GENETIC ALGORITHM APPLIED TO FEATURE SELECTION

This project tackles the feature selection problem in machine learning. The goal is to obtain the subset of features from a dataset that gives the best model performance in terms of accuracy and error rate. Irrelevant features, included in the dataset, require more processing and memory powers than necessary, which is a waste of computing resources. In addition, the feature selection reduces over-fitting which improves the model accuracy and reduces training time. Feature selection may be done manually or by the computer. Some manual techniques include univariate selection, feature importance, and the correlation matrix. The literature presents the usage of optimization techniques to automatically find the best (or a quite good) subset of features. Some of theses methods include: Exhaustive search, Simulated Annealing, Transformation Graph, and Genetic Algorithms.

A genetic algorithm (GA) was developed as a solution to solve the feature selection problem applied to classification tasks, using the decision tree model. The GA procedures involve the following steps: (1) generating a population with a number of chromosomes, where each one of them represents a feature selection; (2) ranking the population according to a fitness value (the F1 score in our case), (3) applying reproduction and mutation processes to the selected chromosomes, and finally, (4) iterating through the previous steps until we reach a stopping rule. The stopping rule may be the number of generations produced or that a target score was reached. In this project, we developed two classes for the genetic algorithm. The Element class is responsible for managing the id, the generation, the genome, and the fitness score of each element of the population for every generation. The GA class includes functions related to the generation of random chromosomes, the mutation, the reproduction, the selection and is also responsible for computing the fitness score. Moreover, the genetic algorithm is configured in nine variations of its parameters: population size, iteration limit, stopping criterion, crossover type, crossover rate, mutation type, mutation rate, and replication rate of the best individuals. The maximum, average and minimum fitness values are considered to compare the various configurations.

In this work, we combine GA with the decision tree technique (it's a supervised learning model). We tackle classification tasks only. No modification was applied to the hyper-parameters of the decision tree model. Also, we only considered tabular datasets. We use the term "tabular data" to designate datasets composed of features that could take numerical or categorical values. They are easy to manipulate and freely available on the Kaggle website. Moreover, there are well known pre-processing libraries to deal with this type of datasets. In total, 9 different datasets were selected. The number of features, from each dataset, varies from 9 to over 30. We initially included a dataset containing over 500 attributes but we ended up discarding it since it required a few days to simulate.

Given the results we obtained from the simulations, we found that GA is very efficient at selecting, combining and trying various subsets of features in order to improve the fitness value (F1 score) and build the best classification model. For the vast majority of the datasets in our project, simply feeding all the features to the decision tree model (the basis model) produces limited results. The genetic algorithm was able to select the most appropriate features with higher fitness values than the basis model. Using the genetic algorithm in order to find the best subset of features, from all the available features, can be an interesting strategy. Our findings are limited to the decision tree technique applied to classification tasks. However, the algorithm can be easily adapted to other fitness functions, and to other machine learning techniques and modelling problems.