A GENETIC ALGORITHM BASED ASSOCIATION RULE LEARNING SYSTEM

PETER EBEAR, DAVID ENGEL, EVAN LOUGHLIN, JAMES MACISAAC, AND SHANE SIMS

1. Introduction

The task of this paper is to satisfy the Winter 2018 CPSC 599.44 Midterm Report requirements, and in doing so, present the proposal for the learning system that we intend to implement. To this end, we will present a sketch of our intended learning system in the following pages. The structure of this paper is as follows. We first present our learning method by means of introducing key parts of our proposed system. We then describe how we intend to include relevant additional knowledge into the learning system. Finally we describe how we intend to deal with the requirement that our system be able to read a list of already known association rules, and how these will be used to prevent our system from learning these rules again.

2. Learning Method

For the task of learning association rules from Statistics Canada's *1999 to 2015 National Collision Database*, we will implement a learning system that is based on a Genetic Algorithm. Our reasoning for this choice, and why we have not chosen to build a system that uses the Apriori algorithm is as follows:

- Kabir et al [5] did a comparative study between the Apriori algorithm, and their proposed genetic algorithm, in the discovery of maximal frequent patterns with varying database sizes and minimum coverage values. The study showed that while Apriori did eventually outperform their genetic algorithm at larger minimum coverage values, the Apriori algorithm devolved to exponential growth at low coverage values, while their genetic algorithm maintained a relatively linear performance across the board. As one objective of this project is to uncover potentially interesting association rules, an algorithm that provides more consistent performance across a variety of parameter ranges is in our opinion, more ideal.
- By using a genetic algorithm as the basis of our learning system, it is conceptually clear how we will be able to meet the requirement that our system be able to use a list of already known rules to prevent these rules from being learned again. Our system will be able to use this list to prevent repeated learning of these rules in a more intelligent way than simply filtering out already learned rules during post-processing (See section 4).
- It is likely that some interesting association rules will include feature values that are within a range of accepted values (ex. $18 \leq P\_AGE \leq 25$), and

not just discrete values. A genetic algorithm allows us the flexibility to structure individuals to capture such rules (see Section 2.1).
- The ability to define the fitness function provides an additional way to add knowledge to our learning system. Instead of using just coverage and accuracy as a measure of fitness, we will also add additional terms to our fitness function to encourage the production of novel association rules (see Subsection 2.4). We can also use the fitness function as a means of directing the search to areas that we wish to explore; for example by specifying the consequent, or a mandatory part of the antecedent (see Section 3).

Having provided our justification for choosing to build a genetic algorithm based learning system, a high-level description of key parts of our system will provide clarity on our intentions.

2.1. **Individual Representation.**

An individual in our system will represent an association rule. We will represent individuals of a given state in our learning system according to the following high level schematic:

Status	Status	...	Status	Status
C_YEAR_upper	C_MNTH_upper	...	P_USER_upper	C_CASE_upper
C_YEAR_lower	C_MNTH_lower	...	P_USER_lower	C_CASE_lower

Naturally, this representation requires further comment:

- 'Status' can hold a value of either 0, 1, or 2. This represents if a feature is not included in the rule, is part of the antecedent, or is part of the consequent, respectively.
- *_upper and *_lower hold the upper and lower bounds on the value of *, respectively. If a feature has a discrete value, this is represented with *_upper = *_lower.
- As none of the features have a negative number as an allowable value [3], we will represent null with -1.
- Allowable but non-integer values such as NN,UU,M,F etc., will be mapped to unique integers (outside of the domain of the respective feature), so that type remains consistent in the implementation.

This representation of an individual corresponds to a single association rule. It includes an index for every possible feature of an arbitrary example in our dataset [3]; its presence and role in the given rule are indicated by the 'Status' number. This representation allows for simple implementation of genetic operations (described in section 5), as will be demonstrated with an example in the final section of this paper. With this representation scheme in mind, we would represent the association rule $(C\_WTHR = 1) \Rightarrow (C\_SEV = 2)$ as:

0	0	0	0	**2**	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-1	-1	-1	-1	**2**	-1	-1	-1	**1**	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	**2**	-1	-1	-1	**1**	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

2.2. **Initial Population.**

The initial population for our learning system will consist of m individuals. The non-null features of a given individual will be chosen randomly so that even weight is given to rules of all sizes. For our purposes, the minimum rule size will be that with one feature in the consequent and one in the antecedent. The value of each included feature in a given rule will also be chosen randomly from among the allowed values. The value of m will be determined empirically once our system is implemented.

2.3. **Genetic Operators.**

Our system will utilize the two usual genetic operations, namely crossover and mutation. How each of these will operate on individuals selected for the given operation is outlined as follows:

- $Crossover(rule_1, rule_2)$: This operation works by obtaining a random number r between 1 and 21[1]. This number r then forms the crossover point of the operation. Then indices 0 to r of $rule_1$ will be prepended to indices $r+1$ to 22 of $rule_2$.
- $Mutate(rule)$: This operation requires four random numbers to be generated. First we choose a random number r_1 between 0 and 22. This corresponds to the index of our mutation point. Let s be the current value (0, 1, or 2) of 'status' at index r_1 of $rule$. We randomly generate another number, r_2 as either 1 or 2. Next we compute $s' \equiv s + r_2 \pmod 3$; forcing the status bit to change while allowing for either of the possible alternative values with equal probability. Finally, if $s' \in \{1, 2\}$, we generate r_3 and r_4 (with $r_3 \leq r_4$) as random values in the domain of the feature corresponding to r_1, and we set the triple at index r_1 in $rule$ to be (s', r_3, r_4). Otherwise, we set 'status' to 0.[2]

2.4. **Fitness Function.**

In this section we describe our proposed fitness function, $fitness$. Some terms of $fitness$ provide a way to add additional knowledge to the system (and is thus listed in section 3).

We begin with the base term of our fitness function. Let $|D|$ be the number of items in our dataset. The basic version of our fitness function is composed of two terms, which are themselves functions:

$$fitness_{base}(rule) = (\tfrac{coverage(rule)}{|D|} + accuracy(rule)) \times \tfrac{1}{2}$$

where $rule$ is the individual under consideration. Here *coverage* and *accuracy* are computed in the regular ways. Depending on the results of our initial system runs,

[1]There are 23 features in our basic data items.

[2]We acknowledge that this scheme may add to much randomness to the mutation operation, and are prepared to experiment with mutation of fewer elements of the triple at index r_1.

this base function may be modified to return 0 if a minimum coverage is not met.

Part of the purpose of the learning system is to discover 'interesting' association rules. Though the term 'interesting' is relative, we choose to define it in accordance with the word 'novel'. To this end, we will add a term to our fitness function that favours individuals that are different from common or expected association rules. We will define a rule as expected if it is contained in a list of association rules output by WEKA via the Apriori algorithm.[3] To measure the novelty of an association rule, we will compute the Hamming distance[4] between the given rule and every rule in the WEKA produced list. We will then penalize the rule based on the lowest of these values (that is, the closer a rule is to one already produced by WEKA, the more it is penalized). We define this part of the fitness function as follows:

Let W be the list of association rules produced by a standard run of WEKA's Apriori algorithm.

Let $ham(w, rule)$ be the number of clauses (and their presence in the antecedent or consequent) common to $w \in W$ and $rule$.

Let c be the number of features in the dataset[5]. Then

$$fitness_{ext_1}(rule) = \frac{min(ham(w_0,rule),\ldots,ham(w_{|W|},rule))}{c}$$

We use a dividend of c to normalize this term to the other terms of $fitness$.

Another measure that will be considered as contributing to *interestingness*, is the completeness factor[4] of a rule. The simplicity of our base fitness function, when considering the accuracy of a rule, potentially favors rules that overfit the data. Given a rule with antecedent A and consequence C and coverage(A) = coverage(A and C), while the rule clearly has 100 percent accuracy it doesn't tell us anything about how often C occurs without A. We will introduce a completeness factor, which measures the coverage of C that is actually predicted by A, in essence penalizing rules that incompletely predict C. Rules found with a low completeness factor are either potential artifacts in the data set, or at the very least uninteresting given their low completeness rating. In either case a reduced fitness value is suitable for our purposes.

Let $comp(rule)$ be the coverage of C predicted by A, relative to the coverage of C, with the range being between [0,1]:

$$comp(rule) = \frac{coverage(A \cup C)}{coverage(C)}$$

Then

[3]Another option is to use a list of rules produced by a previous run of our system using only the base fitness function.

[4]We acknowledge that there may be more appropriate distance measures, but will start our experimentation with this one and change to another measure if needed.

[5]While there are 23 features in the basic dataset, this number will change as extend the dataset with new knowledge.

$$fitness_{ext_2}(rule) = accuracy(rule) \times comp(rule)$$

Note that since $accuracy(rule)$ and $comp(rule)$ are both normalized values, the range of $fitness_{ext_2}$ is also [0,1].

Our overall fitness function is then just the average of our base fitness function and our defined extensions:

$$fitness(rule) = \begin{cases} 0 & \text{; no clause in antecedent or consequent} \\ (fitness_{base}(rule) + fitness_{ext_1}(rule) + fitness_{ext_2}(rule)) \times \frac{100}{3} & \text{; otherwise} \end{cases}$$

2.5. **Search Control.**
As our system will essentially be searching for association rules, it is important that we have a clear and well defined search control. The search control is based around a single function, f_{select}. This function will use an element of randomness (so that even the least fit individuals in a state have *some* chance to propagate if their fitness value is above 0), and $fitness$ to determine which individuals in a given state will be input to a genetic operator. Our search control operates by simply evoking f_{select} to produce generation upon generation until our end condition is met. For our purposes, the end condition will be some number of generations, $g \in \mathbb{N}$, which will be given at runtime. This parameter will be established empirically.

We describe the operation of f_{select} in a given state s_j where $1 \leq j \leq g$ ($j \in \mathbb{N}$), as follows:

(1) Order all individuals of s_j in descending order of fitness to get a vector, fit, of size $|s_j|$.
(2) Let $FIT = \Sigma_{i=1}^{|s_j|} fitness(rule_i)$ represent the total fitness of the population of state s_j.
(3) Associate with each individual a part of the interval $[1, FIT]$ as follows: $fit[i]$ gets $fitness(fit[|s_j| - i])$ out of the FIT number of positions in the interval.
(4) If the current operation is a crossover, select two random numbers with $RNG(1, FIT)$. If mutation, select one. The corresponding individuals on the $[1, FIT]$ interval are those that have been selected.
(5) Perform the crossover or mutation operation until a valid (and unique) offspring is found.
(6) Check that the offspring is not in the list of already known rules. If so go back to step 5. Otherwise add this individual to the population.
(7) If goal state reached, stop searching.
(8) If operation results in a population of size greater than ℓ, (the maximum allowed at the beginning of a state), then delete the least fit individuals until the population is again equal to ℓ.

We intend to establish the crossover to mutation ratio and ℓ empirically.

3. Adding Knowledge

We propose the following four ways of adding knowledge to our learning system that is not already contained in our dataset:

(1) Twilight hours are those that occur during dusk and dawn. The onset and length of dusk and dawn, depend on location and month of the year. While we do not have sub-national location information in our dataset, we do have month and local time features. Local dusk and dawn times vary by about an hour in local time across Canada. Then we can use this information [1] to establish an average Canadian local twilight time for each month of each year in our dataset. Using this information, we can add a feature that holds a boolean value that is True whenever the corresponding example occurred during our averaged twilight time, and False otherwise.
(2) Accepting command line arguments specifying clauses that must be present in the antecedent and/or consequent of any learned rule. This would provide us with a means to explore certain areas of the association rule solution space as guided by the knowledge of the system operator at runtime. Examples of this type of knowledge could be the belief that accidents are more likely to occur when multiple young male passengers are present in the vehicle (distraction), or that a young male is less likely to drive riskily with an adult passenger in the vehicle. There are many more assumptions that could be easily investigated using our system.
(3) Adding a new term to our fitness function that captures a measure of how "interesting" a given association rule is. The way we propose to do this was described in section 2.4 above, with the definitions of $fitness_{ext_1}$ and $fitness_{ext_2}$.
(4) Given a reliable source of monthly Canadian deaths[2] along with the figures pertaining to fatalities that can be extracted from the collision dataset could produce very interesting association rules. To this end, we will add an additional feature by first calculating how many collision related deaths occur in a given month (from our dataset) and divide this by the monthly deaths (from [2]). This gives us a ratio of Canadian deaths explained by our collision-related dataset, for each month. Using these values, we will calculate a mean value for monthly deaths for the 1999-2014 period. With this mean value, we will then categorize each item in our dataset as belonging to an *above average* or *below average* category.

    This feature could provide insight into the overall trend of motor vehicle deaths, and identify some possible outliers. These could be as simple as the month of first snowfall, as well as many other possibilities that we may not have considered. By using *above average* in the consequent of a cmdline argument (see 2) above), we could direct our search to look for rules pertaining to this. These rules could confirm our initial suspicions, as well as prompt an investigation into those that are unexplained. Investigation of unexplained above average deaths could lead to the inclusion of a host of additional knowledge sources we may not have originally considered.

4. Detecting Applicable Knowledge

The requirement that our system be able to read a list of already known rules, and avoid producing those same rules throughout the course of learning, is relatively simple when using a genetic algorithm based learner. Our system fulfills this requirement by cross checking this list upon the creation of any new individual (be it

during the creation of the initial population, or upon applying a genetic operator), and discarding any that is found to already exist in this list.

We will accomplish this by taking an input file containing the list of known rules and parsing the list to convert the rules to the format of our internal rule representation structure. Once converted, we will keep them as a persistent list for the duration of program execution. Any rule produced by our genetic operations can then be cross-checked against this list and kept or discarded. This may add a significant overhead to processing a newly bred generation of rules, therefore we will exercise caution when deciding how many generations to allow to pass before re-examining the pool of rules.

5. Example Association Rule Learning

The following example demonstrates the functionality of the most basic version of our system. Namely, it demonstrates the individual representation, genetic operations, and the base version of the fitness function.

Let the schema of a dataset D be described as:

$D(C\_Year, C\_Month, C\_Day, P\_Sex, P\_Age)$

Let $D =$

1999	1	1	1	60
2000	1	2	1	63
2000	1	6	2	16
1999	2	7	2	16
1999	2	5	1	17
1999	4	2	1	72
2000	5	6	2	17
2000	11	6	2	17
2000	10	1	2	63
2000	10	5	2	62
2013	4	5	1	30

We will let $F = 1$ and $M = 2$ for the values of feature P_Sex. An individual in our system would then be represented as:

Status	Status	Status	Status	Status
C_YEAR_upper	C_MNTH_upper	C_Day_upper	P_Sex_upper	P_Age_upper
C_YEAR_lower	C_MNTH_lower	C_Day_lower	P_Sex_lower	P_Age_lower

Suppose for demonstration purposes that we perform one cross-over operation before each mutation.
Let $\ell = 10$ be the maximum size of the population at any given time.
Let $m = 3$ be the size that the population is reduced to once the maximum population size is reached.
Let s_0 be the start state composed of three 'randomly' generated individuals:

$s_0 = \left\{ \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & 1 & 2 & 1 \\ \hline -1 & -1 & 7 & 2 & 23 \\ \hline -1 & -1 & 5 & 2 & 10 \\ \hline \end{array}, \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & 1 & 2 & 1 \\ \hline -1 & -1 & 7 & 1 & 50 \\ \hline -1 & -1 & 2 & 1 & 99 \\ \hline \end{array}, \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & 1 & 2 & 1 \\ \hline -1 & -1 & 5 & 2 & 30 \\ \hline -1 & -1 & 4 & 2 & 31 \\ \hline \end{array} \right\}$

Call these individuals $rule_1, rule_2$, and $rule_3$, respective to the order in which they appear above. These individuals represent the following association rules:

- $rule_1 : (5 \leq C\_Day \leq 7) \wedge (10 \leq P\_Age \leq 23) \rightarrow (P\_Sex = 2)$
- $rule_2 : (2 \leq C\_Day \leq 7) \wedge (50 \leq P\_Age \leq 99) \rightarrow (P\_Sex = 1)$
- $rule_3 : (4 \leq C\_Day \leq 5) \wedge (30 \leq P\_Age \leq 31) \rightarrow (P\_Sex = 2)$

Given this start state, the search control operates as follows:

(1) Calculate the $fitness$ value of each individual and place in increasing order in a vector, fit:
   - $fitness_{base}(rule_1) = (\frac{5}{11} + \frac{4}{5}) \times \frac{1}{2} = 0.63$
   - $fitness_{base}(rule_2) = (\frac{4}{11} + \frac{2}{4}) \times \frac{1}{2} = 0.43$
   - $fitness_{base}(rule_3) = (\frac{1}{11} + \frac{0}{1}) \times \frac{1}{2} = 0.05$

   Then $fit = (rule_1, rule_2, rule_3)$.

(2) Then $FIT = (0.63 + 0.43 + 0.05) \times 100 = 111$

(3) $fit[1]$ gets from 1-63, $fit[2]$ gets from 64 - 106, and $fit[3]$ gets from 107-111.

(4) After randomly obtained the numbers 10 and 109, individuals $fit[1]$ and $fit[3]$ are selected for the crossover operation.

(5) Suppose $Crossover(rule_1, rule_3)) = rule_4 =$
0	0	1	2	1
-1	-1	7	2	30
-1	-1	5	2	31

, when the crossover point $r = 3$ was used. This gives the next state:

$s_1 = \left\{ \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & 1 & 2 & 1 \\ \hline -1 & -1 & 7 & 2 & 23 \\ \hline -1 & -1 & 5 & 2 & 10 \\ \hline \end{array}, \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & 1 & 2 & 1 \\ \hline -1 & -1 & 7 & 1 & 50 \\ \hline -1 & -1 & 2 & 1 & 99 \\ \hline \end{array}, \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & 1 & 2 & 1 \\ \hline -1 & -1 & 5 & 2 & 30 \\ \hline -1 & -1 & 4 & 2 & 31 \\ \hline \end{array}, \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & 1 & 2 & 1 \\ \hline -1 & -1 & 7 & 2 & 30 \\ \hline -1 & -1 & 5 & 2 & 31 \\ \hline \end{array} \right\}$

(6) To update fit, we calculate $fitness_{base}(rule_4) = (\frac{1}{11} + \frac{0}{1}) \times \frac{1}{2} = 0.05$. This gives $fit = (rule_1, rule_2, rule_3, rule_4)$.

(7) Then $FIT = (0.63 + 0.43 + 0.05 + 0.05) \times 100 = 116$.

(8) $fit[1]$ gets from 1-63, $fit[2]$ gets from 64-106, $fit[3]$ gets from 107-111, and $fit[4]$ gets from 112-116.

(9) After randomly obtaining 81, $fit[2]$ is selected for the mutation operation.

(10) Suppose $Mutate(rule_2) = rule_5 =$
2	0	1	2	1
1999	-1	7	1	50
2000	-1	2	1	99

, when $r_1 = 0, r_2 = 2, r_3 = 1999$, and $r_4 = 2000$.This gives the next state:

$s_2 = \{ \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & 1 & 2 & 1 \\ \hline -1 & -1 & 7 & 2 & 23 \\ \hline -1 & -1 & 5 & 2 & 10 \\ \hline \end{array}, \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & 1 & 2 & 1 \\ \hline -1 & -1 & 7 & 1 & 50 \\ \hline -1 & -1 & 2 & 1 & 99 \\ \hline \end{array}, \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & 1 & 2 & 1 \\ \hline -1 & -1 & 5 & 2 & 30 \\ \hline -1 & -1 & 4 & 2 & 31 \\ \hline \end{array},$

$\begin{array}{|c|c|c|c|c|} \hline 0 & 0 & 1 & 2 & 1 \\ \hline -1 & -1 & 7 & 2 & 30 \\ \hline -1 & -1 & 5 & 2 & 31 \\ \hline \end{array}, \begin{array}{|c|c|c|c|c|} \hline 2 & 0 & 1 & 2 & 1 \\ \hline 1999 & -1 & 7 & 1 & 50 \\ \hline 2000 & -1 & 2 & 1 & 99 \\ \hline \end{array} \}$

(11) The process continues from here in much the same way. The fitness value for the newly introduced rule will be calculated, then steps 3-10 are repeated until such time as a stopping condition is reached.

References

[1] Government of Canada. *Sunrise/Sunset Calculator*. URL: https://www.nrc-cnrc.gc.ca/eng/services/sunrise/index.html.

[2] Statistics Canada. *Deaths, by month, Canada, provinces and territories*. URL: http://www5.statcan.gc.ca/cansim/a26?lang=eng&id=1020502.

[3] Transport Canada. *Data Dictionary NCDB*. URL: https://open.canada.ca/data/en/dataset/1eb9eba7-71d1-4b30-9fb1-30cbdab7e63a.

[4] Alex A. Freitas. *A Survey of Evolutionary Algorithms for Data Mining and Knowledge Discovery*. URL: http://neuro.bstu.by/our/Data-mining/fereitas-ga.pdf.

[5] Mir Jahangir Kabir et al. "Comparative Analysis of Genetic Based Approach and Apriori Algorithm for Mining Maximal Frequent Item Sets". In: (2015). DOI: 10.1109/CEC.2015.7256872.