Project spec:

Project spec has changed

I have a NN and GP,

Whatkind of NN, how many layers,

Design:

I have decided to implemtn two different things

Intention to create two methods to compare them against each other and already made models

Testing -> what kind of testing , validation

How am I comparing the output for the two differnet hing

What kind of crossovers and mutations im doing.

Tree vs infix notations

Data structure and notation

Fitness function

Selection and replacement procedures

Process:

Agile dev

Validation – how do I know it is working

This is not a deterministic process

Testing:

Xvalidation

Since very difficult to test, I tested on small programs that you know it should get the right answer

Test plan- code testing, different functions to you

Product

Is it split into good functions, good clean code?

Testing:

Tuning parameters

What could I have done better

Why testing code is hard because it is not deterministic

Design:

I am planning on making two models as described in the product spec

Before starting project:

Create the financial ratios using the data that I have from the data set

GP’s:

Research the most appropriate GP to use.

Usin gregression trees to produce an optimal function

Research data structures

Chose binary trees

Terminal and functional sets

Understand the full process of a genetic program

Made a flow chart to understand the steps I could take .

Define and research the differnet types of genetic operators

What GO’s will I start with

Why am I using these one

Tournament selection

Selecting best ones

Selecting random ones each time – brute force

types of crossover single point subtree crossover

mutation – single point mutation

fitness function – number of hits technique

terminating conditions – why I have chosen thesw.

Reinstate the final aim as was stated initially – to produce a function tthat will predict CF.

Design of tests:

What kind of testing, valdation

ANN’s:

What libarary to use.

What dataset to use.

Determine what type of ANN to use.

For this, to avoid over complexity, feed forward network used.

Determine how many inputs

Determine how many layers and nodes

What activation function to use

What learning algorithm to use

 A sufficiently wide neural network with just a single hidden layer can approximate any (reasonable) function given enough training data.

 shallow networks are very good at memorization, but not so good at generalization.

advantage of multiple layers is that they can learn features at various levels of abstraction

 Multiple layers are much better at generalizing because they learn all the intermediate features between the raw data and the high-level classification.

the wider your network, the longer it will take to train

How am I comparing the outputs

* Once crossover node had been selected, I had to find locate that subtree
* Created function – find subtree which.
* Function uses depth first search recursively to the node and the specific node id. If the node is not in the list, it throws an error to indicate that the node is not within the list.
* Now that node selected, and subtree found, crossover can occur.
* Made function – swap nodes to perform crossover.
* Uses both trees, their list of nodes and the random nodes selected as the trees and list of nodes are going to be updated.
* Locates the child to be swapped and swaps the links over with the other parent, such that the subtree detatches itself from its current parents and is attatched to the new substree.
* Used a clone of the parents to do this, since wanted to keep parents in the population initially.
* Crossover done with corss over rate based on the user choice.
* Next to be completed was mutation.
* Mutation generally comes after crossover
* Mutations for this project have smaller effects on the population, however can still make changes to the population fitness of individual
* To compelte this, again, selected random value for the each child, and find the subtree for that node.
* Perfrom to check if the node selected is a genetic operator. If it is, then select another genetic operator with the same arity, and return this new tree.
* If value is in X1, X5, then select another variable based on that
* If in neither, then check to see fitness of individual,
* If greater than 0, then reduce value by 0.1.
* Based on mutation rate which was relatively low

***Prefix to infix conversion***

Now that the children trees were made, they now had to be put back into population, therefore they had to go back to original state of being in infix notation.

Created another class – ToInfixParser to do exactly that.

First method – decondstruct tree was used to take all the list node values, which were identcle to the tree, and put them directly back into prefix notation.

Next function made was taken from elsewhere to convert the prefix to infix conversion.

Initially, convert\_to\_infix reverses the prefix notation and starts to add the values into stack

It checks to see if the genetic operators are in the list, if they are, then pop off the top two values from the stack and append them to format – <oerand><operator><operand>. Keep doing this for all values in the prefix expression till it ends, then return the infix expression

Updating population

Fitnesses of both children could now be computed easily and population updated.

Within population class, created method – update population.

Function takes the current popn, their fitnesses, the new children

The function finds the worst member of the current population and gets its index.

This value is then compared against the first child. If it it better, then this value is removed from the population. This process is reeated to prvent child enetering population or replacing the worse individual in the population.

Eventually the new population is created with new fitness values.