LAB N° 4 : Classification trees

Framework and notation

We consider the setup of multiclass classification, with the usual notation (see the KNN Lab). We assume the data falls into K classes. The training set has size $n : \mathcal{D}_n = \{(\mathbf{x}_i, y_i), i = 1, \dots n\}$ comprising n observations (the $\mathbf{x}_i$) and their label (the y_i). We recall that $\mathbf{x}_i = (x_1, \dots, x_p)^\top \in \mathcal{X} \subset \mathbb{R}^p$ is one observation, and in the bidimensional case $p = 2$.

Using `sklearn.tree`

For a reminder, you can read the following webpages ;

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

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

Synthetic data simulation

We use the same functions as in the KNN Lab. To plot the datasets, you can use the functions `plot_2d` or `plot_2d_simple` of `tp_arbres_source.py`

Decision trees - CART Algorithm

For more details about trees, one can read [2, Chapitre 9.2]. The most detailed source is the seminal book [1].

We recall here how a decision tree works (see also the figures below). For simplicity reasons, we only consider binary trees : a node only has two children, except if it's a leaf, in which case it has none.

To a partition of the data is associated a tree representation. At the first step, the tree is reduced to a single node, the root, which represents the whole space $\mathcal{X}$

Recursively, at each step one picks :

— a variable $j \in \{1, \dots, p\}$ (out of the p possible ones),

— a threshold $\tau \in \mathbb{R}$

and one splits the space of explicative variables $\mathcal{X}$ in two subspaces which are represented by two nodes in the tree : $G(j, \tau) = \{x = (x_1, \dots, x_p)^\top \in \mathbb{R}^p : x_j < \tau\}$ and $D(j, \tau) = \{x = (x_1, \dots, x_p)^\top \in \mathbb{R}^p : x_j \geqslant \tau\}$. At each step, the number of leaves (and the number of components in the partition of the space) in the tree is thus incremented. The process is repeated until a termination criterion is reached, for example :

— the depth of the tree reaches a prescribed level

— the number of points falling into into a node goes below a threshold

— the number of leaves reaches a threshold

A visual representation of such a condition is given in Figure 1.

We must now define a rule to decide where the new splitting will occur. This choice is crucial, and there is no unique answer. In order to split, we use a function measuring the "impurity", that we denote H, associated to a partition. We will look for the best split (variable/threshold) as the one which produces the purest partition possible according to H.

Formally, we 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 write

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

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

For any set $R \subset \mathbb{R}^p$ and any label k we write $\widehat{p}_k(R)$ for the proportion of observations which are labelled k (k varying from 1 to K), *i.e.,*

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

In CART, we will 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 a regression framework (*i.e.,*when the target variable is a numerical variable Y and not a class), propose another measure of homogeneity. Justify.

One can construct decision trees using `sklearn.tree`, and in particular `tree.DecisionTreeClassifier`.

2) With `rand_checkers` simulate datasets of size $n = 456$ (be careful not to have imbalanced classes). Create two curves showing the percentage of mistakes made, as a function of the maximal depth of the tree (one curve for Gini, one curve for entropy). For other parameters, use the default ones.
3) Display the obtained classification when the max depth is the one which minimizes the percentage of error obtained with entropy (use `plot_2d` and `frontiere` of the source file).
4) Create $n = 160 = 40 + 40 + 40 + 40$ new data points with `rand_checkers`. For the decision trees previously created, compute the proportion of mistakes made on this test dataset. Comment.

Parameter tuning - Model selection

In practice, one seldom has access to a test set (it is preferable to include as much data as possible in the training set). To select a best model or best parameter while using as many training samples as possible, one usually resorts to Cross Validation (CV). For every parameter used, an estimation of the empirical error of the classifier is obtained with the following procedure. First an integer N is fixed, usually 5 or 10, representing the numbers of *folds*. Then, we pick a grid of candidate values for the parameter we want to tune.

— the training set is partitioned into N folds of size n/N
— for each of the N subsets of the partition, one measures the error obtained by the classifier *when trained on the remaining* $N-1$ *folds.*
— the estimated error for this parameter is the average of the N errors obtained.

This procedure can be repeated for each candidate value of the parameter. We thus obtain an error easure for each parameter on the grid, and we can pick the best parameter as the one minimizing this quantity.

For more details (e.g. on variants), read http://scikit-learn.org/stable/modules/cross_validation.html.

5) Use `sklearn.cross_validation.cross_val_score` and test it on the dataset ZIPCODE (`sklearn.datasets.load_digits`), when the depth of the tree is varied. This function can then be used to choose the optimal depth of the tree.
6) Drawing inspiration from http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.learning_curve.html display the learning curve for decision trees on the same dataset.

- Logistic Regression -

From `sklearn.linear_model` import the class `LogisticRegression` that we use in the sequel.

One can draw inspiration from the following example http://scikit-learn.org/stable/auto_examples/linear_model/plot_iris_logistic.html

7) Go back to the dataset generated by `rand_checkers`. Train a logistic regression classifier on it, and report the proportion of errors *on the train dataset*. Compare to the performance of a decision tree. Comment.

8) What is the variable `coef_` of the model ? `intercept_` ?

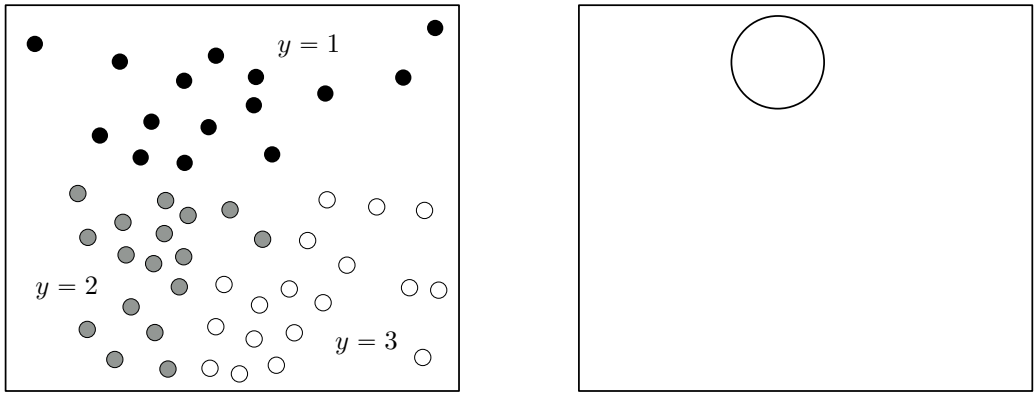
9) What does `score` return ?

10) Propose three ways to perform classification with logistic regression to the dataset ZIPCODE. Detail. Test two of them, using the procedures used above.

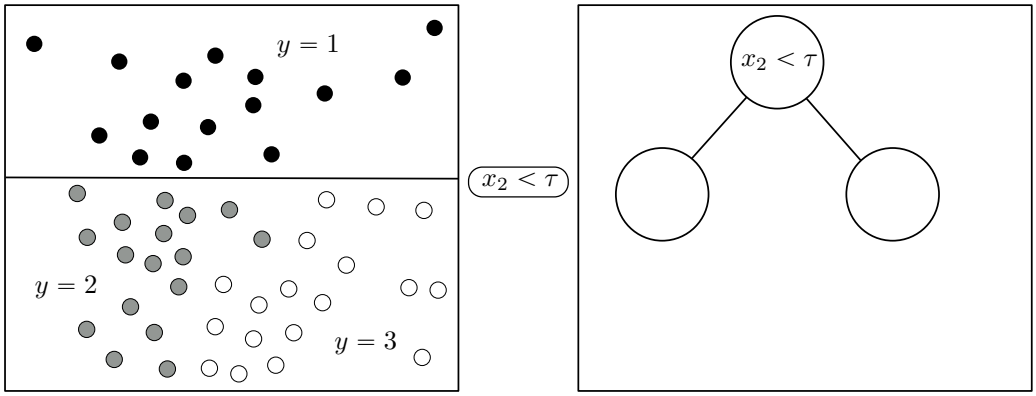
Références

[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

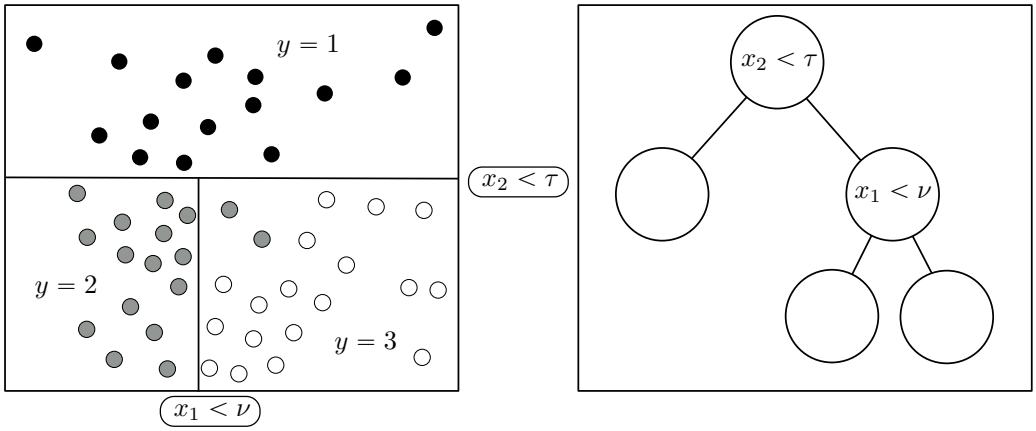
[2] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*. Springer Series in Statistics. Springer, New York, second edition, 2009. http://www-stat.stanford.edu/~tibs/ElemStatLearn/. 1



(a) zéro découpe



(b) une découpe



(c) deux découpes

FIGURE 1 – Example of decision tree creation process