Decision Trees

K. Gibert[(1)]

[(1)]*Department of Statistics and Operation Research*

Knowledge Engineering and Machine Learning group at

Intelligent Data Science and Artificial Intelligence Research Center

Universitat Politècnica de Catalunya, Barcelona

karina.gibert@upc.edu

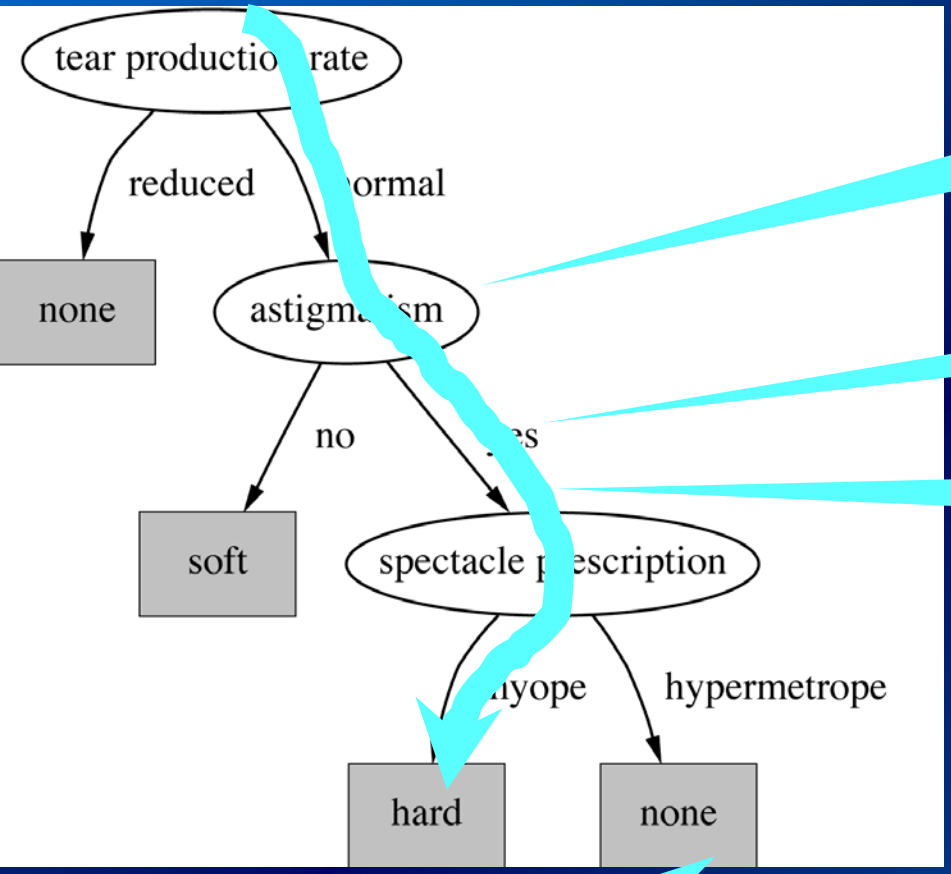
https://www.eio.upc.edu/homepages/karina

Decision Trees

- **Predict a class variable**
- Visual output self-understandable

Decision trees

Model the process of deciding to which class belongs a new example of the domain



Internal nodes: Variables

Branches: modalities

Paths: Inducted decisions

Associated classification rules

If tear prod rate=normal ***and***

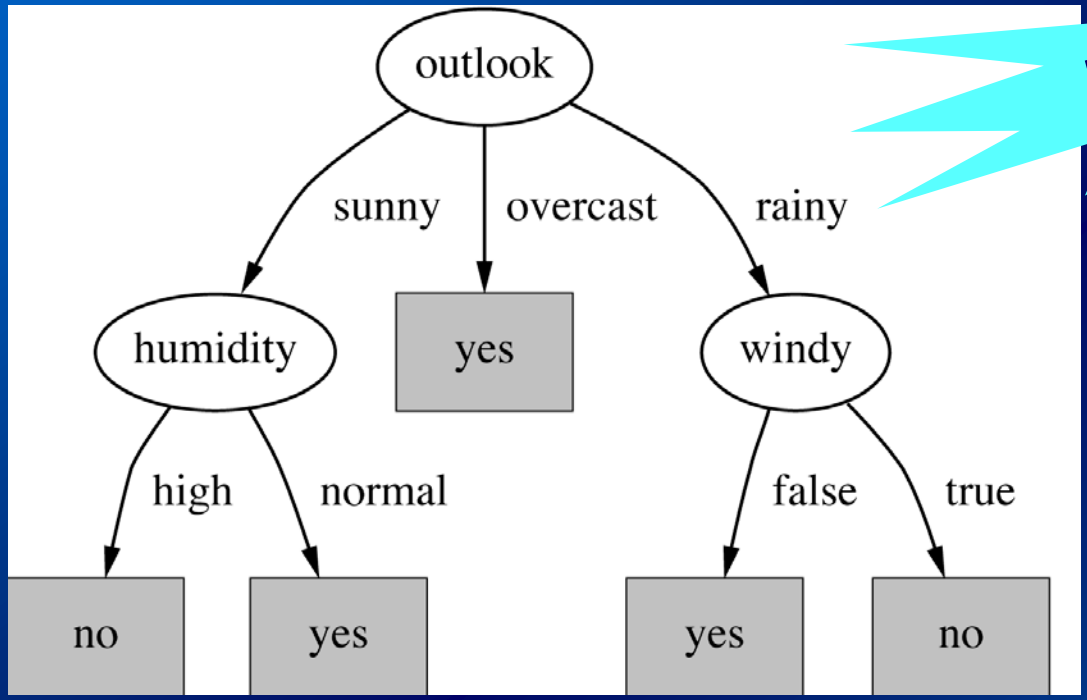
astigmatism= yes ***and***

spectacle presciption=myope

then hard lent

Leafs: Predicted class

Decision Tree: example 2



Easy to understand

Decision tree for the weather data

Construction:

- Top-down strategy
- Which spliting variable?
- Which branches?
- How to label leafs?

Leafs: Voting

Decision Trees

- Many models:
  - For numerical/qualitative response variables
  - For binary, nominal, ordinal, numerical explanatory variables
  - Binary tree/multiway-tree
  - Split criterion (information gain, gini index...)
  - Stop criterion (pre-prunning, post-prunning)

ID3, CART (previous AID, CHAID...)

Decision Trees

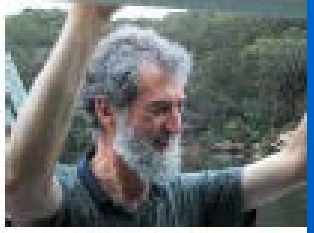
- General algorithm:
  - Set all individuals at the root node
  - Optimize split in children nodes
  - For every child: decide between
    - Continue splitting (repeating process)
    - Stop

Decision Trees

- Number of splits in a node:

  Depends from type of variable and type of tree

	Multi way	*Binary tree*
Binary	*1*	*1*
Nominal	*1*	$2^{q-1}-1$
Ordinal	*1*	$q-1$
Continuous	n_t-1	n_t-1

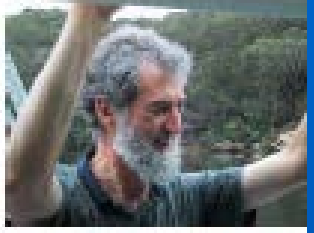

Ross Quinlan
American

ID3 algorithm

- **ID3** ≡ Induction Decision Tree [Quinlan, 1979]
- Machine Learning Technique
- Decision Tree Induction
- Top-Down strategy
- Qualitative variables
- Multi-way trees
- Don't repeat variables
- From a set of **examples (instances** + belonging class) build up the *best* decision tree which explains the instances

Criteria

- **Compactness**→ the more compact the better
- **Goodness** → predictive/discriminant ability


Ross Quinlan
American

C4.5 algorithm

- C4.5: Refines ID3 [Quinlan, 1986]
- Machine Learning Technique
- Decision Tree Induction
- Top-Down strategy
- Qualitative variables
- Multi-way trees
- Uses impurity ratio
- Heuristics for prunning
- From a set of **examples (instances** + belonging class) build up the *best* decision tree which explains the instances

Criteria

- **Compactness**→ the more compact the better
- **Goodness** → predictive/discriminant ability

ID3

Which variable in decision node?

It is a *greedy (voraz)* algorithm

the *best* attribute at each step

is the most discriminant one (potentially more useful)

Maximize a certain G(I,X)

ID3: Notation

- I ≡ set of examples ≡ i={$i_{1..}$ i_n}
- X ≡ variable with values {$x_{1..}$ x_{nx}}
- P ≡ set of classes ≡ {$c_{1.....}$ c_p}
- # ≡ Cardinality

$X^{-1}(x_i)$ ≡ set of elements in I with X=xi

ID3: splitting criteria

- Select variable X which *maximizes* the information gain

$$G(I, X) = H(I) - H(I,X)$$

where P response variable (class)

Divide by H(I)
Avoid effect of n_x

Information of parent node

$$\mathbf{H(I)} = -\sum_{c \in P} p(c|I) * \log_2 p(c|I)$$

Empirical probability of being in class C

$$\mathbf{p(c|I)} = \#c \,/\, \#I$$

Entropy of splitting by X

$$\mathbf{H(I, X)} = \sum_{i=1:n_x} p_X(x_i) * H(X^{-1}(x_i))$$

Probability that one example has the value x_i for the variabe X

$$\mathbf{p_X(x_i)} = \#X^{-1}(x_i) \,/\, \#I$$

Termination criteria

- All instances in a node belong to same class
- A maximum depth of the tree is achieved

  (maxdepth control parameter in R)
- Splits do not open a minimum number of branches

  (only for n-ary trees)
- Next split produces too small leafs

  (minsplit control parameter in R)
- Improvement of termination function is not enough

  (cp control parameter in R, termination fumction = splitting criterion)

Post-prunning criteria

- Avoids overfitting and redundancy
- Evaluate the tree with/without the branch
  - If performance is maintained, prune
  - Performance can be measured in different ways
    - Different error functions
    - Statistical tests

Algorithm BuildDecisionTree($\mathcal{D}, \mathcal{A}$)

input: training data $\mathcal{D}$, set $\mathcal{A}$ of available attributes
output: a decision tree matching $\mathcal{D}$, using all or a subset of $\mathcal{A}$

```
1     if all elements in D belong to one class
2       return node with corresponding class label
3     elseif A = ∅
4       return node with majority class label in D
5     else
6       select attribute A ∈ A which best classifies D
7       create new node holding decision attribute A
8       for each split v_A of A
9         add new branch below with corresponding test for this split
10        create D(v_A) ⊂ D for which split condition holds
11        if D(v_A) = ∅
12          return node with majority class label in D
13        else
14          add subtree returned by calling
                BuildDecisionTree(D(v_A), A\{A})
15        endif
16      endfor
17      return node.
18    endif
```

ID3: EXAMPLE (9)

	Eye Colour	Hair Colour	Height	Class
E1	Blue	Blonde	Tall	C+
E2	Blue	Brown	Medium	C+
E3	Brown	Brown	Medium	C-
E4	Green	Brown	Medium	C-
E5	Green	Brown	Tall	C+
E6	Brown	Brown	Low	C-
E7	Green	Blonde	Low	C-
E8	Blue	Brown	Medium	C+

ID3: example (10)

I={E1,....E8} C+={E1, E2, E5, E8) C-={E3,,E4,E5,E7}

G(I, Eye-Color) = H(I) - H(I,Eye-Color)

$$\mathbf{H(I)} = -\sum_{c_i \in C} p(c_i|I) * \log_2 p(c_i|I) = \mathbf{-p(c+|I)*} \log_2 p(c+|I) \mathbf{-p(c-|I)*} \log_2 p(c-|I)$$

p(C+|I) = p (C-|I)= 1/2

$$H(I) = \mathbf{-1/2 \log_2 1/2 -1/2 \log_2 1/2 = 1}$$

$$\mathbf{H(I, Eye\text{-}Color)} = \sum_{x_i} p_X(x_i) * H(X^{-1}(x_i)) = \sum_{x_i} p_{Eye\text{-}Color}(x_i) * H(Eye\text{-}Color^{-1}(x_i))$$

$P_X(x_1) = P_{Eye\text{-}Color}(Blue) = 3/8$

$H(Eye\text{-}Colour^{-1}(Blue)) = - p(C+|Blue) * \log_2 p(C+|Blue) -- p(C-|Blue) * \log_2 p(C-|Blue)$

$= - 1 \log_2 1 - 0 \log_2 0$

$H(I, Eye\text{-}colour) = \mathbf{3/8} (-1 \log_2 1 - 0 \log_2 0) + \mathbf{2/8} (-0 \log_2 0 - 1 \log_2 1) +$

$\mathbf{3/8} (-1/3 \log_2 1/3 - 2/3 \log_2 2/3) = 0.344$

ID3: example (10)

$$H(I) = -1/2 \log_2 1/2 - 1/2 \log_2 1/2 = 1$$

$$H(I, \text{Eye-colour}) = \frac{3}{8}(-1 \log_2 1 - 0 \log_2 0) + \frac{2}{8}(-0 \log_2 0 - 1 \log_2 1) + \frac{3}{8}(-1/3 \log_2 1/3 - 2/3 \log_2 2/3) = 0.344$$

$$H(I, \text{Hair-colour}) = \frac{2}{8}(-1/2 \log_2 1/2 - 1/2 \log_2 1/2) + \frac{6}{8}(-3/6 \log_2 3/6 - 3/6 \log_2 3/6) = 1$$

$$H(I, \text{Height}) = \frac{2}{8}(-1 \log_2 1 - 0 \log_2 0) + \frac{4}{8}(-1/2 \log_2 1/2 - 1/2 \log_2 1/2) + \frac{2}{8}(-0 \log_2 0 - 1 \log_2 1) = 0.5$$

ID3: example (11)

G(I, Eye-colour) = 1 - 0.366 = **0.656**

G(I, Hair-colour) = 1 - 1 = 0

G(I, Height) = 1- 0.5 = 0.5

	Hair-colour	Height	Class
E4	Brown	Medium	C-
E5	Brown	Tall	C+
E7	Blonde	Low	C-

ID3: example (12)

H(Green) = -1/3 $\log_2$ 1/3 -2/3 $\log_2$ 2/3 = 0,918

(E_5) (E_4,E_7)

Blonde

H(Green, Hair-colour) = **1/3** (- $\log_2$ - 1 $\log_2$ 1) +

2/3 (-1/2 $\log_2$ 1/2 - 1/2 $\log_2$ 1/2) = 2/3

H(Green, Height) = 1/3 (-0 $\log_2$ 0 - 1 $\log_2$ 1) +

1/3 (-1 $\log_2$ 1 - 0 $\log_2$ 0) +

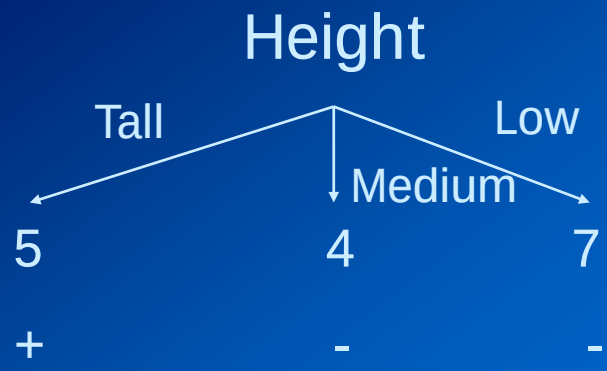
1/3 (-0 $\log_2$ 0 - 1 $\log_2$ 1) = 0

G(Green, Hair-colour) = 0,918 - 0,666 = 0,252

G(Green, Height) = 0,918 - 0 = **0,918**

ID3: example

Recursive Implementation



ID3: example (14)

Eye-colour = Blue $\rightarrow$ C+

Eye-colour = Brown $\rightarrow$ C-

Eye-colour = Green $\wedge$ Height = Tall $\rightarrow$ C+

Eye-colour = Green $\wedge$ Height = Medium $\rightarrow$ C-

Eye-colour = Green $\wedge$ Height = Low $\rightarrow$ C-

ID3: example (15)

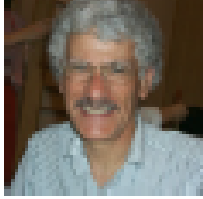
Optimizing the set of rules

(Eye-colour = Blue) or (Eye-colour = Green $\wedge$ Height <> Tall) $\rightarrow$ C+

(Eye-colour = Brown) or (Eye-colour = Green $\wedge$ Height = Tall) $\rightarrow$ C-

Leo Breiman,
American, 1928-2005

The CART solution



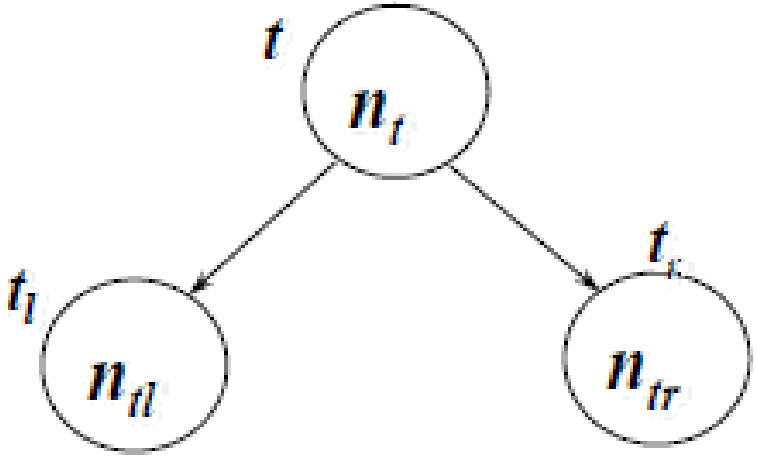
Jerome Friedman,
American

Developed 1974-1984 by 4 statistics professors: Leo Breiman (Berkeley), Jerry Friedman (Stanford), Charles Stone (Berkeley), Richard Olshen (Stanford). Distributed by Salford Systems http://salford-systems.com/

- 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

Selection of the optimal partition

Maximize the decrement of impurity between the parent and its children



$$\Delta i(t) = i(t) - \frac{n_{tl}}{n_t} i(t_l) - \frac{n_{tr}}{n_t} i(t_r)$$

Impurity of a node

For categorical responses:

- Gini

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

- Information (Entropy)

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

For continuous responses:

- Variance

$$i(t) = \frac{\sum_{i \in t}(y_i - \bar{y}_t)^2}{n_t}$$

Gini(Node labelled as C) = p(c|I) * (1-p(c|I)) two cl

Missclassification tax

Gini(Node labelled as C) = 1-$\sum_{c' \in P,\, c \neq c}$ p(c'|I)^2 , more than 2 classes

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 $p(j/t)$

$$p(j/t) = \frac{p(j,t)}{p(t)} = \frac{\pi_j \, n_{tj}/n_j}{n_t/n}$$

If classes j autorepresentatives $\quad if\ \pi_j = \frac{n_j}{n} \quad p(j/t) = \frac{n_{tj}}{n_t}$

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)

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

Criterion to optimize: $Min\ R(T)$

Cost of a regression tree

- In a regression tree, every individuals is assigned with the mean value of the leave.
- Cost of the tree = residual variance

$$r(t) = \frac{1}{n_t} \sum_{i=1}^{n_t} (y_{it} - \bar{y}_t)^2$$

Penalization for complexity

Criterion to optimize: $Min\left(R(T)+\alpha\left|T\right|\right)$

Where α is the *complexity parameter*, expresses the penalization for building large trees

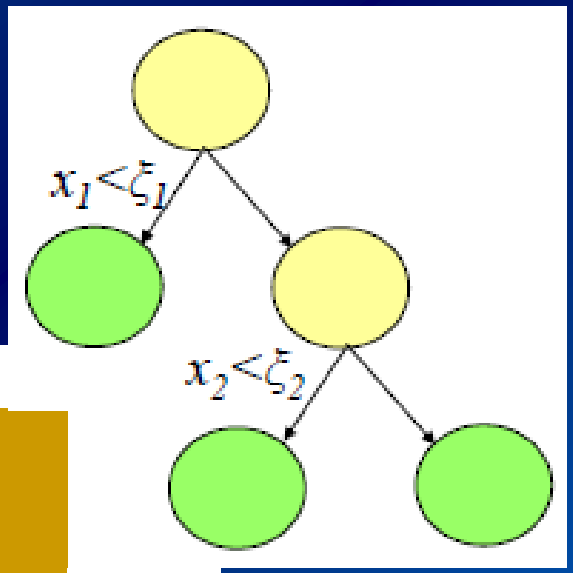
Run with increasing complexity and

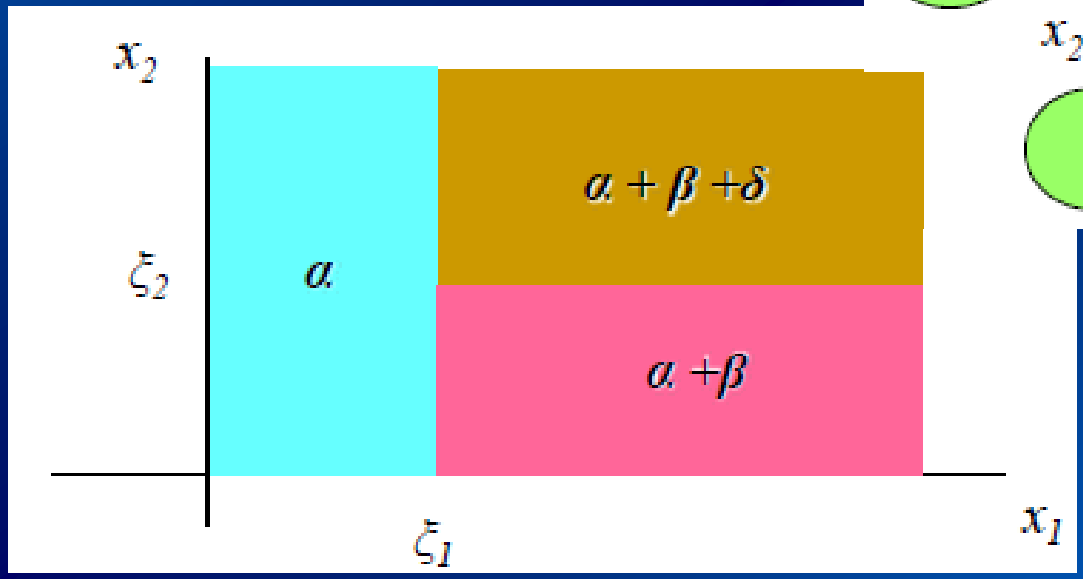
Get a sequence of optimal trees of increasing size....

Evaluate and select the best

Decision trees are simple additive models

$$f(x) = \alpha + \beta I(x_1 > \xi_1) + \delta I(x_2 > \xi_2)$$





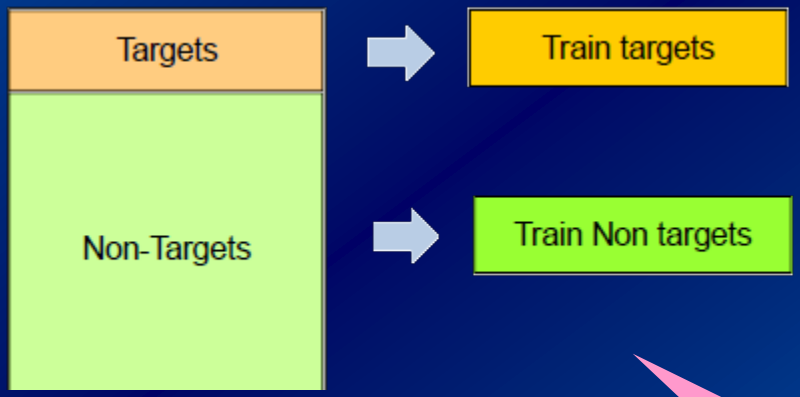
Unbalanced Data

- Unequal class frequencies
  - 99% don't buy, 1% buy
  - 90% healthy, 10% disease
- Getting a majority class of 97% accuracy is useless
- Decision Trees assume balanced classes

  (minority class hardly become a majority in a node)
- Work with balanced traninig datasets

Unbalanced training Data

Bad performance



Otherwise, weight non targets:

$$\sum_{targets} weights = \sum_{non\ targets} weights$$

Keep Proportions

After Learning: Using

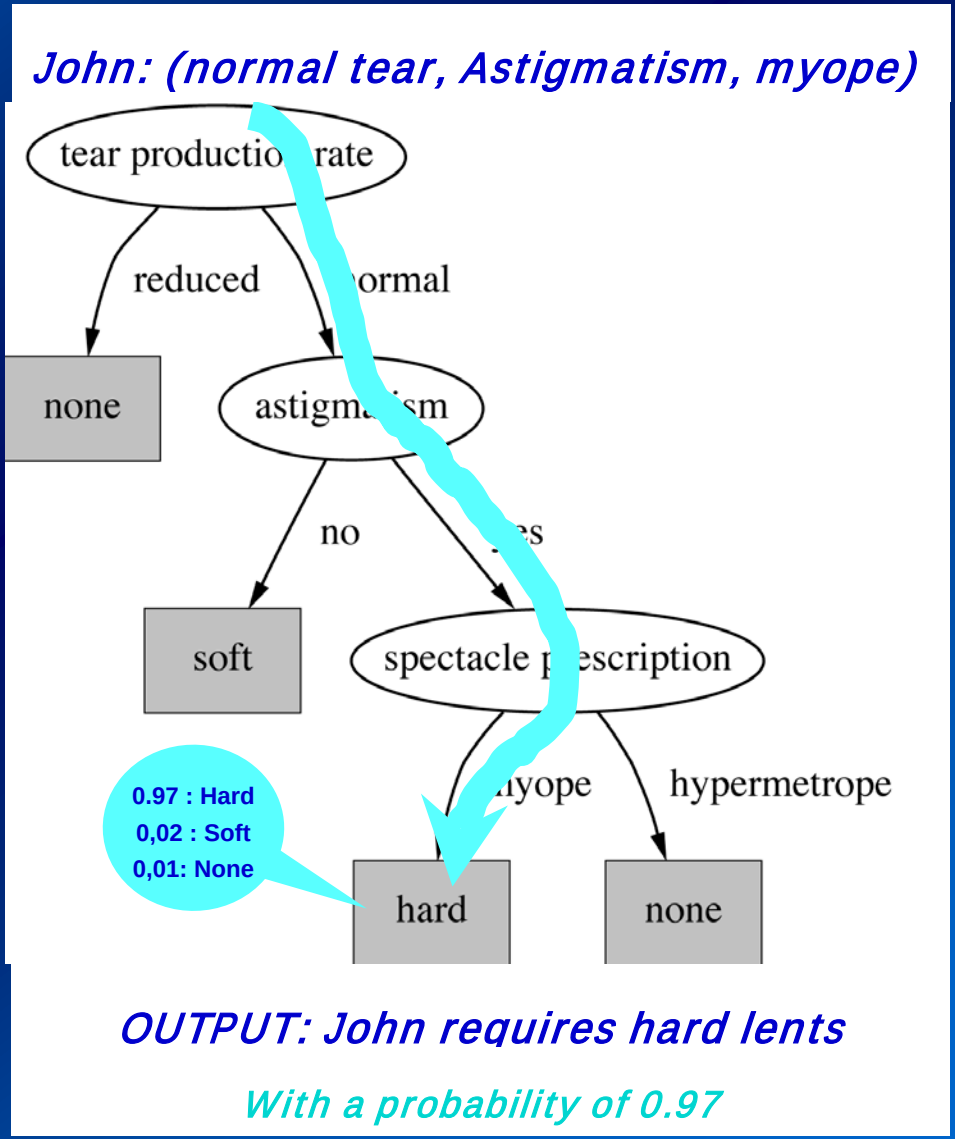
- Use the inducted DT to assign the class of a new object:

  PREDICTION

- Evaluate node-queries over the object and descend to a leaf

- Assign the label of the leaf as the class of the object

Evenctually, consider the purity of the leaf as the probability of the assignment



Validation in Decision Trees

Missclassification rate

Confusion matrix

Cross-validation

ROC curves: (DT provides single points for ROC space)

- use probability of assignment to run ROC
- or move some parameter (cp, minleaf....)

Confusion matrix and error rate

	Predicted class YES	Predicted class NO
Real class YES	n_{TP}	n_{FN}
Real class NO	n_{FP}	n_{TN}

Type-I error

Type-II error

$$Error\ rate = \frac{n_{FN} + n_{FP}}{n}$$

Missclassification rate depends on DT parameters

cpTable

	CP	nsplit	rel error	xerror	xstd
1	0.30312557	0	1.0000000	1.0405435	0.02080410
2	0.04100900	1	0.6968744	0.7092905	0.01917514
3	0.02817402	2	0.6558654	0.6547873	0.01854656
4	0.02200789	5	0.5596869	0.6307159	0.01895873
5	0.01433087	6	0.5376790	0.6151649	0.01925346
6	0.01419604	7	0.5233481	0.6048818	0.01967947
7	0.01372608	8	0.5091521	0.6034818	0.01967728
8	0.01133242	9	0.4954260	0.5893790	0.01932271
9	0.01000000	11	0.4727612	0.5694968	0.01876309

Cp: complexity parameter

Rel error: relative missclassification rate of the tree at each cp

Xerror: Crossvalidated missclassification error

Xstd: standard deviation of crossvalidated missclassification error

Pros and cons of DT

- Easy to interpret
- Branches provide assignment rules (Results operationalized)
- Branches provide good support to decision making
- Automatic detection of complex interactions among variables
- Computationaly efficient .....
- But.... Rough approach to the response (approached by leaps (discontinuities)
- But... worse performance than other classifiers

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 intellegent 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.
cp: 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