**Optimisation the Disease: Randomness the cure**

**Midhath Ahmed & S1700908.**

**1 INTRODUCTION**

In Computer Science, methods of finding optimisations has been vital part of finding solutions to problems. In General, these Optimisation methods allow generic problems to solved easily. This document outlines how I have attempted to find a general optimisation method, involving classifying three separate data sets into two categories using a neural network and training (adjusting weights) using evolutionary algorithm approach, namely a genetic algorithm instead of using the venerable feed-forward and backpropagation approach. In addition, I have compared the two approaches and have tested more data sets to validate the integrity and flexibly of the approach I have come up with.

**2 BACKGROUND RESEARCH**

Evolutionary Computation is a subset of artificial intelligence, that merges the trial and error optimisation methods with the famous theory of evolution from Charles Darwin. All these methods take the concepts found in nature in finding a good enough solution to a problem in reasonable and cost-effective time frame using evolutionary theory of survival of the fittest: i.e.: Selecting the current best individuals for reproduction and produce new individuals with crossover and mutation, and Finally replace the least fit individuals with the new offspring’s.

**Ant Colony Optimisation**

Ant Colony Optimisation is a probabilistic method of finding solutions to problems that can be solved by finding good paths in traversing a graph(). As the name suggests this method inspired by the ants’ foraging behaviour in finding the best path of travelling to food back and forth and searching for it.

It is interesting to see that the relatively trivial path optimisation problems such as Traveling Salesmen and robotic pathfinding() can be solved with great results, this method can also find solutions to some unconventional problems, as a group of researchers managed to find a method to maximize social network profit by identifying the most influential people in a mobile network and theoretically maximise the effectiveness of viral marketing().

**Particle Swarm Optimisation**

As discussed, previously path finding and navigation is a relatively trivial problem, how ever what if there are constraints such as time and other particles. I.e.: in the case of road traffic, such problems are highly dynamic, adaptive versions of Particle Swarm Optimisation can give solution to these dynamic and everchanging problems. PSO is inspired by the behaviour of swarm animals such as bees and birds, this method is used successfully to not only find paths for UAVs() formations, but also to manage and regulate traffic as stated by ().In addition, PSO is also a very viable method of finding optimised architectures for deep convolutional neural networks used in cases like image classification(), significantly reducing the cost of training such models and improves feasibility of such approaches greatly.

**Genetic Algorithm**

Genetic Algorithm is another Evolutionary algorithm approach that treats solutions having genetics that can undergo mutation and crossover, paralleling the natural genetic operations in biological sexual reproduction. This approach seems highly compatible with many problems including optimisations and even classifications, the requirement being the problem domain being able to represent a solution as consisting of many atomic parts called genes.

Researchers have been able to use this approach in creating various models()and even a fraud detection model(). In addition, some researchers () also managed to get a highly performant classifier when they used a hybrid approach of GA with Artificial Neural Network.

All in all, from my research, the hybrid GA + ANN approach seems to be the most flexible and efficient approaches to my classification problem.

**3 EXPERIMENTATION**

In order to implement the hybrid ANN + GA, it is essential to understand how an ANN work. The atomic part of an ANN is a neurone, each neurone has many inputs and many outputs. At its core a neurone is summation operator that produces its output by adding all of the values

**3.1 Data Set 1**

Describe your basic algorithm so anyone who knows about basic GAs could hopefully repeat what you have done. Describe the representation, the parameters, the fitness calculation, etc.

Present your results as averaged behaviour over more than one run, eg (graph not of assignment):

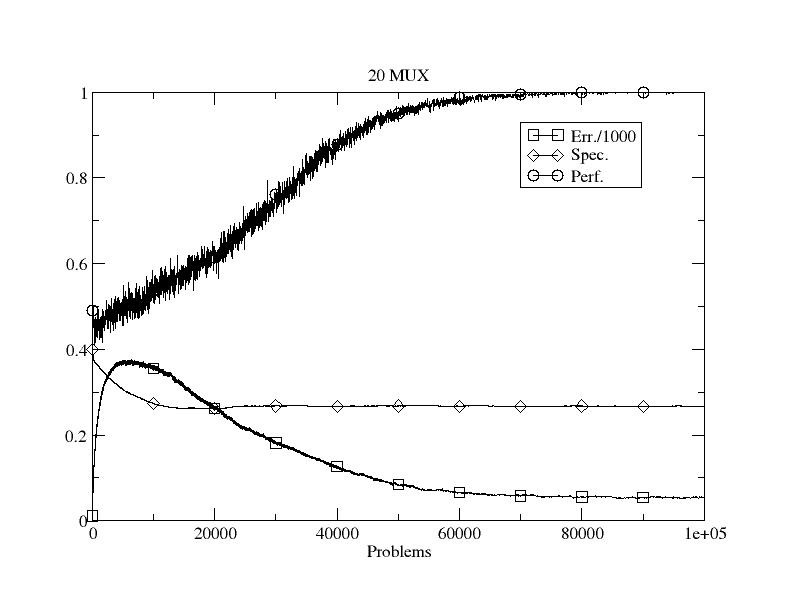


Figure 1: Initial performance on dataset 1.

Show the effects of varying parameters and give potential explanations as to why the behaviour/performance changes. Eg vary the mutation rate or population size.

**3.2 Data Set 2**

If not in the previous section, this should include how you allow generalisation in the representation. Again, show results from varying parameters/aspects and explain them. The use of evolution to learn how many rules to use would be good to include here and/or other modifications.

**3.3 Data Set 3 (and UCI data)**

Clearly describe how you changed the representation to deal with real-valued data. And then present results from its use, with graphs and explanations.

For the very keen, it would be great to see comparative performance on the UCI dataset too and/or the use of another representation scheme, eg, evolving neural networks, or use of Weka, etc.

4 CONCLUSIONS

Concise summary of what you found and learned. Identification of ways you might do things differently next time, and why.

REFERENCES

Bernado Mansilla, E. & Garrell, J. (2003) Accuracy-Based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks. *Evolutionary Computation* 11(3): 209-238.

**And please include either a link to your source code so that more than one person can access it over the next few months or as a full listing in an Appendix. The second marker and external examiner need to be able to see what you actually produced.**