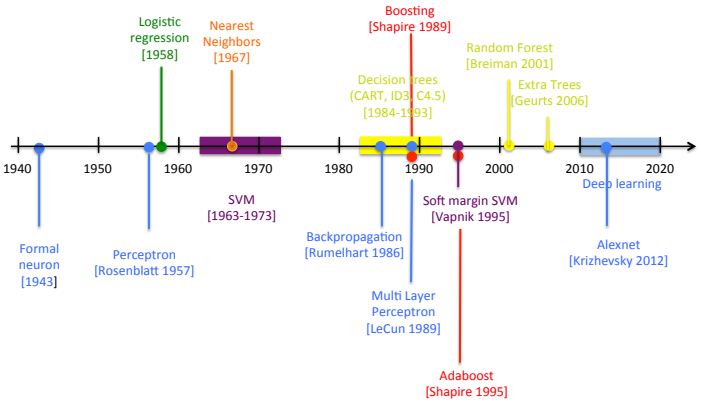
Supervised Learning
Decision Trees

Ewa Kijak

AS, M2RI, Université de Rennes 1

Partial history

- 1763 : Bayes Theorem underpinnings



Outline

First contact
Classification of diseases

Tests on the description

- Do you have fever ?
- Do you cough ?
- Do you have headaches ?
- Do you have muscle pain ?
- Do you have joint pain ?
- Do you have a sore throat ?
- Vomit you ?
- Vomit you blood ?
- Have you met some birds ?
- ...

First contact
Classification of diseases

Decision

- given the answers of the previous questions, you have contracted the *Avian influenza*.

Remarks

- how to formalize this process of classification...
- ... and s.t. it is cheap (fewest tests), understandable...

First contact

Why a decision tree ?

- organize the tests
- lower the average number of tests
- one tree for n classes

How to build a decision tree ?

- by supervised machine learning ; several existing algorithms
  - CART(Classification and Regression Tree) [Breiman et al. 84], binary, gini index
  - ID3 (Inductive Decision Tree) [Quinlan 86], nominal features only, entropy, information gain
  - C4.5, and C5, refinements of ID3 [Quinlan 93], numeric features, missing features, pruning

Definitions

Vocabulary

- node : contains a selector leading to one of the branches beneath the node
- selector : test on one feature, contained in a node
- leaf : end of branch, where the class is given

Coverage

- an instance of data is covered by a node (or a leaf) if its description leads from the root to this node (leaf)
- all data are covered by the root of the tree

Outline

Induction

Building Principle

- iteratively choose a feature and a test on this feature according to a given criterion
  - the one that best separates the classes
- at the end, we obtain a maximal tree $T_{\max}$ in which each leaf contains only data from one class
- this tree is pruned using a validation set

Remarks

- $T_{\max}$ is almost learning by-heart the data
- pruning allows to generalize

Building $T_{\max}$

Data

- from a training set S containing m examples with d features $\{A_i, i \in \{1..d\}\}$ and a class $\omega \in \{\omega_1, ..., \omega_C\}$
- we want a tree having no apparent error (ie empiric risk = 0)
- for now, we only consider binary features (cf. later for nominal or continuous features)

Building $T_{\max}$

algorithm Build_tree(X)
begin
if every elements in X have the same class **or** no more test possible **then**
  create a leaf labelled by the (majority) class
else
  choose the *best* selector to create a node separating the examples in X_d and X_g
  Build_tree(X_d)
  Build_tree(X_g)
end if
end

Building $T_{\max}$

Remarks

- when the tree is partially built, each node covers a subset of the training examples : those satisfying all the (binary) tests from the root to this node
- while this subset is not composed only by elements from the same class, the building continues
- **hot point : how to choose the best test...**

Example for T_{max}

Description

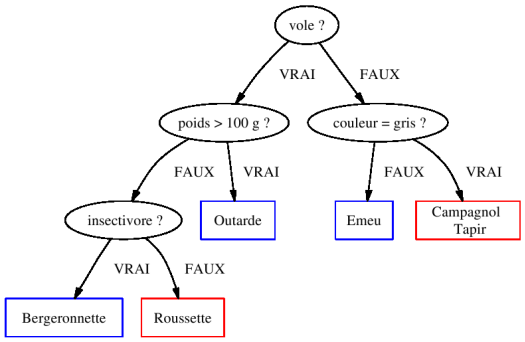
features					class
fly	weight	color	food	skin	animal
yes	1 kg	roux	granivore	plumes	Outarde
yes	20 g	gris et jaune	insectivore	plumes	Bergeronnette
no	100 kg	noir et blanc	omnivore	plumes	Emeu
no	5 g	gris	granivore	poils	Campagnol
no	40 kg	gris	herbivore	poils	Tapir
yes	60 g	noir	frugivore	poils	Roussette

2 classes : birds, mammals

Example for T_{max}

Goal : distinguish between birds (blue) and mammals (red)

- is it a good or bad T_{max} ?



Choosing the selector 1/4

Let W et A, be 2 discrete random variables taking their values in $\omega \in \mathcal{D}_W = \{\omega_1, \omega_2, \cdots, \omega_n\}$ et $a \in \mathcal{D}_A$.

Entropy - informal definition

- measure heterogeneity of a population with respect to a certain criterion ;
- also called *quantity of information* or *impurity* and written H (or sometimes E)
- if H is high, the system is heterogeneous, *i.e.* messy, disordered

Choosing the selector 2/4

Entropy - formal definition

- for a random variable W taking n distinct values :
- $H(W) = -\sum_{\omega \in \mathcal{D}_W} P(\omega) log P(\omega)$

Conditional entropy

$$\begin{aligned} H(W \mid A) &= \mathbb{E}_A[H(W \mid a)] = \sum_{a \in \mathcal{D}_A} H(W \mid a) P(a) \\ &= \sum_{a \in \mathcal{D}_A} P(a)[\sum_{\omega \in \mathcal{D}_W} -P(\omega \mid a) log(P(\omega \mid a))] \end{aligned}$$

Choosing the selector 3/4

Mutual information (Information gain)

- measure of the mutual dependence between the two variables :

$$I(W; A) = \sum_{\omega, a \in \mathcal{D}_W \times \mathcal{D}_A} P(\omega, a) log \frac{P(\omega, a)}{P(\omega) P(a)}$$

$I(W; A)$ is :

- minimal (null) if $P(\omega, a) = P(\omega) P(a) \ \ \forall (\omega, a) \in \mathcal{D}_W \times \mathcal{D}_A$, *i.e.* if W and A are independent
- maximal when the two distributions are perfectly correlated
- intricately linked to the concept of entropy of a random variable

Choosing the selector 4/4

The mutual information (expected information gain) $I(W; A)$ is the change in information entropy H of W from a prior state to a state that takes some information, as the state of the random variable A :

$$I(W; A) = H(W) - H(W \mid A)$$

Among the d features (v.a. $\{A_i\}_{i=1...d}$), we want the feature A_i that minimizes the conditional entropy (meaning that it has the highest correlation with the class distribution)

$$\text{Maximize } I(W; A) \leftrightarrow \text{Minimize } H(W \mid A)$$

The selected feature A_i is such that :

$$i^* = \arg\min_{i=1,\ldots,d} H(W \mid A_i)$$

Case of binary features 1/4

Notations :

- consider a node with m examples attached : m_1 examples from class ω_1, ... m_C from class ω_C
- consider a binary feature A splitting each m_j in two : G_j examples for $A = \text{true}$ and D_j for $A = \text{false}$

$A \backslash$class	ω_1	...	ω_C	
true	G_1	...	G_C	G
false	D_1	...	D_C	D

with $G = \sum_{i=1}^{C} G_i$ et $D = \sum_{i=1}^{C} D_i$

Case of binary features 2/4

Links with probabilities

- the values G_j/G and D_j/D are estimates of the probabilities $P(W = \omega_j | A = \text{true})$ and $P(W = \omega_j | A = \text{false})$
- the values G_j/m et D_j/m are estimates of the probabilities $P(W = \omega_j, A = \text{true})$ and $P(W = \omega_j, A = \text{false})$
- similarly, G/m and D/m are estimates of $P(A = \text{true})$ and $P(A = \text{false})$
- similarly, m_j/m is an estimate of $P(W = \omega_j)$

Case of binary features 3/4

Principle

- among the d features ($\{A_i\}_{i=1...d}$), we look for the one with the minimal conditional entropy $H(W|A)$ of the examples vs. the classes
- when splitting the m examples according to feature A, the amount of information brought by the fact that $A = \text{true}$ is

$$H(W|A = \text{true}) = -\sum_{j=1}^{C} \frac{G_j}{G} \log_2 \frac{G_j}{G}$$

- the total amount of information brought by A is

$$H(W|A) = \frac{G}{m} H(W|A = \text{true}) + \frac{D}{m} H(W|A = \text{false})$$

$$H(W|A) = -\sum_{j=1}^{C} \frac{G}{m} \frac{G_j}{G} \log_2 \frac{G_j}{G} + \frac{D}{m} \frac{D_j}{D} \log_2 \frac{D_j}{D}$$

Case of binary features 4/4

Summary

- for each feature A_i, build the table of the joint distribution of the examples attached to the current node

$A_i \setminus W$	ω_1	ω_2	total
true	3	2	5
false	1	2	3
total	4	4	8

- compute $H(W|A_i)$ (for each feature A_i) on the current node
- keep the feature with the smallest $H(W|A_i)$ as the selector for this node

Example

Can I play ?

- a boy, back at home after school, wants to know if he can go and play with his neighbor
- he has some experience, based on the decision of the 8 preceding days
- formally, this machine learning problem consists in finding a binary decision rule from 8 examples described by 4 features
- the 8 preceding days (*i.e.* the examples) are described in the following table

Example

Description of the examples

	my homework is done or not	mom is in good spirits	the weather is nice	my lunch is taken	Decision
1	done	false	true	false	Yes
2	not done	true	false	true	Yes
3	done	true	true	false	Yes
4	done	false	true	true	Yes
5	not done	true	true	true	No
6	not done	true	false	false	No
7	done	false	false	true	No
8	done	true	false	false	No

Example

Joint distributions

homework $\setminus W$	Yes	No	total
done	3	2	5
not done	1	2	3
total	4	4	8

lunch $\setminus W$	Yes	No	total
taken	2	2	4
not taken	2	2	4
total	4	4	8

spirits $\setminus W$	Yes	No	total
good	2	3	5
not good	2	1	3
total	4	4	8

weather $\setminus W$	Yes	No	total
nice	3	1	4
not nice	1	3	4
total	4	4	8

Example

Entropy computation

- $H(W|homework) = \frac{5}{8}H(W|homework = \text{done}) + \frac{3}{8}H(W|homework = \text{not done})$
  - with $H(W|homework = \text{done}) = -\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5}$
  - with $H(W|homework = \text{not done}) = -\frac{1}{3}\log\frac{1}{3} - \frac{2}{3}\log\frac{2}{3}$
- thus, $H(W|homework) \approx 0.95$
- similarly, we obtain $H(W|weather) \approx 0.80$, $H(W|spirits) \approx 0.95$, $H(W|lunch) = 1$

Example

First split (first node)

- thus, for the root, we choose the test *Is the weather nice ?*
- on the left branch (with the value true) we have the following examples (those having the feature *Is the weather nice = true*)

	my homework is done	mom is in good spirits	my lunch is taken	Decision
1	done	false	false	Yes
3	done	true	false	Yes
4	done	false	true	Yes
5	not done	true	true	No

Example

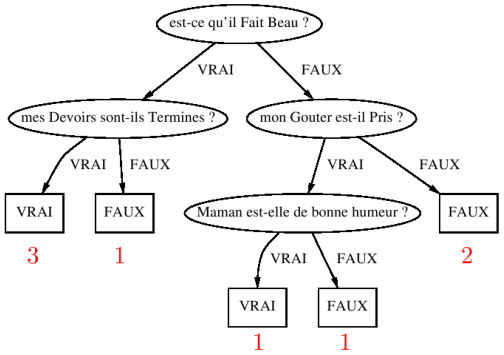
First split (first node)

- on the right branch (with value false) we have the following examples (those having the feature *Is the weather nice = false*)

	my homework is done	mom is in good spirits	my lunch is taken	Decision
2	not done	true	true	Yes
6	not done	true	false	No
7	done	false	true	No
8	done	true	false	No

Example

Final tree (T_{max})



Choosing the selector - conclusion

Other criteria

- many variants to entropy have been tested
- always the same principle (minimize a numerical criterion) and the same philosophy (the criterion represents impurity of the classes of the examples)

Criterion of Gini

- most known variant of entropy
- $Gini(X) = 1 - \sum_{i=1}^{n} p_i^2$

Nominal feature 1/3

Simple solution : back to the binary case

- to get back to the binary case : oppose one value vs. the others
  - consider a feature *color* that can take its value in the set {blue, red, green, yellow}, it's simple to break it into 4 binary features, like red-color which is true or false
- then, it is the same process than explained in the previous section : a k-value nominal feature is transformed into k binary features
- drawback : we forget the global signification of the feature

Nominal feature 2/3

Other solution

- directly compute the mutual information between the k-value feature and the class
- if this is the lowest value among the features, we create a non-binary node in the decision tree

Nominal feature 3/3

Example

- consider a feature A_i with $\mathcal{D}_{A_i} = \{blue, white, red\}$
- consider its distribution

$A_i \setminus W$	ω_1	ω_2	total
blue	2	3	5
white	4	0	4
red	3	2	5
total	9	5	14

- $H(W|A_i) = \frac{5}{14}\left(\frac{-2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) + \frac{4}{14}\left(\frac{-4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}\right) + \frac{5}{14}\left(\frac{-3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right)$

Continuous feature

Problem

- what can we do with a continuous numeric feature ?
- how to switch back to a binary decision problem ?

Solution

- since the number of examples is finite, the number of values for the feature is finite in practice
- moreover, if one sorts these values, the selector only consists in setting a threshold = binary test

Continuous feature

Solution in practice

- to sum up, a continuous feature X_i is processed as follows
  - sort the examples according to the values of the feature
  - look for the threshold $s(X_i)$ minimizing the entropy
  - test all the possible thresholds : for n examples, we have at most $n-1$ thresholds (*e.g.* medians between two consecutive values in the sorted list)

Remark

- to the contrary of the preceding cases, the feature is kept since it can be used, with another threshold, later in the tree

Continuous feature

Example

- *cf. Notes de cours de Ph. Preux*

Day	Outlook	Temperature	Humidity	Windy	Play tennis ?
1	Sunny	27.5	85	Weak	No
2	Sunny	25	90	Strong	No
3	Overcast	26.5	86	Weak	Yes
4	Rain	20	96	Weak	Yes
5	Rain	19	80	Weak	Yes
6	Rain	17.5	70	Strong	No
7	Overcast	17	65	Strong	Yes
8	Sunny	21	95	Weak	No
9	Sunny	19.5	70	Weak	Yes
10	Rain	22.5	80	Weak	Yes
11	Sunny	22.5	70	Strong	Yes
12	Overcast	21	90	Strong	Yes
13	Overcast	25.5	75	Weak	Yes
14	Rain	20.5	91	Strong	No

Continuous feature

Example

- suppose that the first node (*i.e.* the root) has the feature `Outlook` as selector
- let's focus on the branch for which `Outlook = Sunny`
- draw the table of examples concerned by this node
- consider the feature `Temperature` and find the best threshold (maximal information gain)

Continuous feature

Example

- table with the examples concerned with `Outlook = Sunny`

Day	Outlook	Temperature	Humidity	Windy	Play tennis ?
1	Sunny	27.5	85	Weak	No
2	Sunny	25	90	Strong	No
8	Sunny	21	95	Weak	No
9	Sunny	19.5	70	Weak	Yes
11	Sunny	22.5	70	Strong	Yes

Continuous feature

Example

- let's have a look at the feature `Temperature` ; we sort it by increasing order

Temperature	19.5	21	22.5	25	27.5
Day	9	8	11	2	1
Play tennis ?	Yes	No	Yes	No	No

- we can cut between 19.5 and 21, 21 and 22.5, 22.5 and 25, 25 and 27.5
- but since ex2 and ex1 have the same class, cutting between 25 and 27.5 cannot be optimal for the entropy
- we decide that the threshold chosen between 2 temperatures x and y is min(x,y) (we could take any value between x and y, *e.g.* moy(x,y))

Continuous feature

Example

- let's examine the threshold t=21 and compute $H(W|seuil_t = 21)$
- joint distribution table :

temp $\backslash W$	Yes	No	total
$t < 21$	1	0	1
$t \geq 21$	1	3	4
total	2	3	5

  - $H(W|t < 21) = -\frac{1}{1}\log_2\frac{1}{1} - 0$
  - $H(W|t \geq 21) = -\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4}$
- $H(W|seuil_t = 21) = \frac{1}{5}H(W|t < 21) + \frac{4}{5}H(W|t \geq 21)$

Continuous feature

Example

- let's now consider the threshold t=22.5 and compute $H(W|seuil_t = 22.5)$
- ...
- let's now consider the threshold t=25 and compute $H(W|seuil_t = 22.5)$
- ...

Continuous/nominal feature

Gain ratio

- in the case of numerical or continuous features with high arity (*i.e.* many possible values), these ones are mechanically favored by computing the information gain
- it is possible to prevent that side effect by choosing the selector that maximizes the gain ratio (*rapport de gain*) and not only the gain :

$$\text{Gain Ratio(W,A)} = \frac{\text{Gain(W,A)}}{\text{SplitInfo(W,A)}} = \frac{H(W) - H(W|A)}{-\sum_{a \in \mathcal{D}_A} \frac{|S_a|}{|S|} \log_2 \frac{|S_a|}{|S|}}$$

  - Gain(W,A) =mutual information
  - $|S_a|$ is the subset of examples having the attribute $A = a$.
  - SplitInfo(W,A) = $H(A)$ is also called *intrinsic information*.
  - Discourages the selection of attributes with many uniformly distributed values. Attributes with higher intrinsic information are less useful.

Multiple continuous features

Manage jointly these features

- if all the features are elements of $\mathbb{R}^d$, we've seen that each test is a comparison to a threshold
- thus, we obtain separative surfaces composed of hyperplanes orthogonal to the axes
- instead, one can compare a combination of the features to a threshold (*oblique trees*)
- unfortunately, this way cannot be used in general : the search space (separative surface) is infinite (each has $d+1$ value)
- several heuristics have been proposed (*e.g.* OC1)

Outline

Pruning and number of nodes

Basics

- the tree building algorithm ends by building $T_{\max}$
- this is almost learning-by-heart, thus it may overfit the training set and under-estimate the error rate (0 in many case)
- the number of nodes is a simple criterion to evaluate the complexity of a tree
- remember Occam's razor, we want a better compromise generalization/complexity
- thus, we search for an optimal value k_0 of the number of nodes

Simple pruning technique

A first (but bad) solution : pre-pruning

- stop split a node when its learning examples are not *completely* homogeneous, but *sufficiently* homogeneous
- once the best feature has been selected, one may check that the criterion (*e.g.* entropy) is below a certain threshold
- according to the criterion used, several heuristics were proposed to fix this threshold
  - its value can change according to the node considered, or to the a priori probability of the class, or to an empirical estimate of how hard splitting the class is

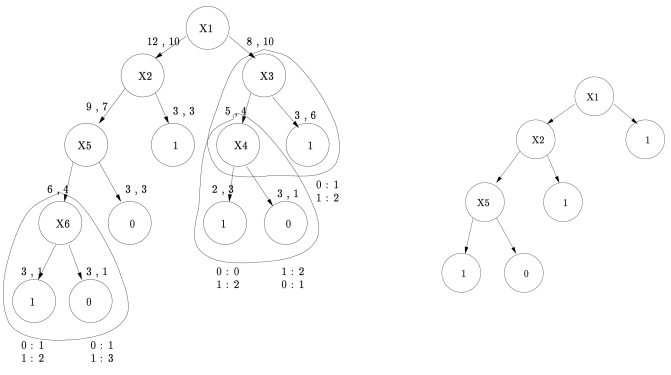
Pruning with a validation set

Post-pruning : rinciple

- best technique in theory and practice
- $T_{\max}$ is built completely
- then, a series of simpler trees are built by recursively grouping some nodes
- last, the best tree is chosen with a validation set

Pruning with a validation set

Example



Pruning with a validation set

In practice

- split the learning set of examples into two parts : one is used to build T_{max} (the real training set), pruned trees are then built, and the other part (validation set) is used to select the best pruned tree
- the problem is just to replace some nodes of T_{max} by leaves
- the majority class (among the examples) is attached to this leaf f newly created this way

Pruning with a validation set

Generating pruned trees

- the optimal algorithm would be to compute the error rate on the validation set on all the possible trees that can be generated from T_{max}
- but their number increases fastly when T_{max} is big (*i.e.* many nodes)

In practice

- we use a sub-optimal solution, only some pruned trees are generated
- build a sequence of trees by successive pruning $S = \{T_{max}, T_1, ..., T_k, ...T_m\}$, were T_m contains only one leaf containing the m learning points
- to go from T_k to T_{k+1}, one has to transform one or more nodes into one or more leaves in T_k

Pruning with a validation set

Cost Complexity Pruning criterion

- choose the node of T_k that minimizes the following criterion :

$$\varpi(T_k, d) = \frac{MC(d, T_k) - MCT(d, T_k)}{n(T_k).(nt(d, T_k) - 1)}$$

  - $MC(d, T_k)$ is the number of learning examples misclassified by node d of T_k supposing it is transformed into a leaf
  - $MCT(d, T_k)$ is the number of learning examples misclassified by the leaves of T_k beneath node d
  - $n(T_k)$ is the total number of leaves of T_k
  - $nt(d, T_k)$ is the number of leaves of T_k beneath node d

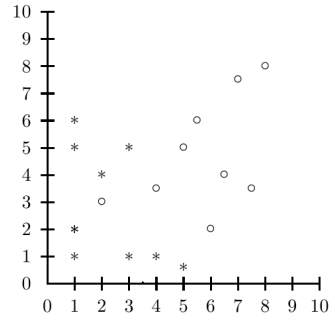
Pruning with a validation set

Pruning criterion

- this criterion allows to prune one or more nodes of T_k s.t. T_{k+1} has a better balance between size and error rate
- finally, in the series $\{T_{max}, T_1, ..., T_k, ...T_m\}$ that we built, we look for the tree T_{k0} with the minimal error rate on the validation set
- this tree is the final result

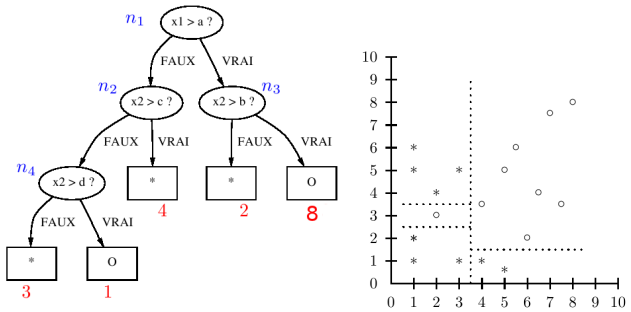
Pruning Example

Training set (18 examples, 2 classes, 2 numeric features)



Pruning Example

Decision tree T_{max} with the previous examples



Pruning Example

Computing the pruning criterion

- let's note n_1 the root node of T_{max}, n_2 and n_3 its left and right sons and n_4 its last inner node (left son of n_2)
  - $\varpi(T_{max}, n_1) = \frac{MC(n_1, T_{max}) - MCT(n_1, T_{max})}{n(T_{max}).(nt(n_1, T_{max}) - 1)} = \frac{9-0}{5.(5-1)} = 9/20$
  - $\varpi(T_{max}, n_2) = \frac{1-0}{5.(3-1)} = 1/10$
  - $\varpi(T_{max}, n_3) = \frac{2-0}{5.(2-1)} = 2/5$
  - $\varpi(T_{max}, n_4) = \frac{1-0}{5.(2-1)} = 1/5$
- T_1 is built by pruning node n_2 in T_{max}

Pruning Example

Next step of pruning

- let's now try to prune T_1
  - $\varpi(T_1, n_1) = \frac{9-1}{3.(3-1)} = 4/3$
  - $\varpi(T_1, n_3) = \frac{2-0}{3.(2-1)} = 2/3$
- T_2 is built by pruning node n_3 in T_1
- T_2 thus contains only the root node with one leaf for each class

Pruning Example

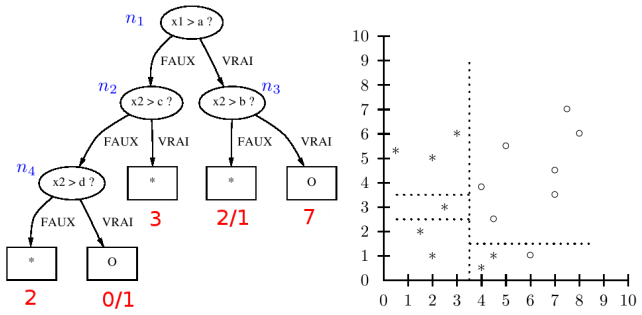
Choosing the best pruned tree

- we test the validation set examples on the trees T_{max}, T_1 and T_2 we just generated
- we also test the tree T_m, composed of one unique leaf with the majority label of the training examples
- we choose the tree with the lowest error rate ; the pruning process is over

Pruning Example

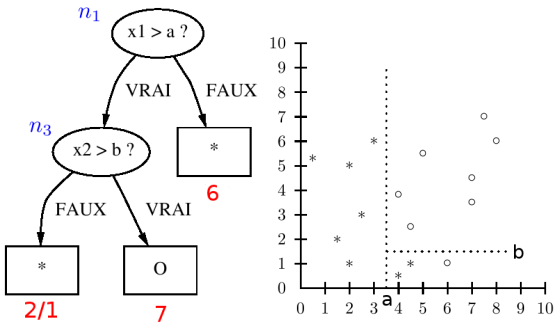
T_{max} results on the validation set

Validation set : 16 examples



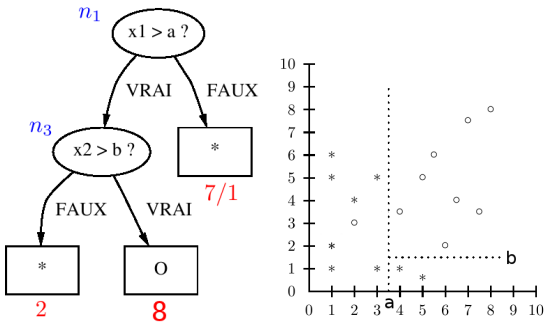
Pruning Example

T_1 results on the validation set



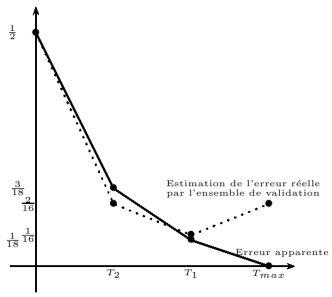
Pruning Example

T_1 results on the training set



Pruning Example

Apparent error and real error



Tree obtained after pruning step : T_1

Pruning Example

Fisher Iris dataset

- 2 numeric attributes : `sepal length` and `sepal width`,
- 3 classes denoted in the following by the numbers $\{1, 2, 3\}$,
- 150 examples :
  - 100 are used for the training set
  - the remaining 50 for the test set

Outline

Case of missing attribute values

Context

- compute the Gain(W,A) for the attribute A at node n
- $< x, \omega(x) >$ is one of the training examples in $\mathcal{S}$ and the value $A(x)$ is unknown
- not handled in ID3, ad-hoc mechanism in C4.5

Case of missing attribute values

Several strategies

- discard examples with missing values $\Rightarrow$ decrease the number of examples
- consider the fact that the value is missing as an information $\Rightarrow$ add a specific new value for this attribute
- assign the value that is the most common among training examples at node n
- assign the value that is the most common among training examples at node n of the same class
- assign a probability to each of the possible values of A, estimated based on the observed frequencies of the various values for A among the examples at node n. Example :
  - Example : boolean attribute A.
  - node n contains 6 examples with $A = 1$ and 4 examples with $A = 0$
  - A fractional 0.6 and a fractional 0.4 of instance x are distributed down the 2 branches $A = 1$ and $A = 0$ and are used for the purpose of computing information gain.

Decision tree advantages

- Easy to interpret the decision rules
- Non parametric so it is easy to incorporate a range of numeric or categorical data layers and there is no need to select unimodal training data
- Classification is fast once rules are developed

Decision tree drawbacks

- Decision trees tend to overfit training data which can give poor results when applied to the full data set
- Not possible to predict beyond the minimum and maximum limits of the variables values in the training data
- Splitting perpendicular to feature space axes is not always efficient

Multi-class evaluation

Confusion matrix

Real \ Predicted	A	B	C	D
A	15	9	1	0
B	7	24	6	2
C	11	4	9	1
D	0	0	3	4

Evaluation measures

- Accuracy $= \frac{15+24+9+4}{25+39+25+7}$
- **Per-class** precision : $P_A = \frac{15}{33}$, $P_B = \frac{24}{37}$,...
- **Per-class** recall : $R_A = \frac{15}{25}$, $R_B = \frac{24}{39}$,...

Multi-class evaluation

Averaging the evaluation measures

3 types of averaging :

macro : unweighted mean of the metrics computed for each class.
macro-P$= \frac{1}{4}(P_A + P_B + P_C + P_D)$

weighted : the metrics computed for each class are weighted by support (number of true instances for each class).

$$\begin{aligned}\text{weighted-P} &= \frac{|A|}{N}P_A + \frac{|B|}{N}P_B + \frac{|C|}{N}P_C + \frac{|D|}{N}P_D \\ &= \frac{25}{96}P_A + \frac{39}{96}P_B + \frac{25}{96}P_C + \frac{7}{96}P_D\end{aligned}$$

micro : calculate metrics globally by counting the total true positives, false negatives and false positives.
micro-averaging $\Leftrightarrow$ accuracy, loosing precision/recall difference

Multi-class evaluation

Several equivalences exist between these measures
e.g. accuracy=weighted-R

Balanced dataset

- no differences between macro and weighted averaging
- macro-R=weighted-R=accuracy

Unbalanced dataset

- large classes dominate small classes in micro-averaging (accuracy) $\rightarrow$ micro-averaged results = measure of effectiveness on the large classes