Intelligent Systems

Escuela Técnica Superior de Informática

Universitat Politècnica de València

Block 2 Chapter 3
Classification Trees.

Index

Decision and Classification Trees (DCTs)

Classification trees (also called **Decision trees** or Identification trees) is a **non-parametric approach** to Pattern Recognition; that is, the probability density of the features are not known a priori so we cannot make assumptions regarding the distribution of the features.

A classification tree is a structure that results from a recursive splitting of the representation space from a pattern or learning sample. Classification trees are a simple and effective knowledge representation method.

Method for classifying a pattern through a decision tree: ask a sequence of questions (queries), in which the next question depends on the answer to the current question. Questions are about the value of a feature (attribute or property) of the sample.

The sequence of questions is displayed in a directed *decision tree*, where the *root node* is displayed at the top, connected by successive (directional) links or branches to other nodes. These are similarly connected until we reach the terminal or *leaf nodes* which have no further links.

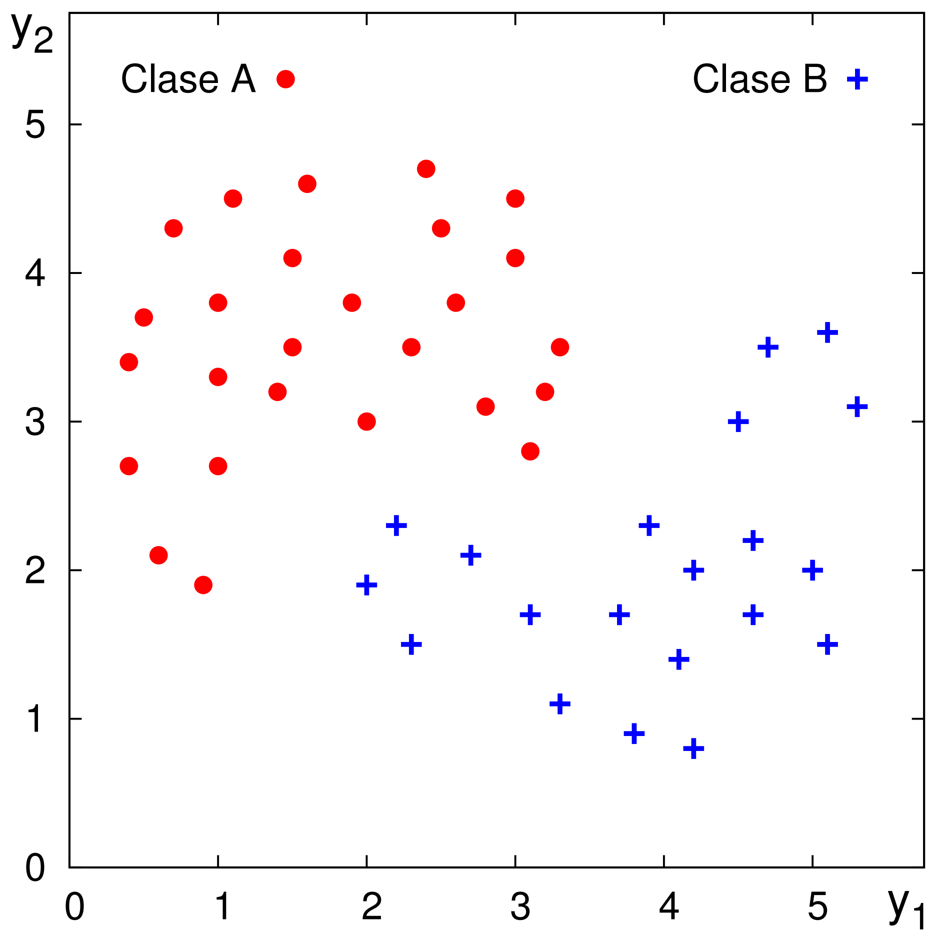
The root node and intermediate nodes contain a question about a particular feature (with a branch for each possible answer). Leaf nodes are labeled with a class. The reached leaf node will determine the class of the sample, i.e., the final classification decision.

Continuation ...

Classifiers based on decision trees: ID3, C4, C4.5, bayesian trees, **CART (Classification and Regression Trees)**[3]

CART: partitions or splits are exclusively based on binary-valued properties, whose values are supported by solid statistical information

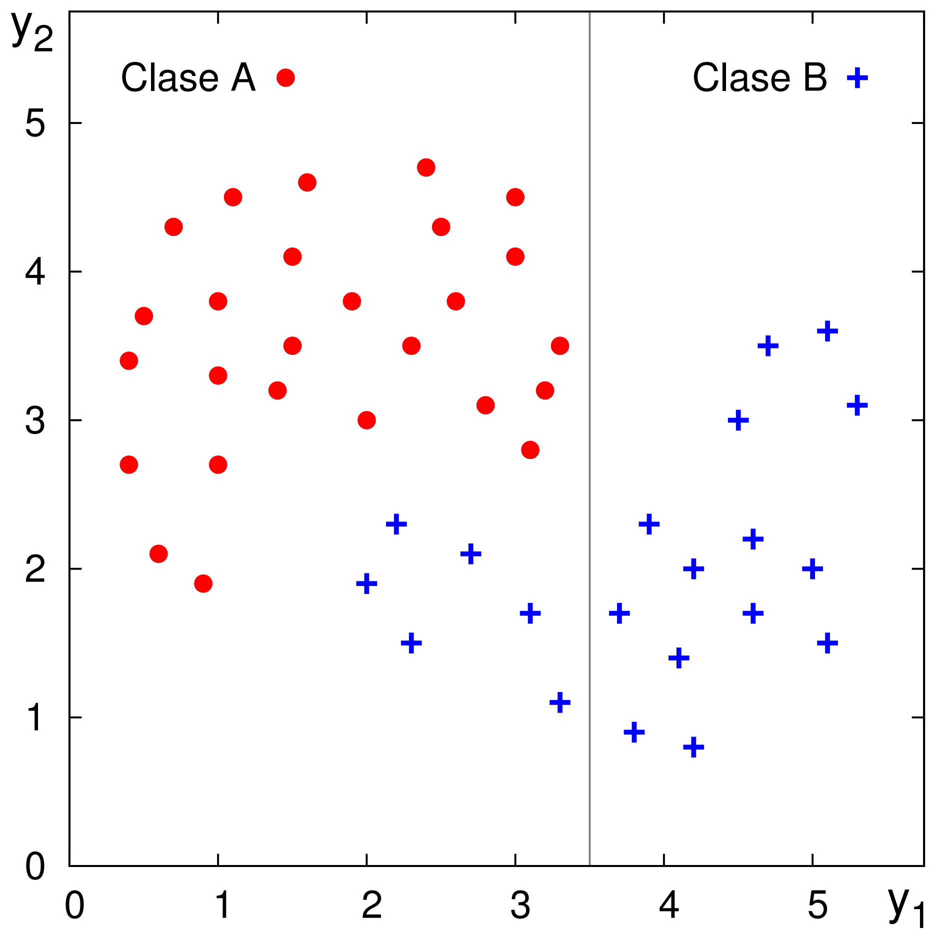
Example

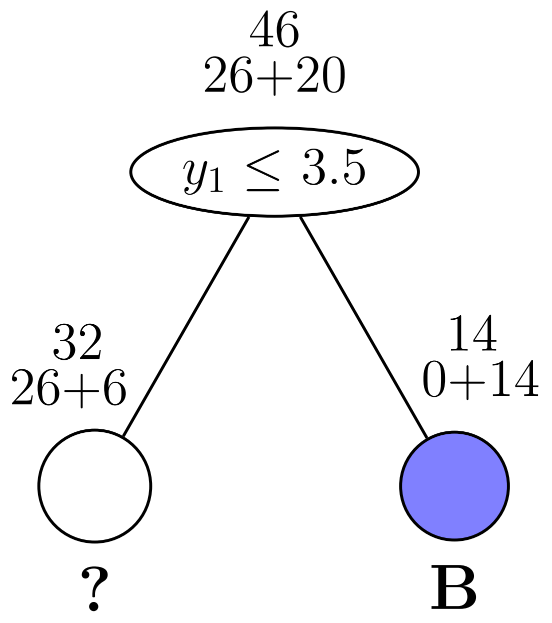


Simple task:

- Two-dimension representation ($E = \mathbb{R}^2$)
- Classification in two *non-linearly* separable classes
- 46 feature vectors (data)
  - 26 belong to class A
  - 20 belong to class B

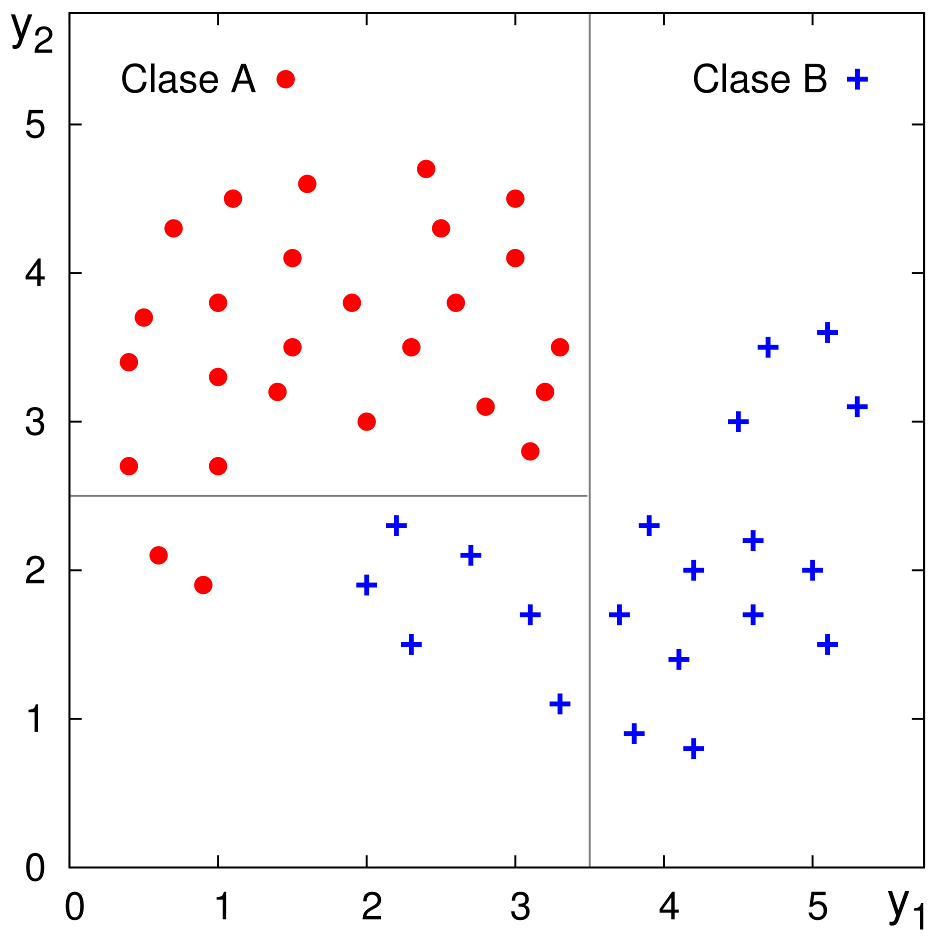
Example: First split (partition)

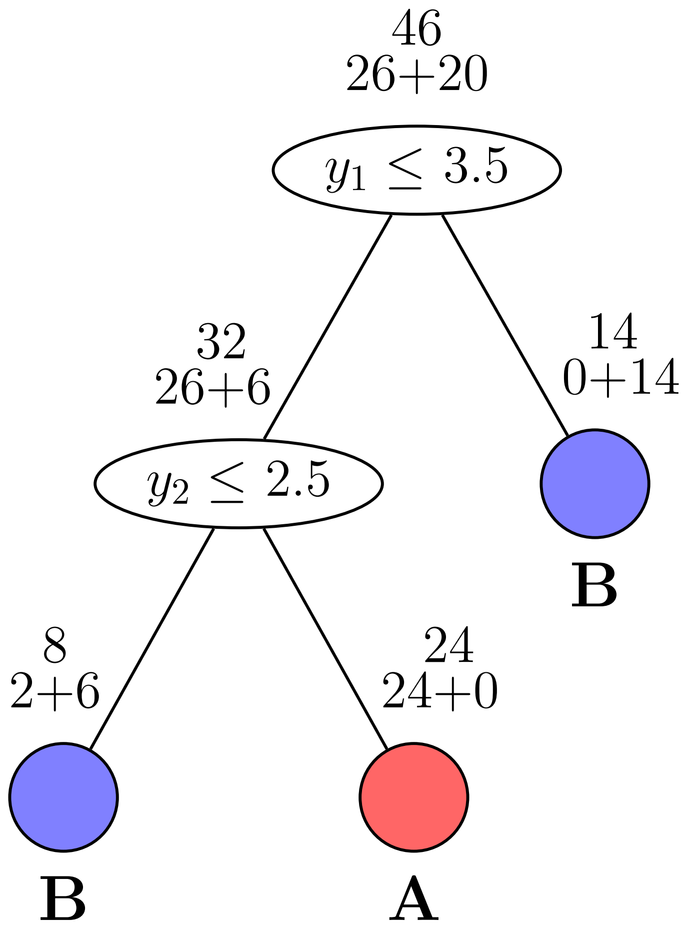




The root node splits the full training set; each successive decision outcome splits a proper subset of data. The first split is made on the basis of the query: $y_1 \leq 3.5$?. The right node contains 14 training samples that are all of class B. We say the node is 'pure' and it is labeled as a *terminal node* of class B.

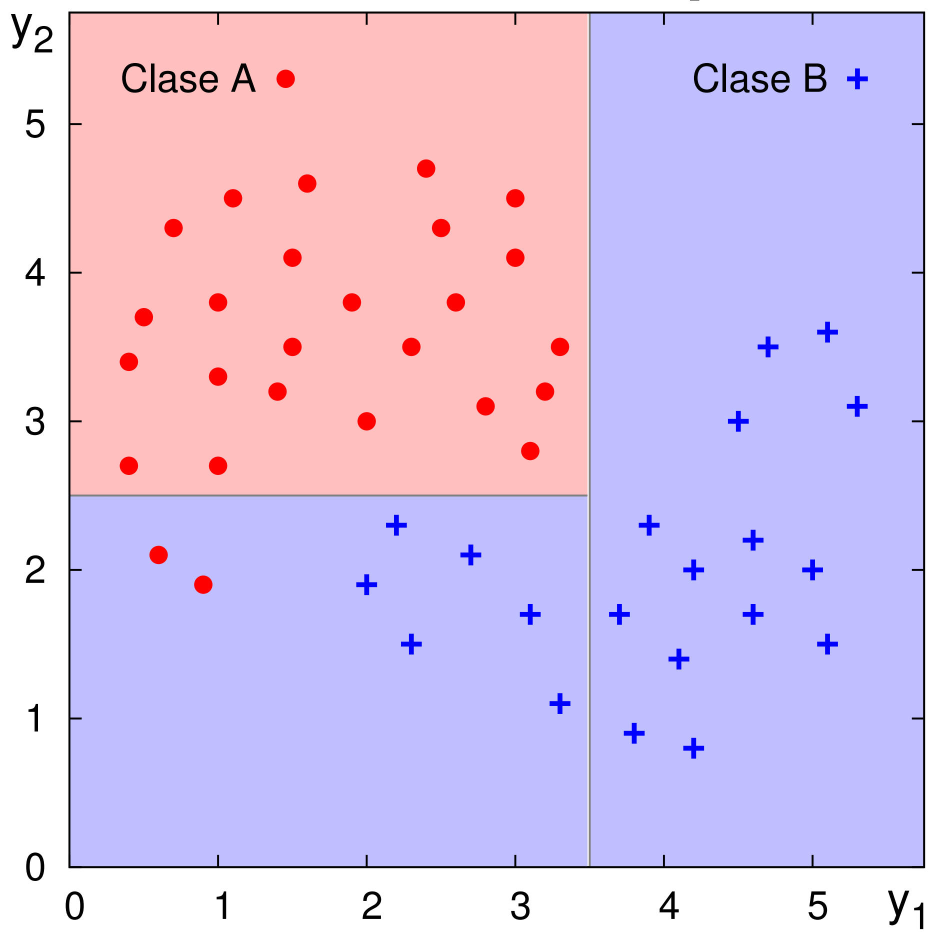
Example: second split and decision boundaries

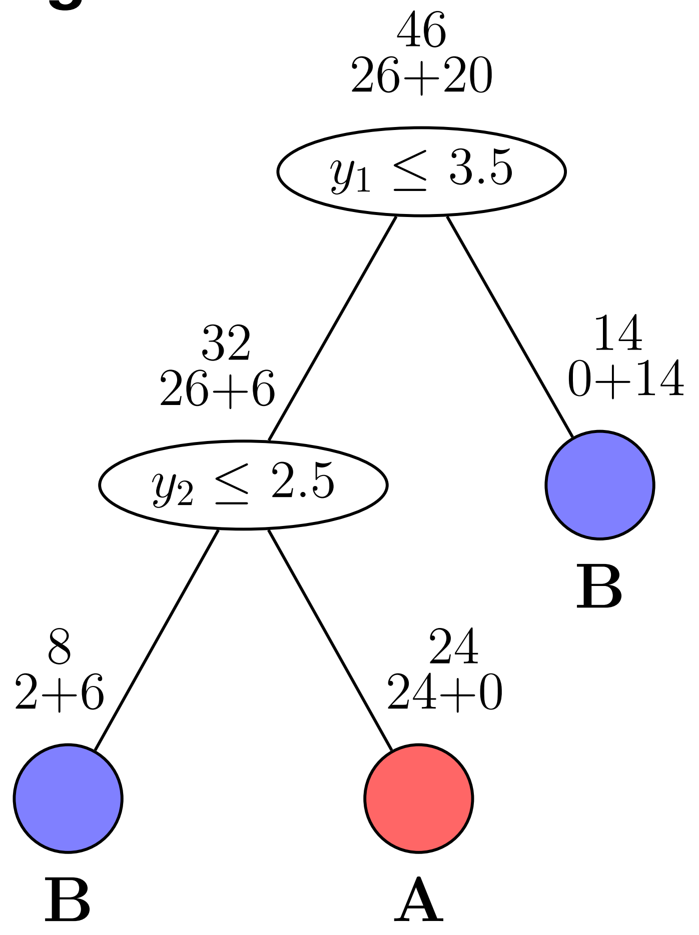




A second split is performed in the left node through the query $y_2 \leq 2.5$?. The right descendent node is pure and it is labelled as a terminal node of class A. We might proceed splitting the left descendent node until we get pure nodes; but we opt for stopping and labelling the node with the mostly represented class: class B.

Example: decision region



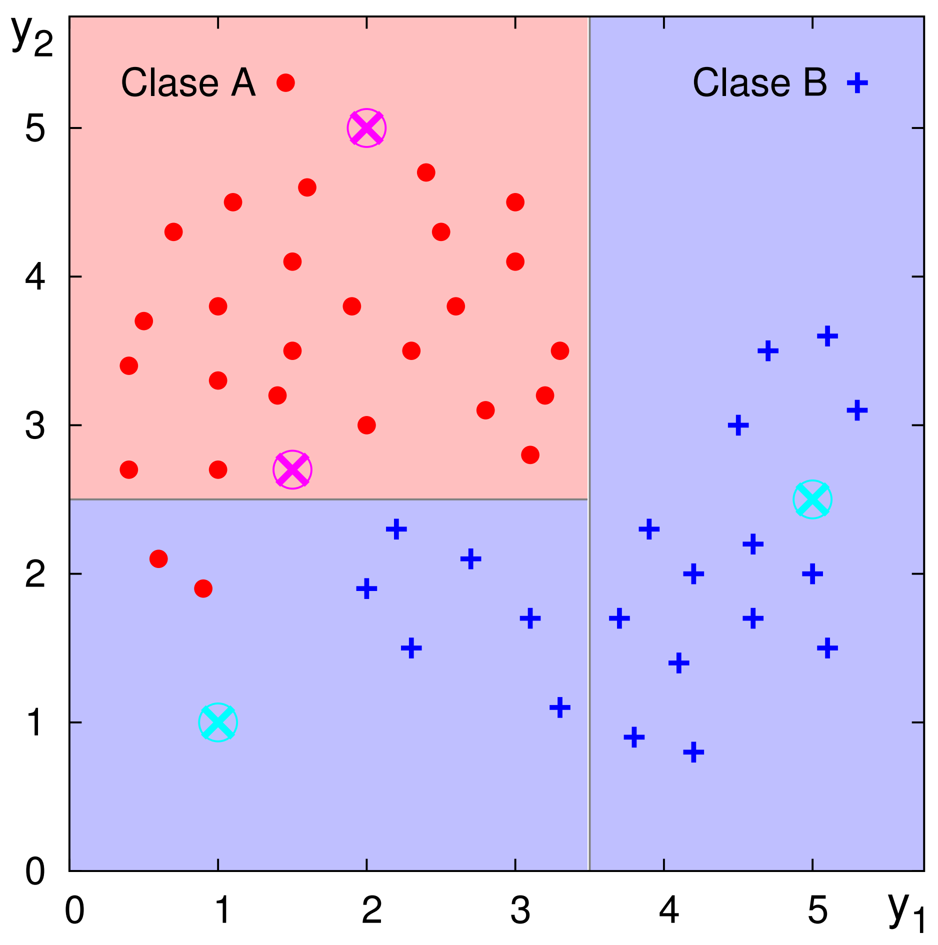


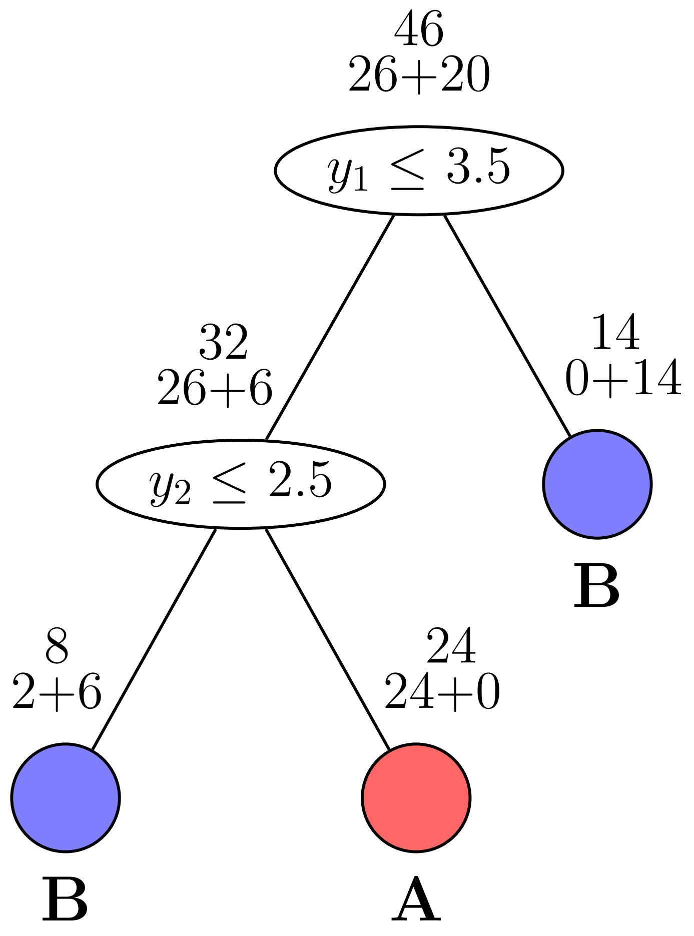
The decision regions are formed by shaped rectangular blocks as the decision boundaries are always parallel to the axes.

Resubstitution error: misclassification error, the proportion of misclassified observations on the training set.

The probability of resubstitution error is $2/46 = 0.0435 \rightarrow 4.35\,\%$

Example: classification of new data





The decision tree allows us to classify new samples:

$(1.0, 1.0)^t$: $y_1 \leq 3.5, y_2 \leq 2.5 \rightarrow$ class B
$(5.0, 2.5)^t$: $y_1 > 3.5 \rightarrow$ class B
$\ldots$

$(1.5, 2.7)^t$: $y_1 \leq 3.5, y_2 > 2.5 \rightarrow$ class A
$(2.0, 5.0)^t$: $y_1 \leq 3.5, y_2 > 2.5 \rightarrow$ class A
$\ldots$

Notation

- Representation space: $E \equiv \mathbb{R}^D$; $\boldsymbol{y} = (y_1, y_2. \dots, y_D)^t \in E$
- Training set /N learning samples: N feature vectors along with its correct classification (correct class): $(\boldsymbol{y}_1, c_1), \dots, (\boldsymbol{y}_N, c_N)$, $\boldsymbol{y}_i \in E$, $c_i \in \mathbb{C} = \{1, 2, \dots, C\}$, $1 \leq i \leq N$
- A tree is denoted by T (*“Tree”*)
  - a node is denoted by t
  - the left child of t is denoted by t_L
  - the right child of t is denoted by t_R
  - the set of terminal or leaf nodes by $\tilde{T}$
- T is a binary tree: the outcomes of a query in a node t are ’yes/no’
- A binary split is denoted by s and the set of admissible splits by S

Evaluating probability estimates from a DCT

Let be:

- N the number of training samples
- N_c the number of training samples that belong to class c
- $N(t)$ the number of training samples represented in node t
- $N_c(t)$ the number of training samples in node t that belong to class c

Prior probability of class c : $$\hat{P}(c) = \frac{N_c}{N}$$

Conditional (posterior) probability of class c in node t : $$\hat{P}(c \mid t) = \frac{N_c(t)}{N(t)}$$

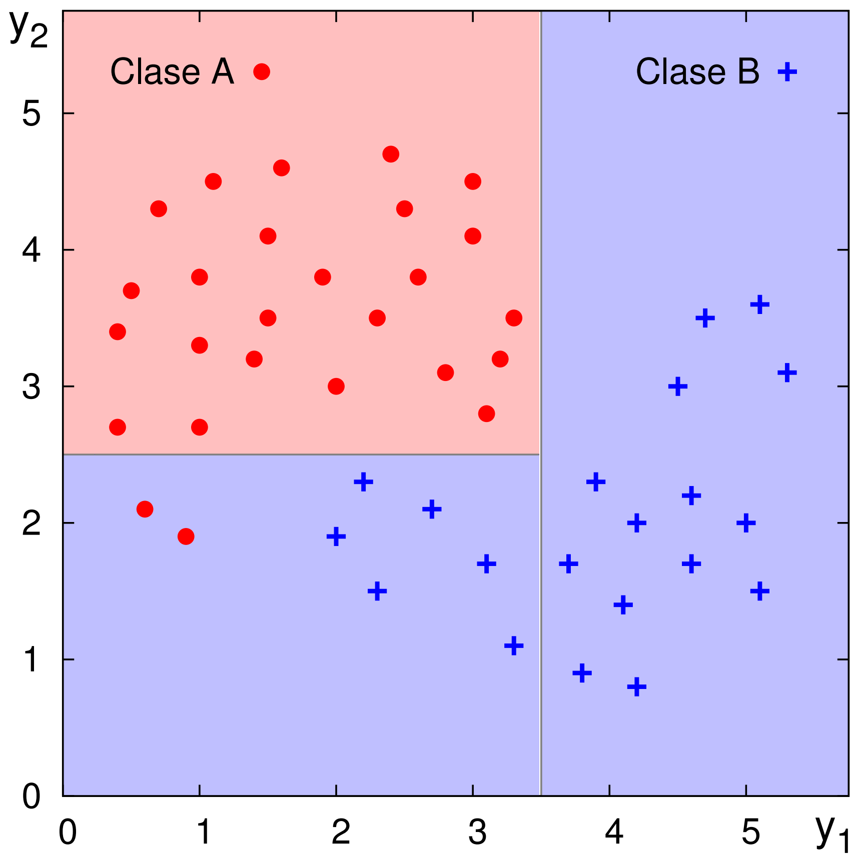
Probability of a terminal node $t \in \tilde{T}$: $$\hat{P}(t) = \frac{N(t)}{N}$$

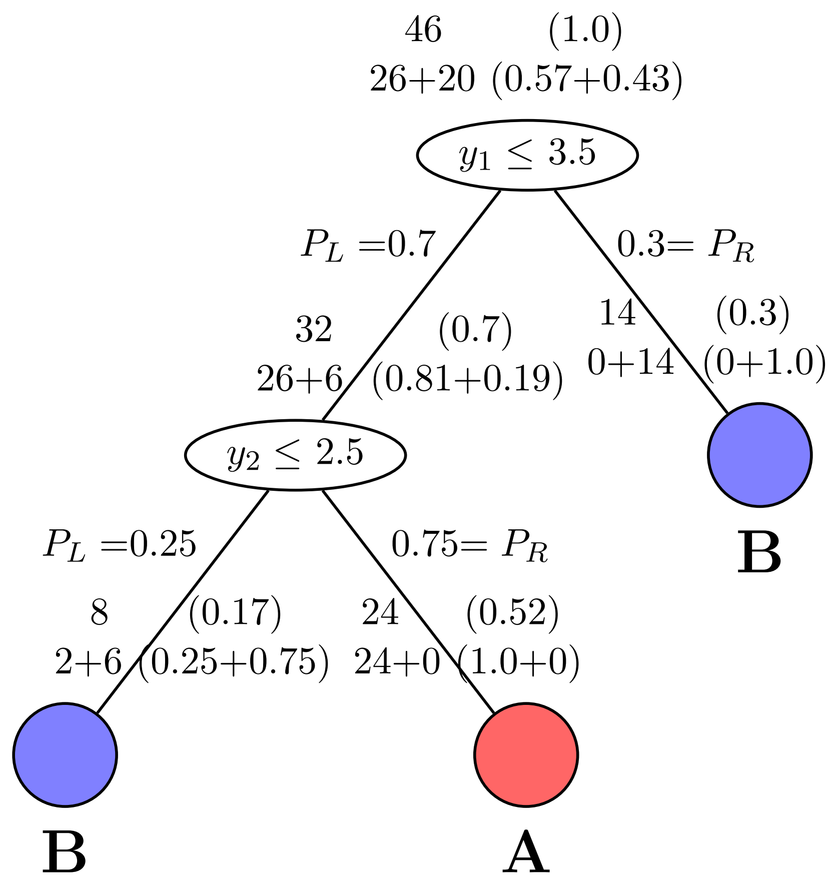
Probability of choosing the left node of t : $$\hat{P}_t(L) = \frac{N(t_L)}{N(t)}$$

Probability of choosing the right node of t : $$\hat{P}_t(R) = \frac{N(t_R)}{N(t)}$$

Exercise: calculate $\hat{P}(c)$ and $\hat{P}(c \mid t)$, $\hat{P}(t)$, $\hat{P}_t(L)$, $\hat{P}_t(R)$ $\forall t$, for the decision tree in page 7.

Solution to the exercise in page 10





Nodes:	$\hat{P}(t_i)$	$\hat{P}(A \mid t_i)$	$\hat{P}(B \mid t_i)$	$\hat{P}_{t_i}(L)$	$\hat{P}_{t_i}(R)$
t_1 (root)	–	0.565	0.435	0.696	0.304
t_2 (intermediate)	–	0.813	0.187	0.250	0.750
t_3 (leaf B)	0.304	0.000	1.000	–	–
t_4 (leaf B)	0.174	0.250	0.750	–	–
t_5 (leaf A)	0.522	1.000	0.000	–	–

Index

Building a DCT from a learning sample

Necessary elements to build a decision tree:

1. The method for splitting and selecting the best split; particularly:
   - Query selection: decide which property test or query should be performed at each node. Without loss of generality, queries will have the form: $\boldsymbol{y} \in B?,\ B \subseteq E$
   - Evaluation and optimization of the quality of a split
2. Criteria that makes the data in a node t as pure (homogeneous) as possible so as to declare t as *terminal node*. We will define the **impurity**, rather than the purity of a node.
3. Criteria to assign a class label to a terminal node.

Query selection for splitting a data set

- A query involves only one property j of E, $1 \leq j \leq D$
- Queries like $\boldsymbol{y} \in B$? are actually queries of the form $y_j \leq r$? (the value of property or feature y_j is lower or equal than r?)

  That is, a split is a pair $s = (j, r)$ made up of a component, $j \in \{1, \ldots, D\}$, and its corresponding threshold, $r \in \mathbb{R}$.
- Since splits define hyperplanes parallel to the feature axes, the resulting partitions are hyper-parallelepiped blocks (B), which are rectangular in the case $E = \mathbb{R}^2$.
- Since we work with a finite number of learning samples (N), there are only finitely many splits. For a node t with $N(t)$ elements:
  - We have to explore each feature j, $1 \leq j \leq D$, de E
  - For each j, we have to analyze (at least) $N(t)$ possible values of r

  Consequently, for each node t, we have to explore at least $\mathcal{O}(D\,N(t))$ splits.

Entropy

- It is a measure of the ***uncertainty*** associated to a decision between k classes (quantitative measure of the information not available when taking a decision):

$$H = -\sum_{i=1}^{k} P_i \log_2 P_i \qquad (0\ \log 0 \overset{\text{def}}{=} 0)$$

  - In the case of two classes ($k=2$) that have the same probability, the value of H is 1, the largest possible value of entropy (*one bit, information associated to a binary decision*).
  - The minimum value of H is 0 and corresponds to a decision for which there is only one possible choice.
  - The maximum value of H is $+\infty$. This happens when taking a decision between k classes all of them with the same probability and with $k \to \infty$:

- Examples:

If $P_1 = P_2 = 1/2, \qquad H = -(0.5(0-1)+0.5(0-1)) = 1$

If $P_1 = 1,\ P_2 = 0, \qquad H = -1 \cdot 0 + 0 = 0$

If $P_i = 1/k,\ 1 \le i \le k, \qquad H = log_2 k;\ \ H \to \infty$ If $k \to \infty$

Exercise: according to Eq.(1), calculate $\mathcal{I}(t)\ \forall t$ for the decision tree in page 7.

Evaluating the split quality (node impurity)

- We will use the concept of **node impurity** to evaluate the quality of a decision outcome at a node (split)
- Several different mathematical measures of impurity have been proposed, all of which have basically the same behaviour. Let $\mathcal{I}(t)$ denote the impurity of a node t. In all cases we want $\mathcal{I}(t)$ to be 0 if all of the patterns that reach node t bear the same category.

  The most popular measure is the **entropy impurity** (page. 15):

$$\mathcal{I}(t) = -\sum_{c=1}^{C} \hat{P}(c \mid t) \log_2 \hat{P}(c \mid t) = -\sum_{c=1}^{C} \frac{N_c(t)}{N(t)} \log_2 \frac{N_c(t)}{N(t)} \qquad (1)$$

  Other definitions of $\mathcal{I}(t)$: *Gini impurity* and *misclassification impurity* (see [1,2,3])

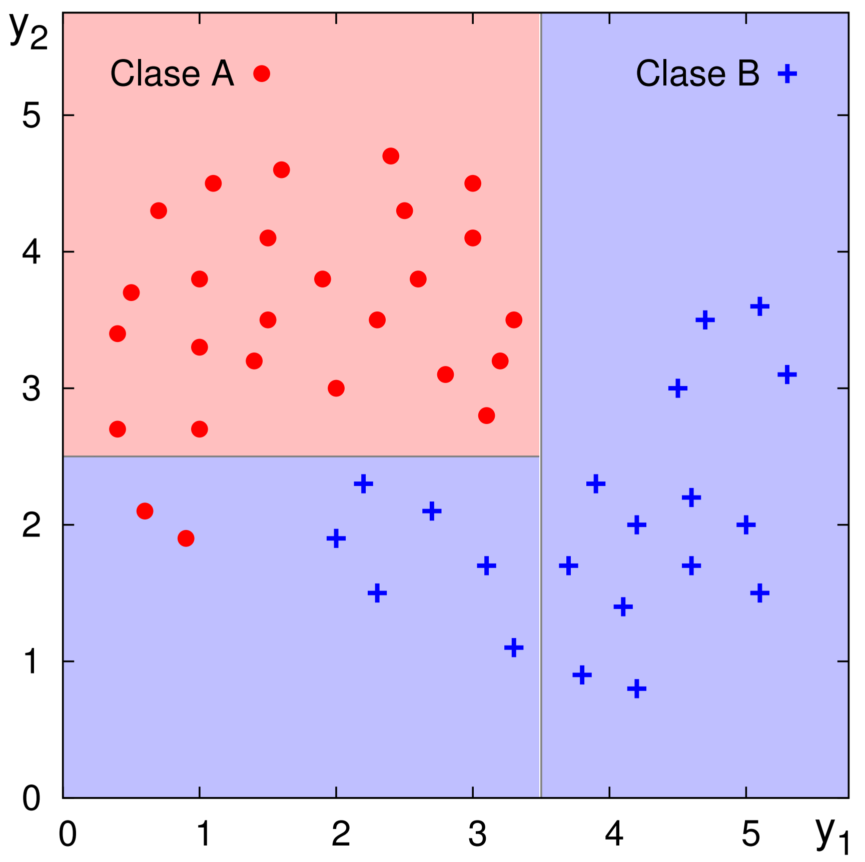
- Key question: given a partial tree down to a node t, what split $s = (j, r)$ should we choose for node t (i.e., which property query and value)?. The split quality is measured as the drop in impurity (***impurity reduction***):

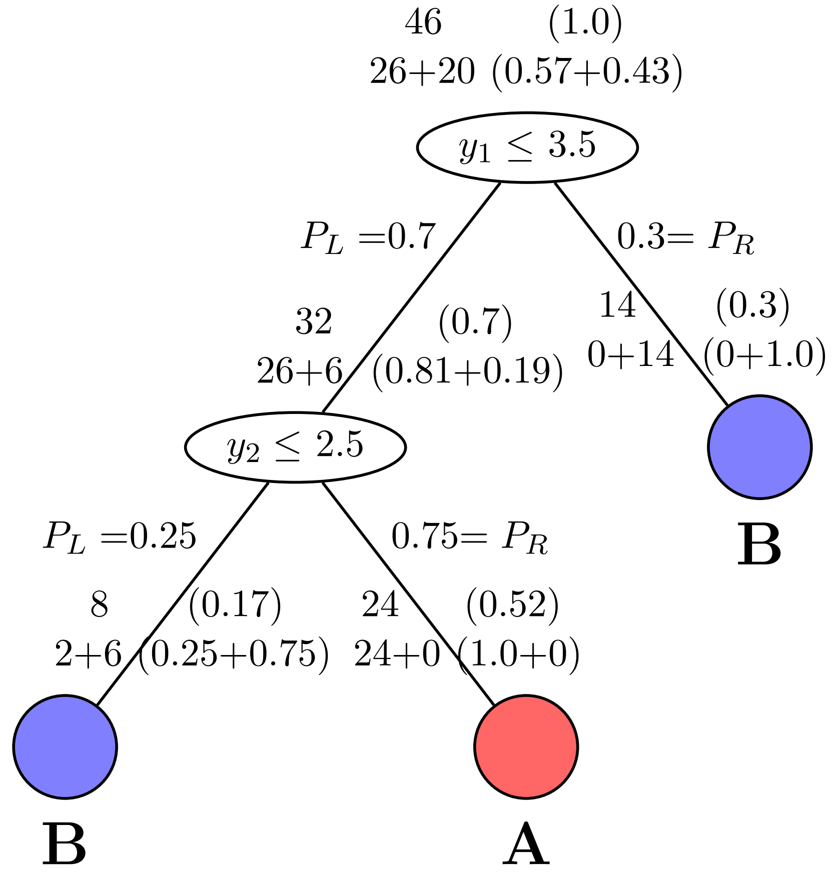
$$\Delta\mathcal{I}(j, r, t) \overset{\text{def}}{=} \mathcal{I}(t) - \hat{P}_t(L)\mathcal{I}(t_L) - \hat{P}_t(R)\mathcal{I}(t_R) \qquad (2)$$

- The best split is the choice for t that maximizes the impurity reduction $\Delta\mathcal{I}(j, r, t)$:

$$(j^\star, r^\star) = \underset{\substack{1 \le j \le D \\ -\infty < r < +\infty}}{\operatorname{argmax}} \Delta\mathcal{I}(j, r, t) \qquad (3)$$

Solution to the exercise in page 15





Nodes:	$\hat{P}(t_i)$	$\hat{P}(A \mid t_i)$	$\hat{P}(B \mid t_i)$	$\hat{P}_{t_i}(L)$	$\hat{P}_{t_i}(R)$	$\mathcal{I}(t_i)$
t_1 (root)	–	0.565	0.435	0.696	0.304	**0.988**
t_2 (intermediate)	–	0.813	0.187	0.250	0.750	**0.695**
t_3 (leaf B)	0.304	0.000	1.000	–	–	**0.000**
t_4 (leaf B)	0.174	0.250	0.750	–	–	**0.811**
t_5 (leaf A)	0.522	1.000	0.000	–	–	**0.000**

Purity criteria at terminal nodes

Simplest criterion: declare node t as a terminal node if the maximum impurity reduction is too small, smaller than a given threshold ϵ:

$$\max_{\substack{1\leq j\leq D\\ -\infty<r<+\infty}} \Delta\mathcal{I}(j, r, t) < \epsilon \tag{4}$$

where ϵ is a small constant selected empirically.

Another criterion: declare t as a terminal node only when t is totally pure (force every node to be a totally pure node). Inconvenient: trees are too large and too specific (little capacity for generalization).

In this case, we can resort to pruning methods. Let the tree grow fully. Then, all pair of neighboring leaf nodes (i.e, ones linked to a common antecedent node, one level above) are considered for elimination. Any pair whose elimination yields a satisfactory (small) increase in impurity is eliminated, and the common antecedent node is declared a leaf. Clearly, such *merging or joining* of the two leaf nodes is the inverse of splitting.

Assignment of leaf node labels

Simple criteria: each leaf should be labeled by the category (class) that has most points represented:

$$c^\star(t) = \operatorname*{argmax}_{1\leq c\leq C} \hat{P}(c \mid t), \quad \forall t \in \tilde{T} \tag{5}$$

Exercise (to do in class)

Following the example in page 7:

- According to equation (2), compute the impurity reduction in each of the two non-terminal nodes.
- The two splits in this example are the result from a pure geometric intuition ... (that is, they do not result from the optimization of an impurity expression).

  According to Eq. (3), analyze the second split $s = (2, 2.5)$; i.e., $y_2 \leq 2.5$, and say if any of the following splits is better for this node: $(y_1 \leq 1.95)$, $(y_2 \leq 1.8)$
- In this example, one of the terminal nodes is not pure. According to Eq. (4), which is the minimum value of ϵ for which the decision of declaring this node as a *terminal node* is correct?

Solution to the exercise in page 19

- Impurity reduction for t_1 and t_2:

Nodes:	$\hat{P}(t_i)$	$\hat{P}(A \mid t_i)$	$\hat{P}(B \mid t_i)$	$\hat{P}_{t_i}(L)$	$\hat{P}_{t_i}(R)$	$\mathcal{I}(t_i)$	$\Delta\mathcal{I}(t_i)$
t_1 (root)	–	0.565	0.435	0.696	0.304	0.988	**0.504**
t_2 (intermediate)	–	0.813	0.187	0.250	0.750	0.695	**0.492**
t_3 (leaf B)	0.304	0.000	1.000	–	–	0.000	–
t_4 (leaf B)	0.174	0.250	0.750	–	–	0.811	–
t_5 (leaf A)	0.522	1.000	0.000	–	–	0.000	–

- Alternative *Splits* in t_2:

$$(y_1 \leq 1.95) : \mathcal{I}(t_L) = 0, \ \mathcal{I}(t_R) = -(11/17)\log(11/17) - (6/17)\log(6/17) = 0.937$$
$$(y_2 \leq 1.80) : \mathcal{I}(t_L) = 0, \ \mathcal{I}(t_R) = -(26/29)\log(26/29) - (3/29)\log(3/29) = 0.480$$

$$\Delta\mathcal{I}(1, 1.95, t_2) = 0.695 - 0 - (17/32) \cdot 0.937 = 0.197 < 0.492$$
$$\Delta\mathcal{I}(2, 1.80, t_2) = 0.695 - 0 - (29/32) \cdot 0.480 = 0.260 < 0.492$$

Then none of these *splits* would be better than $(y_2 \leq 2.5)$.

- The impurity of t_4 es $\mathcal{I}(t_4) = 0.811$. The max value of $\Delta\mathcal{I}(t_4)$ would be with a split that would yield two pure nodes; for example, $s = (1.1.5)$ ($y_1 \leq 1.5$). In this case, $\mathcal{I}(t_{4L}) = \mathcal{I}(t_{4R}) = 0$, and the max impurity reduction for t_4 would be $0.811 - 0 - 0 = 0.811$. Consequently, for t_4 to be considered terminal, ϵ should be higher than 0.811. Evidentemente el ejemplo no corresponde a un resultado real de aprendizaje de ADC, ya que con este valor de ϵ tanto el nodo interno como el mismo nodo raiz se habrían considerado terminales).

Algorithm DCT

Tree BuildNode(*class* c, *component* j, *threshold* r, *nodes* t_L, t_R) //builds a tree (↑node); j is the property

```
Tree DCT(sample $\mathcal{Y} \equiv (\boldsymbol{y}_1, c_1), \ldots, (\boldsymbol{y}_n, c_n)$) {        // learns a classification tree
    $(j^\star, r^\star, \delta)$ = BestSplit($\mathcal{Y}$)        // by Eq. (1,2,3); $\delta$ is the impurity reduction
    if ($\delta < \epsilon$) {        // if $\delta$ is too small – Eq. (4)
        $c$ = DominantClass($\mathcal{Y}$)        // Eq. (5)
        return BuildNode($c$, –, –, NULL, NULL)        // builds a terminal node and is assigned class $c$
    } else {
        $\mathcal{Y}_L = \mathcal{Y}_R = \emptyset$        // $\emptyset$ is the empty set
        $\forall(\boldsymbol{y}, c) \in \mathcal{Y}$ {        // split according to $j^\star, r^\star$
            if ($y_{j^\star} \leq r^\star$)    $\mathcal{Y}_L = \mathcal{Y}_L \cup \{(\boldsymbol{y}, c)\}$
            else /*($y_{j^\star} > r^\star$)*/    $\mathcal{Y}_R = \mathcal{Y}_R \cup \{(\boldsymbol{y}, c)\}$
        }
        $t_L$ = DCT($\mathcal{Y}_L$)        // recursively builds the left subtree
        $t_R$ = DCT($\mathcal{Y}_R$)        // recursively builds the right subtree
        return BuildNode($0, j^\star, r^\star, t_L, t_R$)
    }
}
```

Complexity (I)

Temporal Complexity:

Let's assume a node t has m training samples. *BestSplit( )* has to explore m values of the threshold (r) for each of the D components (j). That is, Dm queries or splits.

For each split (j and r), we need the entropy impurity of t_L and t_R in order to compute best split (impurity reduction). Then, we need to calculate $N(t_L)$, $N_c(t_L)$, $N(t_R)$ and $N_c(t_R)$. An easy way to calculate these values is to sort the m samples according to j so that the calculations can be done incrementally. Sorting a list can be done in $\Theta(m \log m)$ steps.

Consequently, to compute the *BestSplit( )* of a node t, we need:

- to make Dm queries
- to sort the m samples D times with a cost of $\Theta(Dm \log m)$

Therefore, the cost of ***BestSplit( ) for a node*** t is $\Theta(Dm + Dm \log m)) \;=\; \Theta(D\, m \log m)$.

Complexity (II): complexity of a tree

Once we know the cost of *BestSplit( )*, we will see now the cost for the entire tree. We need to compute:

- how many nodes in the tree
- how many samples at each node

Let's assume a balanced tree, where roughly half the training samples are sent to each of the two branches. If the root node has N samples, then these samples go through all the way down the tree until they all reach a terminal node. Then, there will be N samples at each level of the tree. Consequently, we have:

- tree depth = $\log N$ (the depth of a binary tree with N leaves is at least $\log N \rightarrow \Omega(\log N)$)
- the number of nodes at depth h is 2^h nodes
- a node at a depth h has $N/2^h$ samples.

Then, the total temporal cost is (see [1]):

$$\mathcal{O}(\sum_{h=0}^{logN} 2^h \, D \, \frac{N}{2^h} \log \frac{N}{2^h}) \quad = \quad \mathcal{O}(D \, N \, (logN)^2 \,)$$

In practice, trees are never completely balanced nor completely unbalanced and the cost is usually closer to the best case.

Spatial complexity: A tree has less than $2N$ nodes, each using a fixed and small amount of memory. Therefore, the spatial cost is $\mathcal{O}(N)$

Resubstitution estimate of the misclassification error

Let's see now how to estimate the misclassification error of a classification tree. Since all of the N samples of the root node have to reach a terminal node, in order to compute the misclassification error of a tree we need:

- misclassification error of a terminal node
- probability of a terminal node

According to the *statistical decision theory*, the probability of misclassification error of a terminal node t estimated by resubstitution, is:

$$\hat{P}_t(\text{error}) \;=\; 1 - \max_{1 \leq c \leq C} \hat{P}(c \mid t)$$

And for a tree T:

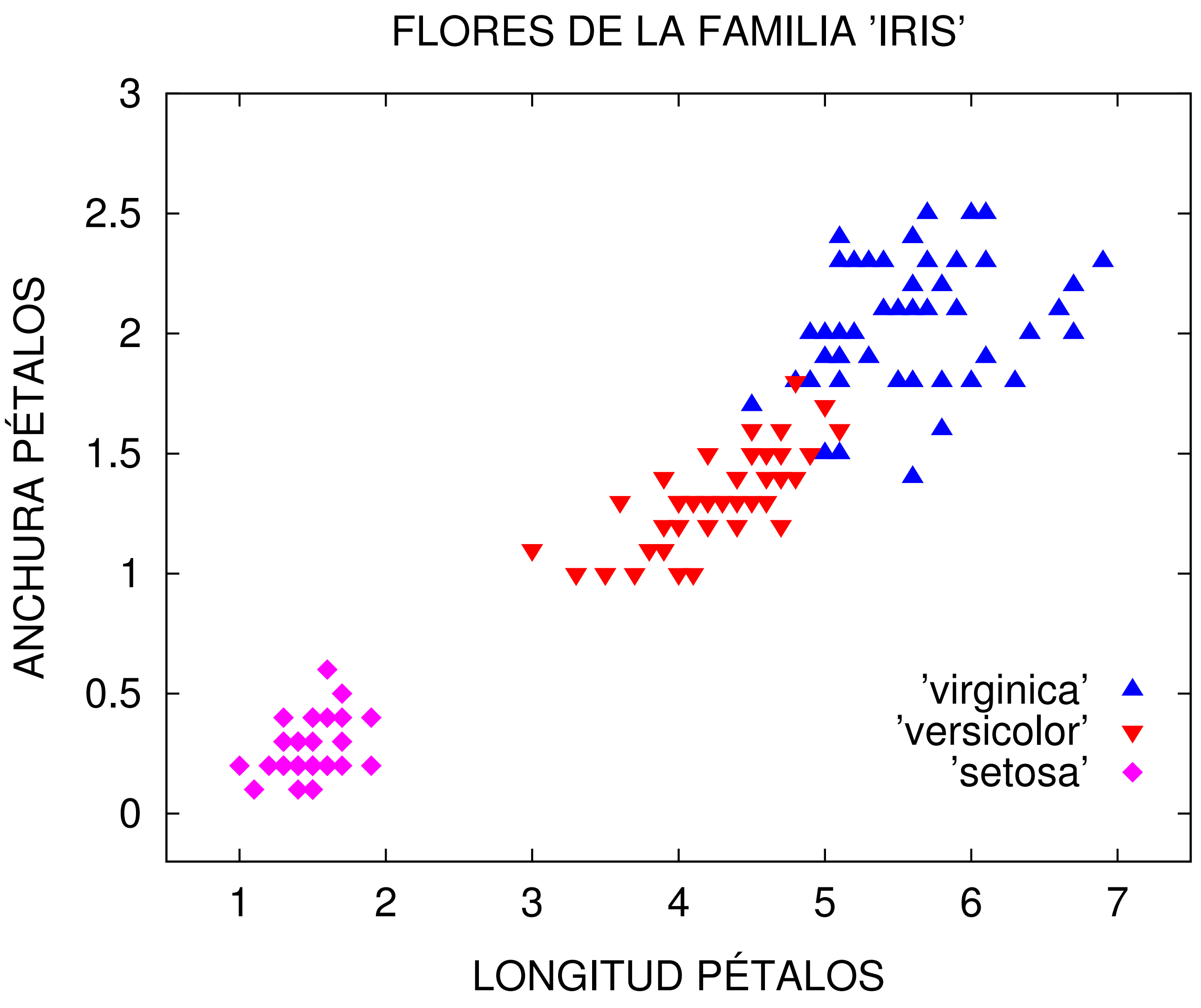
$$\hat{P}_T(\text{error}) \;=\; \sum_{t \in \tilde{T}} \hat{P}(t)\hat{P}_t(\text{error})$$

In the example of page 7 (3 terminal nodes, 2 pure nodes):

$$\hat{P}_T(\text{error}) \;=\; \frac{14}{46} \cdot 0 + \frac{8}{46} \cdot \frac{2}{8} + \frac{24}{46} \cdot 0 \;=\; \frac{2}{46} \;\approx\; 0.0435 \;\rightarrow\; 4.35\,\%$$

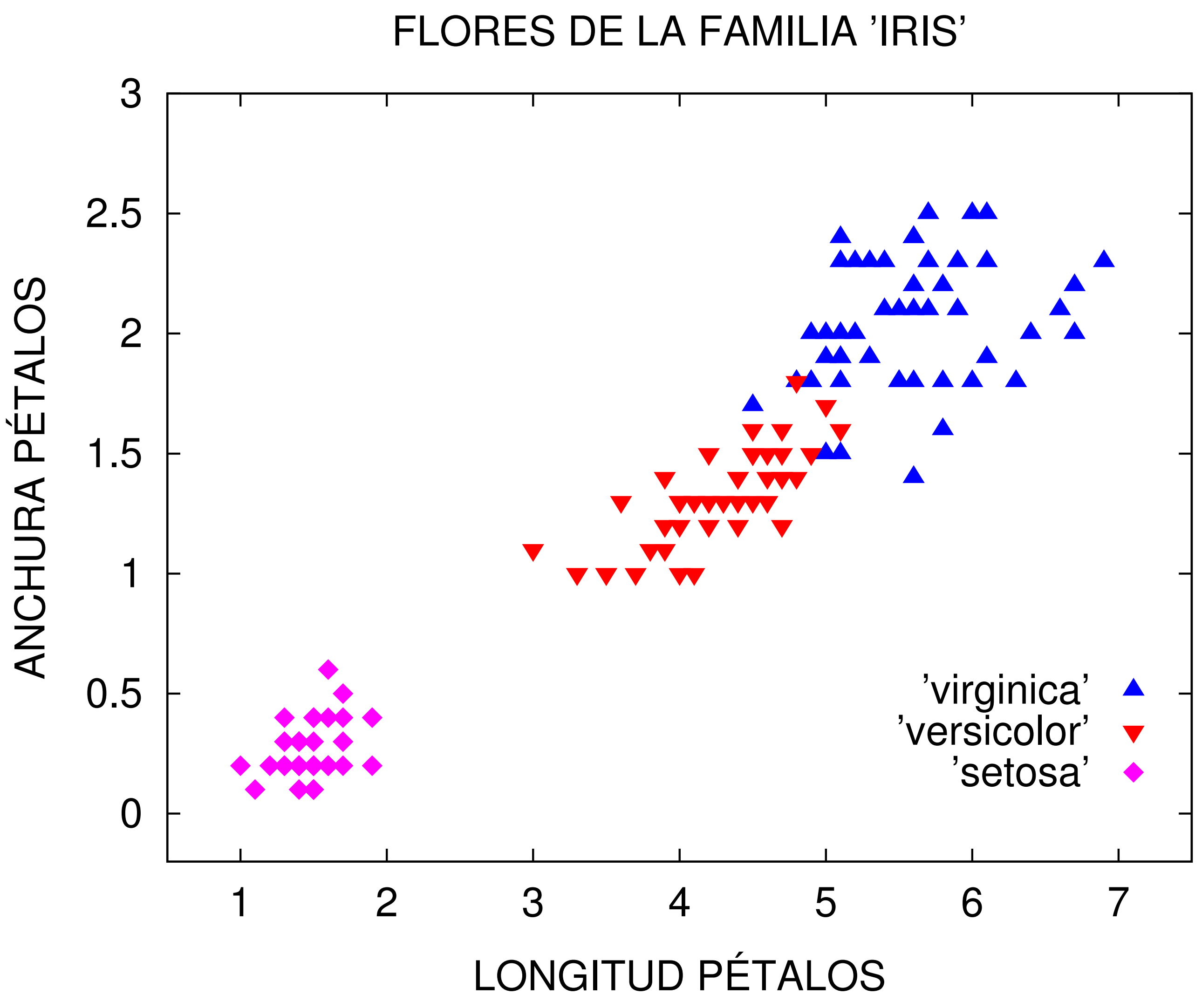
If the tree grows fully up to pure terminal nodes, the resubstitution error estimate will be 0 since $\hat{P}_e(t) = 0 \; \forall t \in \tilde{T}$. An extremely small resubstitution error is not necessarily desirable because it may be an indication that the tree is overfitting the training data (too specific tree, lack of generalization).

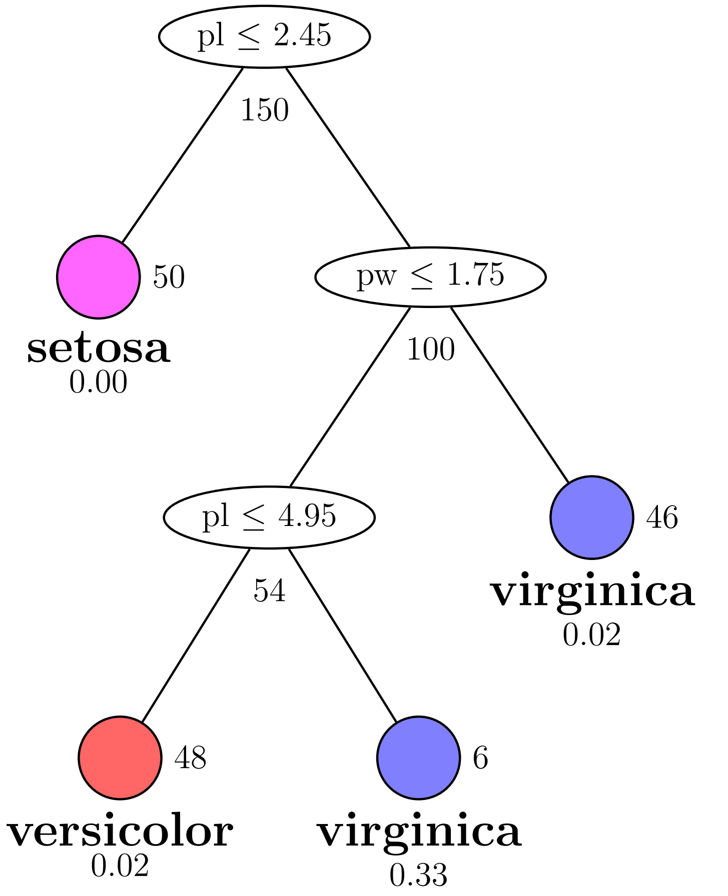
Example: learning a DCT to classify iris flowers



- 3 classes,
  one class is easily separable
- 4 dimensions ($E = \mathbb{R}^4$)
- 150 vectors,
  50 of each class

Example: learning a DCT to classify iris flowers

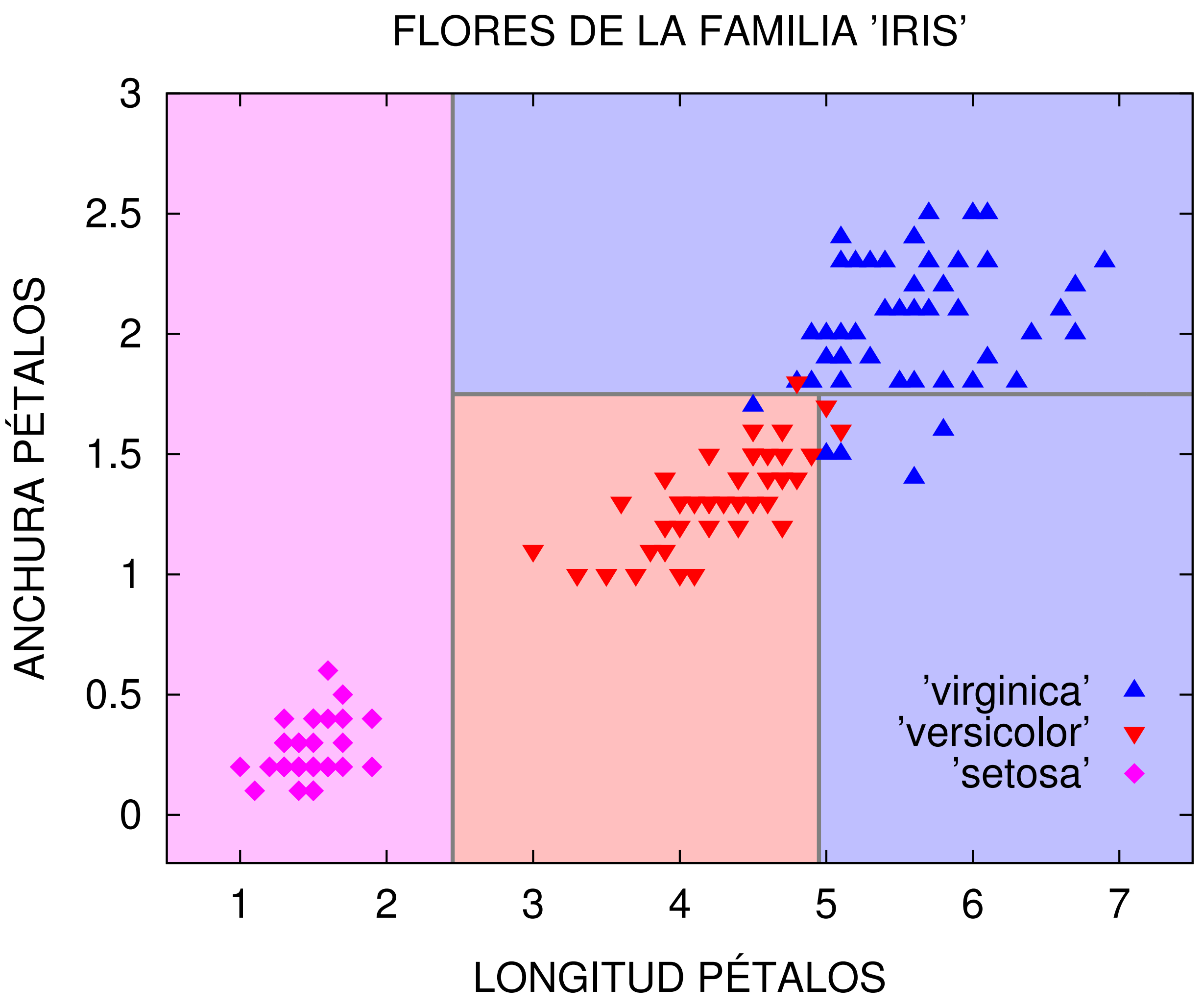


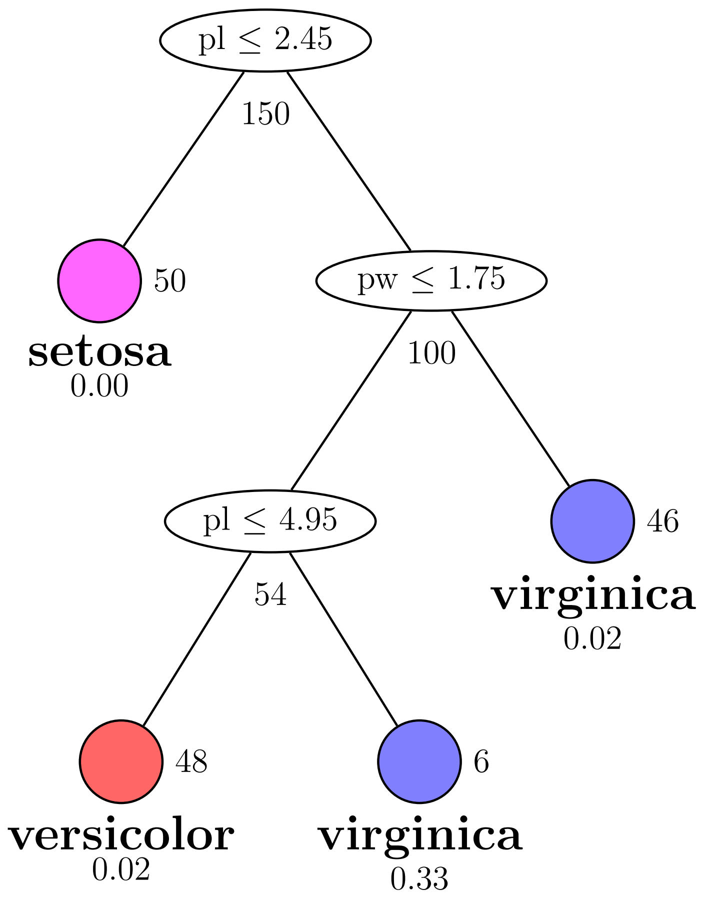


Tree learnt:

We only use 2 out of the 4 dimensions

Example: learning DCTs to classify iris flowers





Boundaries and classification error. Estimated resubstitution error:

$$(50/150)0.0 + (46/150)0.02 + (48/150)0.02 + (6/150)0.33 = 0.02 \rightarrow 2.6\,\%$$

Example: learning DCTs to classify iris flowers

0.02 in virginica: 1 misclassified sample $(1/46 = 1 - 45/46 = 0.02)$

0.02 in versicolor: 1 misclassified sample $(1/48 = 1 - 47/48 = 0.02)$

0.33 in virginica: 2 misclassified samples $(2/6 = 1 - 4/6 = 0.33)$

Estimated resubstitution error:

$$(50/150)0.0 + 0.02(46/150) + 0.02(48/150) + 0.33(6/150) = 0.02 \rightarrow 2.6\,\%$$

Overall, 4 misclassified samples out of 150: $(4/150 = 0.026)$

Other approaches and criteria for DCTs

See details in [1,2,3]:

- Discrete attributes and categorical attributes (attributes whose domain is not numerical)
- Pure terminal nodes and posterior pruning
- Other impurity measures: *Gini impurity* and *misclassification impurity*
- Splits that are not parallel to the axes: splits along arbitrary directions in the feature space (multivariate trees)
- Non-binary splits
- Use of the *twoing* criterion in multiclass binary tree creation
- Improve misclassification error: cross validation
- Handling missing attributes in patterns
- Regression trees (instead of classification trees)

Index

Bibliography

[1] R.O. Duda, D.G. Stork, P.E. Hart. Pattern Classification. Wiley, 2001.

[2] A. R. Webb, K. D. Copsey. Statistical Pattern Recognition. Wiley, tercera ed., 2011.

[3] Classification and Regression Trees by L. Breiman, J.H. Friedman, R.A. Olshen y C.J. Stone. Chapman & Hall, 1984.

The material of this chapter is mostly based on [1] and [2].