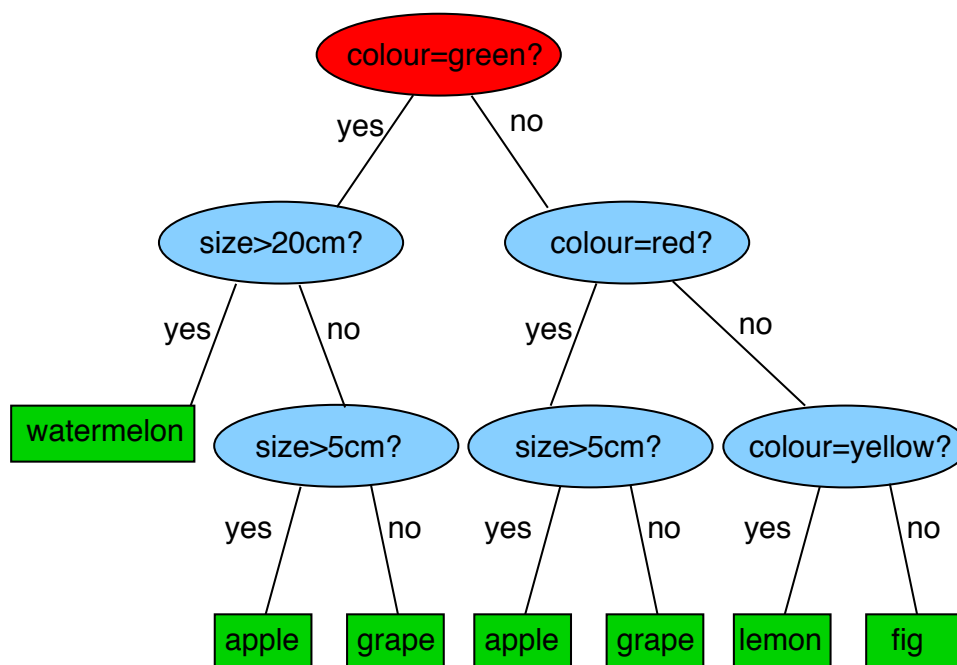


University of Cambridge
Engineering Part IIB

Paper 4F10: Statistical Pattern
Processing

Handout 10: Decision Trees



Mark Gales
mjfg@eng.cam.ac.uk
Michaelmas 2015

Improved Detection of Prostate Cancer

The aim is to build a classifier for patients suspected of having prostate cancer. The data collected included:

- serum prostate-specific antigen (PSA) level
- presence of hypoechoic lesion (binary)
- PSA density (PSAD)
- prostate volume
- demographic information (e.g. age)

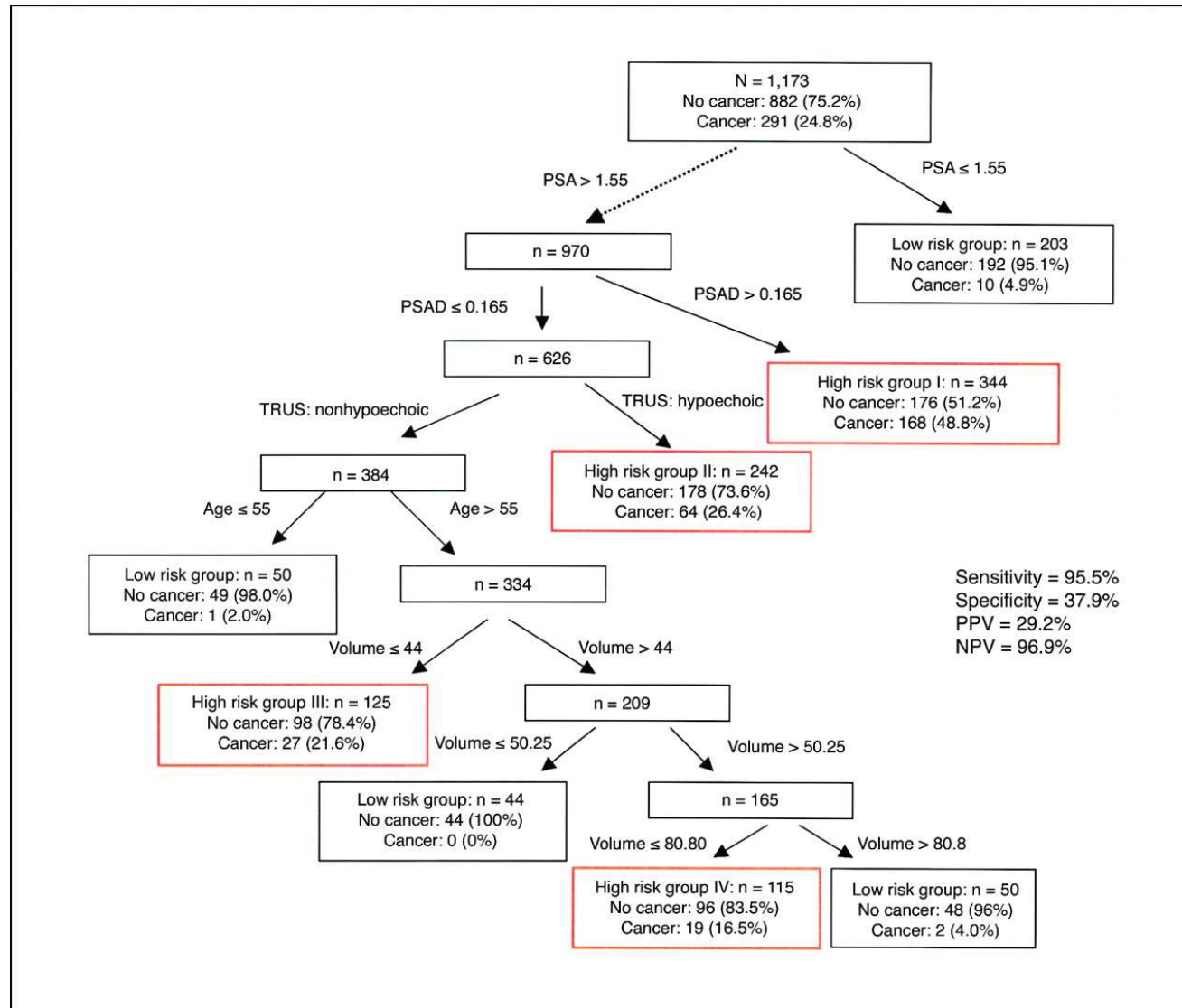
Using this information a CART decision tree was constructed.

$$\text{specificity} = \frac{\# \text{true_negatives}}{\# \text{true_negatives} + \# \text{false_positives}}$$

$$\text{sensitivity} = \frac{\# \text{true_positives}}{\# \text{true_positives} + \# \text{false_negatives}}$$

$$\text{PPV} = \frac{\# \text{true_positives}}{\# \text{true_positives} + \# \text{false_positives}}$$

$$\text{NPV} = \frac{\# \text{true_negatives}}{\# \text{true_negatives} + \# \text{false_negatives}}$$



Introduction

So far we have only considered situations where the feature vectors have been real valued, or discrete valued. In these cases we have been able to find a natural distance measure between the vectors.

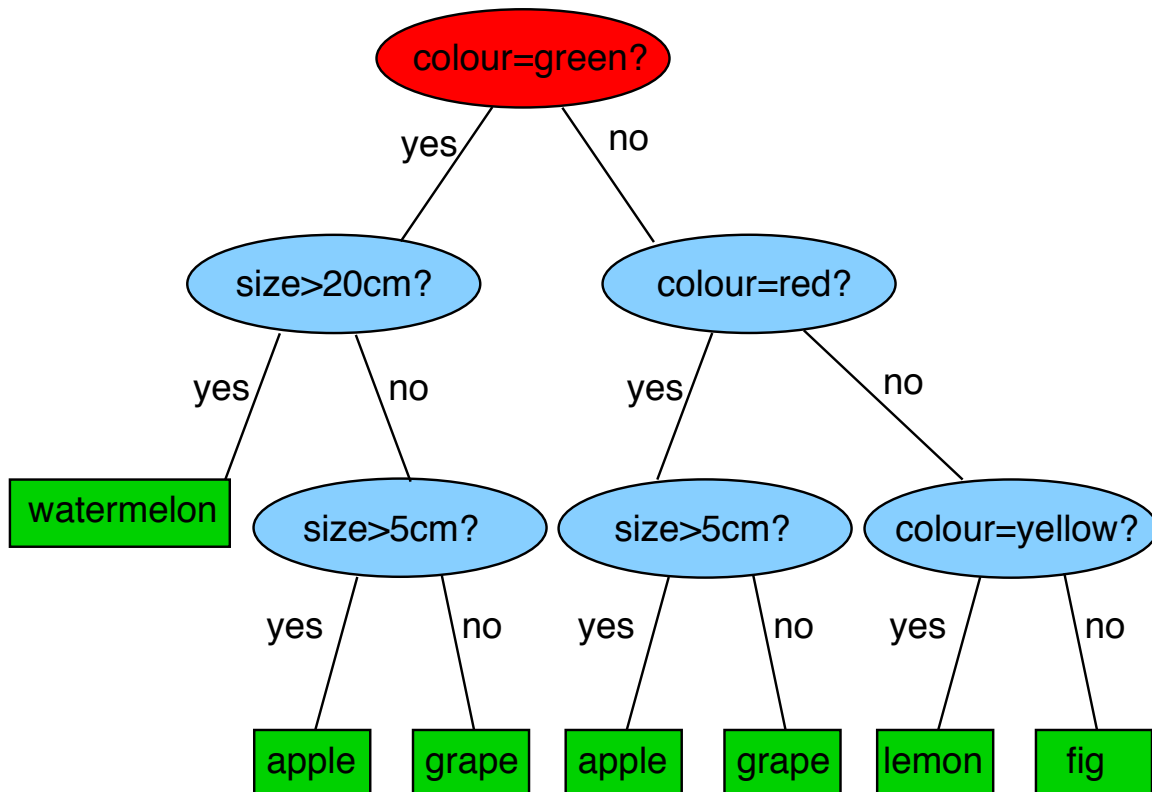
Consider the situation where we have **nominal** data in addition to the numerical data. For instance descriptions that are discrete and there is no natural concept of ordering. The question is how can such data be used in a classification task. Most importantly how can we use such data to train a classifier.

One approach to handling this problem is to use **decision trees**, an example of a **non-metric** classifier - they elegantly handle nominal data. Multiple levels of classifier are used to obtain a final decision. In contrast to the multi-layer perceptrons (another multi-layer classifier) the training of these decision trees is relatively simple and the classification speed is fast.

We will look at the issues behind **training** and **classifying** with decision trees. As with the other classifiers **supervised training data** will be used to train the decision tree.

Example Decision Tree

Consider a simple task to classify fruit according to its colour and size. A reasonable decision tree would be:

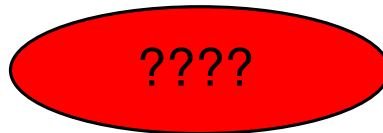


The feature-vector, [red, 3.0cm], is observed, what is the fruit?

1. Does colour=green? **NO**
2. Does colour=red? **YES**
3. Is size > 5cm? **NO**

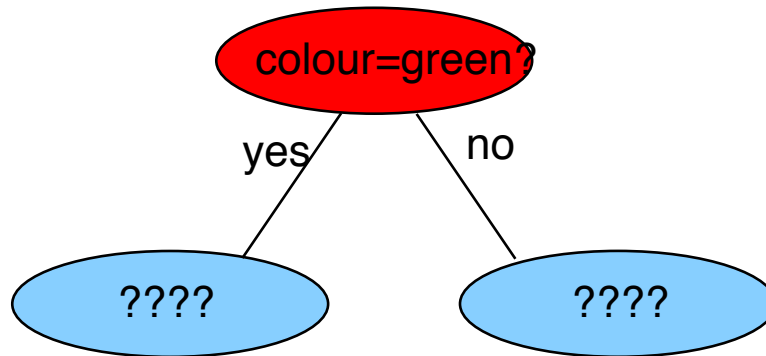
It's a **grape!**

Example Tree Construction (1)



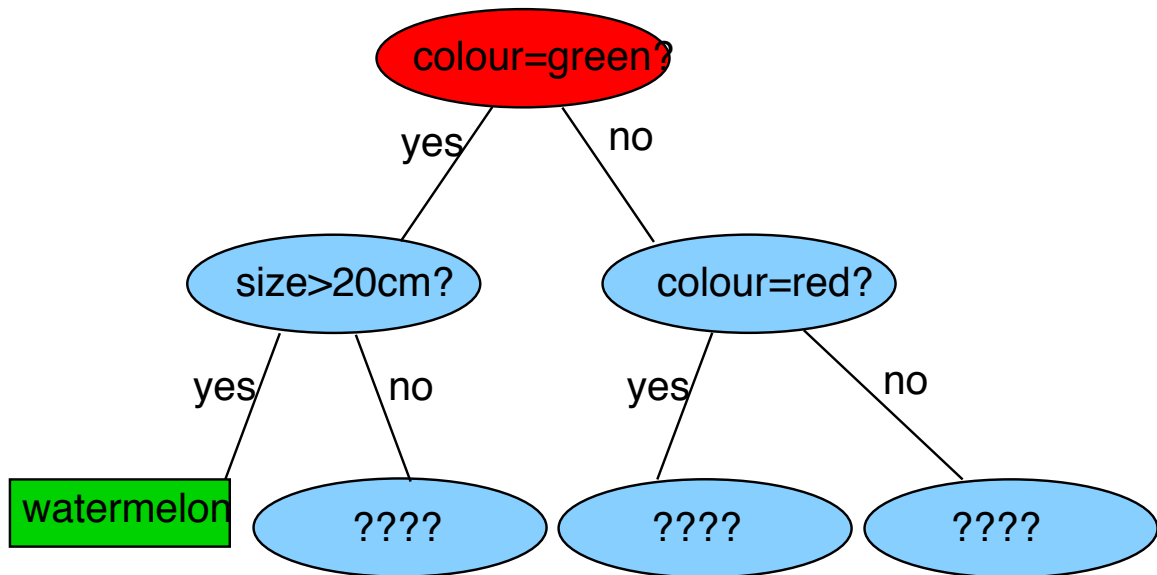
mellon1	green	25.0cm
mellon2	green	21.1cm
grape1	green	2.5cm
grape2	green	4.5cm
grape3	red	2.5cm
lemon1	yellow	8.0cm
fig1	black	4.3cm
apple1	green	7.0cm
apple2	red	8.2cm

Example Tree Construction (2)



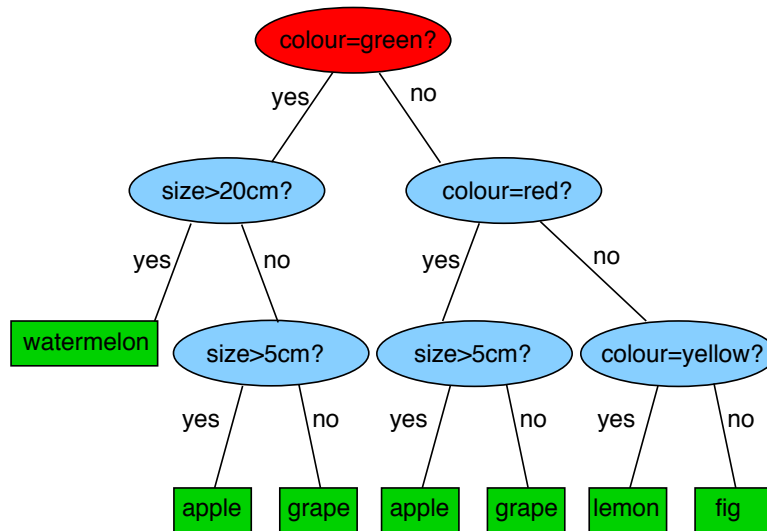
mellon1	green	25.0cm
mellon2	green	21.1cm
grape1	green	2.5cm
grape2	green	4.5cm
apple1	green	7.0cm
grape3	red	2.5cm
lemon1	yellow	8.0cm
fig1	black	4.3cm
apple2	red	8.2cm

Example Tree Construction (3)



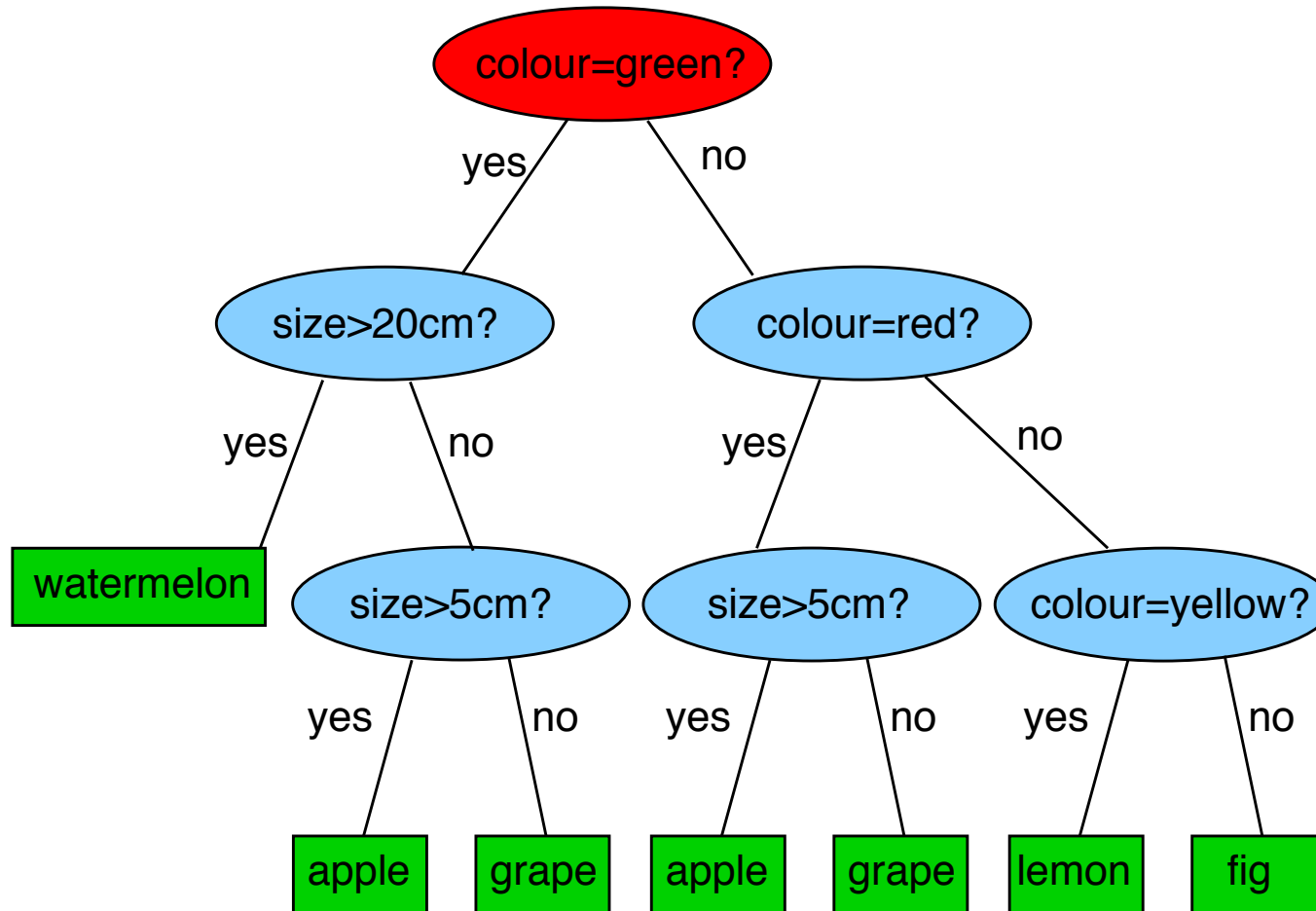
mellon1	green	25.0cm
mellon2	green	21.1cm
grape1	green	2.5cm
grape2	green	4.5cm
apple1	green	7.0cm
grape3	red	2.5cm
apple2	red	8.2cm
lemon1	yellow	8.0cm
fig1	black	4.3cm

Example Tree Construction (4)



mellon1	green	25.0cm
mellon2	green	21.1cm
grape1	green	2.5cm
grape2	green	4.5cm
apple1	green	7.0cm
grape3	red	2.5cm
apple2	red	8.2cm
lemon1	yellow	8.0cm
fig1	black	4.3cm

Example Decision Tree



Decision Tree Construction Issues

Decision tree comprises:

- **root node**: the top node/question in the tree (red).
- **internal nodes**: consists of questions (blue)
- **terminal nodes** (or leaves): consists of a class label (green)

When constructing decision trees a number of issues must be addressed:

1. how many **splits** should there be at each node (**binary?**)?
2. what question for root/internal nodes?
 - what order should the questions be asked in?
3. when should a node be declared a **leaf**?
4. how should a leaf be labelled?

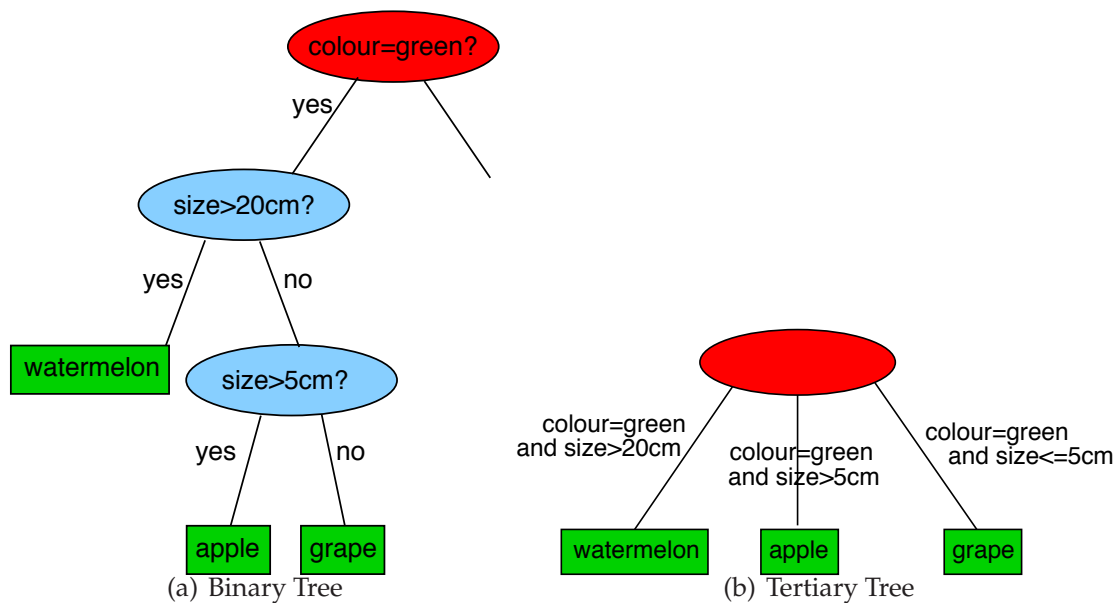
For more general decision trees there are other issues that may be addressed for example handling missing data and pruning trees. These are not considered in this lecture.

Number of Splits

When the question associated with a node is applied the outcome is called a **split** - it splits the data into subsets. The root node splits the complete training set, subsequent nodes split subsets of the data.

How many splits to have at each node?

Normally the number is determined by an **expert**. Note **every** decision and hence **every** decision tree can be generated just using binary decisions.

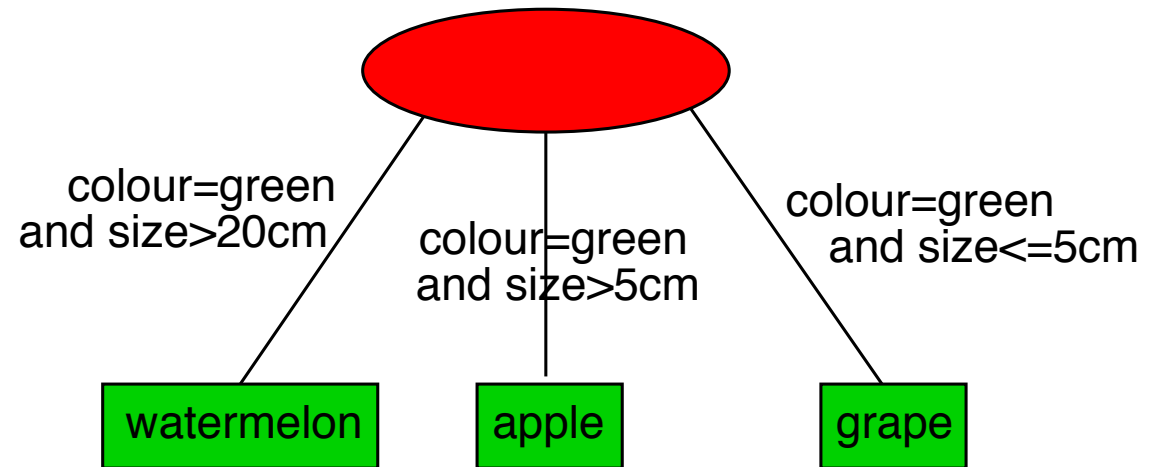
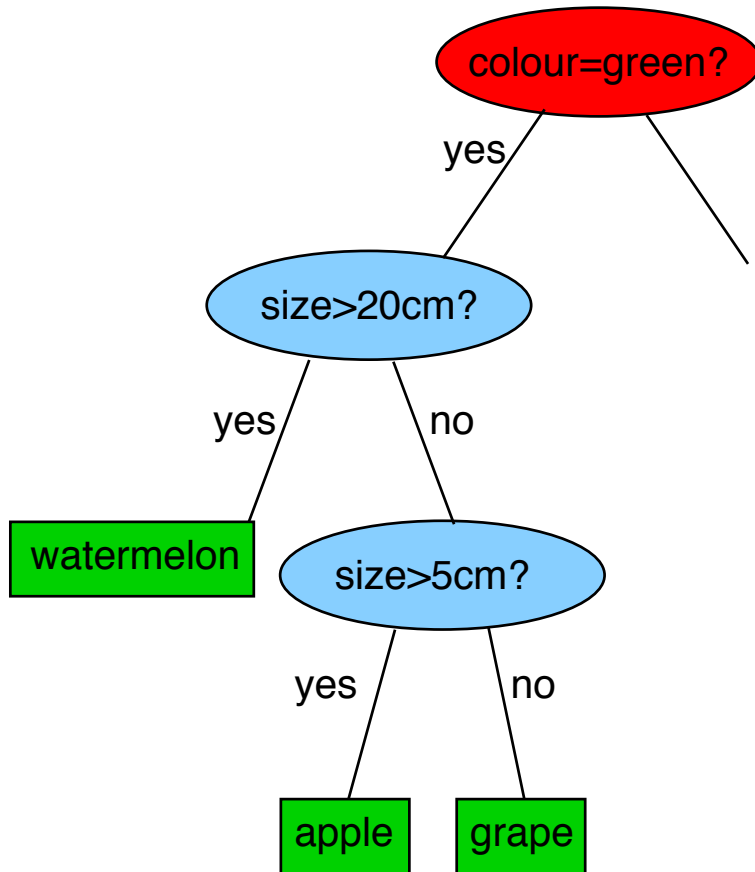


Two forms of query exist

- **monothetic**: (colour=green?)
- **polythetic**: (size>20cm?) AND (colour=green?)

Polythetic queries may be used by can result in a vast number of possible questions.

Number of Splits



Question Selection

The basic training paradigm for designing a **binary** decision tree automatically from training data is

1. generate a **list of questions**
2. **initialise a root node** containing all training data
3. **sequence through all questions** and for each:
 - (a) apply the question to the node training data
 - (b) partition the data and compute the **purity**
 - (c) record the question with the highest purity
4. **attach the best question** to the current node and create two offspring nodes
5. **split** the data assigning as follows:
 - **yes** data in the left node
 - **no** data in the right node
6. **recursively apply** the above algorithm to each new node

Note that this is a so-called **greedy** algorithm because it only computes a locally optimal partitioning of the data. The complete tree is not guaranteed to be an optimal classifier.

Purity Measures

The aim of a split is to yield a **pure** node (nodes only containing one class), or get as close as possible.

To assess the purity of the nodes a **node impurity** function at a particular node N , $\mathcal{I}(N)$ is defined

$$\mathcal{I}(N) = \phi(P(\omega_1|N), \dots, P(\omega_K|N))$$

where there are K classification classes.

The function $\phi()$ should have the following properties

- $\phi()$ is a maximum when $P(\omega_i) = 1/K$ for all i
- $\phi()$ is at a minimum when $P(\omega_i) = 1$ and $P(\omega_j) = 0, j \neq i$.
- It is symmetric function (i.e. the order of the class probabilities doesn't matter).

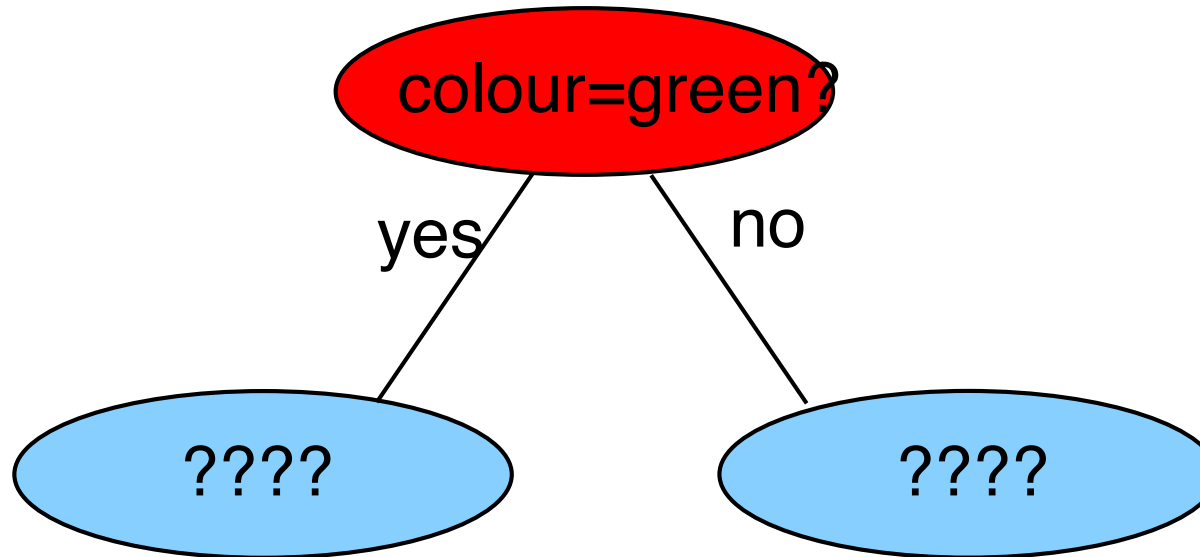
The obvious way to use the purity when constructing the tree is to split the node according to the **question** that maximises the increase in purity (or minimises the impurity). The drop in impurity for a binary split is

$$\Delta\mathcal{I}(N) = \mathcal{I}(N) - \frac{n_L}{n_L + n_R}\mathcal{I}(N_L) - \frac{n_R}{n_L + n_R}\mathcal{I}(N_R)$$

where:

- N_L and N_R are the left and right **descendants**;
- n_L is the **number** of the samples in the left descendant;
- n_R is the **number** of the samples in the right descendant;

Purity Measures



$$\Delta \mathcal{I}(N) = \mathcal{I}(N) - \frac{n_L}{n_L + n_R} \mathcal{I}(N_L) - \frac{n_R}{n_L + n_R} \mathcal{I}(N_R)$$

Purity Measures

There are a number of commonly used measures:

- **Entropy** measure:

$$\mathcal{I}(N) = - \sum_{i=1}^K P(\omega_i|N) \log(P(\omega_i|N))$$

- **Variance** measure: for a two class problem a simple measure is

$$\mathcal{I}(N) = P(\omega_1|N)P(\omega_2|N)$$

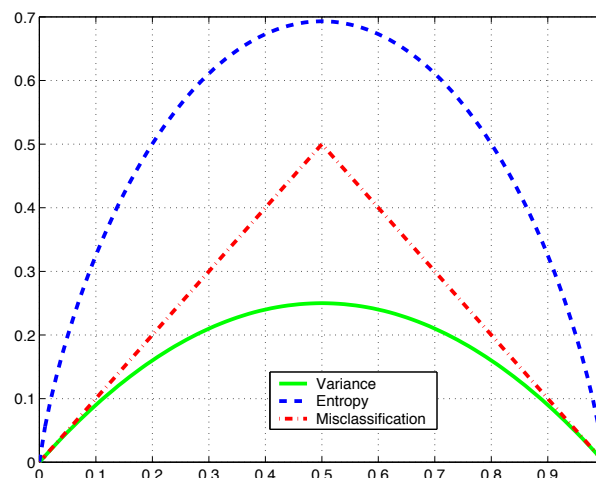
- **Gini** measure: this is a generalisation of the variance measure to K classes

$$\mathcal{I}(N) = \sum_{i \neq j} P(\omega_i|N)P(\omega_j|N) = 1 - \sum_{i=1}^K (P(\omega_i|N))^2$$

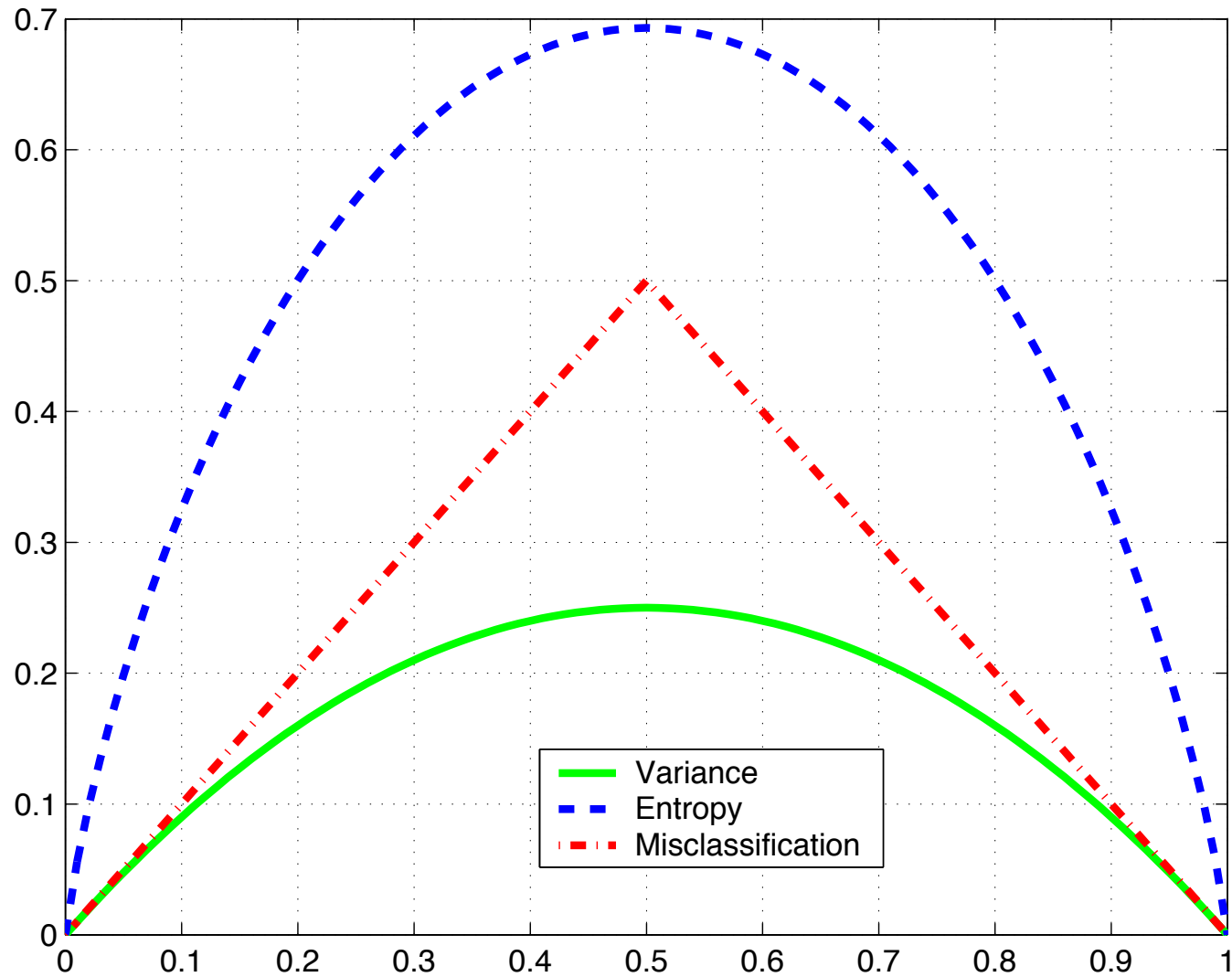
- **Misclassification** measure:

$$\mathcal{I}(N) = 1 - \max_k (P(\omega_k|N))$$

The purity measures for a two class problem are shown below.



Purity Measures



Non-Binary Purity Increase

We may also want to extend the binary split into a multi-way split. Here we can use the natural extension

$$\Delta\mathcal{I}(N) = \mathcal{I}(N) - \sum_{k=1}^B f_k \mathcal{I}(N_k)$$

where f_k is the fraction of the samples in descendant k

$$f_i = \frac{n_i}{\sum_{k=1}^B n_k}$$

The direct use of this expression to determine the number of splits B has a drawback:

- large B are inherently favoured over small B .

To avoid this problem the split is commonly selected by

$$\Delta_B \mathcal{I}(N) = \frac{\Delta\mathcal{I}(N)}{-\sum_{k=1}^B f_k \log(f_k)}$$

This is the **gain ratio impurity**

Using the splitting rules described before results in a **greedy** algorithm. This means that any split is only **locally** good. There is no guarantee that a global optimal split will be found, nor that the smallest tree will be obtained.

Gain Ratio Impurity

$$\Delta\mathcal{I}(N) = \mathcal{I}(N) - \sum_{k=1}^B f_k \mathcal{I}(N_k)$$

$$f_i = \frac{n_i}{\sum_{k=1}^B n_k}$$

$$\Delta_B \mathcal{I}(N) = \frac{\Delta\mathcal{I}(N)}{-\sum_{k=1}^B f_k \log(f_k)}$$

Question Set Generation

Only consider binary questions with n training samples. The question set generation can be split into two parts.

- **Nominal data:** a set of questions associated with specific nominal attribute needs to be specified, e.g.
 - fruit colour: (colour=green?), (colour=red?)

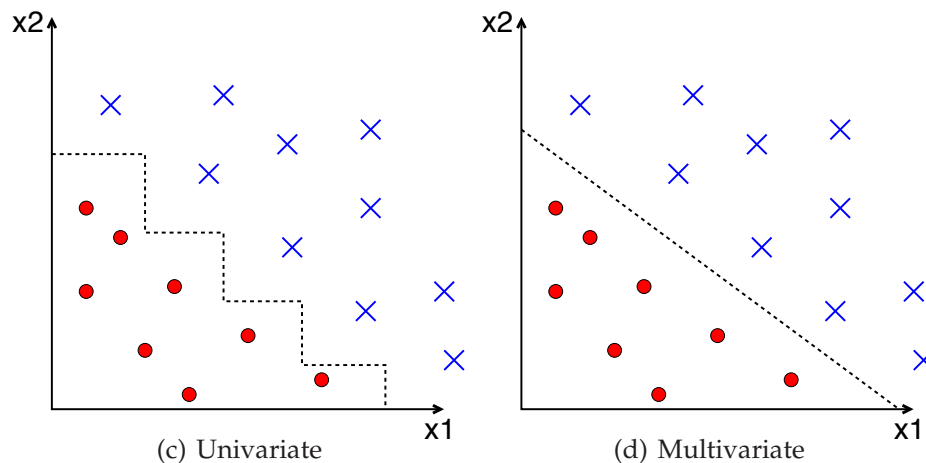
For many tasks, such as clustering the phonetic context in speech recognition, the exact set of questions is not found to be important, provided a **sensible** set of questions is used.

- **Naturally ordered data:** Only considering splits parallel to the axis (combinations are possible) The choice of where to split for a particular dimension is determined by:
 1. sorting the n -samples according to the particular feature selected ;
 2. examining the impurity measure at each of the possible $n - 1$ splits;
 3. selecting the split with the lowest impurity;

Each of the possible d -dimensions should be examined.

Multivariate Numerical Questions

In some situations the proper feature to use to form the decision boundary does not line up with the axis.



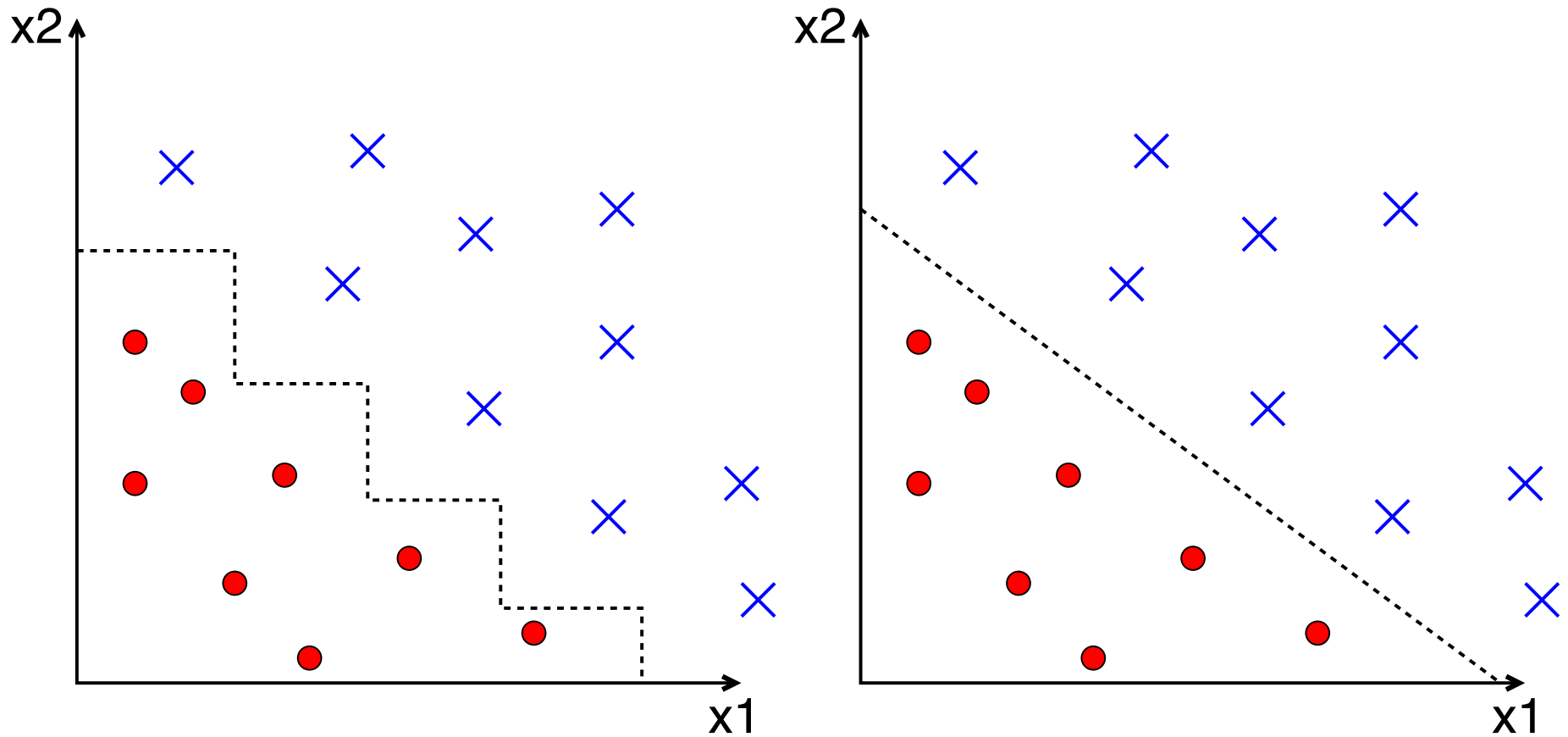
This has led to the use of **multivariate** decision trees. Here decision do not have to be parallel to the axis. The question is how to select the direction of the new split.

This may be viewed as producing a **linear hyperplane** that correctly partitions the data. This is the same problem as generating a linear decision boundary. We can therefore use for example:

- perceptron criterion;
- logistic regression;
- support vector machines.

There is a problem with this form of rule as the training time may be slow, particularly near the root node.

Multivariate Numerical Questions



Declaring a Leaf Node

If the splitting is allowed to carry on we will eventually reach a stage where there is only one sample in each node.

Pure nodes can always be obtained!

However these pure nodes are not useful. It is unlikely that the tree built will generalise to unseen data. The tree has effectively become a lookup-table. There are a number of techniques that may be used to stop the tree growth at an appropriate stage.

- **Threshold**: the tree is grown until the reduction in the impurity measure fails to exceed some specified minimum. We stop splitting a node N when

$$\Delta\mathcal{I}(N) \leq \beta$$

where β is the minimum threshold. This is a very simple scheme to operate. It allows the tree to stop splitting at different levels.

- **MDL**: a criterion related to minimum description length (MDL) may also be used. Here we look for a global minimum of

$$\alpha \text{size} + \sum_{\text{leaf nodes}} \mathcal{I}(N)$$

size represents the number of nodes and α is a positive constant. The problem is that a tunable parameter α has been added.

Other generic techniques such as cross-validation and held-out data may also be used.

Labelling Impure Nodes

If the generation of the tree has been very successful then the nodes of the tree will be pure. By that it means that all samples at a terminal node have the same class label. In this case assignment of the terminal nodes to classes is trivial.

However extremely pure trees are not necessarily good. It often indicates that the tree is over tuned to the training data and will not **generalise**.

For the case of impure nodes they are assigned to the class with the greatest number of samples at that node. This is simply **Bayes' decision rule** (pick the class with the greatest posterior).

$$\hat{\omega} = \arg \max_i \{P(\omega_i|N)\}$$

where

$$P(\omega_i|N) \approx \frac{n_i}{\sum_{k=1}^K n_k}$$

and n_i is the number of the samples assigned to node N that belong to class ω_i .

Classification Trees?

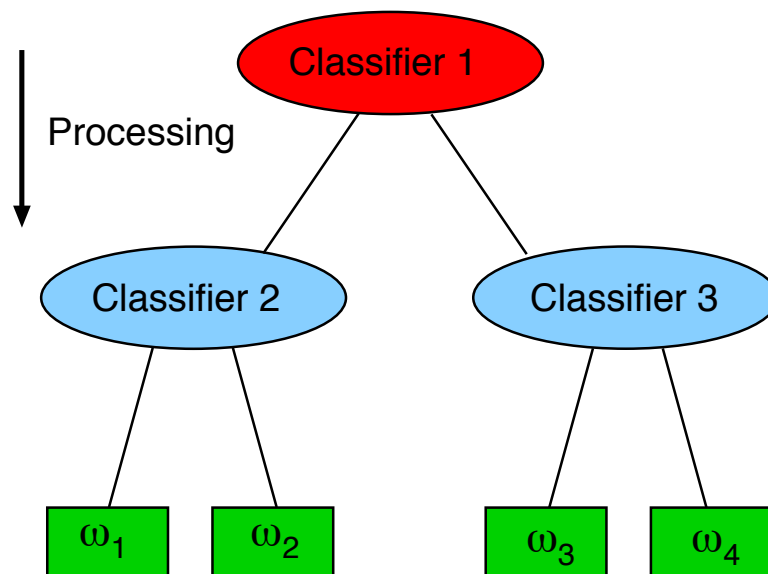
The classification tree approach is recommended when:

1. Complex datasets believed to have non-linear decision boundaries, which may be approximated by a sum of linear boundaries.
2. Data sets with nominal data.
3. There is a need to gain insight into the nature of the data structure. In multi-layer perceptrons it is hard to interpret the meaning of a particular node or layer, due to the high connectivity.
4. Need for ease of implementation. The generation of decision is relatively simple compared to, for example, multi-layer perceptrons.
5. Speed of classification. Classification can be achieved in cost $\mathcal{O}(L)$ where L is the number of levels in the tree. The questions at each level are relatively simple.

Standard forms of classification tree implementations are: [CART](#), [ID3](#), [C4.5](#).

Tree-Based Reduction

Given a tree-structure, it is possible to use this tree-structure as an approach to combine multiple **binary** classifiers to form a **multi-class** classifier without the “no-decision” regions of simple voting.



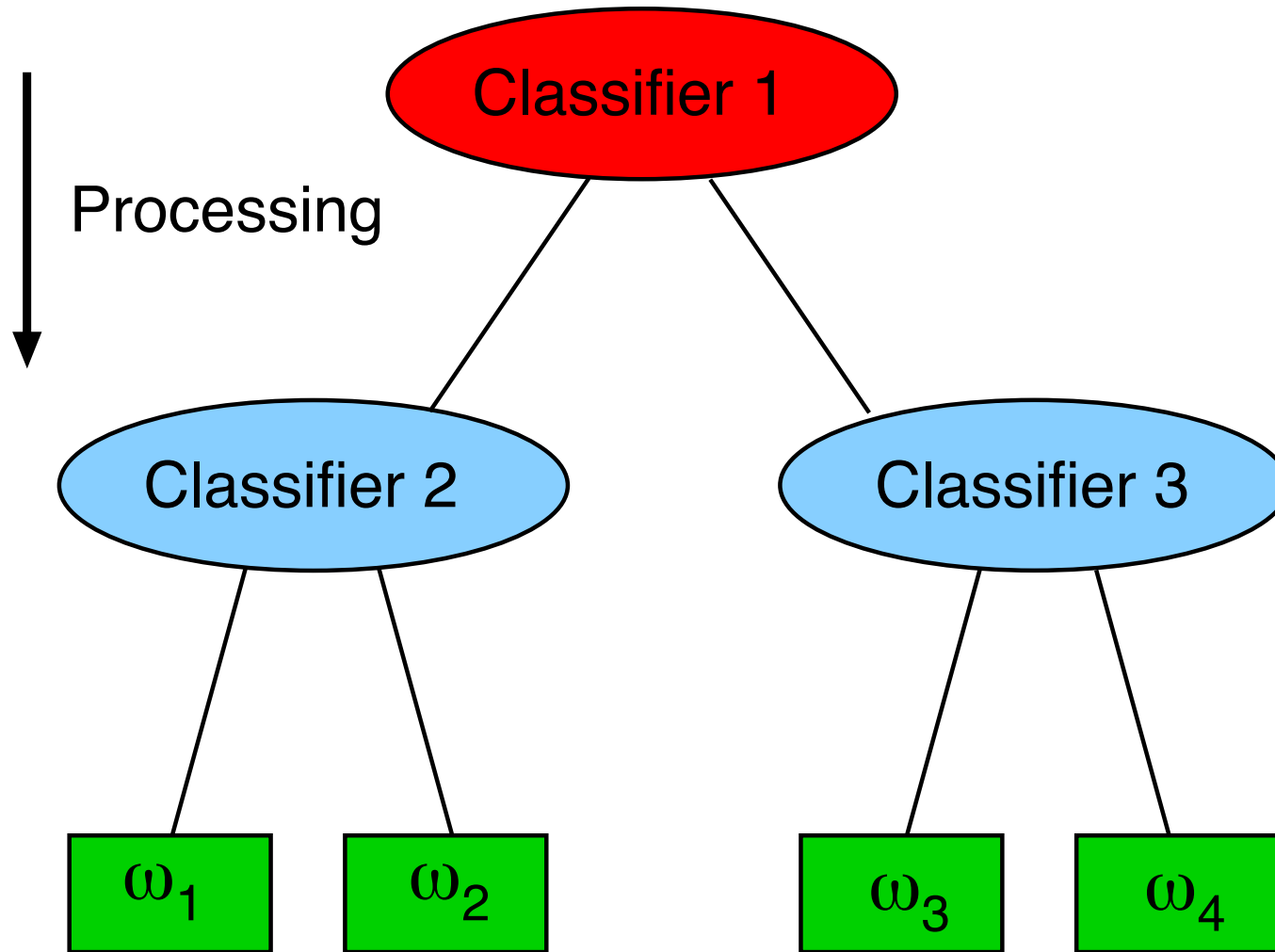
The above **Directed Acyclic Graph** (DAG) is one approach:

- classification/training works from root node **down**
- top classifier classifies $\{\omega_1, \omega_2\}$ vs $\{\omega_3, \omega_4\}$
- performance will depend on ordering of classes
- possible to use **purity** measures previously described

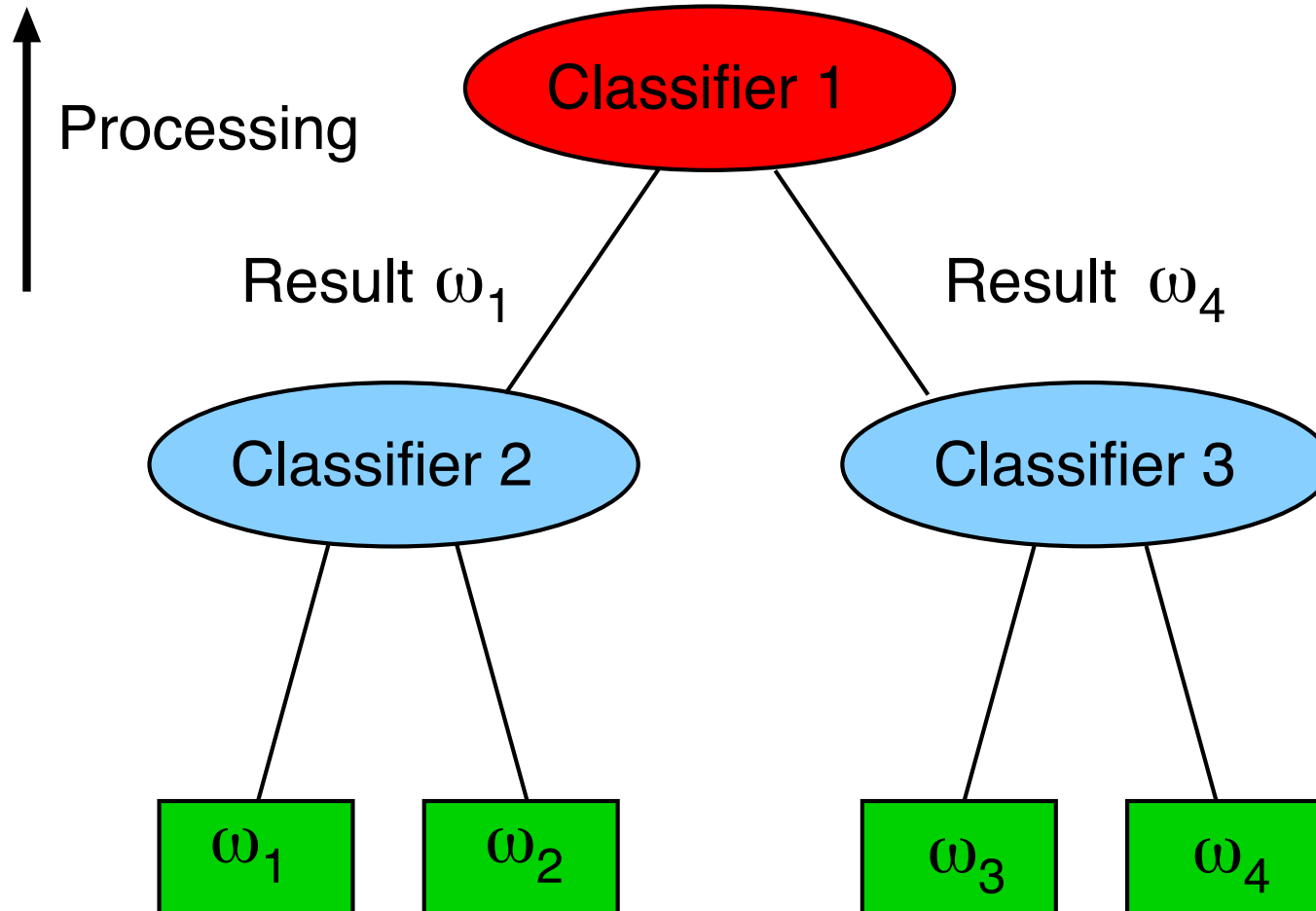
Mistakes at the top-level **cannot** be corrected by lower levels

- top-classifier task highly complicated

Tree-Based Reduction



Adaptive Directed Acyclic Graph

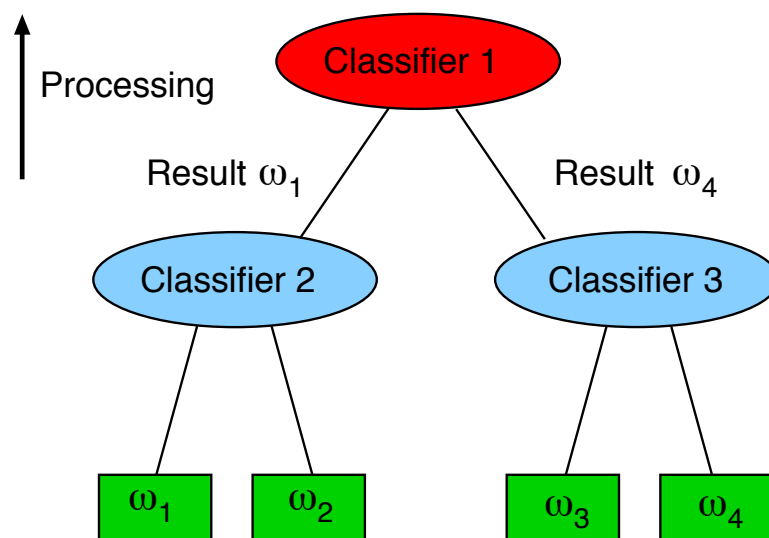


Adaptive Directed Acyclic Graph

Adaptive DAG classifiers are an alternative process:

- classification works from leaf nodes **up**
- classifiers decide between **results** from **lower** classifiers

Simple “knockout” competition



Above diagram, the actions are for a particular observation:

- **Classifier 2** applied to split ω_1 vs ω_2 - yields ω_1
- **Classifier 3** applied to split ω_3 vs ω_4 - yields ω_4
- **Classifier 1** applied to split ω_1 vs ω_4 to yield final result

Classifier 1 changes depending on the class outputs!

DAG versus ADAG

System	# classifiers	# operations
DAG	$K - 1$	$\log_2(K)$
ADAG	$K(K - 1)/2$	$K - 1$