Bio Computation Assignment

Ahmed Aslah (S1700918)

1 INTRODUCTION

The purpose of this assignment is to play with an evolutionary algorithm and record the attempt we made to solve a classification problem. A classification problem can be defined as the output variable or the final outcome variable of the problem is a category. That is whether it can be either "0" or "1" or "positive" or "negative", etc. Classification problems can get a conclusion from a given data set after examining the values in that data set. There are different types of classification models.

2 BACKGROUND RESEARCH

The world we are living in is a place where a huge amount of data is collected day by day. Examining the collected data is an essential requirement (Han, et al., 2011). Today data mining is the most imperative tool used to separate data and to handle data. Also, data mining is used to make patterns to get the most useful data for making decisions (Guruvayur & Suchithra, 2017).

In general technology terms, as long as the data has a meaning related to the target application, any type of data can be a target for data mining. The most common data which are mined was data from databases, data from data warehouses and transactional data. Data mining can also be targeted for other types of data which include spatial data, data streams, multimedia data, sequenced or ordered data, text data, World Wide Web, networked or graph data, etc (Tiwari, et al., 2017) & (Han, et al., 2011). Recently, there has been a huge development in data gathering innovation such as standardized bar-code scanners in business spaces and sensors in both logical and modern parts. This has led to an era of huge data measurements. (Guruvayur & Suchithra, 2017).

Classification is one of the types used in data mining to group objects into identified classes Guruvayur & Suchithra, 2017). The process of classification analysis is done by examining the data in the demonstration database, by creating an exact description or by founding an exact model or by mining the classification rule for every category and using the classification rule to classify the data in other databases (Zhang, et al., 2008). To find the faults in fields like medicine, biology, geography, business and many more, the techniques for classification have been used broadly. To address the matter of classification, various approaches and algorithms have been used. Performance and interpretability are the most important measure of classification. So huge efforts are made to minimize and understand the resulting rule bases. Therefore, the main objective of major model classification is accurate and interpretable rule sets (Zhang & Liu, 2010).

If the quality of the data is bad, the whole thing can become a failure, though data mining can be used to do amazing things.

Data mining can be done using several methods, but this paper focuses on the use of the Genetic Algorithm (GA). The GA is a methodology of optimization and search based on the principles of genetics and natural selection (Haupt & Haupt, 2004).

By taking direct cues from nature, a GA allows the creation of a fittest solution from the use of the following methods of evolution:

- Offspring: creation of a high population from the combination of parents.
- Mutation: occur across offspring to help survive in the environment.
- Selection: if the offspring are strong they will survive to become parents to pass.

The GA is based on these three stages of evolution.

3 EXPERIMENTATION

3.1 Data Set 1

Representation:

The data set 1 (data1.txt) contains 32 rows of data and each row holds a length of six bits. The first five bits in each row represent the conditions and the last bit is the type. For example, let's assume that each of the conditions in the data set is the result of a test. 1 is considered as positive and 0 is considered as negative, the combination of the conditions will produce an output of the type (last bit). The output is also again either positive or negative with 1 or 0. The input data to this is binary, so we only had to generate the initial population with the genes being met from a random selection of 0 and 1s.

Fitness calculation:

The fitness of the solution is calculated by comparing the conditions, which is in each row of the given data set against the rules which are generated in the possible solution. When the data set is passed through the rules, if it gets a match the quality of the rule will be recorded and then the algorithm breaks out from the loop. This helps the algorithm to compare that set of data against any other rule in the solution. This theory can guarantee the fittest among the solution will contain an ordered list of rules which can be followed easily.

The rules should not be hardcoded and it has to be generalized in a way that it can be fitted to more than one row in the given data set. For example, in the data set if the first and last condition (which is the first bit and the fifth bit) contains 1, the result is always 1. The system needs a way to remember this, so later while it is running it can ignore the middle part (which is from second bit to fourth bit). To achieve this condition a third character # was used. So when the system gets a condition that includes # it can safely ignore the character and it can continue the algorithm.

With the help of wildcards, the system can now classify multiple lines of data using a single rule.

- Rule: 1###1 = 1
- Data 1: 10001 = 1
- Data 2: 11101 = 1

Results:

It seems that data 1 does not have a real structure or pattern. While running data set 1 it gives poor results. The highest fitness the system gets is equal to:

- (Number of rules – 1) + 17

When examining the data in data set 1 it came to know that results from most of the rows has a type 0. When running the algorithm to find a single rule it returns:

- ##### = 0

The above rule gives a fitness of 17, which is that it can successfully classify 17 rows of data out of the 32 rows. Any other rules added to this lead to a copy of the data row itself.

Parameter Testing:

With the help of the information from data set 1, it is possible to test different parameters in order to find the optimal search parameter.

The fitness for the following series of results is taken from the average fitness of each generation over 100 runs of the algorithm.

3.2 Data Set 2

The data set 2 seems very much similar to data set 1 but with a minor change. Data set 1 has five conditions, but data set 2 has six conditions. So the calculation for the gene has changed:

- (6 + 1) * 10

Instead of this change, there were no other changes made in order to run the data set 2.

Since the data set has an extra condition compared to data set 1, it doubles the size of the input data to 64 lines. When running the data set 2 the GA analyzed all the 64 rows within 10 rules, without having any problems. After running 100 GA's, 66 solutions correctly classified all the 64 items in the data set 2 and returned the best set of rules as:

- Rule 1: 110### = 0
- Rule 2: 01##0# = 0

- Rule 3: 10#0## = 0
- Rule 4: 11000# = 0
- Rule 5: 00###0 = 0
- Rule 6: ###### = 1
- Rule 7: 1#1011 = 1
- Rule 8: #01000 = 1
- Rule 9: 100011 = #
- Rule 10:010#1#= 1

The scoring system works in such a way that the rules are presented in the form of an ordered list and should be read from first to last. Within this in mind, the above set of rules includes four needless rules. Due to rule 6, rule 7 – 10 are never seen by the data set. Once the data has gone through rules 1 – 5 and directly linked all data sets which ends with 0 all else must be a 1. This mean s that all the 64 rows can be classified by a set of 6 rules. This is not the least amount of rule for this data set.

After 100 GA has been configured to examine for 5 rules 21 accurately identified all the 64 items in the data set. Which is approximately 1 in 5 chance of finding the below set of rules:

- Rule 1: 00###0 = 0
- Rule 2: 01##0# = 0
- Rule 3: 110### = 0
- Rule 4: 10#0## = 0
- Rule 5: ###### = 1

After repeating the test it came up with another set of 5 rules:

- Rule 1: 111### = 1
- Rule 2: 00###1 = 1
- Rule 3: 01##1# = 1
- Rule 4: 10#1## = 1
- Rule 5: ###### = 0

The two rules sets look similar in structure. Among the five rules, the first four rules classify the conditions for the return of a specific type, while the fifth rule is covered by the other type.

3.3 Data Set 3 (and UCI data)

Representation:

Different from Other data sets (Data set 1 and Data set 2) this data set (Data set 3) has floating-point numbers in it. The data set consists of 2000 rows.
Example of a row from data set 3: (0.059994 0.534671 0.952161 0.245603 0.661871 0.353233 1)
So, to overcome this situation a new way has to be implemented. At the moment the formula that is used to find the length of the gene is based on the assumption that each condition or a bit in the data set can be represented by using 1 of 3 different characters. But in this data set, each bit can have any value between 0.0 and 1.0. So we have to come up with a way to encapsulate those values.
To overcome this situation we can use ranges. It allows floating-point numbers to be calculated easily. If a number falls between two values it can be considered as a machine. To proceed from the previous example of the floating-point conditions:

- (0.2 – 0.9), (0.6 – 0.7), (0.1 – 0.4) = 1

This will be a valid rule that fits the example given above. Through doubling the size of the conditions in the gene it is possible for the gene to come up with a suitable range that can be determined by the GA. The data set 3 has six floating-point conditions and one binary bit for the type the gene length will follow the rule:

- ((6*2) + 1) * the number of rules.

We already changed the formula for determining the size of the gene length. The function which populates the gene also has to be changed. Rather than populating the genes with binary numbers (0s and 1s), it has to populate with floating numbers between 0.0 and 1.0 randomly. Another most important changes to the algorithm should be updating the mutation and fitness scoring functions.

Mutation:

Now it is possible to represent any number between 0.0 and 1.0. So there is no use of the wild card. Yet the process of mutation is still needed to avoid stagnation of the solutions. By retaining the method for calculating the percentage of mutation per bit, when the mutation is needed instead of adjusting the value to one of the other characters (as per binary configuration) the values are +/- by a set amount. It is very beneficial to add a range limit (lower limit and upper limit). So when the number is > 1.0, it will round off the number to 1.0 and is the number is < 0.0,

it will round off the number to 0.0.

Fitness Scoring:

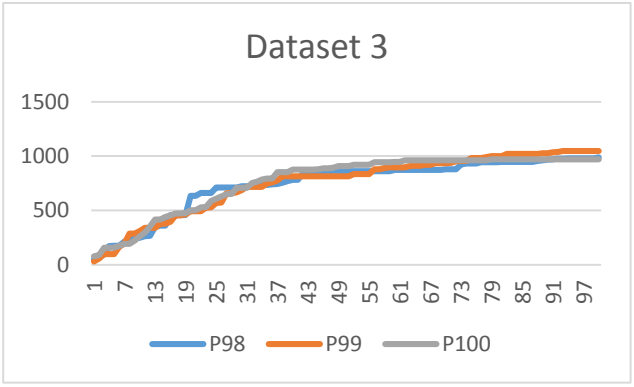
The way that fitness needs to be scored is another change we have to make. The function in the previous GA compared the value to either 0, 1 0r #. This method requires a change. This time we have to check whether the condition in the data fits between 0.0 and 1.0. The new matching condition requires the two numbers in the gene to decide the range to be either (min, max) or (max, min). The rest of the matching algorithm remains the same to construct an ordered rule set. To compensate for the additional conditions in the gene, a counter is needed.

Training and testing:

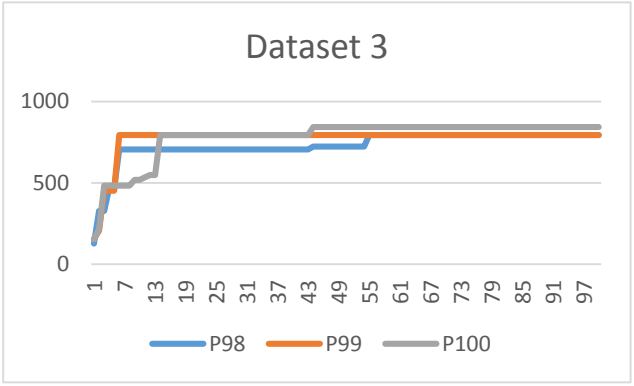
Data set 3 contains 2000 rows of data. To test the efficiency of the GA the above-mentioned data set has been split into two data sets. That is a training set and a test set. The idea was to use a training set of known data to create a solution. Then it is possible to test the validity of the solution by comparing it with the remaining known data in the test set.

Results:

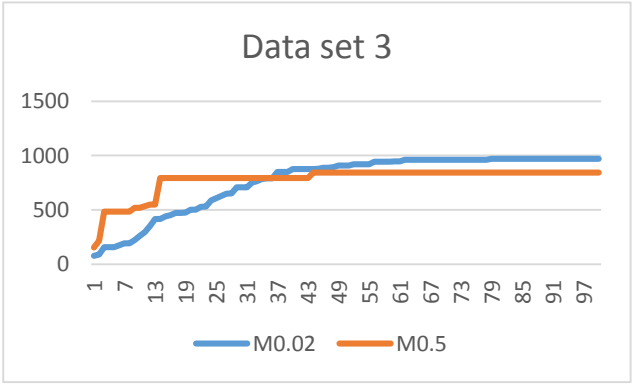
When working with a huge amount of data, the probability of being able to get a 100% match is very low. But the % can be high if we use high-level resources and the processing time is used to find the perfect parameters. Often, by using such a high level of data, it is very likely that there may be one or two incorrect pieces of data that may twist the results. In the case of the data set 3, it came up with a result of a 100% match rate after testing out different parameters. With a size of mutation to 0.5, the algorithm was able to correctly classify 100% of both the training and test data set. After running the GA multiple times across the data set 3, it came to know that when the GA score was > 95 percent on the training set, the rule would fit into the set that scored 100 percent. Since the rules generated by the GA included ranges for each data state, it is desirable to translate them into a more readable format. If it takes that "< 0.5 = 0" and ">= 0.5 = 1" It is very easy to change these ranges to bit strings like the one in the data set 1 and data set 2.



Mutation rate 0.02, Population 100, Generation 100.



Mutation rate 0.5



The figure shows the best fitness of Mutation rate at 0.02 and mutation rate at 0.5.

4 CONCLUSIONS

As the Genetic Algorithm is very powerful in creating patterns and rules hidden within data sets it seems that there may have more powerful and more efficient ways of data mining. It can conclude that it is possible to get a solution to the problem as a process, although it is obvious that it may fail when scaled. The outcome after the experiments from data set 1 came to know unless a certain range for population size and mutation rate has reached a change of finding an optimum solution within a reasonable time, it will be severely reduced, and while it is possible to find the most effective set of rules, eg: data set 2 and data set 3, which can be reduced to 5 rules, there is no guarantee that the GA will be able to find them for the first time. It is very likely that the extension of GA may lead to the use of another GA to settle on the specific values found in the database of the optimal population size and mutation rate.

Source code of the project:

https://github.com/ahmed-aslah/bio_computation_assiginment.git

REFERENCES

Guruvayur, S. R. & Suchithra, R., 2017. *A detailed study on machine learning techniques for data mining.* India, IEEE.

Han, J., kamber, M. & Pei, J., 2011. *DATA MINING Concepts and Techniques.* 3rd ed. USA: Elsevier Inc.

Haupt, R.L. & Haupt, S.E., 2004. *Practical Genetic Algorithms.* 2nd ed. New Jersey: John Wiley & Sons, Inc.

Tiwari, M., Dixit, R. & Kesharwani, A., 2017. *DATA MINING PRICIPLES, PROCESS MODEL, AND APPLICATIONS.* New Delhi: Educreation Publishing.

Zhang, L., Chen, Y., Liang, Y. & Li, N., 2008. *Application of Data Mining Classification Algorithms in Customer Membership Card Classification Model.* Taiwan, IEEE.

Zhang, Y. & Liu, H., 2010. *Classification Systems Based on Fuzzy Congnitive Maps.* China, IEEE.