the information gain is small (although not 0)
- We can define a threshold for the gain (absolute or percentage)
- If there is no attribute to exceed the threshold, we stop the recursion

Problem with information gain

- A is an attribute with a set of possible values: $\{A_1, \ldots, A_v\}$
- Create a new attribute B by splitting one of the values into two (eg. A_v splits into B_v and B_{v+1}, proportion does not matter)
- It can be shown, that $gain(B) \geq gain(A)$ always
- So ID3 prefers attributes with more values $\rightarrow$ builds flat trees
- It's better to use the gain ratio as attribute selecting function
- $IV(A) = \sum_{i=1}^{v} \frac{p_i+n_i}{p+n} \log \frac{p_i+n_i}{p+n}$

$$gain\ ratio(A) = \frac{gain(A)}{IV(A)}$$

Problem of overfitting

- The training data may contain noise
- We want to learn the general distribution, to reduce classification error on any data
- Error of hypothesis h on training data: $error_{train}(h)$
- Error of hypothesis h on entire distribution: $error_D(h)$
- Hypothesis h **overfits** the training data, if there is an alternative hypothesis $h' \in H$, for which

  $$error_{train}(h) < error_{train}(h')$$

  but

  $$error_D(h) > error_D(h')$$

Avoiding overfitting

Pruning:

- Prepruning: stop growing, when the data split is not significant (covered earlier, problem of irrelevant attributes)
- Postpruning: grow full tree, and then substitute some subtrees by single nodes. A few methods:
  1. Reduced Error Pruning
  2. Cost-Complexity Pruning
  3. Pessimistic Pruning
- We usually split the input set into 3 parts
  1. Training set: used for constructing the tree
  2. Pruning set: independent from training set, used by some pruning methods
  3. Test set: estimation of accuracy on a real data

Reduced Error Pruning

Algorithm:

1. Let S be a subtree in T
2. T': S is substituted with the best leaf (majority voting in training set)
3. E is the number of missclassified items of T on the pruning set
4. E' is the number of missclassified items of T' on the pruning set
5. If E' $\leq$ E, then prune S
6. Repeat this, until possible

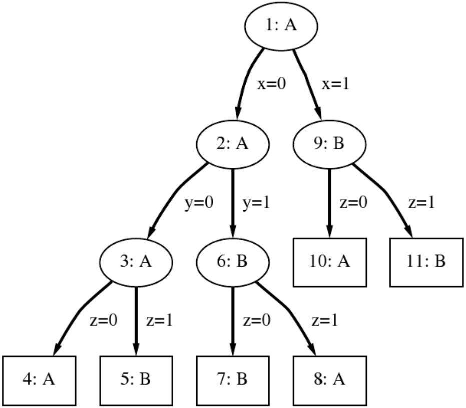
What is the order of the nodes to consider?

- Bottom-up

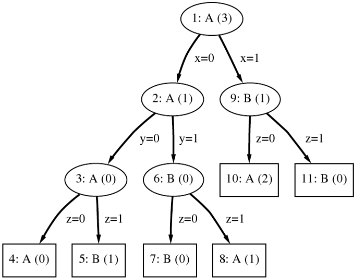
Example of REP

An example tree and pruning set:

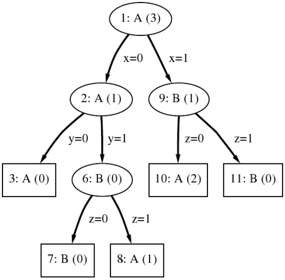
x	y	z	class
0	0	1	A
0	1	1	B
1	1	0	B
1	0	0	B
1	1	1	A



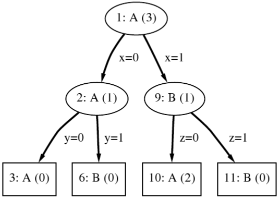
Example of REP



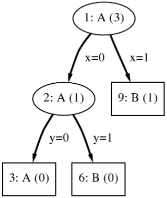
(a)



(b)



(c)



(d)

Cost-complexity pruning

Considers

- error on training set
- error on pruning set
- size of the tree

Tries to minimize error and complexity

Notations

- T is a decision tree
- N is the size of the training set used for generating T
- L(T) is the number of leaves in T
- E is the number of missclassified training examples
- Cost-complexity (or total cost) = Error cost + Cost of complexity
- $CC(T) = \frac{E}{N} + \alpha \cdot L(T)$
- α is the cost of complexity per leaf

CC Pruning

- S is a subtree of T, we substitute it with a single leaf by majority voting beetwen its descendant leaves
- M = (the new number of misclassified training examples) - E
- The new tree has $L(T) - L(S) + 1$ leaves.
- If $CC(T_{old}) = CC(T_{new})$, then

$$\alpha = \frac{M}{N \cdot (L(S) - 1)}$$

CC Pruning - Algorithm

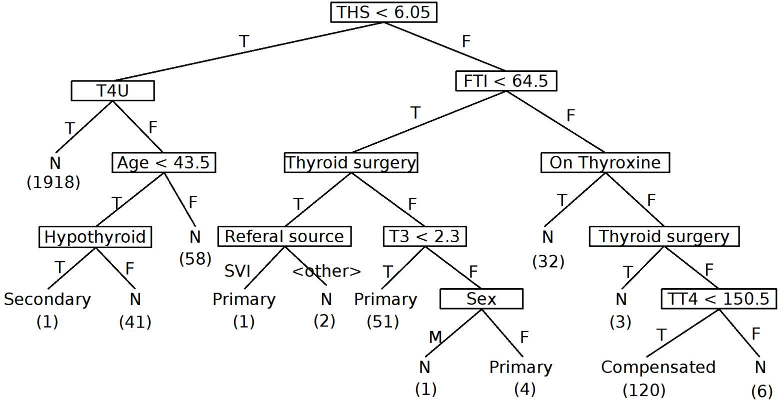
Algorithm:

1. Compute α for each node,
2. Find the minimum, prune subtrees having this value
3. Repeat steps 1, 2 until one leaf is left, this gives a series of trees: $T_0, \ldots, T_k$
4. Standard error (N' is the size of the *pruning* set, and E' is the smallest error of all the trees on the *pruning* set):

$$se = \sqrt{\frac{E' \cdot (N' - E')}{N'}}$$

5. Choose the smallest tree, whose observed error on the pruning set does not exceeds $E' + se$ (=not very far from the minimum error)

A medical example



- 4 categories: *primary, secondary, compensated, negative*
- The numbers of training examples, used for generating the tree, are shown in parenthesis

Example of CCP

Let's consider the subtree of the node [T4U].

- There is only one example, which is not *negative*
- If we replace this subtree, with a leaf (of course with *negative* label), then:
  - M=1
  - L(S)=4
  - N=2018
  - $\alpha = \frac{1}{2018 \cdot 3} = 0.00013$, this will be the least in the whole tree
- So at first, we shall replace this subtree by a leaf labelled as negative

Pessimistic Pruning

Notations again:

- N is the number of *training* examples
- K of them corresponds to a specific leaf
- J of them is missclassified, if we use a majority elected label (J = K - number of majority examples)
- Estimation of error on the leaf: $\frac{J}{K}$
- A better one (with the continuity correction of binomial distribution): $\frac{J+1/2}{K}$
- Expected error for K unseen samples: $J + 1/2$

Pessimistic Pruning

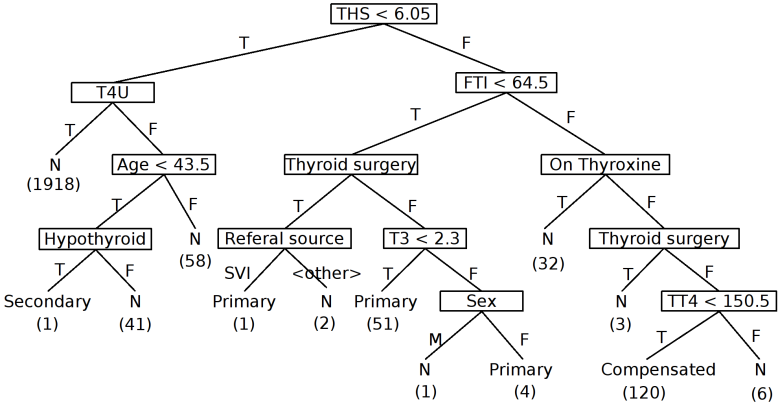
If we think pessimisticly:

- S is a subtree, contains L(S) leaves
- $\sum J$ and $\sum K$ are the sums of J and K over these leaves
- Expected error on S for $\sum K$ unseen samples: $\sum J + L(S)/2$
- Let E be the number of misclassified training examples, if we substitute S by majority vote
- If $E + 1/2 \leq (\sum J + L(S)/2) + se$ then prune S (se is the standard error, as before)

Algorithm:

1. Top-down checking of nodes (all none-leaf nodes are examined just once)
2. If possible, then substitute subtree with majority labelled leaf

A medical example - revisited



Example of PP

For the node [T4U]:

- $\sum K = 2018$
- $\sum J = 0$
- $L(S) = 4$
- Estimate of error on $S = 0 + 4/2 = 2$
- Standard error $= \sqrt{\frac{2 \cdot (2018-2)}{2018}} = 1.41$
- Result of election: *negative*, $E = 1$
- $1 + 1/2 < 2.0 + 1.41 \Rightarrow$ substitute subtree of [T4U] with [negative] leaf

Summary

- Decision trees for classification
- ID3 uses entropy based information gain for selecting best attribute to split on
- Problem of irrelevant attributes
- Overfitting can be avoided by postpruning
  1. Reduced error pruning
  2. Cost-complexity pruning
  3. Pessimistic pruning

Thank you for your attention. Questions?