Multi-class classification on gene expression data*

Simone Daniotti
Physics Department, University of Milan.†

Riccardo Castelli‡
Informatics Department, University of Milan
(Dated: September 26, 2019)

An article usually includes an abstract, a concise summary of the work covered at length in the main body of the article.

Usage: Secondary publications and information retrieval purposes.

PACS numbers: May be entered using the `\pacs{#1}` command.

Structure: You may use the `description` environment to structure your abstract; use the optional argument of the `\item` command to give the category of each item.

CONTENTS

I. DATASET DESCRIPTION

The dataset for this work is taken from UCI Machine Learning Repo (available at `https://archive.ics.uci.edu/ml/datasets/gene+expression+cancer+RNA-Seq`) [**?**]; this is part of the RNA-Seq (HiSeq, a tool for measuring gene expression levels) PANCAN data set. It is a collection of gene expression levels of patients having different types of tumor: BRCA(breast), KIRC(kidney), COAD(colon), LUAD(lung) and PRAD(prostate). These data represent the quantity of gene information used in the synthesis of a functional gene product. For further information, we refer to [**?**].

II. DATASET MANIPULATION

A. Preliminary manipulation

Both dataset and labels can be downloaded in a csv ('comma separated value') format, then can be easily imported in a Pandas Dataframe (`https://pandas.pydata.org/`). The first column represents patient's ID: it has been removed because it is useless for our purposes.

B. Label handling

The labels are strings representing the five types of cancer. Learning models can be created using raw features(in the case of trees and forests), using Label Encoding or One-hot Encoding. Label Encoding creates a map between the string and an ordered sequence of natural numbers, from 0 to 4 in our case. One-hot encoding creates a single binary label for each class, changing the task of the learning model to a multi-label problem.

C. Train and Test Set

For training the net and then evaluating it, we split the dataset in training and test set using the Sklearn library *train_test_split* (`https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`). We set the seed of the random split and the proportions between the two sets: test set is 0.15 of the entire database.

D. Class Weighting

III. ARCHITECTURES

There are lots of books reviewing these architectures and concepts. Here we refer to [**?**] , [**?**] [**?**].

A. Support Vector Machines (SVM)

B. Decision Tree Classifier and Random Forests

Decision Trees are versatile Machine Learning algorithms that can perform both classification and regression tasks, and even multioutput tasks. They are particularly useful in treating with complex data, such as a dataset that can hardly be represented by a vector in a multi-dimensional space: this is not our case, but we think it is useful to approach our problem with more simple models and evaluating them before *deeping* in

* A footnote to the article title
† daniottisimone@gmail.com
‡ riccardo.castelli3@studenti.unimi.it

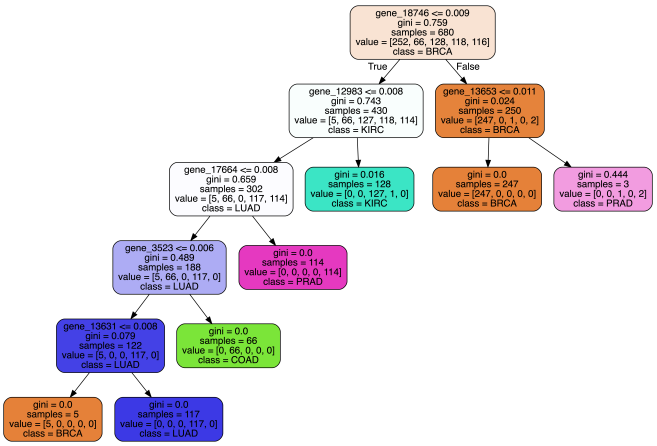

FIG. 1. jnjcnkd

more difficult models. Tree Classifiers have the structure of an *ordered and rooted tree*. It is *ordered* because the children of any internal node are numbered consecutively, and *rooted* because splitting starts from only one node. From that node, the model is built following the attribute selection measure. Attribute selection measure is a heuristic for selecting the splitting criterion that partition data into the best possible manner.

Scikit-Learn uses the *Classification And Regression Tree (CART) algorithm* to train Decision Trees.It works as follows: it first splits the training set in two subsets using a single feature k and a threshold t_k . How does it choose k and t_k? It searches for the pair (k, t_k) that produces the purest subsets. The cost functions that the algorithm tries to minimize are different: most used are *Gini Impurity* and *Entropy*, tunable in Scikit-learn by the hyperparameter *criterion*. Entropy hyperparameter measures Shannon's Entropy, a concept taken from information theory. Each leaf of the tree corresponds to a possible classification label, so inserting datas of a patient from the root, the model *spits* its classification. This can be a modality for building a predictor.

If you aggregate the predictions of a group of predictors (regulated by certain rules, such as majority rule..), you will often get better predictions than with the best individual predictor. A group of predictors is called an *ensemble*; thus, this technique is called *Ensemble Learning*, and an Ensemble Learning algorithm is called an *Ensemble method*. Training a group of Decision Tree Classifiers and gathering into one single predictor is called a *Random Forest*. The Random Forest algorithm introduces extra randomness when growing trees; instead of searching for the very best feature when splitting a node (as in the tree case), it searches for the best feature among a random subset of features. This results in a greater tree diversity.

C. Deep Learning

IV. PARAMETER OPTIMIZATION AND VALIDATION

A. Cross-Validation

B. Grid Search CV (and Random Search)

C. Architecture setting

1. *Parameters*

2. *Optimizers*

3. *Losses*

V. MODELS EVALUATION

A. PCA and Permutation Importance

B. Metrics

VI. CONCLUSIONS AND OUTLOOK

[1] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[2] John N Weinstein, Eric A Collisson, Gordon B Mills, Kenna R Mills Shaw, Brad A Ozenberger, Kyle Ellrott, Ilya Shmulevich, Chris Sander, Joshua M Stuart, Cancer Genome Atlas Research Network, et al. The cancer genome atlas pan-cancer analysis project. *Nature genetics*, 45(10):1113, 2013.

[3] Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.* " O'Reilly Media, Inc.", 2017.

[4] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[5] John Hertz, Anders Krogh, Richard G Palmer, and Heinz Horner. Introduction to the theory of neural computation. *Physics Today*, 44:70, 1991.