Jurgen Palsma
jurgen.palsma@gmail.com

December 2017

Rule Discovery with Evolutionary Algorithms

With the recent trend in Data Analysis, scientists are urged to develop tools and techniques used to extract or abstract information from large data sets, a process which we commonly call "Data Mining". One set of techniques that can be used in this process is the field of evolutionary algorithms and genetic algorithms (GAs), inspired by the natural process of genetic selection and reproduction.

Indeed, GAs can be used in data mining as either 1) a sort of heuristic approximation to guide and parametrise other data mining algorithms, 2) as a tool to sharpen the results of other algorithms by for example "pruning classifiers", and 3) as a classification algorithm for data. In this essay, we will explore one application of GA in classification, called rule discovery.

Rule discovery[1] allows notably the discovery of correlations or links between attributes, called prediction rules. Prediction rules can often been seen in the form of "IF condition THEN class", where "condition" is one or more condition for attributes in a data sample to fit a "class". In order to have a functioning GA for rule discovery, one must have a way to represent a candidate solution through individuals, a way to operate these individuals through genetic operations, and a way to evaluate these individuals, through a fitness function.

To represent a candidate solution through an individual, there are two main approaches: an individual can be represented either as an individual rule (Michigan approach[2]) or as a set of rules (Pittsburgh approach[2]). In either approach, a rule can be represented in two parts[1]: an "*antecedent*", which is the set of condition(s) that apply to the rule, and the "*consequent*", which is the class in which the data set applying to the condition(s) will be assigned to.
Typically, an antecedent can be represented with binary encoding. For example, if a discrete attribute is distributed through *k* possible values, we could encode those values through a sequence of *k* bits. This way, the sequence of bits represents the value of the attribute, and conditions can be made this way: if we take a discrete attribute "*color*" which values can be "*red*", "*green*", and "*blue*", mapping the values in a binary sequence (alphabetically) could give the sequence "010" a condition like "*color* EQUALS *green*".

The main component of a GA is the way individuals perform "genetic operations" to produce new (ideally more performant) individuals. Although classic genetic operators such as single-point crossover can be applied in the case of rule discovery, it is important to note that these methods have to take into account that in some cases they can produce invalid individuals, who have contradictory antecedents. To prevent this, an effective method to avoid having to verify the validity of each new individual, we can format our new individual's parent's binary sequence in order to have only overlapping attributes, and use binary padding if an attribute is not present in one parent.

Once candidates solutions can be represented through either the Michigan or Pittsburgh approach, and genetic operators have been applied, it is important to be able to quantify and evaluate them in order to assess the quality of our algorithm and to be able to parametrise and guide our genetic operations. One way to evaluate our candidates is by stating that “the discovered rules should have a high predictive accuracy, be comprehensible, and be interesting”[1] and try to quantify that[1]. A fitness function that would address these problems would be of the form of:

$$w_{acc}.accuracy + w_{comp}.comprehensibility + w_{int}.interest$$

Where each problem is represented as a parameter weighted by its weight (defined beforehand), tuned considering the importance of each quality of the rule. As comprehensibility and interest of a rule depend highly on the application of the GA, we will only develop the $acurracy$ quality of the rule, which can be calculated with the following: $acurracy = CF.Comp$, with CF being the confidence factor, or the number of examples satisfying the antecedent of the rule divided by the number of examples satisfying both the antecedent and consequent of a rule, and Comp being the completeness of a rule, the number of examples satisfying both the antecedent and consequent of a rule (true positive, TP) divided by the sum of true positives and false negatives (number of examples satisfying only the consequent).

As a result, it is possible to combine these techniques of individual representation, genetic operation, and fitness evaluation to discover rules in a data set using GAs. One of the strength of GAs in this context is that it can be extremely performant on large sets of redundant data, by encoding data and operating it on a bitwise level. Additionally, GAs can be used to provide a set of interesting rules rather than a local optimum: because of the stochastic-ness nature of the algorithm, and the ease of combining different rules, it operates on a potentially larger search space (opposed to a greedy algorithm for example[3]). GAs can also be fairly simple to implement, if one contents himself with only the $acurracy$ quality of the fitness function, to get a first idea, a simple set of rules from his data set. In addition, if we reduce the complexity of the fitness function, we tackle a paint-point of genetic algorithms which is the computational power required to compute fitness functions, and can thus provide a good candidate algorithm for resolving rule discovery problems.

If computational power is not an issue, it is possible to provide even higher quality results using distributed GAs[4]: by running multiple GAs and “migrating” individuals sequentially during the execution of the GAs, it is possible to combine high quality individuals from very distant points on a search space, and thus each GA can focus on exploiting its individuals rather than on exploring other solutions (in our fitness function, this would be reducing accuracy and augmenting interest).

In conclusion, despite the lack of literature on the subject, GAs can be efficient tools for rule discovery. As they can be tweaked easily (notably via the fitness function) and as they can simplify complex data (by translating data to discrete, binary sequences), they allow for efficient scaling. In addition, the ease of implementation of a simple GA with a simple fitness function allows a data scientist to start examining data quickly and get some basic acceptable results easily.

References and links

1. Most of the concepts and solutions talked about in this paper come from: *A Survey of Evolutionary Algorithms for Data Mining and Knowledge Discovery,* A. Freitas, published in *Advances in Evolutionary Computing,* Springer (2013)
Available online via: http://neuro.bstu.by/our/Data-mining/fereitas-ga.pdf

2. *Comparison of the Michigan and Pittsburgh Approaches to the design of Fuzzy Classification Problems,* H. Ishibuchi, T. Nakashima, T. Murata, *Electronics and Communications in Japan, Vol 80* (1997)

3. *A Greedy Classification Algorithm Based on Association Rule*, F.A. Thabtah, published in *Applied Soft Computing, Volume 7,* Elsevier (2007)
Available online via: http://www.sciencedirect.com/science/article/pii/S1568494606000718

4. *Discovery of Classification Rules Using Distributed Genetic Algorithm,* P. Sharma, published in *Procedia Computer Science,* Elsevier (2015)
Available online via:
https://ac.els-cdn.com/S187705091500085X/1-s2.0-S187705091500085X-main.pdf?_tid=e2196c0a-ddbc-11e7-893f-00000aacb35d&acdnat=1512919077_a5b550d6d99eb047cd7fae4292db7c00