TELECOM ParisTech **SD210**
2017/2018 **M. Djerrab, A. Garcia, P. Gori, A. Lambert, G. Varni**

TP n° 5 : Classification Trees

Mathematical framework and notations

Here, we place ourselves in the framework of multi-class classification with the following notations :

— $\mathcal{Y}$ is the set of *labels*. Here we work with L classes and choose $\mathcal{Y}$ to be $\mathcal{Y} = \{1, \dots, L\}$ where each element of $\mathcal{Y}$ corresponds to a different label. (note that $L = 2$ in the case of binary classification).

— $\mathbf{x} = (x_1, \dots, x_p)^\top \in \mathcal{X} \subset \mathbb{R}^p$ is an observation, an example, a sample. The j^{th} coordinate of $\mathbf{x}$ is the value taken by the j^{th} variable and corresponds to a feature.

— $\mathcal{D}_n = \{(\mathbf{x}_i, y_i), i = 1, \dots n\}$ is the learning set containing the n examples and their corresponding labels.

— There exists a probabilistic model that generates the observed samples through the random variables X and Y : $\forall i \in \{1, \dots, n\}, (\mathbf{x}_i, y_i) \overset{i.i.d}{\sim} (X, Y)$.

— We want to learn a classifier from the learning set $\mathcal{D}_n$ which is a function $\hat{f} : \mathcal{X} \mapsto \mathcal{Y}$ that is able to predict a label $\hat{f}(\mathbf{x}_{\text{new}})$ from a new point $\mathbf{x}_{\text{new}}$

Discovering the module `tree` of scikit-learn

Visit the following pages in order to get a fresh start in python or to revisit some concepts

★★★ http://scikit-learn.org/stable/modules/tree.html

★★★ http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

★★ IPython Notebook :
http://nbviewer.ipython.org/github/ipython/ipython/tree/1.x/examples/notebooks/

★★ Fore more details on trees (especially for the `R` language) :
http://www.stat.cmu.edu/~cshalizi/350/lectures/22/lecture-22.pdf

Generating synthetic data

We can use the set of functions from the file `tp_arbres_source.py` . In order to visualize the datasets, you can also use or modify the functions `plot_2d` or `plot_2d_simple` from the same file.

Decision tree - CART algorithm

You can read [2, Chapter 9.2] for more details about trees. The most detailed ressource being the seminal book [1].

We recall the basic principles of decision trees. Note that for the sake of simplicity, we only consider binary trees : a node can only have 2 children, except in the case of leaf nodes that have no children.

We associate a tree representation to any partition of the data. At the beginning, the tree is restricted to a single node which is its root and represents the complete $\mathcal{X}$ space. Recursively, at each step we choose :

— a variable $j \in \{1, \dots, p\}$ (among the p available),

— a threshold $\tau \in \mathbb{R}$

and we build a partition of the space of input variables $\mathcal{X}$ in two subsets that are represented by two nodes in the tree $G(j, \tau) = \{x = (x_1, \dots, x_p)^\top \in \mathbb{R}^p : x_j < \tau\}$ et $D(j, \tau) = \{x = (x_1, \dots, x_p)^\top \in \mathbb{R}^p : x_j \geq \tau\}$. At each step, we incrementally increase the number of components in the partition and by the same process the number of leaves in the tree. This process is repeated until one reaches a stopping criterion that can be :

— when the depth of a tree reaches a given threshold,

— when the population of a node (*i.e.,* the number of observations placed in the corresponding partition) passes below a given threshold,

— when the number of leaves of a tree reaches a given threshold.

— etc.

A visual example of such a construction is given in the Figure 1.

Now we need to define a splitting rule. This choice is crucial and different methods are available. We use a function that will measure the "impurity", that we denote H associated to a partition. Then we look for the cut (corresponding to a threshold) that will produce the partition as pure as possible under the H criterion.

Mathematically we want to solve :

$$\underset{j\in[\![1,p]\!],\tau\in\mathbb{R}}{\arg\min}\ \widehat{q}_{j,\tau}H(G(j,\tau))+(1-\widehat{q}_{j,\tau})H(D(j,\tau)), \tag{1}$$

where we denoted

$$\widehat{q}_{j,\tau}=\frac{|\{i\in[\![1,n]\!]:x_i\in G(j,\tau)\}|}{|\{i'\in[\![1,n]\!]:x_{i'}\in G(j,\tau)\cup D(j,\tau)\}|}, \tag{2}$$

the proportion of observations that are in $G(j,\tau)$. Note that $|\cdot|$ represents the cardinal of a set.

For each set $R\subset\mathbb{R}^p$ and every label k we note $\widehat{p}_k(R)$ the proportion of observations that are labelled k (counted from 1 to K), *i.e.,*

$$\widehat{p}_k(R)=\frac{|\{i\in[\![1,n]\!]:x_i\in R \text{ et } y_i=k\}|}{|\{i\in[\![1,n]\!]:x_i\in R\}|} \tag{3}$$

With CART we consider the following impurity measures H :

- the Gini index : $\sum_{k=1}^{K}\widehat{p}_k(R)(1-\widehat{p}_k(R))$
- the entropy : $-\sum_{k=1}^{K}\widehat{p}_k(R)\log(\widehat{p}_k(R))$

1) In the regression framework (*i.e.,* when we try to predict a numerical value Y instead of a class), propose an other homogeneity measure. Justify your choice.

With `scikit-learn` we can build decision trees thanks to the package `tree`. We obtain a classifier with `tree.DecisionTreeClassifier`.

```
from sklearn import tree
```

2) Simulate with `rand_checkers` a sample of size $n = 456$ (be careful to keep balanced classes). Display 2 charts that indicate the percentage of errors as a function of the maximal depth of the tree (one for Gini and one for the entropy). Keep the default values for the other parameters.

3) Display the classifier obtained using the depth that minimizes the error percentage with the entropy (you can use the functions `plot_2d` and `frontiere` from the source file).

4) Export a figure of the tree obtained at the previous question in the `pdf` format. We can for example call the `export_graphviz` function from the module `tree`.

5) Create $n = 160$ new samples with `rand_checkers`. For the decision trees obtained in question 2, compute the proportion of errors on this second sample. Comment.

6) Do the question 2 again for the dataset ZIPCODE. It is available in the module `sklearn.datasets`. One can import it with the function `load_digits` (for more details see http://scikit-learn.org/stable/auto_examples/classification/plot_digits_classification.html) :

```
digits = datasets.load_digits()
```

Parameter choice methods - Model selection

In practice we rarely have access to a test set (we prefer to include a many samples as possible in the training set) and it is the scientist's role to keep some of its samples to test the model. In order to select a model or a parameter while keeping as many examples as possible for the learning step, one usually performs a cross-validation selection. For each parameter used, an estimation of the empirical error of the classifier is obtained with the following procedure : First we fix and integer N, often $N = 5$ or $N = 10$ which represents the number of folds and a set of candidate parameters for the model :

- The learning set is splitted in N folds of size n/N
- For each subset, we measure the error obtained by the classifier (for a fixed set of parameters) learnt on the $N - 1$ other folds.
- the error estimated is the mean of the error of the different classifiers learnt

We can repeat this procedure for the whole parameter grid. This allows one to obtain an estimation of the error for each set of parameters. Finally we choose the set of parameters that minimizes this error.

For more details concerning the different variants of this principle implemented in `scikit-learn`, visit the page http://scikit-learn.org/stable/modules/cross_validation.html.

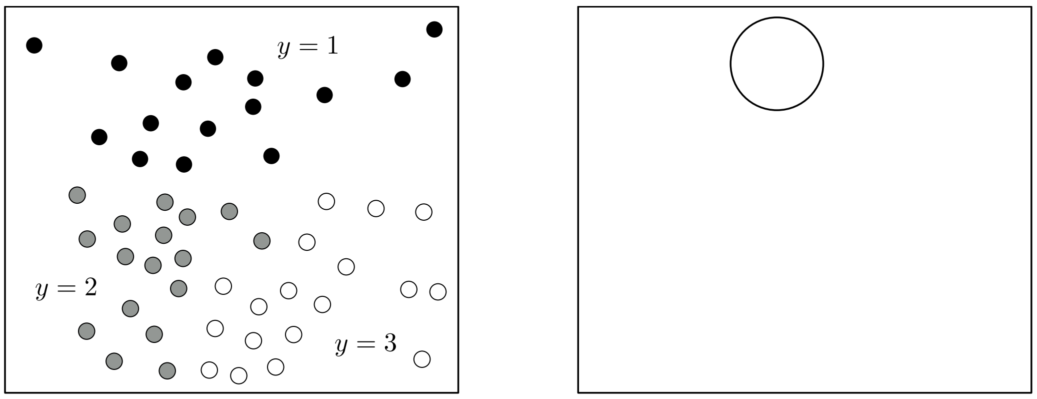
7) Use the function `sklearn.cross_validation.cross_val_score` and test it on the dataset ZIPCODE by varying the depth of the decision tree. You can reuse this function to choose the optimal depth of the tree.
8) Inspire yourself from http://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html to plot the learning curve (en : *learning curve*) for decision trees on the same dataset[1].

Références

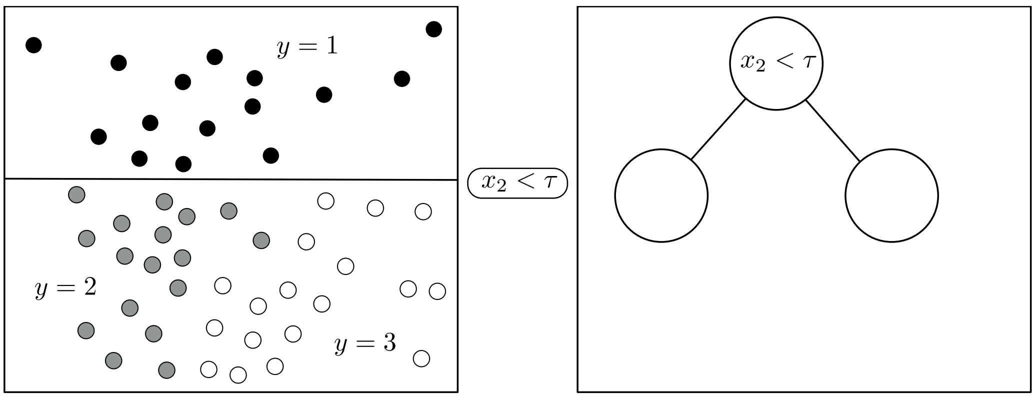
[1] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees.* Wadsworth Statistics/Probability Series. Wadsworth Advanced Books and Software, Belmont, CA, 1984. 1

[2] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning.* Springer Series in Statistics. Springer, New York, second edition, 2009. http://www-stat.stanford.edu/~tibs/ElemStatLearn/. 1

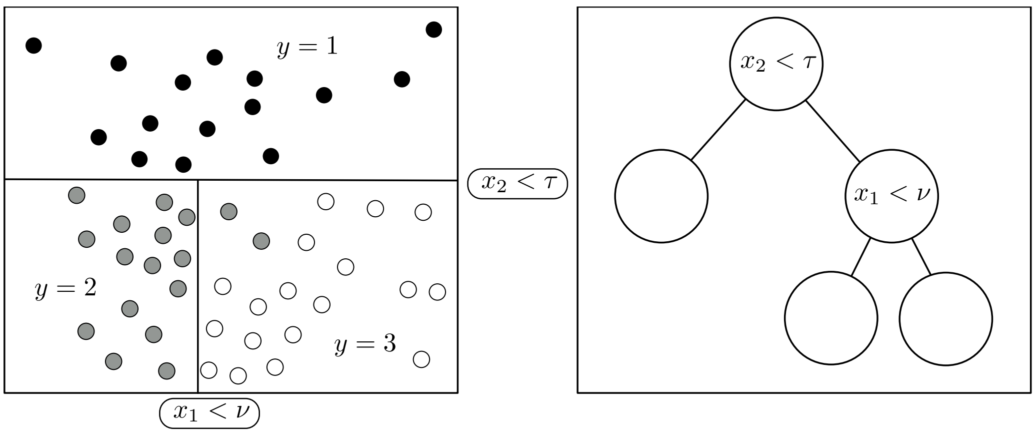
1. pour une version de `sklearn` $>$ version 0.18, voir http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.learning_curve.html



(a) zero split



(b) one split



(c) 2 splits

FIGURE 1 – Example of decision tree creation