Biologically Inspired Computation

Coursework 1

F20BC

Deaven Howarth and Kimberley Stewart

H00158684 & H00156061

Introduction

We built an artificial neural network using a Genetic Algorithm. We chose to use the sphere function. We chose to create our neural network in Java, using Eclipse. Our neural network can learn the sphere function for the pairs (1,1), (1,2) ... (9,9). In its current state, our neural network can only learn this range of data. It also calculates the error rates and the recall number. The recall number for all of our results is very close to one.

We were unable to get COCO to run, therefore we don't have any results to compare. We also ran out of time trying to get crossover and mutation working. We currently only use one generation because the crossover and mutation has not been successfully implemented.

The link to our GitHub is: https://github.com/dh134/BiologicallyInspiredComputation

Setup of Experiments

Due to the limited nature of our program, we could not run any experiments on it. In its current state, the program learns a pre-determined list of pairs along with the results for these. As crossover and mutation have yet to be implemented, the neural network can only run for one generation.

We had difficulty understanding how COCO was meant to be used in the coursework. We voiced our concerns regarding COCO in a number of emails to try to aid in our understanding of what we were being asked to do. Despite the assistance we received, we still could not manage to get COCO to run and continued to get errors whenever we tried. Part of our problems with COCO was that the coursework descriptor only ever referenced Matlab, with small hints towards Python. As we have never learned Matlab, we chose to do the coursework in Java. The files in the Java version of COCO were very different to the ones referenced in the Matlab version which made trying to run it almost impossible.

The results our program can generate are quite limited. The only thing it really produces is an error rate and recall number. If we had more time, we would have continued to try to improve our program so it would create a wider range of results.

Results

Due to the difficulty we had attempting to run COCO, we were unable to generate any results which could be used as a comparison to the results our program currently generates. At the moment, when our program is run, it generates a list of recall numbers for the data inputted to it. We are aiming for the recall number to be as close to 1 as possible to show that the program has learned. The exact results vary each time the program is run.

```
Recall:
1:1:=0.9999999999977411
1:2:=0.9999999999989762
1:3:=0.9999999999993752
1:4:=0.9999999999995219
1:5:=0.9999999999995819
1:6:=0.9999999999996081
1:7:=0.9999999999996199
1:8:=0.9999999999996252
1:9:=0.9999999999996276
2:1:=0.9999999999984492
2:2:=0.9999999999991993
2:3:=0.9999999999994551
2:4:=0.9999999999995539
2:5:=0.9999999999995957
2:6:=0.9999999999996143
2:7:=0.9999999999996225
2:8:=0.9999999999996265
2:9:=0.9999999999996283
```

Figure Example of some of our results

The main issue we had when attempting to generate COCO results was that our program always learns the same information. We could not find a way to generate results with COCO, and whenever we tried to run it, would get an unsatisfied link error.

Discussion

There are a lot of ways we could have improved our results. Firstly, we could have tried to get crossover and mutation implemented. We had planned on doing this but as the deadline was really close, we felt it was more important to make sure that the implementation we had actually worked, rather than breaking it in an attempt to add more features.

Additionally we could have spent longer trying to make COCO work to get comparison results. However with the way we had implemented our program and the lack of clarity in exactly how to run COCO, we prioritised the report over continuing to struggle with COCO. If we had more time, we may have asked for more help. The advice that we did receive generally wasn't very relevant to the problem we were having. We understand that the answers were never going to be an exact explanation of what to write, but on some occasions the answers we received did not answer the question we asked.

To try to improve our program, we tried a few different approaches to the problem. Our first choice was to randomly generate pairs and add the solutions to the current population. This worked initially, however it confused us when we came to trying to add in crossover and mutation. This code is now commented out in the code we have submitted. We decided to change our approach when we realised that it didn't seem right, however this left us with little time to get the new version working fully.

Conclusions

In conclusion, the program we built does not do everything it was supposed to. Initially we tried a different method, which we later found to be unsuitable. By the time we realised that we would have to change our approach, it left us with very little time to complete it. Due to the limitations, it was very difficult to produce results.

COCO added to the difficulties we had. It was not very user friendly and it was not explained very well. We did try getting help with COCO but even after this, it would not run for us.