Genetic Algorithm Rule based Classifiers

George Ellicott & 160190

1 INTRODUCTION

The focus of this paper is to demonstrate the effectiveness of evolutionary intelligence to solve a set of classification problems. The classification problems were, dataset one and two which are in binary string representation and dataset three which consisted of real number values.

The given datasets did not need of any preparation, data cleaning nor have any missing data.

Research on basic concepts and different techniques of data mining and classification are provided within the background research.

The preliminary step was to construct a Simple Genetic Algorithm (GA). A simple GA is the use of chromosomes representation as a binary string that imitates evaluation. Dudek (2013) States a GA is an adaptive and parallel search technique based on natural section, reproduction and mutation. To enable the GA to work efficiently, parameters such as; population size, crossover and mutation rate, need to be manipulated so that candidate solution traverse within the search space effectively.

After the initial setup, the GA needed amendment so that it could evolve to classify data. The approach taken was to build up a set of condition action rules that encapsulate a model classifier. As rules are a useful technique to represent bits of knowledge or information Han, Kamber and Pei (2015). This extracted knowledge can then be inferred to predict and categorize the data.

The model was constructed by N number of rules, these rules consisted of chromosomes (genes) of an individual solution produced by the GA. These genes were partitioned to match the dataset to correctly classify the data, therefore giving each individual a higher fitness, when a precondition was matched.

On all datasets, the parameters; selection, generation, population, crossover, mutation and in the case of dataset 3 mutation range, have been experimented on to find the optimal ranges for that dataset.

2 BACKGROUND RESEARCH

Agarwal (2013) states data mining is seen as the intersection of computer science and statistics, used to discover patterns in information and then mould it into an understandable structure.

Labelled and Unlabelled Data

Data from a dataset are called instances, each of these instances can contain several variables, which are generally referend to as attributes. There are two types of data, labelled and unlabelled, which are handled in different ways Bramer (2016).

Labelled data has the intention of predicting value of designated attribute that have not yet been seen, this is known as supervised learning. Unlabelled data does not have any specially designated attribute. Therefore, the aim is to extract the most information from data available, this is known as unsupervised learning.

Classifying: Basic Concepts

Han, Kamber, Pie (2012) refer to Classification as highly popular application for data mining, which is a form of data analysis by extracting models describing data classes. These models can be referred to as classifiers, predict categorical class labels.

Naïve Bayes

Naïve Bayes is an algorithm is a non-explicit representation of the classifier. Therefore, does not uses rule based or decision trees, instead Naïve Bayes uses mathematics. Probability theory is used to classify possible classifications Bramer (2016).

Decision trees

Decision trees are a method of categorising large datasets. A decision tree is a classifier that uses recursive partition of the instance space. The decision tree consists of nodes that form a rooted tree, from a root node to an internal node to an end node. Internal node separates the instance space into two or more branches of the sub spaces. Each leaf or terminal node is assigned to one class representing the most appropriate target value. Dahan *et al.* (2014) states that decision trees are similar to rule induction, as a path from the root to a leaf node can be conjoined test through the tree to form the antecedent and the leaf node precondition as the class value. Apté and Weiss (1997) explain that decision trees are created from training data in a top-down, general-to-specific direction. The start state of a decision tree is the root node that is assigned examples from the training set.

Random forest

Random forest classifiers are several decision trees used on the same dataset, then merges them together to achieve a more accurate and stable prediction. They operate by passing the data to a multitude of decision trees and then use and algorithm to decide on selected nodes, such as, the highest chosen leaf node class. Another approach could be having a preference from one tree over another. Random forest is a good alternative to deep decision trees, as they are susceptible over fitting. Angshuman et al. (2018) states how an improve random forest (IRF) can be implemented for classification, the implementation starts with a small number of decision trees which identify important features in the dataset. This is iterated to find both important and unimportant data in the dataset, with the unimportant data being discarded. This data is used to calculate upper and lower bound for the number of trees to implemented to the forest. The conclusion drawn, show that a dynamically changing random forest, improves efficiency and accuracy. This attributed to the removal of unimportant features and an optimal number of trees.

Rule-Base Classification

Automatically generating classification rules has been used as alternative approach to Expert Systems, as no previous knowledge is needed.

Rule-base classifiers are a model that is built up of IF-THEN rules. The precondition (IF) holds a condition, that if true, the rule consequent is satisfied. Rules can be accessed by the amount of coverage and accuracy they have over a dataset.

Weiss *et al*. (2010) paper on Rule-based data mining for yield improvement in semiconductor manufacturing, explains how the inference of patterns to form rules that demonstrate the performance of tools. The rules are used to as alterative to decisions tress as they are stated as being too complex to cover the insufficient number of IBM wafers. The rules are developed, then filtered to cover a significant number of wafers from many lots. A parallel can be drawn to the classification of the dataset one and two, where generalisation, using wildcards aim is to reduce the number of rules, while still classifying the dataset.

Neural Networks

Neural networks (NN) works are based upon the neural connections from biology, such as the connections in the brain. Neural networks are not specifically programmed for a task, they are trained. Gurney (1997) states that neural networks emphasis on learning is from experience, this is made possible by the modification of weights with a defined threshold (bias) and use of backpropagation. Weights adjust throughout the learning process, they increase or decrease, which fire a signal from neuron to the next layer in the NN. The topography of a NN is an input layer, hidden layer and output layer. Backpropagation algorithm is the ability to distribute the error, back through the layers, changing each weight when an error is received at the output layer. NN are known as supervised learning.

NN can be used in varying classification techniques, Kuruvilla, Jinsa (2013) paper classifies the detection of lung cancer. The input to the NN was CT scans with an intensity greater than a threshold level are labelled with a ‘1’ and all pixel values with an intensity less than threshold are labelled with a ‘0’threshold. This representation makes it able to feed in the information from the scans, to the NN. The NN was trained on 70% of the data, using different types of back propagation, then tested on the remaining 30%. An accuracy of over 90% is achieved from classifying lung cancer from the input.

A GA rule base algorithm could have been used classify and detect patterns from the CT scans, then once a rule had been hit this could have been fed into the NN, this approach could have been used to optimally search through a CT which has large quantities of unimportant features.

Hybrid classifiers

A hybrid classifier is the combination of one or more classification techniques. They are a power approach to classification, as all techniques have their advantages and disadvantages, when used in conjunction they can become optimal or offset their disadvantages.

An example of a hybrid classifier is Farid *et al*. (2014) paper on a hybrid between a decision tree (DT) and naïve Bayes (NB). Farid *et al.* (2014) explains that both are efficient classifiers. DT does suffer from overfitting, which causes its accuracy to drop. Farid et al (2014) explains the use of the NB is to remove the noisy instances from the training set. The DT is used to reduce computation of the NB to use induction to select comparatively more important subsets of attributes. The hybrid was concluded to have made progress upon single action classifiers, thus the use of hybrid classifiers is a powerful tool.

3 EXPERIMENTATIONS

3.1 Data Set 1

The approach taken to alter the Simple GA, to a rule base classifier was first to create a class Rule, this held an integer array for the condition and integer for the output.

The fitness method needed modification to include the initialisation of rules and the evaluation of the rules. This was achieved by passing individual solution, which formed a rule. This string of binary was generated from the GA which was then partitioned to match the dataset condition length and output. The gene length of individual solution had to adapted so that the N number of rules would fit the length of a solution. For dataset one, the starting point was 10 rules, this amounted to 5 genes in the precondition and 1 for output. So, for the 10 rules this amounted to 60 gene length (10*(5+1)).

Once The rules had been formed, evaluation of the rules was achieved by looping over the data instance’s attributes and output, comparing to the rule condition and output respectively. If the rule matched the dataset, the fitness would then be incremented, this was repeated for N number of times, depending on the amount of data in the dataset.

Pseudo Code Fitness Function

```
Fitness Function
Set Rule[number of Rules]
Loop For(number of rules;j++)
        Loop For(condition length)
                Rule[i].condition[j] = Condition[i].gen[k++]
        End Loop
End Loop
         Rule[i].output[k++]
For(length of data;i++)
        For(Number of rules;j++)
                If( data[i].postion[j] equals Rule[j].condition )
                        If (Rule.output equals data.output
                                Fitness++
        End Loop
                Break – to get next item after match
End Loop
```

Parameters	Description
Population	The amount of individual solutions
Selection	The method of selecting solutions, for recombination. **Tournament selection** is selecting two random solutions and picking the solution with highest fitness **Selection wheel** is selecting a solution from the population by associated probability to each solution
Recombination	Recombination/Crossover is a method of combining two sections of different solutions by selecting one (or more) point in which the section of genes is swapped
Mutation	The process of changing part of the solution, for example in a binary string to flip a 0 to 1. In real value number this could be change the value within a range
Generation	The Number of times the Genetic algorithm repeats the process of selection, recombination mutation.

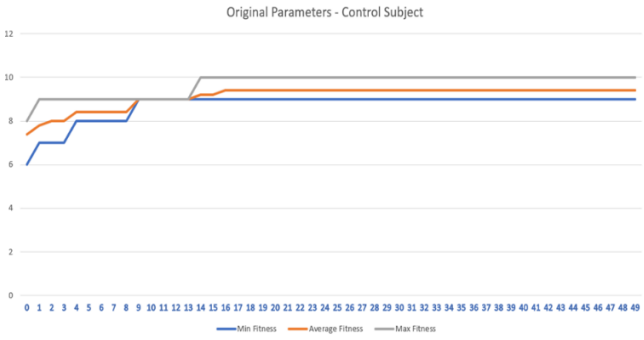


Figure 1

The GA rule base classifier was executed five times, so that an average could be obtained this is displayed as average fitness (the average of the max fitness over 5 executions), the best result from the five runs is displayed as max fitness and the worst performing run from the five is referred to as min fitness. This was performed on all data for this research, which predominantly shown in graphs.

The parameters have been set to an arbitrary value so that with experiment can be conducted to view the results of varying the parameters. The parameters are set to; Selection type: Tournament, Population:50, Generation:50, Crossover:60% and Mutation:10% The results show that 10 rules were classified, indicated by the max fitness reaching 10 and an average over 5 executions show that an average of 9.4 was achieved, this is shown in figure 1.

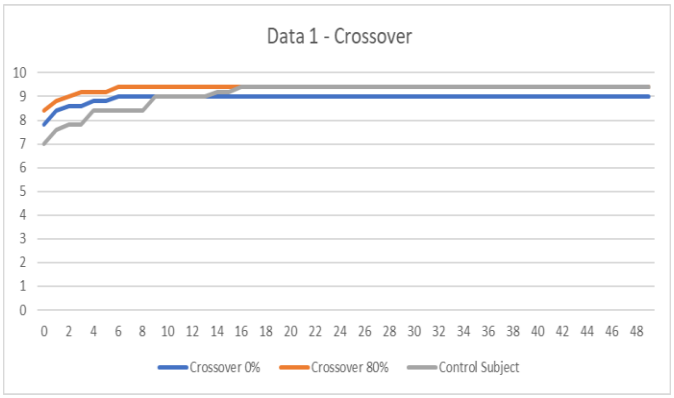


Figure 2

Figure 2 shows the results of varying crossover rates, which are set at; 0%, 80% and the control subject (60%). The graph shows that 0% performed the least efficiently, as it initialised with a higher fitness than the control subject, yet it finished with a lower fitness. The results of the 0% show premature convergence, without crossover to help the solution move throughout search space, only mutation can provide diversity. On the other end of the results, was crossover at 80% was an improvement over the control subject. Crossover on two fittest solutions either side of a peak in the search space landscape such as a unimodal would return a better candidate solution. The reason for this is solution one will be correct up to the crossover point and solution two correct after the point, the recombination of the offspring will result in a higher fitness.

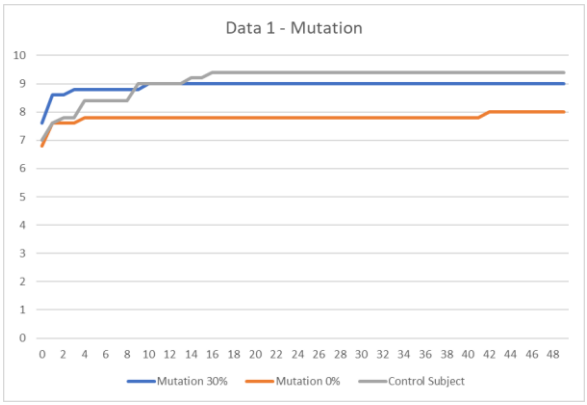


Figure 3

The rates of mutation are shown in figure 3, which are 0%, 30% and the control subject (10%). It is clear the importance of setting mutation at the right percentage, as 0% displays no ability to move in the local search space, it is visible that this produced premature convergence, as crossover could only swap bits from one another. Alternatively, If the mutation is set too high, as shown by the 30% mutation the candidate solution will act erratically within the landscape, mutating an individual solution too often will create a random population of solutions.

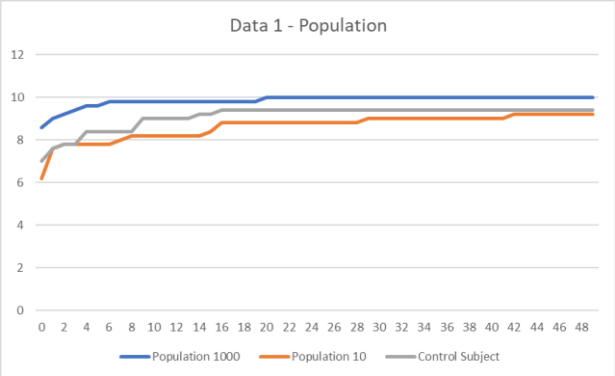


Figure 4

The size of the population has a clear impact on the result of the fitness, this revealed in figure 4. The sizes compared were; 10, 50 and 1000. The population of size 1000 outperformed the control subject and the population of 10. A reason for this is that the larger the population, the more solutions there is in a search space, which benefits the algorithm map the search space. A population which is more diverse uses selection to retains the fitter individuals. It also reduces the chance of premature convergence of individuals as there is a wider selection of solutions. This is shown in figure 4 as a population with 1000 initialised a higher fitness, increased faster and reach 10 fitness on all attempts. There is also a correlation between the size of the population and its initialised fitness, the larger the population the higher the initial fitness.

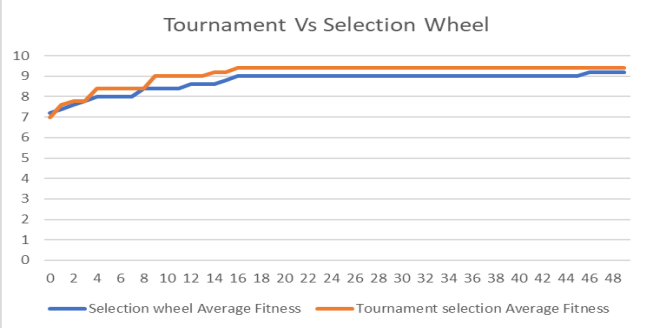
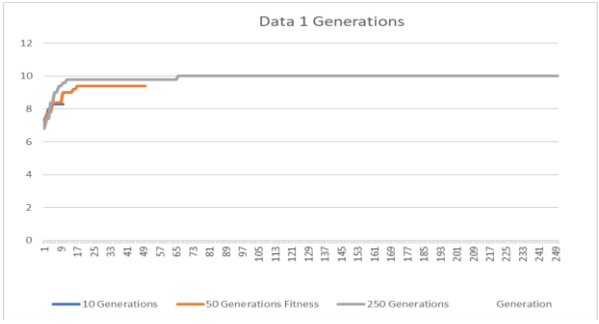


Figure 5

The results of selection wheel compared to tournament selection are shown in Figure 5. Tournament selection outperforms selection wheel, with a fitness of 9.4 to 9.2 respectively. This reveals that tournament selection is better at selecting individuals for this search space.



	10 GENERATIONS	50 GENERATIONS	200 GENERATION
MAX FITNESS AVERAGE OVER 5 EXECUTIONS	8.3	9.4	10

The effect of generation change showed that the more generations executed the high the fitness. The different attributes were, 10 generations, 50 generations and 200 generations. 10 generations finished with a fitness of 8.3, this was below the control subject which scored 9.4 and the 200 generations reached 10 on all 5 attempts.

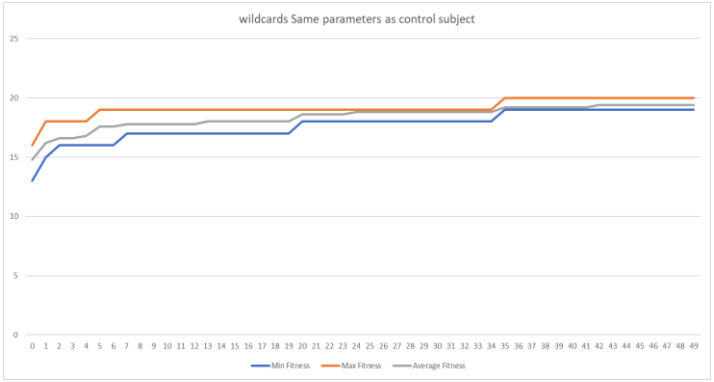


Figure 6

A wildcard enables generalisation, the wild card is represented by a ‘2’. The wildcard will be accepted by the precondition as either a ‘0’ or ‘1’.

Figure 6 demonstrates the introduction of the wildcard. The wildcard provides an opportunity for a rule to match more than one dataset. Therefore, the fitness of an individual can increase past 10, the previous limitation. For example, the rule 22222 0 matches 17 data items from dataset 0, this leaves the other rules to be set and match the dataset. The wildcard introduces generalisation, which is useful for complex problem spaces. The wildcard makes it possible to classify a range in the search space. Therefore, the rule with all wildcards can classify more than one rule as it is occupying more than one space in the landscape.

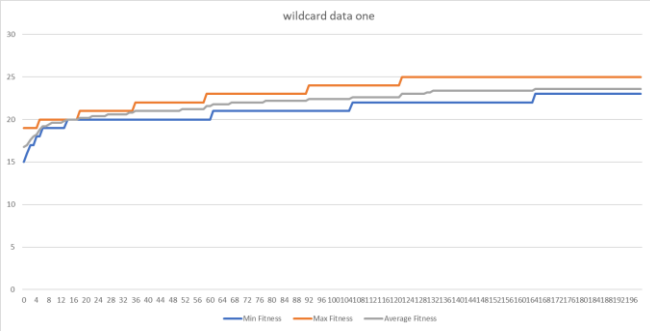


Figure 7

The next step in improving upon the figure 6 results, was achieved by using the knowledge gained in previous outcomes from the changes in parameters. The population was increased to 200, crossover to 80% and mutation to 3%. Figure 7 shows the improvements of the combined parameter changes, the max fitness was 25, with an average of 23.6. From analysis of the model the max fitness for dataset one with 10 rules is 25, this amounts to 78.13% of the dataset being classified from 10 rules. This is because one rule handles half the dataset scoring 17 fitness (sometimes less depending on position in individual string), this is the rule full of wildcards. The other rules match 8/9 from the dataset. The reason for this is that no more generalisation can performed, otherwise the same rule would be hit twice, causing the fitness function to break out, as the length of dataset condition only permits a certain placement of wildcards. Rule conditions must be able to identify descriptive features of the data attributes and ignore others, thus the model for dataset one does not offer a suitable structure that can be classified.

Data 1	Rule
00101 1	00101 1
01100 1	01100 1
01010 1	01010 1
11011 1	11011 1
10111 1	10111 1
01001 1	01001 1
11110 1	11110 1
11101 1	11101 1
11011 1	11011 1
00000 0 01000 0 10000 0 10110 0 00111 0 00001 0 01011 0 10011 0 11010 0 00010 0 01101 0 10101 0 11100 0 00100 0 01110 0 11001 0 11111 0	22222 0
00011 1 01111 1 10010 1 11000 1 00110 1 10001 1 10100 1	Unclassified

Figure 8

The table above shows an example of the rules that were classified, as the rules were dynamic this is just one representation of the rules that were established by the GA, they show nine rules handling 8 instances, and the other handling seventeen. Seven rules are unclassified.

3.2 Data Set 2

Dataset 2 is in binary representation and has consists of 64 data instances, these instances consisted of 7 attributes and one output. The gene structure had to be changed so that it could correctly be partitioned, the gene length was changed to 80 (10*(7+1)).

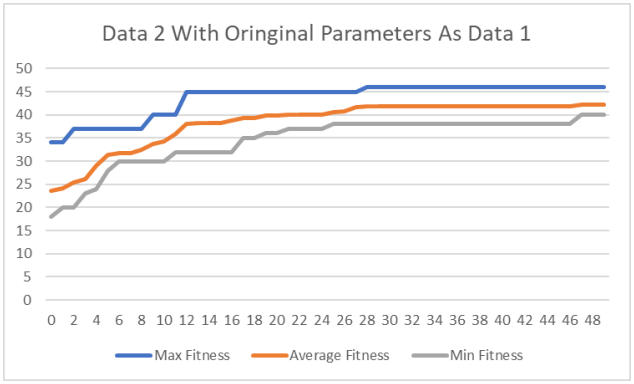


Figure 9

Figure 9 shows the performance of the of the GA classifier, the parameters here are the same as the control subject from dataset 1; Selection type: Tournament, Population:50, Generation:50, Crossover:60% and Mutation:10% and 10 Rules. The average fitness has 42.2 with a max of 46. With a possible fitness of 64, these parameters were producing unsatisfactory results, thus parameters will be changed to same figures that concluded dataset one.

Figure 10 shows the results of dataset 2 with 10 rules with the improved parameters; generation: 200 population: 200, crossover: 80%, mutation: 3% and tournament selection. All data items have been classified by the 10 rules, shown by the max fitness of 64 and an average of 61 over five executions. When comparing Figure 9 and figure 10, shows that the similar results in changing parameters as in dataset 1. The improvement of almost 20 on average and the max fitness being found shows the effects of parameter manipulation.

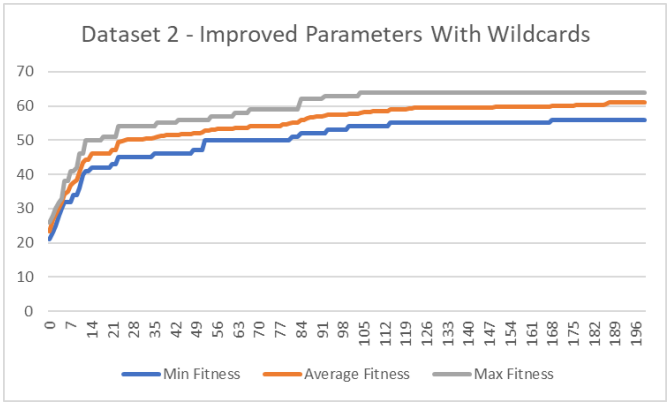


Figure 10

Figure 11

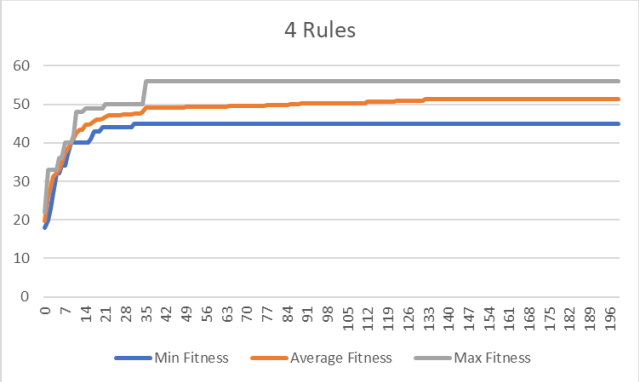


Figure 12

The gene length was changed depending on the number of rules, were N is number of rules (N*(7+1)).
To maximise the generalisation of the classification, the rule base classifier was executed on reducing number of rules until the max fitness was unable to be reached. Figure 11 and 12 shows that this separation, Figure 11 shows that 5 rules was able to correctly classify all 64 data items. Figure 12 shows that 4 rules were unable to reach a fitness of 64, reaching a maximum of 56 and an average of 51.4. Figure 13 shows the reason for this, the structure of the rules and how these rules have classified the data. Figure 13 shows that there is a rule which is just wildcards, which represents half the dataset. The other rules contain wildcards in last five attributes. The rules with the output 0 shown in figure 13, the top 4 rules in the table, are separated by the first two digits or attributes. They are 11, 01, 00 and 10, as each of these data instances sets beginning with each one respectively has 8 data items. Therefore, at least 4 rules are needed to represent this half of the dataset, as there are only four combinations of two binary digits. The completely generalised wildcard rule to represent the other half, determines that at least 5 rules are needed.

Dataset 2	Rules
1100000 0 1100100 0 1101000 0 1101101 0 1100011 0 1100111 0 1101011 0 1101110 0	1102222 0
0111001 0 0110000 0 0101001 0 0100000 0 0111010 0 0110010 0 0101011 0 0100010 0	0122022 0
0011101 0 0010101 0 0001100 0 0001000 0 0011001 0 0010000 0 0000100 0 0000001 0	0022202 0
1010001 0 1010101 0 1000000 0 1000100 0 1010011 0 1010111 0 1000010 0 1000110 0	1020222 0
0000011 1 0100101 1 1001000 1 1110000 1 0000111 1 0100110 1 1001011 1 1110010 1 0001011 1 0101100 1 1001101 1 1110101 1 0001110 1 0101110 1 1001111 1 1110111 1 0010011 1 0110100 1 1011000 1 1111000 1 0010111 1 0110111 1 1011010 1 1111010 1 0011011 1 0111100 1 1011100 1 1111101 1 0011110 1 0111110 1 1011110 1 1111111 1	2222222 1

Figure 13

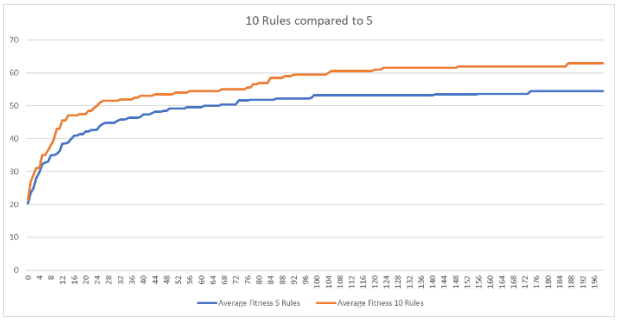


Figure 14

Figure 14 shows the comparative of results from 10 rules compared to 5 rules. Although both hit the max at least once in the 5 executions, 10 rules perform better on average. To perform max generalisation of the model, reduces the efficiency. This is apparent in the result of figure 14, the model needs to find rules that are correctly generalised so that a rule can match 8 different data items, unlike 10 rules were 4 is needed. Although, the 10 rules may evolve a rule that matches 8 and have rules that match 0, this gives 10 rules more room in the search space.

3.3 Data Set 3

Dataset 3 is a large set of real number values. The number of instances is 2000 and has 7 attributes of decimal numbers between 0 and 1. The output is represented by a 1 or a 0.

As the representation has changed to real value numbers, the fitness function, mutation and initialisation of genes had to be adapted. The the rules also had to be adapted, a rule would now be a range for each instance between two genes. Therefore, the length of the gene would be twice the number of instances plus the output multiplied the number of rules, 150 ((7*2)+1)*10). The initialisation of genes needed to real value numbers and every 15th gene a 0 or 1 to represent the output. The rule class also needed to be modified, so that it had two arrays of type double to hold the lower and upper bound of the of the range. Mutation was altered to moving the decimal within a range, that was between 0 and 0.1, either adding or subtracting if it did not

exceed 1 or go below 0. Finally, the fitness function was changed so that it would increment the fitness is a data instance was in between the lower and upper bounds, which were set by the rules.

Pseudo code – matches condition (within Fitness function)

```
Matches condition Function
    int amountCorrect = 0;
    For (dataset length; i++) {
      IF (conditionGenelowerBound[i] >= dataSet[i] && dataSet[i] <= conditionGeneUpperbound[i])
        amountCorrect++;
      end IF
    IF (amountCorrect == dataSet.length) {
      return 1;
  ELSE
      return 0;
```

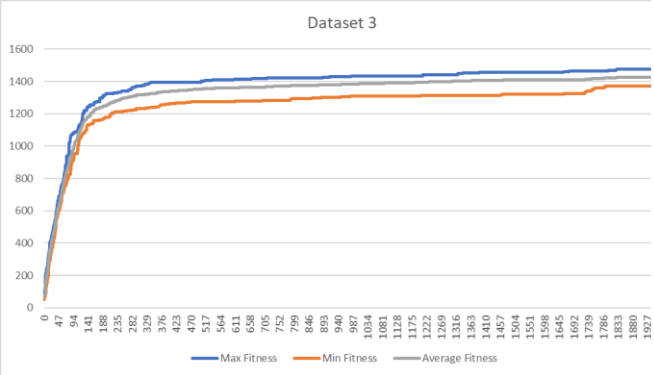


Figure 15

Figure 15 shows the result of dataset 3. The parameters are set to; generation 2000; population 1000, crossover 80%, mutation 4% (Range set to 0.0-0.1) and tournament selection. As the dataset is a lot larger this has impacted the results of the fitness, the average of the max fitness is correctly classifying 71.25% of the data, with the max run at 73.75%.

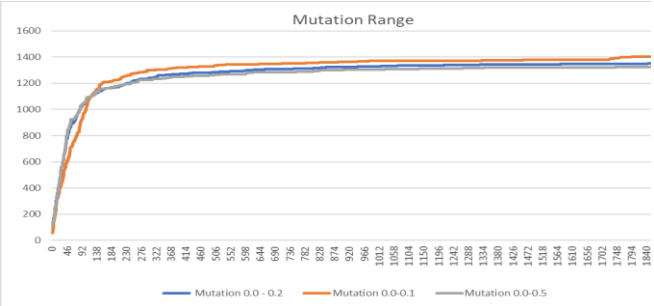


Figure 16

The range in which a gene can mutated was an important parameter to examine. The ranges that were experimented upon are; 0.0-0.1, 0.0-0.2 and 0.0-0.5. Figure 16 shows that 0.0-0.1 having reached the highest fitness, it also shows that it performed slower in the earlier generations, then performed better once the population had converged. A reason for this is that the smaller range only nudges the gene up or down slightly, thus on genes that already have a higher fitness would only want a small change, only slightly moving within the landscape.

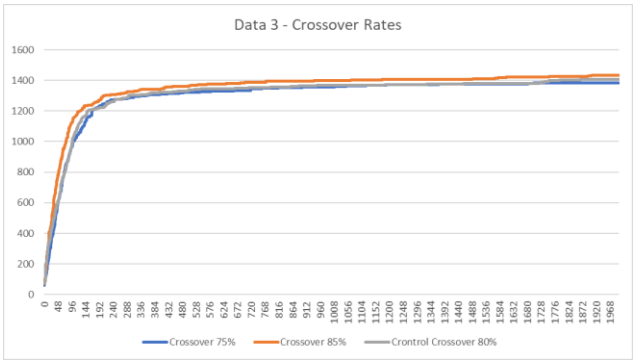


Figure 17

The results presented in figure 17 show crossover at 85% outperforms 75% and 80%. This leads to the conclusion that a larger dataset performs better when crossover is set at a high rate.

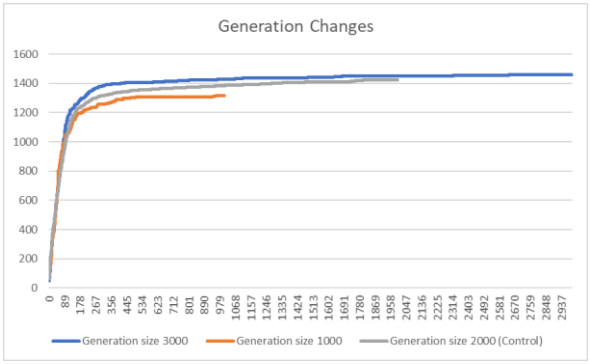


Figure 18

The results presented in figure 18, is consistent with data set 1 and 2, that the longer a GA is run, the better the results are achieved.

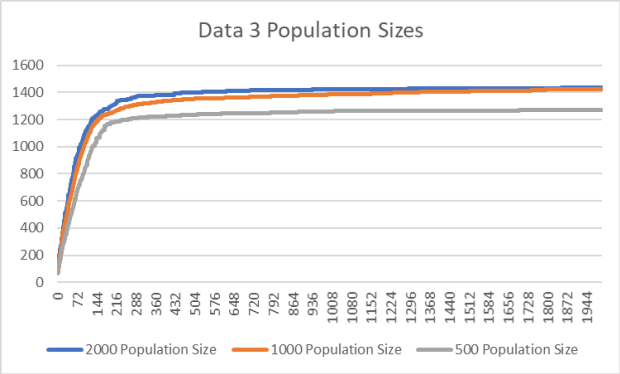


Figure 19

The graph in figure 19, shows the effects of population sizes on dataset 3. The larger the population the higher chance of producing fitter solutions. Although has the figure 19 shows once the generation hits 1000 there is little different between a population of a 2000 or 1000. A population which is a lot smaller, such as population 500 does not perform to the same standard. Which shows that there is range in which the GA performs at optimal standard for a population size.

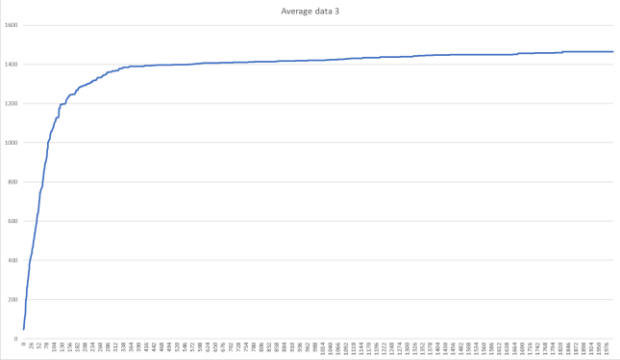


Figure 20

Figure 20 shows dataset 3, after parameters have been changed to optimally improve the GA performance. This has resulted in 73.22% of the data instance being correctly identified, which is an improvement compared 71.25.

4 CONCLUSIONS

In conclusion, the experimentation on dataset one has revealed that by varying the parameters has improved the max fitness and the effectiveness of the GA to construct the rules effectively which attempted model the data. The use of wildcards has advanced the fitness, as the rules have become generalised, therefore covering wider range in the search space. From the data collected on dataset, shows that it is not possible to use ten rules to classify 100% of the data, as only 78.13% of the data was correctly classified.

Dataset two experimentation has provided further knowledge on fluctuating the parameters, to produce an effective rule base GA. The use of wildcards enabled the model to correctly classify all the data in dataset two and permitted the number of rules to be reduced, which help revealed the true structure of the modelled data.

The assessment into adjusting the parameters for dataset three has proven inconclusive. The testing proved that adjusting separate parameters had a positive effect on the results, yet when these results were pulled together to produce effective parameters for the GA results were negligible. The results of the rule base approach to dataset three was just over 73.22, which is improvement of 1.98% compared to the initial run of dataset three.

If the experiment was to be repeated, an approach to systemically increase and decrease the parameters would have produce the ability to hone in on and locate the point in which they performed optimally. Another approach would be to execute the rule-based GA more than five times, as the initialisation had a large impact efficiency and on the out of the outcome. For dataset three a different classification, such as a neural network, might have improved the results. As a rule, based GA was unable produce highly successful results.

REFERENCES

Apte, C. and Weiss, S. (1997) Data mining with decision trees and decision rules. *Future Generation Computer Systems* [online]. 13 (2), pp.197-210. [Accessed 22 November 2018].

Bramer, M. and SpringerLink (Online service) (2016) *Principles of Data Mining* [online]. 3rd 2016. ed. London: Springer London. [Accessed 02 November 2018].

Dahan, H., Cohen, S., Rokach, L. and Maimom, O. (2014) *Proactive Data Mining with Decision Trees* [online]. New York: Springer. [Accessed 26 November 2018]

Dudek, G. (2013) Genetic algorithm with binary representation of generating unit start-up and shut-down times for the unit commitment problem. *Expert Systems with Applications* [online]. 40 (15), pp.6080-6086. [Accessed 15 November 2018]

Farid, D.M., Zhang, L., Rahman, C.M., Hossain, M.A. and Strachan, R. (2014) Hybrid decision tree and naive Bayes classifiers for multi-class classification tasks. *Expert Systems with Applications* [online]. 41 (4), pp.1937-1946. [Accessed 22 November 2018]

Gurney, K. (1997) *An Introduction to Neural Networks* [online]. UCL Press. [Accessed 24 November 2018]

Han, J., Kamber, M. and Pei, J. (2012) *Data Mining: Concepts and Techniques* [online]. London; Amsterdam; Morgan Kaufmann. [Accessed 19 November 2018]

Kuruvilla, J. and Gunavathi, K. (2013) Lung cancer classification using neural networks for CT images. *Computer Methods and Programs in Biomedicine* [online]. 113 (1), pp.202-209. [Accessed 22 November 2018]

Angshuman, P., Mukherjee, D.P., Das, P., Gangopadhyay, A., Chintha, A.R. and Kundu, S. (2018) Improved Random Forest for Classification. *IEEE Transactions on Image Processing* [online]. 27 (8), pp.4012-4024. [Accessed 21 November 2018]

Weiss, S.M., Baseman, R.J., Tipu, F., Collins, C.N., Davies, W.A., Singh, R. and Hopkins, J.W. (2010) Rule-based data mining for yield improvement in semiconductor manufacturing. *Applied Intelligence* [online]. 33 (3), pp.318-329. [Accessed 20 November 2018]

Appendix

Project Stored on GitHub:
https://github.com/george98788/-UFCFY3-15-3-Genetic-Algorithm-Rule-based-Classifiers-