5/6/2019

Assignment 2

Comp307

Zane Rawson
300367145

Part 1:

(I did not change the data files for this part)

1. The number of input nodes is equal to the number of independent values in the data set meaning in this case it would be 4 (sepal length, sepal width, petal length, petal width). The number of output nodes should be 3 as we are dealing with 3 classes. The hidden nodes can be between 1 and 5, I found that having 3 hidden nodes gave the greatest degree of accuracy.
2. For the learning rate I used 0.05 as when I increased the learning rate increased very quickly however when I ran the test data it was clear that it was over fitting as the error rate was very high. For the momentum I did not change it and left it at 0.0. I used 0.01 as my weight range as I did not want any random fluctuations to have an affect whenever I changed a parameter.
3. For my Network training termination Criteria I used mean squared error and set it to 0.01 so as to keep the highest level of accuracy I could while making sure that over fitting was not a problem which it would have been had I set it to 0.
4.

Test	Result	Accuracy
1	567 Epoch, 0.013% Test Error, 0.001% Training Error	Training: 73/75 Test: 74/75
2	570 Epoch, 0.008% Training Error, 0.001 Test Error	Training: 73/75 Test: 74/75
3	584 Epoch, 0.0093% Training Error, 0.001% Test Error	Training: 73/75 Test: 74/75
4	577 Epoch, 0.001% Training Error, 0.001% Test Error	Training: 73/75 Test: 74/75
5	566 Epoch, 0.008% Training Error, 0.001% Test Error	Training: 73/75 Test: 74/75:
6	570 Epoch, 0.013% Training Error, 0.011% Test Error	Training: 73/75 Test: 74/75
7	576 Epoch, 0.009% Training Error, 0.017% Test Error	Training: 73/75 Test: 74/75
8	574Epoch, 0.01% Training Error, 0.01% Test Error	Training: 73/75 Test: 74/75
9	590 Epoch, 0.008% Training Error, 0.014% Test Error	Training: 73/75 Test: 74/75
10	583 Epoch, 0.01% Training Error, 0.01% Test Error	Training: 73/75 Test: 74/75

Average Epoch = 576

Average Test Error = 0.0079%

Average Training Error = 0.0077%

Average Training Accuracy = 73/75

Average Test Accuracy = 74/75

Overall I would say these results are satisfactory and that this model works well with the parameters set

5. While K-nearest neighbour does not need to train any data it does use dynamic memory. This means that it my use up more memory than a neural network.
   BP neural networks are good as they do not use up large amounts of memory as while it may take a bit of time to train it so that it achieves a decent error rate it doesn't need the training data once this is achieved.

Part 2:

1. A good terminal set for this task is just the variable X and constants between 2.0 and 10.0
2. Since our goal is to find the relation between X and Y I believe the basic operations (Add, Subtract, Multiply, Divide and Power) to be enough to achieve a desirable result
3. The fitness function I used the calculates the Y value based on the X input and looks at the difference between the calculated Y Value and the actual Y. by doing this we look at how small the difference is to determine the accuracy.
4. Max Initial Depth = 5
   Population = 500
   Max Tree Depth = 10
   Evolutions = 1000

   It repeats the genetic programming until it either reaches the best fitness set(0) or it has been evaluated 1000 times.
5.

Best Solution	Fitness Accuracy
`(5.0 / 5.0) + (((X * (X * X)) * (X - 2.0)) + (X * X))`	`0.00`
`(((X - 2.0) * (X * (X * X))) + (3.0 / 3.0)) + (X * X)`	`0.00`
`((X * X) + ((X * X) * (((X * X) - X) - X))) + (3.0 / 3.0)`	`0.00`

6. Using the fist one

```
(5.0 / 5.0) + (((X * (X * X)) * (X - 2.0)) + (X * X))
```

Lets get a few values from the data:

X	Y
-2	37
-1.75	24.16
-1.5	15.06

a. `(5.0 / 5.0) + (((-2 * (-2 * -2)) * (-2 - 2.0)) + (-2 * -2)) = 37`
b. `(5.0 / 5.0) + (((-1.75 * (-1.75 * -1.75)) * (-1.75 - 2.0)) + (-1.75 * -1.75)) = 24.1601562`
c. `(5.0 / 5.0) + (((-1.5 * (-1.5 * -1.5)) * (-1.5 - 2.0)) + (-1.5 * -1.5)) = 15.0625`

Based of the Y values given from the X input it is clear that these work as they give back the exact Y value.

Part 3:

1. A good terminal for this task is a set of variables a-i and constants between 2.0 and 10.0
2. For this task the Function had to be changed to achieve the best results. For this one I used Add, Add 3, Add 4, Multiply, Multiply 3, Divide and Subtract. From my tests these gave me the best results.
3. For this case I chose to go for a high accuracy Approach. This meant using the approach that was covered in the tutorial and if the output is < 0 it is in class 1 else class 2. With this it selects the generation with the highest accuracy and mutates from there
4. Max Initial Depth = 4
   Population Size = 1000
   Max Crossover depth = 8
   Crossover Prob = 0.6
   Reproduction prob = 0.05
   Mutation prob = 0.1

   The stopping Criteria are that there have been either 1000 generations or that there is a fitness function equal to 100%
5. I chose a split of 75% for the training data and 25% for the test data. This should give enough data to train the program and more then enough to test it while giving us accurate results
6. K

run	Best Solution	Accuracy
1	`((8 - (I + F + A)) * B * G) + ((10 - (B * H)) + 4)`	Training: 96.95% Test: 97.8%
2	`7 + (C + B + ((3 - (B * A * F)) + E + 10))`	Training: 96.95% Test: 97.1%
3	`(((3 - (B * C * F)) - G) + 10) - (-6 - (9 - (H * C * F)))`	Training: 96.95% Test: 98.3%
4	`(((3 - (B * C * F)) - G) + 10) - (-6 - (9 - (H * C * F)))`	Training: 96.95% Test: 98.3%
5	`(((3 - (B * C * F)) - G) + 10) - (-6 - (9 - (H * C * F)))`	Training: 96.95% Test: 98.3%
6	`(10 - A) + 8 + (10 - (F * E * B)) + (9 - (H * E * B))`	Training: 96.95% Test: 98.3%
7	`((E * H) * ((9 / 1) - (F + C + I + B))) + C + D`	Training: 96.95% Test: 98.3%
8	`(10 - (E - (10 - (B * F)))) - (I + C + G + C)`	Training: 97.14% Test: 98.3%
9	`9 + (H * (E + F + B + (E - (B * E * F)))) + B`	Training: 96.95% Test: 98.8%
10	`8 + H + (((6 + (8 - A) + E) - (C * E)) - (B * H * F))`	Training: 97.14% Test: 98.8%

7. The best solutions to use should be the ones that performed the best in the test set.
   a. `8 + H + (((6 + (8 - A) + E) - (C * E)) - (B * H * F))` – 98.8%

b. `9 + (H * (E + F + B + (E - (B * E * F)))) + B` – 98.8%
c. `(((3 - (B * C * F)) - G) + 10) - (-6 - (9 - (H * C * F)))` – 98.3%