Introduction to tree classifiers

Prof. Roberto Garello

Information Theory and Applications
Politecnico di Torino

2019/2020

Statistical Classification

Starting from a **training set** of data it builds a **classifier** that, given a **new instance**'s **features**, identifies which **class** it belongs to.

Data $\equiv$ vector of features

$$\underline{v} = (X_1, \ldots, X_M)$$

Each vector belongs to a class

$$f(\underline{v}) = c_v$$

Training Set $\equiv$ set of vectors and corresponding class

$$T_S = \{(\underline{v}, c_v)\}$$

From the Training set we build a classifier

$$g(\underline{v}) = c_v$$

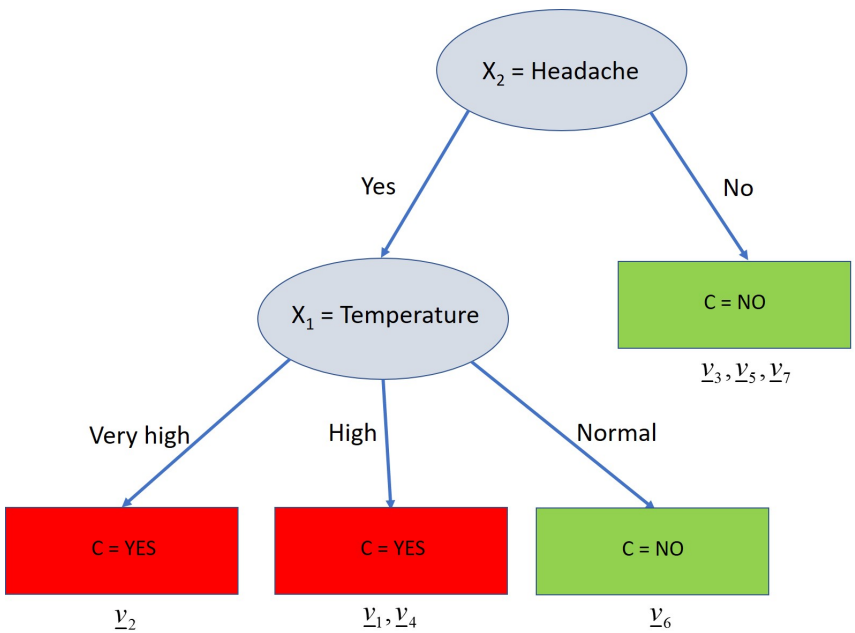
Given a new vector we can compute its class

$$\forall \underline{w} \notin T_S : g(\underline{w}) = c_w$$

Example

Given a record of data a doctor must understand if a patient is healthy or ill

	Attributes			*Decision*
	Temperature	**Headache**	**Nausea**	**Flu**
1	high	yes	no	yes
2	very_high	yes	yes	yes
3	normal	no	no	no
4	high	yes	yes	yes
5	high	no	yes	no
6	normal	yes	no	no
7	normal	no	yes	no



Examples:

- given a record of data a doctor must understand if a patient is healthy or ill.
- given a number of features, an antispam filter must decide if an email is spam or not.
- give a set of observations, we must automatically decide which animal we have observed.

Statistical Classification and Machine Learning

Statistical classification is a branch of Machine learning, in particular **supervised learning** (because we use a training set).

Similar Machine Learning Problems

Clustering: group data into different classes **without a starting training set (unsupervised)**, by using some distance properties.

Regression: while classification assigns to a new vector a class, regression assigns a **real value**.

Statistical Classification and Machine Learning

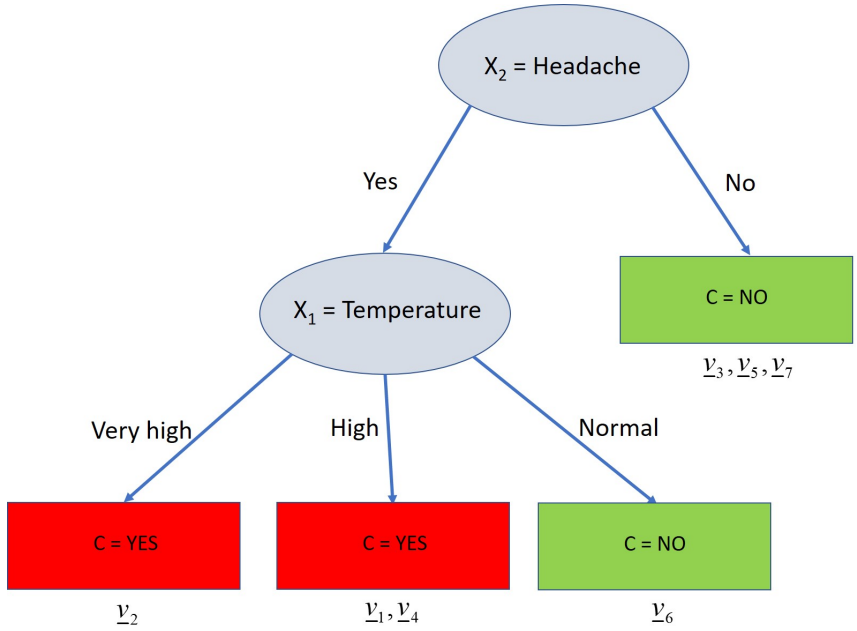
Statistical classification has many different approaches:

- Linear Discriminant Analysis
- Naive Bayes Classifiers
- Nearest neighbor
- Support Vector Machine
- **Decision Trees**
- Neural Networks

Statistical Classification by Decision Trees

Application of information theory (entropy and mutual information) to Statistical classification: Decision tree classifiers.

Decision Tree = decision-making tool based on a **flowchart**-like diagram that, given an object, computes an outcome from **a series of decisions** taken by analyzing the object's features.



For classification we first use a training set to build the decision tree. Then, for any new object, we use the tree to analyze its features and compute the outcome, which is the class assigned to the new object.

Vector

Vector with M elements:

$$\underline{v} = (x_1, \ldots, x_i, \ldots, x_M) \qquad x_1 \in A_1 \ldots x_i \in A_i \ldots x_M \in A_M$$

Alphabet:

$$\underline{v} \in A_{\mathrm{TOT}} \qquad A_{\mathrm{TOT}} = A_1 \times \cdots \times A_i \times \cdots \times A_M$$

with cardinality:

$$N_{\mathrm{TOT}} = |A_{\mathrm{TOT}}| = \prod_{i=1}^{M} |A_i|.$$

Sometimes not all the vectors are admissible, so we denote the set of admissible vectors by $A_v \subseteq A_{\mathrm{TOT}}$ and its cardinality by $N_v \leq N_{\mathrm{TOT}}$.

Alphabets

$$x_1 \in A_1 \quad \ldots \quad x_i \in A_i \quad \ldots \quad x_M \in A_M$$

- Alphabets A_i may be different.
- Alphabets may be categorical (set of elements, e.g., colors, names, $\{\mathrm{Yes}, \mathrm{No}\}$, ...) or numerical (integers, reals, ...).

Class

$$f : A_v \Rightarrow A_c$$

$$f : \underline{v} = (x_1, \cdots, x_i, \cdots, x_M) \in A_v \quad \Rightarrow \quad c \in A_c$$

There is a function f mapping each admissible M-dimensional vector $\underline{v}$ into a class value $c = f(\underline{v})$.

Usually, the alphabet A_c is categorical.

Often, the alphabet A_c is binary ($\{\mathrm{Yes}, \mathrm{No}\}$, $\{\mathrm{Buy}, \mathrm{Sell}\}$, $\{\mathrm{Healthy}, \mathrm{Ill}\}$, $\cdots$)

Terminology (machine learning)

$$f : \underline{v} = (x_1, \cdots, x_i, \cdots, x_M) \in A_v \Rightarrow c \in A_c$$

- Vector $\underline{v} \Rightarrow$ **instance**
- Elements $x_i \Rightarrow$ **features**
- Value $c \Rightarrow$ **class**

Other terminology (statistics)

$$f : \underline{v} = (x_1, \cdots, x_i, \cdots, x_M) \in A_v \Rightarrow c \in A_c$$

- Vector $\underline{v} \Rightarrow$ **observation or example**
- Elements $x_i \Rightarrow$ **variables**
- Value $c \Rightarrow$ **category**

Training set

$$f : \underline{v} = (x_1, \cdots, x_i, \cdots, x_M) \in A_v \Rightarrow c \in A_c$$

$$N_v = |A_v| \leq N_{\mathrm{TOT}} = |A_{\mathrm{TOT}}| = \prod_{i=1}^{M} |A_i|.$$

The total function f over the entire domain A_v of N_v elements is unknown, i.e., the rule that given any vector $\underline{v}$ provides the corresponding class $c = f(\underline{v})$ is unknown.

But we know $N_{TS} < N_v$ vectors $\underline{v}$ and their corresponding classes $c = f(\underline{v})$.

They form a **training set** T_S for the classifier.

Classifier

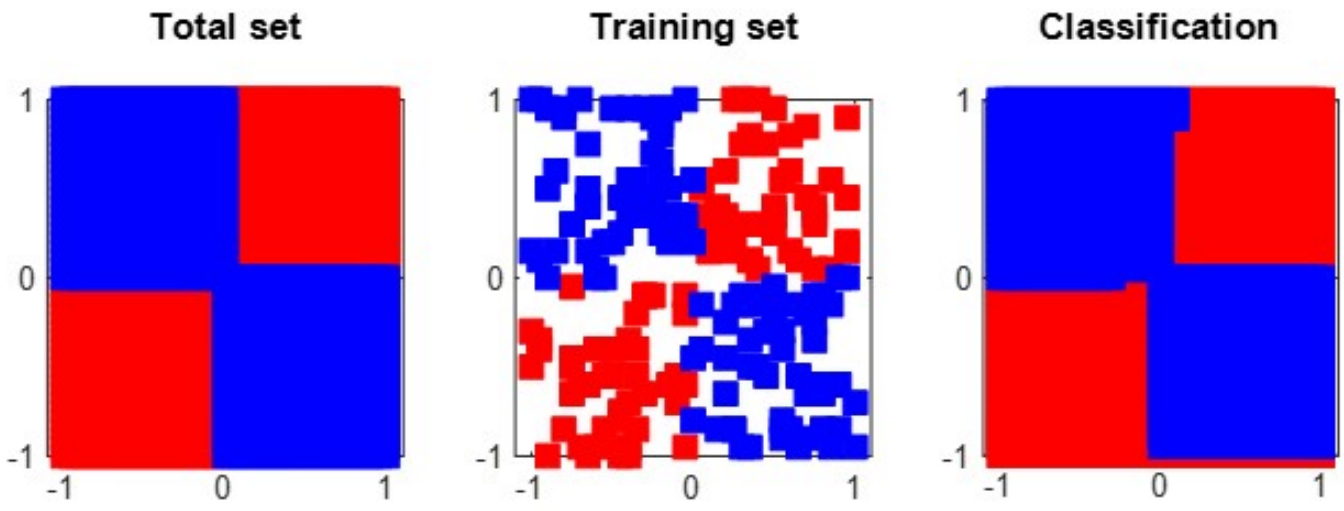
We use the $N_{TS} < N_v$ vectors $\underline{v} \in T_S$ of the **training set** to build a **classifier**.
A classifier is an algorithm that assigns to each vector $\underline{v} \in A_v$ a class $c' = g(\underline{v})$:

$$g : \underline{v} = (x_1, \cdots, x_i, \cdots, x_M) \in A_v \Rightarrow c' \in A_c.$$

Our goal is to create a classifier able to compute the correct class for **all the admissible vectors** (also those outside the training set), i.e., able to learn the **entire true law** f from the **small training set** T_S:

$$\forall \underline{v} \in A_v : g(\underline{v}) = f(\underline{v}),$$

or, at least, to minimize the number of wrong associations.


Total set
Training set
Classification
1
0
-1
-1
0
1

Summary

Original function

$$f : \underline{v} = (x_1, \cdots, x_i, \cdots, x_M) \in A_v \Rightarrow c \in A_c$$

Training set of $N_{TS} \leq N_v = |A_v|$ vectors.

Classifier function

$$g : \underline{v} = (x_1, \cdots, x_i, \cdots, x_M) \in A_v \Rightarrow c' \in A_c$$

Scope

$$g(\underline{v}) = f(\underline{v}) \qquad \forall \underline{v} \in A_v$$

Entropy

Decision Tree Classifiers are built by using Information Theory tools.
Discrete Random Variable A.
A takes values $\{a_i\}_{i=1}^{M}$ with probabilities $p(a_i)$.
We define the **entropy** as:

$$H(A) = \sum_{i=1}^{M} p(a_i) \, log_2 \left(\frac{1}{p(a_i)} \right) \quad \text{bits}$$

The entropy is maximum when the uncertainty is maximum, i.e., all probabilities are the same:

$$\forall i \quad p(a_i) = 1/M \longrightarrow H(A) = \log_2(M)$$

Conditional Entropy

Two discrete Random Variables A and B.
A takes values $\{a_i\}_{i=1}^{M}$ with probabilities $p(a_i)$.
B takes values $\{b_j\}_{j=1}^{N}$ with probabilities $p(b_j)$.
We introduce

$$H(A|B=b_j) = \sum_{i=1}^{M} p(a_i|b_j) log_2 \left(\frac{1}{p(a_i|b_j)} \right)$$

and we define the **conditional entropy** as

$$H(A|B) = \sum_{j=1}^{N} p(b_j) H(A|B=b_j) = \sum_{i=1}^{M} \sum_{j=1}^{N} p(a_i, b_j) log_2 \left(\frac{1}{p(a_i|b_j)} \right)$$

Reduction of uncertainty

We always have:

$$H(A) \geq H(A|B)$$

Entropy is a measure of uncertainty (information) for A.
When we know B we have some information on A, too, then its uncertainty is reduced.

Mutual Information

$$I(A;B) = H(A) - H(A|B)$$

- $H(A)$ = uncertainty on A.
- $H(A|B)$ = residual uncertainty on A when B is known.
- $I(A;B)$ = reduction on A uncertainty provided by B.

In classifier literature, the **mutual information** is often called the information gain and is denoted by $IG(B \longrightarrow A)$.

Information Gain Ratio

It is defined as the ratio between the mutual information (information gain) and the entropy of the conditioning variable :

$$IGR\left(B \longrightarrow A\right) = \frac{I\left(A;B\right)}{H\left(B\right)}$$

The reason to divide by the entropy $H(B)$ is the following. Suppose that A depends on two variables B_1 and B_2. If the size of B_1 is bigger than that of B_2 we can expect a bigger reduction on A uncertainty. By dividing by the entropy of B (that increases with the cardinality), we normalize the information gain and the comparison between variables B with different size is fairer.

Problem

Original function

$$f : \underline{v} = (x_1, \cdots, x_i, \cdots, x_M) \in A_v \Rightarrow c \in A_c$$

Training set of $N_{\mathrm{TS}} \leq N_v = |A_v|$ vectors.

Algorithm function

$$g : \underline{v} = (x_1, \cdots, x_i, \cdots, x_M) \in A_v \Rightarrow c' \in A_c$$

Scope

$$g(\underline{v}) = f(\underline{v}) \qquad \forall \underline{v} \in A_v$$

Example

$$f : \underline{v} = (x_1, x_2, x_3) \in A_v \Rightarrow c \in A_c$$

$\underline{v}$	x_1	x_2	x_3	c
	Temperature	**Headache**	**Nausea**	**Flu**
1	high	yes	no	yes
2	very_high	yes	yes	yes
3	normal	no	no	no
4	high	yes	yes	yes
5	high	no	yes	no
6	normal	yes	no	no
7	normal	no	yes	no

$$A_1 = \{\text{High}, \text{Very high}, \text{Normal}\}$$

$$A_2 = \{\text{Yes}, \text{No}\}$$

$$A_3 = \{\text{Yes}, \text{No}\}$$

$$A_c = \{\text{Yes}, \text{No}\}$$

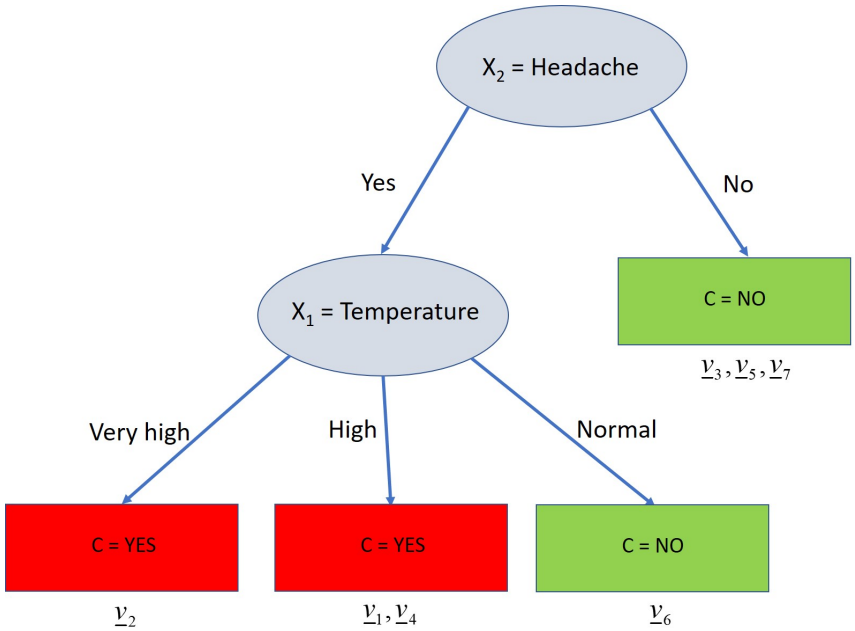
Decision Tree Classifier

- A flowchart-like diagram.
- Each internal node represents a test on a **single feature**.
- Each branch represents the test outcome.
- Each leaf represents the class label.

The path from root to leaf represents the classification function g that, given a vector $\underline{v}$, analyzes its features and associates a class c. The main advantage of tree classifier is the simplicity (for building, interpreting and applying it)

Example

	Attributes			Decision
	Temperature	Headache	Nausea	Flu
1	high	yes	no	yes
2	very_high	yes	yes	yes
3	normal	no	no	no
4	high	yes	yes	yes
5	high	no	yes	no
6	normal	yes	no	no
7	normal	no	yes	no



new vector $\underline{w} = (\text{very high}, \text{yes}, \text{no}) \Longrightarrow c = \text{YES}$

ID3 Decision Tree Classifier: Idea

Iterative Dichotomiser 3 algorithm (by **Ross Quinlan**).
Given a training set, it generates a **Decision Tree Classifier** based on this key idea:

At each step use the feature that provides the largest reduction of the class uncertainty (i.e., the largest Information Gain Ratio).

ID3 Decision Tree Classifier: Algorithm

The algorithm generates a Decision Tree in this way:

- Given a set S of vectors $\underline{v} = (x_1, \cdots, x_i, \cdots, x_M)$ and their classes $c = f(\underline{v})$, compute the feature X_i that maximizes the Information Gain Ratio $IGR(C, X_i)$.
- Create a decision node for this feature X_i and split the set S into subsets according to it.
- Recurse on subsets by using the remaining features.

Example

	Attributes			Decision
	Temperature	Headache	Nausea	Flu
1	high	yes	no	yes
2	very_high	yes	yes	yes
3	normal	no	no	no
4	high	yes	yes	yes
5	high	no	yes	no
6	normal	yes	no	no
7	normal	no	yes	no

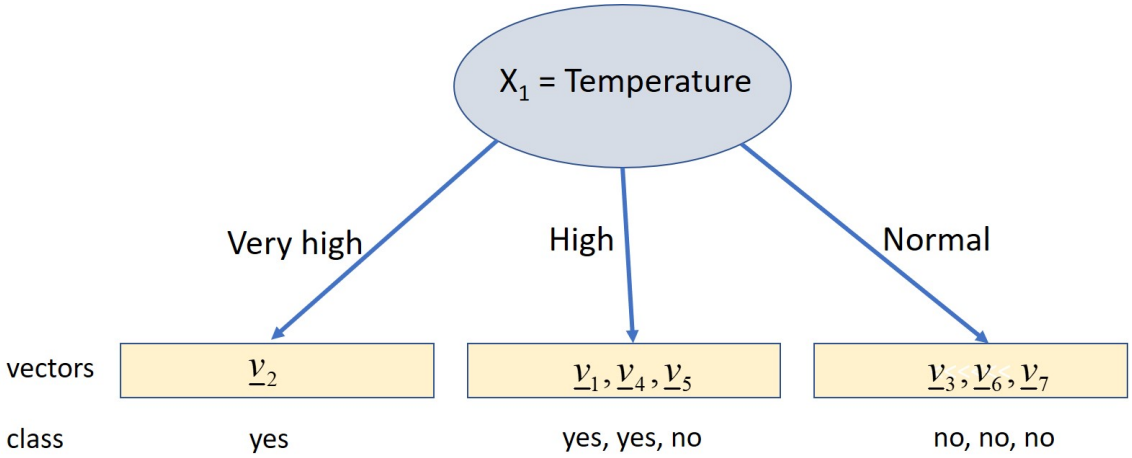
Let us first compute the entropy of the class:

$$P(c = \text{yes}) = \frac{3}{7} \quad P(c = \text{no}) = \frac{4}{7}$$

$$H(C) = 0.9852$$

Example

Let us consider the first feature:



$$H(C) = 0.9852$$

$$H(C|X_1 = \text{Very High}) = 0$$

$$H(C|X_1 = \text{High}) = 0.9183$$

$$H(C|X_1 = \text{Normal}) = 0$$

$$H(C|X_1) = 0.3936$$

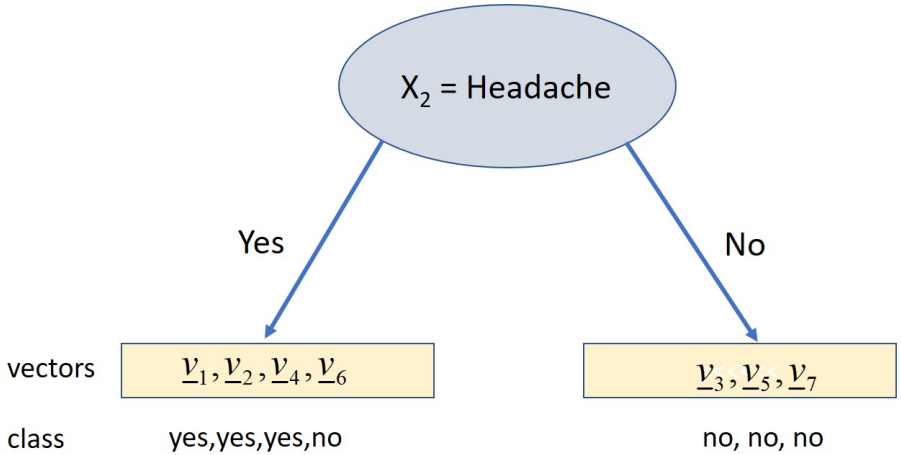
$$I(C; X_1) = 0.5916$$

$$H(X_1) = 1.4488$$

$$IGR(C, X_1) = 0.4083$$

Example

Let us consider the second feature:



$$H(C) = 0.9852$$

$$H(C|X_2 = \text{Yes}) = 0.8113$$

$$H(C|X_2 = \text{No}) = 0$$

$$H(C|X_2) = 0.4636$$

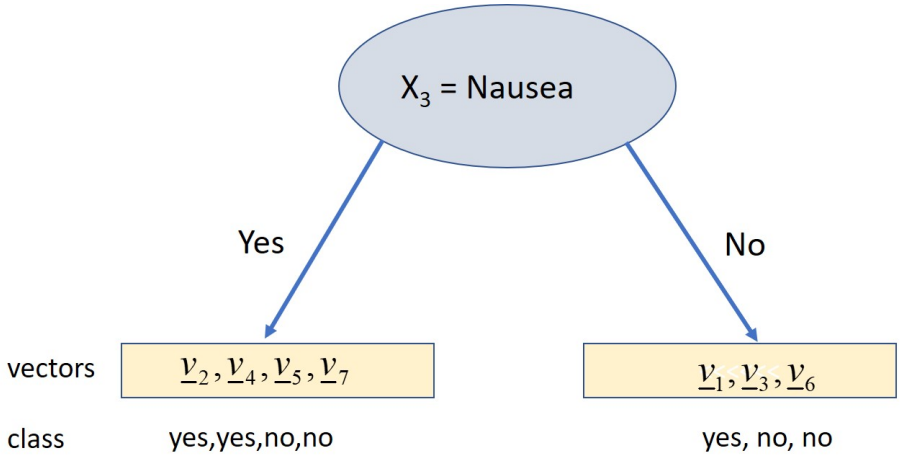
$$I(C; X_2) = 0.5216$$

$$H(X_2) = 0.9852$$

$$IGR(C, X_2) = 0.5294$$

Example

Let us consider the third feature:



$$H(C) = 0.9852$$

$$H(C|X_3 = \text{Yes}) = 1$$

$$H(C|X_3 = \text{No}) = 0.9183$$

$$H(C|X_3) = 0.9650$$

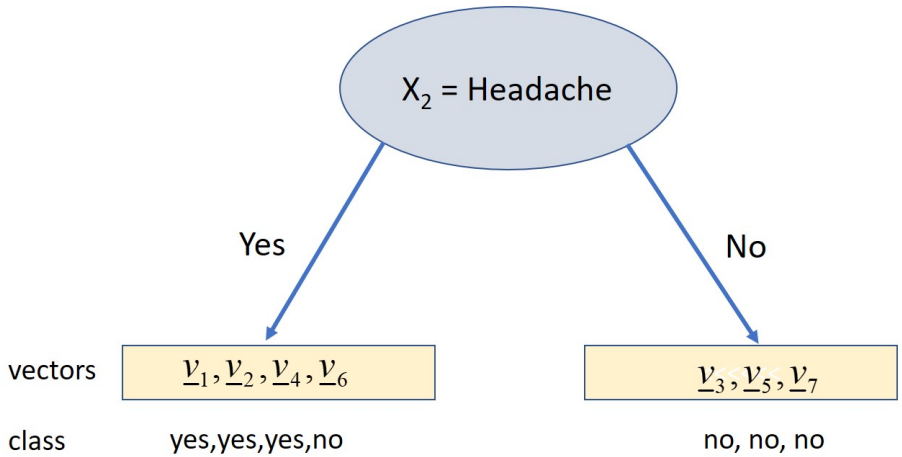
$$I(C; X_3) = 0.0202$$

$$H(X_3) = 0.9852$$

$$IGR(C, X_3) = 0.0205$$

Example

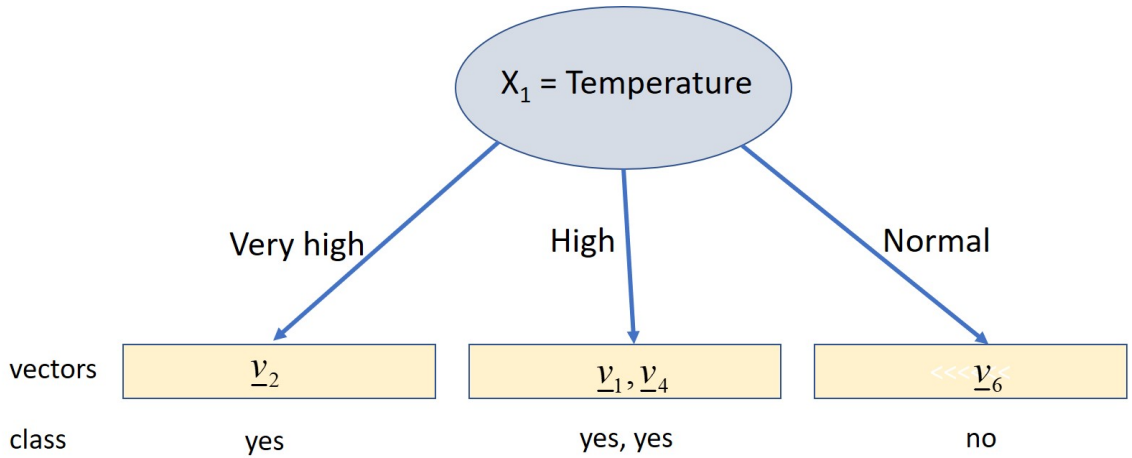
Based on the best Information Gain Ratio, we choose the second feature:



Now we apply the same procedure to the subset $\underline{v_1}, \underline{v_2}, \underline{v_4}, \underline{v_6}$. We consider the remaining features: x_1 =Temperature and x_3 = Nausea.

Example

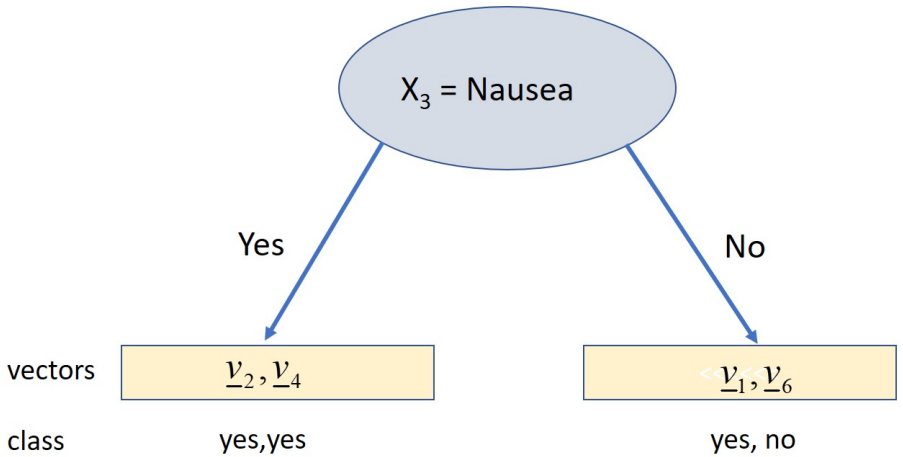
Let us consider the first feature x_1 =Temperature:



$$H(C) = 0.8113$$
$$H(C|X_1 = \text{Very High}) = 0$$
$$H(C|X_1 = \text{High}) = 0$$
$$H(C|X_1 = \text{Normal}) = 0$$
$$H(C|X_1) = 0$$
$$I(C; X_1) = 0.8113$$
$$H(X_1) = 1.5$$
$$IGR(C, X_1) = 0.5408$$

Example

Let us consider the third feature x_3 =Nausea:



$$H(C) = 0.8113$$

$$H(C|X_3 = \text{Yes}) = 0$$

$$H(C|X_3 = \text{No}) = 1$$

$$H(C|X_3) = 0.5$$

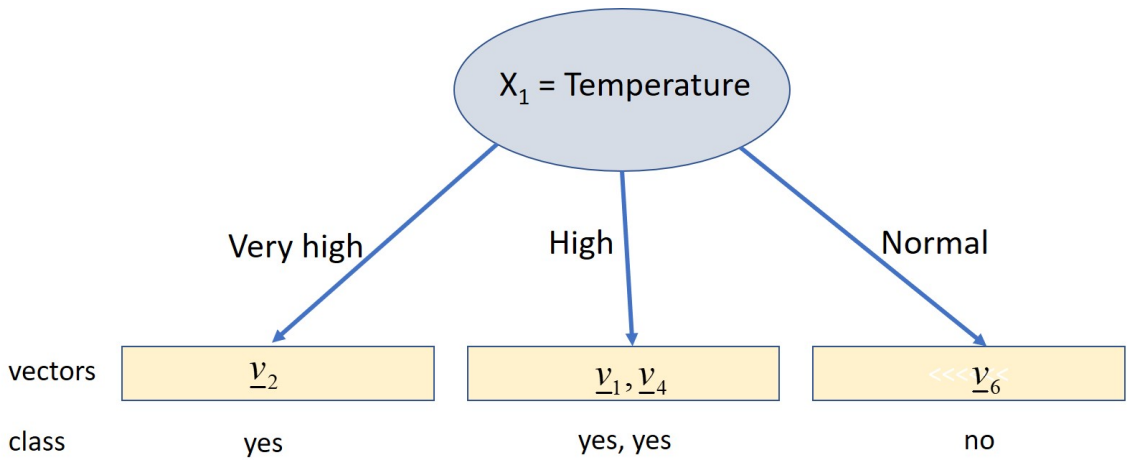
$$I(C; X_3) = 0.3113$$

$$H(X_3) = 0.9852$$

$$IGR(C, X_3) = 0.3160$$

Example

Based on the best Information Gain Ratio, we choose the first feature:



Example

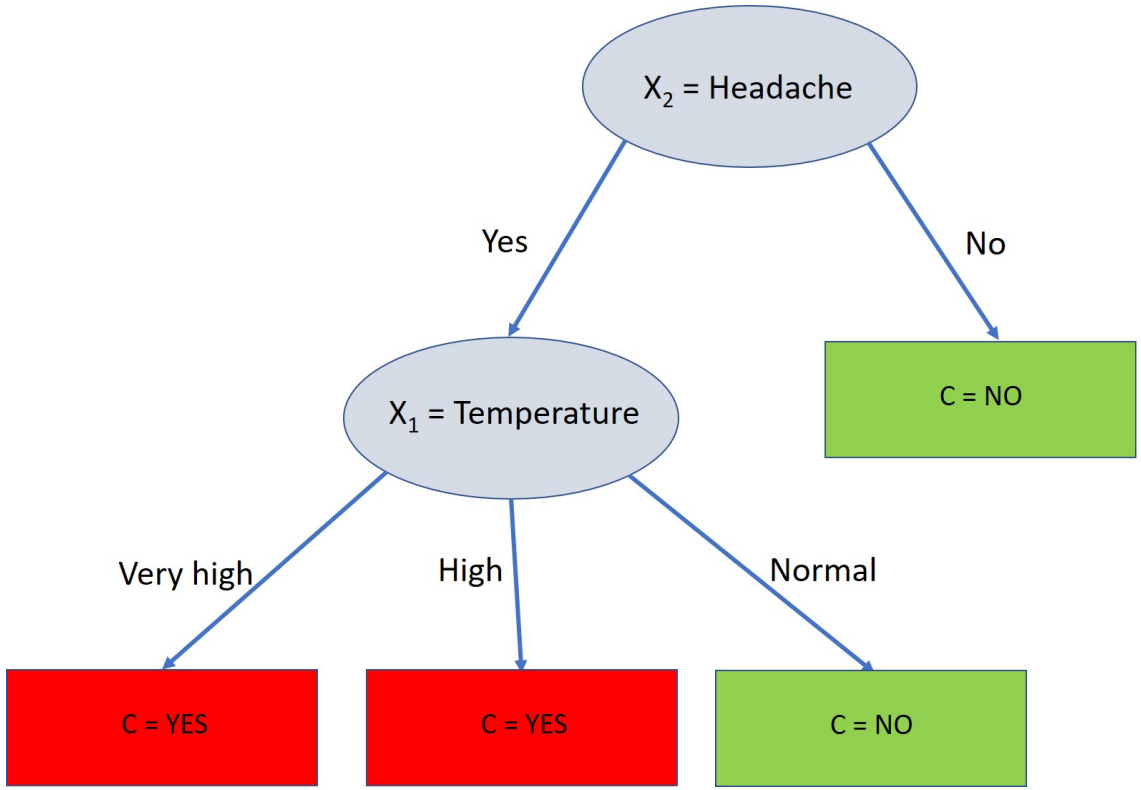
For the other subset it is useless to make a choice because all the vectors already belong to the same class:

$\underline{v}_3, \underline{v}_5, \underline{v}_7$

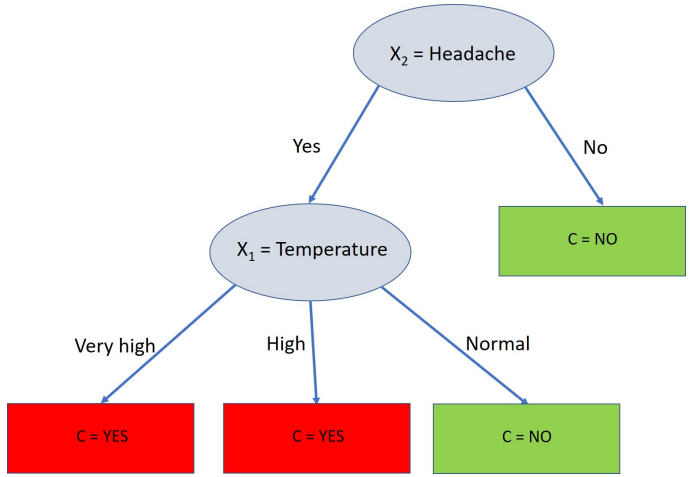
no, no, no

Example

Finally we have our decision tree:



Example



Now we can use the decision tree as a classifier. If we observe a new vector, we apply the rule to compute the class it belongs to. As an example

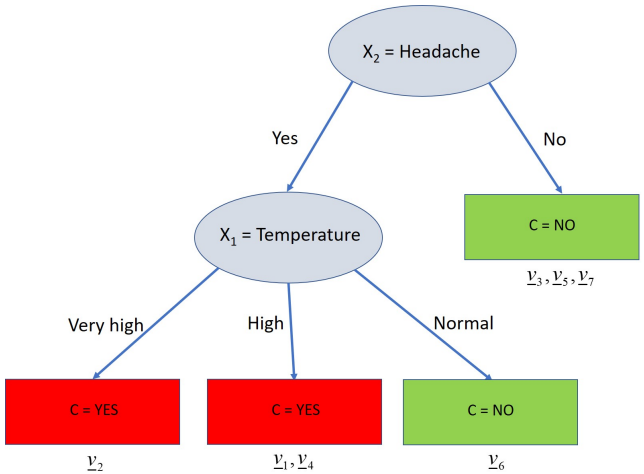
$$\underline{w} = (\text{normal}, \text{yes}, \text{yes})$$

is mapped into class NO

$$\underline{w} = (\text{veryhigh}, \text{yes}, \text{no})$$

is mapped into class YES

Example



Note that we can translate the tree classifier into these logical rules:

$$(\text{Headache} = \text{yes}) \quad \text{AND}\, (\text{Temperature} = \text{veryhigh}) \longrightarrow (\text{Class} = \text{YES})$$

$$(\text{Headache} = \text{yes}) \quad \text{AND}\, (\text{Temperature} = \text{high}) \longrightarrow (\text{Class} = \text{YES})$$

$$(\text{Headache} = \text{yes}) \quad \text{AND}\, (\text{Temperature} = \text{normal}) \longrightarrow (\text{Class} = \text{NO})$$

$$(\text{Headache} = \text{no}) \longrightarrow (\text{Class} = \text{NO})$$

Some stopping criteria

The algorithm generates the Decision Tree in this way:

- Given a set S of vectors $\underline{v} = (x_1, \cdots, x_i, \cdots, x_M)$ and their classes $c = f(\underline{v})$, compute the feature X_i that maximizes the Information Gain Ratio $IGR(C, X_i)$.
- Create a decision node for this feature X_i and split the set S into subsets according to it.
- Recurse on subsets by using the remaining features.

Stopping criterion:

- When a subset contains only vectors of the same class: we create a leaf node labeled by this class.
- When a subset contains vectors of different classes, but all the features have already been considered: we create a leaf node labeled by the most common class among the subset vectors.
- When a subset is empty: we create a leaf node labeled by the most common class of the parent's subset.

Exercise 1

Generate the classification tree for the previous example and verify the two given vector examples.

Exercise 2

Suppose that X_1, X_2, X_3 are three binary variables.
Consider this rule

$$C = (X_1 \ \text{OR} \ X_2) \ \text{AND} \ X_3$$

Generate the eight vectors $\underline{v} = (X_1, X_2, X_3))$ and the corresponding classes $C = f(\underline{v})$.
Use them as training set and build the corresponding classification tree.

Numerical values

C4.5 algorithm

If X_i is a numerical variable, proceed as follows:

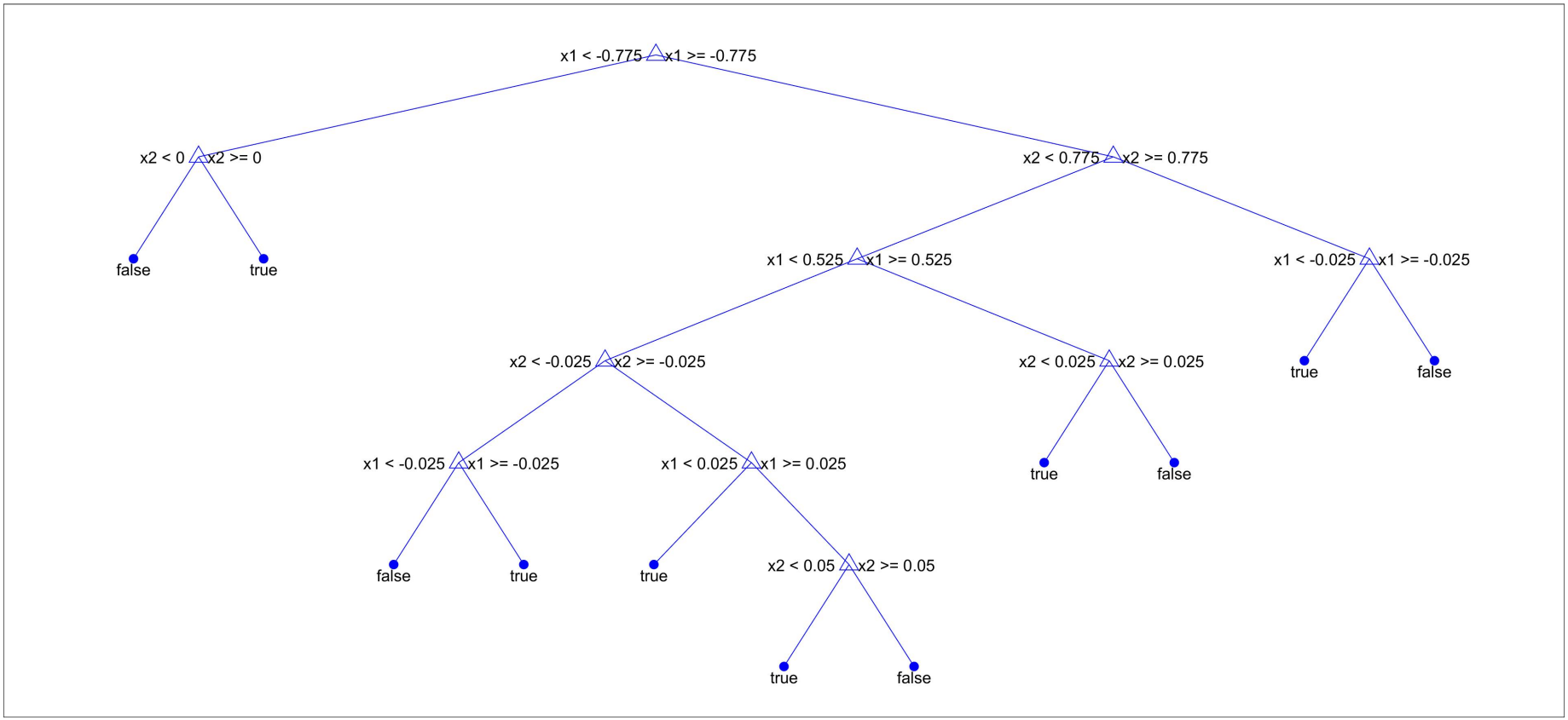
- Fix a threshold t.
- Split the X_i values in two sets for $X_i \leq t$ and $X_i > t$.
- Compute the information gain ratio with respect to this binary representation of X_i.
- Consider all possible threshold values t.
- Consider the threshold giving the maximum information gain ratio and use this binary partition for X_i as in the ID3 algorithm.

C4.5 and Numerical Features

Each categorical feature can be considered only once, as in ID3.

Important: Numerical features can be considered again: if we select a numerical feature at a given level, we can select it again at a lower level, with a different threshold.

Example 1


x1 < -0.775
x1 >= -0.775
x2 < 0
x2 >= 0
x2 < 0.775
x2 >= 0.775
false
true
x1 < 0.525
x1 >= 0.525
x1 < -0.025
x1 >= -0.025
x2 < -0.025
x2 >= -0.025
x2 < 0.025
x2 >= 0.025
true
false
x1 < -0.025
x1 >= -0.025
x1 < 0.025
x1 >= 0.025
true
false
false
true
true
x2 < 0.05
x2 >= 0.05
true
false

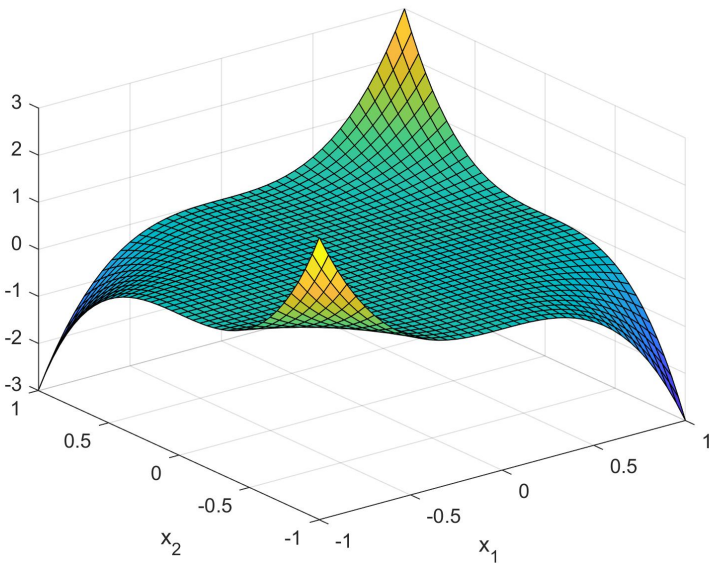
Example 1

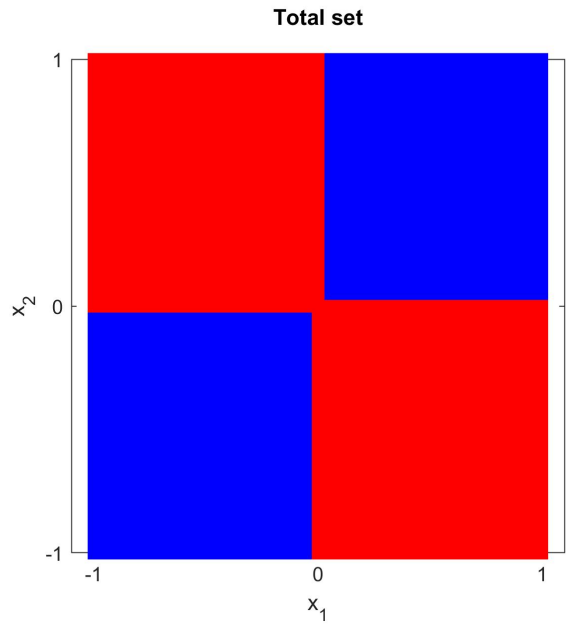
Decision tree for classification
1 if $x_2 \leq -0.125$ then node 2 elseif $x_2 > -0.125$ then node 3 else false
2 if $x_1 \leq -0.025$ then node 4 elseif $x_1 > -0.025$ then node 5 else false
3 if $x_1 \leq 0.025$ then node 6 elseif $x1_{>}0.025$ then node 7 else true
4 class = false
5 class = true
6 if $x_2 \leq -0.025$ then node 8 elseif $x_2 > -0.025$ then node 9 else true
7 if $x_2 \leq 0.025$ then node 10 elseif $x_2 > 0.025$ then node 11 else false
8 class = false
9 class = true
10 class = true
11 class = false

Example 2

$$\underline{v} = (x_1, x_2)$$

$$c = f(\underline{v}) = \text{sign}\left(3x_1^3 x_2^3\right)$$



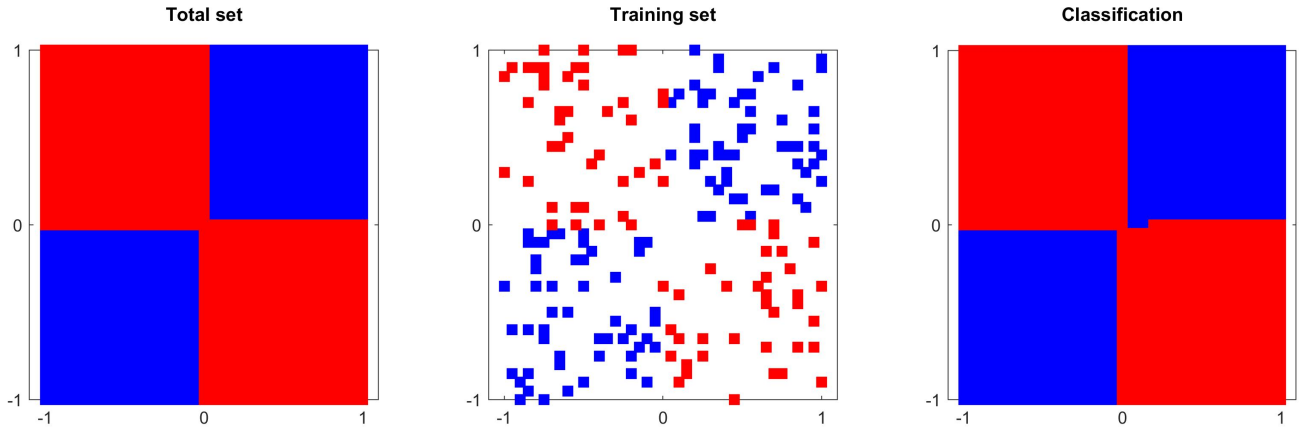


Example 2: training sets 1

$$x_1 = -1 : 0.05 : 1 \quad x_2 = -1 : 0.05 : 1;$$

$$N_v = 1681$$

$$N_{TS} = 300$$



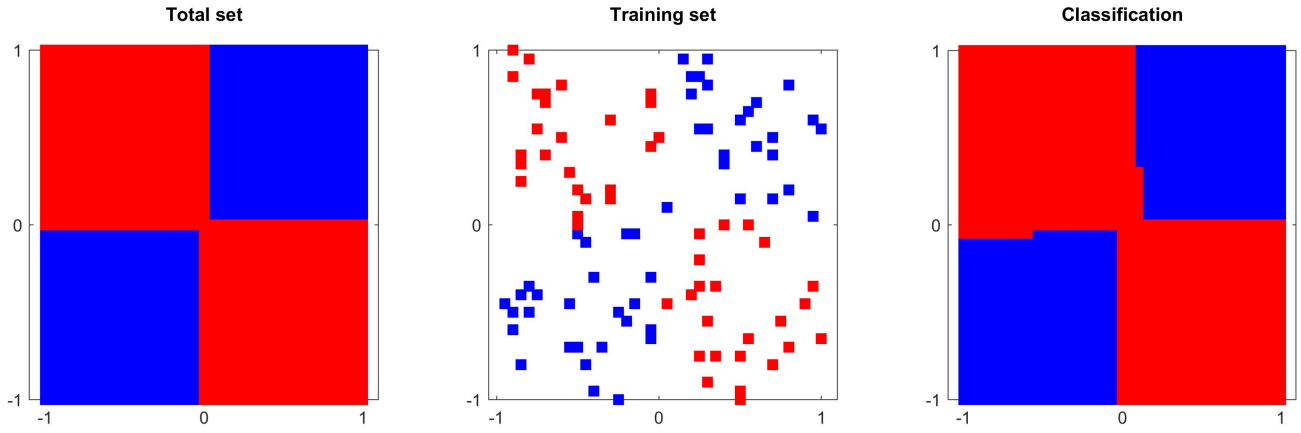
number of errors: 3/1681

Example 2: training sets 2

$$x_1 = -1 : 0.05 : 1 \quad x_2 = -1 : 0.05 : 1;$$

$$N_v = 1681$$

$$N_{TS} = 100$$



number of errors: 35/1681

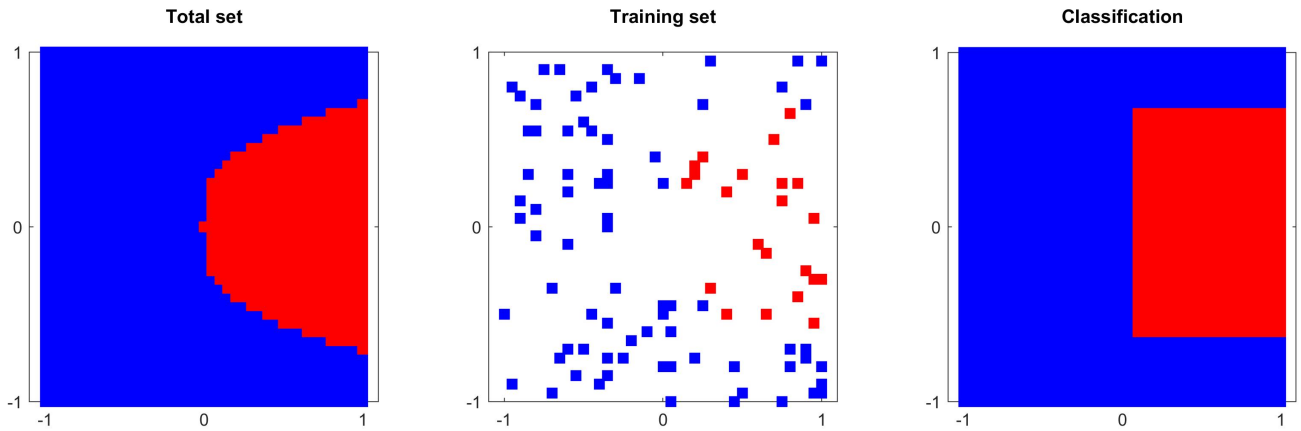
Example 3: training set 1

$$\underline{v} = (x_1, x_2)$$

$$c = f(\underline{v}) = \text{sign}\left(-2\sqrt[3]{x_1^2} + 4x_2^2\right)$$

$$x_1 = -1 : 0.05 : 1 \quad x_2 = -1 : 0.05 : 1;$$

$$N_v = 1681 \quad N_{TS} = 100$$



number of errors: 97/1681

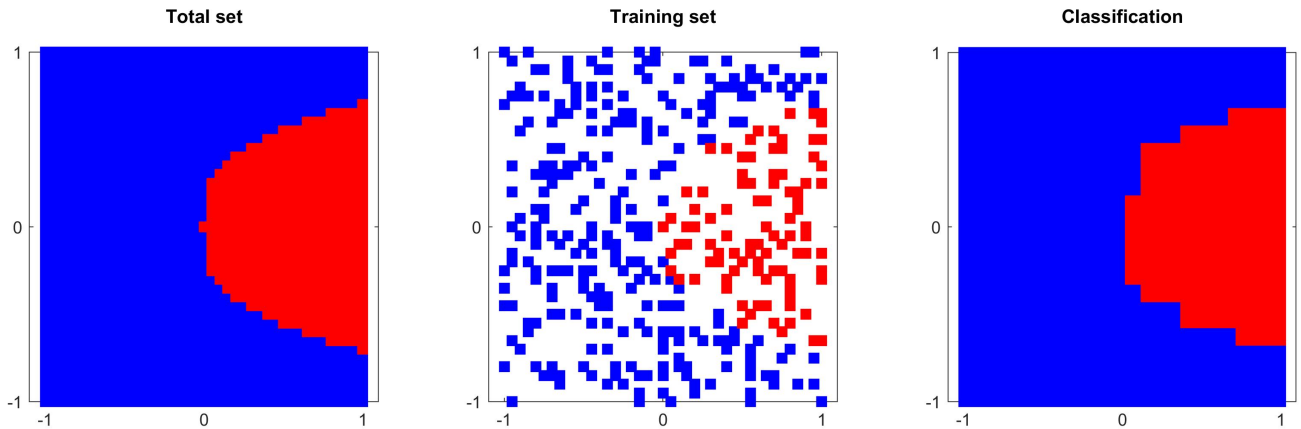
Example 3: training set 2

$$\underline{v} = (x_1, x_2)$$

$$c = f(\underline{v}) = \text{sign}\left(-2\sqrt[3]{x_1^2} + 4x_2^2\right)$$

$$x_1 = -1 : 0.05 : 1 \quad x_2 = -1 : 0.05 : 1;$$

$$N_v = 1681 \quad N_{TS} = 400$$



number of errors: 26/1681

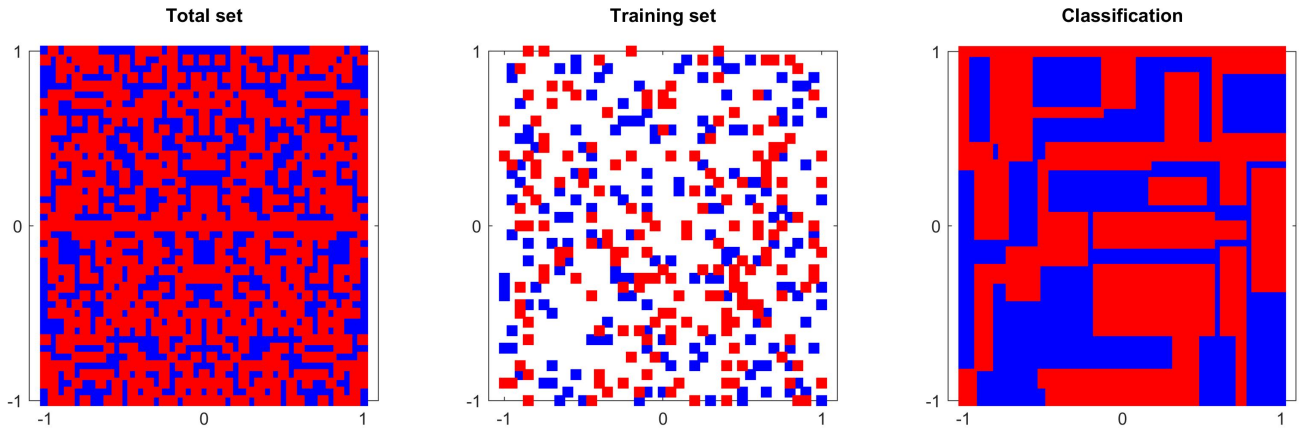
Example 4: training set 1

$$\underline{v} = (x_1, x_2)$$

$$c = f(\underline{v}) = \text{sign}\left(3x_1^3 y_1^3 exp\left(-x_1^2 - x_2^2\right)\right)$$

$$x_1 = -1 : 0.05 : 1 \quad x_2 = -1 : 0.05 : 1;$$

$$N_v = 1681 \quad N_{TS} = 400$$



number of errors: 611/1681

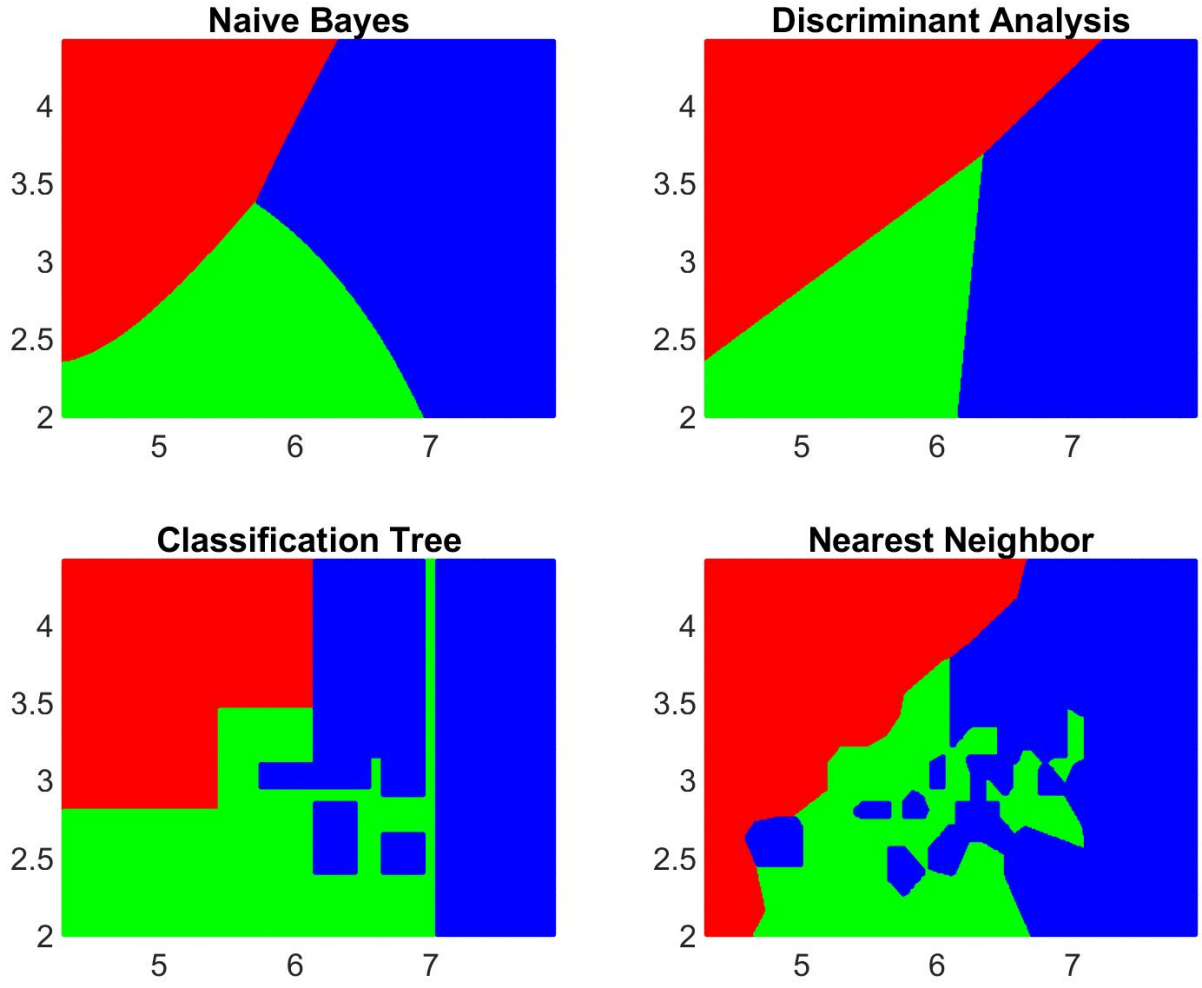
Problem

For numerical features, this Classifier divides the space $A_v = \mathbb{R}^M$ into **hyper-rectangles** of dimension M defined as

$x_{1,inf} < x_1 < x_{1,sup}, x_{2,inf} < x_2 < x_{2,sup} \ldots x_{M,inf} < x_M < x_{M,sup}$.

For each of these hyperrectangles, there is one and only one class. As a consequence, the set of vectors which is mapped by the classification function g into a given class is always the union of hyper-rectangles. If the true set mapped by the original function f into the same class has a different shape we can only approximate it.

Comparison


Naive Bayes
Discriminant Analysis
Classification Tree
Nearest Neighbor
4
3.5
3
2.5
2
5
6
7

Decision Tree Ensembles

To improve the performance of tree classifiers: Tree Ensembles

Main idea: given the training data-set, instead of building a single classification tree we build many and we combine their decisions.

Random forest approach

Bagging: starting from the training data-set T_S we build N bootstrapped data-sets made by randomly extracted vectors of T_S.

Random feature selection: For each bootstrapped data-set, we build a classification tree. When doing this, each time we process a subset to build a node, we only use $M' < M$ randomly extracted features.

Decision: given a new vector, we process it with all the N tree classifiers and we assign the most popular class among the N results.

(To decide the number M' of features, we usually start from a value $M' \simeq \sqrt{M}$. Then we test the accuracy of the forest by using the out-of-bag vectors (the vectors which are outside the different bootstrapped data-sets). We repeat the procedure by increasing and decreasing the values of M' looking for the value with best accuracy.)