Decision trees

Data Mining course
Master in Information Technologies
Enginyeria Informàtica

Tomàs Aluja
Decision trees in Data Mining

• The central task in DM is prediction (classification if response is categorical or regression if response is continuous)

• There are plenty of methods (models) to perform prediction: linear regression, logistic regression, discriminant analysis, Naive Bayes, Knn, Neural networks, SVM, …

• The decision trees are a non parametric method to perform predictions. Its popularity comes from the simplicity of the results (everyone can understand them) and their direct operationally
Purpose

To segment the given data in order to find homogeneous groups with respect to the response variable.

Visual output

Trees may differ:
- Type of the response variable (con/cat)
- Type of explanatory variables (bin, nom, ord, con)
- Binary or multi-way
- Split criterion (information, gini, ...)
- Stop criterion (pre-pruning, post-pruning)
Algorithm (top-down)

1. To set all individuals in the root node
2. Evaluate all possible splits and find the optimal partition in children nodes.
3. For every child node: Decide if we stop the process or we go back to step 2.

We need
• a split criterion
• a stop criterion (if prepruning)
How many splits can we make in a node

- Splits are defined by the number and the type of the explanatory variables and the type of the tree

<table>
<thead>
<tr>
<th></th>
<th>Multi way</th>
<th>Binary tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Nominal</td>
<td>1</td>
<td>$2^{q-1}-1$</td>
</tr>
<tr>
<td>Ordinal</td>
<td>1</td>
<td>$q-1$</td>
</tr>
<tr>
<td>Continuous</td>
<td>$n_t-1$</td>
<td>$n_t-1$</td>
</tr>
</tbody>
</table>

★ Attention

★ Attention
Antecedents

● AID (Sonquist y Morgan, 1964)
  – Response: Continuous
  – Binary tree, but can be extended to multiway
  – Split criterion: Fisher’s F
  – Stop criterion: threshold upon the p.value of the split

● CHAID (Kass, 1980)
  – Response: Categorical
  – Binary tree, but can be extended to multiway
  – Split criterion: $\chi^2$
  – Stop criterion: threshold upon the p.value of the split
AID Automatic Interaction Detection
(Sonquist & Morgan 1963)

AID spit criterion based in the wellknown decomposition of variance, between the nodes and within the nodes. The interest is to maximize the between part.

$$\sum_{i=1}^{n} (y_i - \bar{y})^2 = \sum_{k=1}^{q} n_{t_k} (\bar{y}_{t_k} - \bar{y}) + \sum_{k=1}^{q} \sum_{i \in t_k} (y_{it_k} - \bar{y}_{t_k})^2$$

$$F = \frac{\sum_{k=1}^{q} n_{t_k} (\bar{y}_{t_k} - \bar{y})}{q-1} \sim F_{q-1,n,q-1}$$

The (categorical) variable giving the most significative $F$ defines the optimal split.
CHAID (Kass, 1980)

CHAID split criterion: based in the Chi-square statistic computed crossing the response variable with the tentative partition defined by a explanatory variable.

The most significant is the Chi-square, more different is the distribution of the response in the parent node respect to the children nodes.

The (categorical) variable giving the most significative $\chi^2$ defines the optimal split.

\[
\chi^2 = \sum_{k=1}^{q} \sum_{j=1}^{m} \left( \frac{n_{jk} - n_k \frac{n_j}{n}}{n_k \frac{n_j}{n}} \right)^2
\]

$q$: Number of child nodes (=2 if binary tree)

$m$: number of response classes
The CART solution


- Just perform Binary trees
- Unifies the categorical and continuous responseunder the same framework.
  - Classification tree
  - Regression tree
- Any kind of explanatory variable
- Split criterion: Impurity of the node
- Post pruning (without stop criterion)
- Delivers honest estimates of the quality of a tree
Notation

- $n$: number of individuals.
- $t$: a node of the $T$ tree
- $t_r$: the right child of $t$
- $t_l$: the left child of $t$
- $n_t$: individuals of node $t$
- $n_{jt}$: individuals of node $t$ belonging to the response class $j$
- $p(j/t)$: probability of class $j$ in node $t$
- $T_t$: subtree descendant of $t$
Split criterion

Measure of the quality of a node: the impurity

\[ i(t) = \begin{cases} 
    \varphi(p_j / t) & j = 1, \ldots, m \\
    \psi(y_i / t) & i = 1, \ldots, n_t 
\end{cases} \]

\[ i(t) = \max \quad \text{if} \quad p_j / t = \text{cte.} \]

\[ i(t) = 0 \quad \text{if} \quad p_j / t = 0 \; \forall j \neq k \; p_k / t = 1 \]

\[ i(t) \geq \alpha i(t_r) + (1 - \alpha) t(i_l) \; \quad 0 \leq \alpha \leq 1 \]
**Impurity of a node**

For categorical responses:

- Gini

\[ i(t) = \sum_{i \neq j} p(j/t)p(i/t) = 1 - \sum_j p_j^2 \]

- Information (Entropy)

\[ i(t) = -\sum_j p(j/t) \log_2 p(j/t) \]

For continuous responses:

- Variance

\[ i(t) = \frac{\sum_{i \in t} (y_i - \bar{y}_t)^2}{n_t} \]
Selection of the optimal partition

Maximize the decrement of impurity between the parent and its children

\[
\Delta i(t) = i(t) - \frac{n_{tl}}{n_t} i(t_l) - \frac{n_{tr}}{n_t} i(t_r)
\]
Importance of variables and dealing with missing data

- **Competing partitions:**
  They are the other splits with similar decrement of impurity in a node. They can be used to rank the importance of a variables to predict the response in the whole tree.

- **Substitute partitions:**
  They are splits most similar to the optimum respect to the assignment of individuals between the children. They are used as substitute in case that one individual has a missing value for the variable defining best split.
Stop criterion

There is no stop criterion. CART uses post-pruning, it builds a maximum tree and prunes the non interesting branches.

Absolute maximum tree: Tree with all pure leaves.
Selection of the optimal tree
(from the maximum tree)

We need to define how good (or bad) is a tree:

Obvious criterion = *Probability of misclassification (= cost of the tree)*

How can we compute the Cost of the tree:

We assign every leave to the response class with maximum $p(j/t)$

\[
p(j/t) = \frac{p(j,t)}{p(t)} = \frac{\pi_j n_j/n}{n/n}
\]

If classes $j$ autorepresentatives

\[
if \ \pi_j = \frac{n_j}{n} \ \ \ \ \ \ \ \ p(j/t) = \frac{n_j}{n_t}
\]
Cost of the tree

of a node:

\[ r(t) = 1 - \max_j p(j/t) \]

If there are missclassification costs

\[ r(t) = \min_i \sum_j c(i/j) p(j/t) \]

Cost of the tree:

(decreasing with size)

\[ R(T) = \frac{\sum_{t \in T} p(t) r(t)}{r(root)} \times 100 \]

Criterion to optimize:

\[ \text{Min } R(T) \]
Cost of a regression tree

- In a regression tree, every individual is assigned with the mean value of the leave.

- Cost of the tree = residual variance

\[
\begin{align*}
    r(t) &= \frac{1}{n_t} \sum_{i=1}^{n_t} (y_{it} - \bar{y}_t)^2 \\
    &= \frac{1}{n_t} \sum_{i=1}^{n_t} (y_{it} - \bar{y}_t)^2
\end{align*}
\]
Penalization for complexity

Criterion to optimize: \[ \min \left( R(T) + \alpha |T| \right) \]

Where \( \alpha \) is the complexity parameter, expresses the penalization for building large trees

But, how can we determine \( \alpha \)?
We can build the trees for increasing values $\alpha$, obtaining a sequence of optimal trees (of increasing size) (CART does this pruning from the maximal tree in a more intelligent way)

\[ T_{\text{max}}, T_{\text{max}} - T_{t_0}, T_{\text{max}} - T_{t_0} - T_{t_1}, \ldots, 1 \]

Each subtree is optimum ($\text{min } R(T)$), within the subtrees of complexity $\mathcal{I}(T)$
Selection of the optimal tree

• To obtain a reliable tree, we need to test it with independent data from the one used in the learning phase. Hence, we divide the total data at random in one part (2/3) for training and the remaining for testing.

• We build the sequence of trees using the training data and we compute its goodness $R(T)$ using the test data

$$R^{ts}(T) = \frac{\sum_{t \in \overline{I}} p(t) r^{ts}(t)}{r^{ts}(\text{root})} \times 100$$
Select the tree with optimal \( R^{ts}(T) \)

(in the test sample)

Another way of putting the Cost relative to situation in the root (in this case we look for the maximum)

\[
R^{ts}(T)_{rel} = \frac{r^{ts}(\text{root}) - \sum_{t \in T} p(t)r^{ts}(t)}{r^{ts}(\text{root})} \times 100
\]

Cost
Selection of the optimal tree by crossvalidation

- We divide at random the data matrix in 10 parts (ten-fold cross validation). Let’s call $n_1$, $n_2$, ... the groups formed.
- We compute the maximal tree for the following data sets formed with $n-n_1$, $n-n_2$, ... individuals.
- For a sequence of $\alpha$ values, we compute the cost of the corresponding trees. Then, the $cv$ cost is:

$$R_{\alpha}^{vc}(T_{\alpha}) = \sum_{t=1}^{k} \left( \sum_{l \in T_{\alpha}} p_l(t) r_l^{vc}(t) \right) \times \frac{100}{k}$$

- 1se rule. To have a more prudent tree, it is possible to take, instead the minimum of $R_{\alpha}^{cv}(T)$, the tree corresponding to the minimum value of $R_{\alpha}^{cv}(T) + 1se$

$$Min \left( R_{\alpha}^{cv}(T_{\alpha}) + 1 \times desv\ tip(R_{\alpha}^{cv}(T_{\alpha})) \right)$$
Presenting the results

Selected tree

Table with ranked leaves

<table>
<thead>
<tr>
<th>N</th>
<th>Prob.</th>
<th>Individuos</th>
<th>%Acum.</th>
<th>Exitos</th>
<th>%Acum.</th>
<th>%Exitos</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>100.00</td>
<td>30</td>
<td>3.75</td>
<td>30</td>
<td>11.36</td>
<td>100.00</td>
</tr>
<tr>
<td>5</td>
<td>78.95</td>
<td>19</td>
<td>6.13</td>
<td>15</td>
<td>17.05</td>
<td>91.84</td>
</tr>
<tr>
<td>7</td>
<td>71.94</td>
<td>139</td>
<td>23.50</td>
<td>99</td>
<td>54.55</td>
<td>76.60</td>
</tr>
<tr>
<td>3</td>
<td>62.50</td>
<td>32</td>
<td>27.50</td>
<td>20</td>
<td>62.12</td>
<td>74.55</td>
</tr>
<tr>
<td>8</td>
<td>47.65</td>
<td>149</td>
<td>46.13</td>
<td>71</td>
<td>89.02</td>
<td>63.69</td>
</tr>
<tr>
<td>9</td>
<td>19.61</td>
<td>51</td>
<td>52.50</td>
<td>10</td>
<td>92.80</td>
<td>58.33</td>
</tr>
<tr>
<td>4</td>
<td>17.86</td>
<td>28</td>
<td>56.00</td>
<td>4</td>
<td>94.32</td>
<td>55.58</td>
</tr>
<tr>
<td>6</td>
<td>7.81</td>
<td>64</td>
<td>64.00</td>
<td>5</td>
<td>96.21</td>
<td>49.61</td>
</tr>
<tr>
<td>1</td>
<td>3.77</td>
<td>265</td>
<td>97.13</td>
<td>10</td>
<td>100.00</td>
<td>33.98</td>
</tr>
<tr>
<td>2</td>
<td>0.00</td>
<td>23</td>
<td>100.00</td>
<td>0</td>
<td>100.00</td>
<td>33.00</td>
</tr>
</tbody>
</table>

We rank the leaves according the predicted response.
Quality assessment of the tree

Error rate Given a threshold, compute the ratio of misclassified respect the total of instances

Global measures of quality:

Concentration curve

ROC curve

DM course. Decision Trees. T. Aluja
## Error rate

We apply the model in the test sample, and we classify every individual according to a threshold (usually = 0.5) on the outcome.

<table>
<thead>
<tr>
<th></th>
<th>Predicted class YES</th>
<th>Predicted class NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real class YES</td>
<td>$n_{TP}$</td>
<td>$n_{FN}$</td>
</tr>
<tr>
<td>Real class NO</td>
<td>$n_{FP}$</td>
<td>$n_{TN}$</td>
</tr>
</tbody>
</table>

\[
\text{Error rate} = \frac{n_{FN} + n_{FP}}{n}
\]

It depends on the chosen threshold
Concentration curves

- Represent how concentrated is the response variable in the data
  - Used in Economy

- Coordinates
  - y axis shows the accumulated percentage of true positives in the data
  - x axis shows the accumulated percentage of population

Measure of concentration: Quotient of area under the curve and area of maximum concentration
ROC curves

- Similar to the concentration curve
  - Stands for “receiver operating characteristic”, used in signal detection to show trade off between hit rate and false alarm rate over noisy channel

- Coordinates:
  - y axis shows the accumulated percentage of true positives in the data
  - x axis shows the accumulated percentage of false positives in the data

<table>
<thead>
<tr>
<th>Predicted class</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>TP: True positive</td>
<td>FN: False negative</td>
</tr>
<tr>
<td>No</td>
<td>FP: False positive</td>
<td>TN: True negative</td>
</tr>
</tbody>
</table>
The area under the ROC is equivalent to the area under the Concentration curve

\[ \text{Area Concentration} = 2 \times \text{Area ROC} - 1 \]
but be aware!

For small sample, model A is better, but for large sample model B is better.
Advantages of decision trees

- Very easy to interpret.
- The branches define the assignment rules. Results are directly operational.
- The branches simulate fairly well the human process for taking decisions.
- Minimizing the preprocess, they can work up to certain level of error and missing data.
- Detect automatically complex interactions among variables (no need to specify them).
- Computationally efficient (scalability guaranteed for … )
- But rough approximation to the response, since it approximates the response by leaps (discontinuities) (worse prediction error than other methods)
Decision trees as simple additive models

\[ f(x) = \alpha + \beta I(x_1 > \xi_1) + \delta I(x_2 > \xi_2) \]
How to improve the precision

Performing predictions from consensus bootstrap resampling.

Bagging:
- Extract $M$ bootstrap samples.
- Obtain the optimal tree for each bootstrap resample.
- Predict every individual by the mean of the $M$ trees if continuous response or by the majority vote if categorical response.

Boosting:
- Is equal to the previous, but with bootstrap resampling with probabilities not uniform, but proportional to the previous missclassification error (to force the classifier to focus on the difficult cases.)
Práctica en árboles de decisión

- Inadecuación del criterio de calidad de los árboles. La asignación de un nodo a una clase de respuesta depende de las probabilidades iniciales de cada clase de respuesta.

- El poder de discriminación de un árbol debe medirse relativa a la situación inicial de partida.

- El interés del árbol es la ordenación de la población por grupos según valores crecientes de la variable de respuesta. Estos grupos somos capaces de entenderlos y localizarlos en la base de datos.
Handling unbalanced data

- Sometimes, response classes have very unequal frequency
  - Attrition prediction: 97% stay, 3% attrite (in a month)
  - Medical diagnosis: 90% healthy, 10% disease
  - eCommerce: 99% don’t buy, 1% buy
  - Security: >99.99% of Catalans are not terrorists
- Similar situation with multiple classes
- Majority class classifier can be 97% correct, but useless
- CART cost is intended for balanced classes (it is hard to a minority class to become majority in a node).

- Need to balance response classes in the training data
Balancing the training data

Or if possible; weighing the non targets, in such a way that:

$$\sum_{targets} weights = \sum_{non\ targets} weights$$
Decision trees in R - CART

Library `rpart`

\[
p2 = \text{rpart(dict} \sim \text{edat} + \text{ratfin} + \text{tiptreb, data=dd,parms=list(prior=c(.50,.50), split='gini'))}
\]

\begin{verbatim}
> l6=rpart(medv~lstat)
> l6p=predict(l6)
> points(lstat,l6p,col="violet",pch=20)
> l6
n= 506

node), split, n, deviance, yval
     * denotes terminal node

1) root 506  42716.3000  22.53281
   2) lstat>=9.725 294  7006.2830  17.34354
      4) lstat>=16.085 144  2699.2200  14.26181
         8) lstat>=19.9  75  1214.0820  12.32533 *
         9) lstat< 19.9  69   898.1933  16.36667 *
      5) lstat< 16.085 150 16813.8200  29.72925
         6) lstat>=4.65 162  6924.4230  26.64630
            12) lstat>=5.495 134  5379.1940  25.84701 *
            13) lstat< 5.495  32  2109.5620  37.31562 *
         7) lstat< 4.65  50  3360.8940  39.71800
            14) lstat>=3.325 32  2109.5620  37.31562 *
            15) lstat< 3.325  18   738.3178  43.98889 *
\end{verbatim}
> pl = rpart(dict~edat + ratfin + tiptreb, data=dd[learn,])
> pl
n= 406
node), split, n, deviance, yval
  * denotes terminal node
1) root 406 101.490100 0.4950739
  2) ratfin>=70.5 264 62.359850 0.3825758
      4) tiptreb=altr,temp 46 4.456522 0.1086957 *
      5) tiptreb=auton,fixe 218 53.724770 0.4403670
  10) edat< 50.5 178 42.252810 0.3876404
      20) tiptreb=auton 55 10.909090 0.2727273 *
      21) tiptreb=fixe 123 30.292680 0.4390244
          42) edat< 25.5 21 3.238095 0.1904762 *
          43) edat>=25.5 102 25.490200 0.4901961 *
  11) edat>=50.5 40 8.775000 0.6750000 *
  3) ratfin< 70.5 142 29.577460 0.7042254
      6) tiptreb=altr,auton,temp 61 15.049180 *
      7) tiptreb=fixe 81 12.222220 0.8148148 *
> plot(pl)
> text(pl, use.n=T)
> pl1=predict(pl, data=dd[learn,])
> pl1p=NULL
> pl1p[pl1<0.5]=0
> pl1p[pl1>=0.5]=1
> table(dict[learn],pl1p)
pl1p
  0   1
0 150 55
1 74 127

> plt = predict(pl,newdata=dd[-learn,])
> pltp=NULL
> pltp[plt<0.5]=0
> pltp[plt>=0.5]=1
> table(dict[-learn],pltp)
pltp
  0   1
0 63 35
1 44 59

\[ P_{\text{acierta}} = 60.7\% \]

\[ P_{\text{acierta}} = 68.2\% \]
rpart  package:rpart
Recursive Partitioning and Regression Trees

rpart(formula, data, weights na.action = na.rpart, method, parms, control, cost, ...)

formula: a formula, as in the 'lm' function.
data: an optional data frame in which to interpret the variables named in the formula
weights: optional case weights.
na.action: The default action deletes all observations for which 'y' is missing, but keeps those in which
one or more predictors are missing.
method: one of "anova", "poisson", "class" or "exp". If 'method' is missing then the routine tries to
make an intelligent guess.
parms=list(prior=c(.85,.15), split='information')
control=rpart.control(minsplit=20, minbucket=round(minsplit/3), cp=0.01,maxcompete=4,
maxsurrogate=5, usesurrogate=2, xval=10, maxdepth=30, ...)

minsplit: the minimum number of observations that must exist in a node, in order for a split to be
attempted.
minbucket: the minimum number of observations in any terminal '<leaf>' node.
ep: complexity parameter.
maxcompete: the number of competitor splits retained in the output.
maxsurrogate: the number of surrogate splits retained in the output.
usesurrogate: 1= use surrogates, in order, to split subjects missing the primary variable; if all surrogates
are missing the observation is not split. 2= if all surrogates are missing, then send the
observation in the majority direction.
xval: number of cross-validations
maxdepth: Set the maximum depth of any node of the final tree, with the root node counted as depth 0
C4.5 (ID3, Quinlan 1983)

**Multi-way trees**

Node impurity: Entropy (Information)

*Number of bits to predict an event of a distribution.*

\[
I_t = \sum_{k=1}^{q} p_k \log_2 p_k
\]

\[
p_k = \frac{n_{tk}}{n_t}
\]

**Split criterion:**

Gain of information

Maximize GAIN

\[
GAIN = I_{\text{parent}} - \sum_{j=1}^{m_j} \frac{n_{tj}}{n_t} I_j
\]

DM course. Decision Trees. T. Aluja
A random event provides information, the more when the event is less probable. We can measure this amount of information by the number of bits needed to locate the event in its distribution.

Examples:
Location of a cell in a square of 16 cells: 4 (Why?)
Transmission of a character of the alphabet (26 letters suppose + 6 punctuation): How much is it

The amount of information needed to locate an event is:

\[ I_w = \log_2 n \]

Let's suppose that the events are grouped according a categorical variable \( A \), with levels \( A_k \). If we know that an event belong to \( A_k \). Then the amount of information is:

\[ I_{w \in A_i} = \log_2 n_k \]

Hence, the information of a modality \( A_k \) is:

\[ I_{A_k} = I_w - I_{w \in A_k} = -\log_2 \left( \frac{n_k}{n} \right) \]

and on average:

\[ I_A = -\sum_{k=1}^{q} P(A_k) \log_2 P(A_k) \]
C4.5 (ID3, Quinlan 1983)

The GAIN depends on the number of modalities of the splitting variable, thus the results depend largely on the actual coding of variables.

Remedy. Relativize respect the intrinsic information of the splitting variable which depends on its number of modalities:

\[
GAIN_{rel} = \frac{I_{parent} - \sum_{j=1}^{m} \frac{n_{tj}}{n_t} I_j}{\text{Intr.Info}(t)}
\]

Intr.Info(t) = entropy of the splitting variable

If the variable is continuous, the algorithm detects the cutoff point where the gain is highest. If the variable is categorical, there is a partition in as many classes as children nodes have the explanatory variable.
C4.5

Criterion stop: It stops when the relative gain is below a certain threshold.
Allocation of each node to the response modality with maximum probability
Pruning the branches if the probability of misclassification in descending nodes does not improve the probability of misclassification in the current node in a significant way.