Assignment #2

Neural and Evolutionary Learning

COMP 307 Introduction to Artificial Intelligence

Qiangqiang Li(Aaron)

ID: 300422249

Tutors: Prof. Mengjie Zhang

Dr. Yi Mei

Part 1: Neural Networks for Classification

1. Determine and report the network architecture, including the number of input nodes, the number of output nodes, the number of hidden nodes (assume only one hidden layer is used here). Describe the rationale of your choice.

Reply: Using the neural network simulator of BPNN:

The number of input nodes is 4, because there are four 4 attributes/features in iris dataset.
The number of output nodes is 3, because there are three classes. Each class refers to a type of iris.
The number of hidden nodes is 2. Comparing to the number of the hidden nodes, like the number of the hidden nodes is 1, mean squared error for test data: 0.0231. Like the number of the hidden nodes is 2, mean squared error for test data: 0.016. Like the number of the hidden nodes is 3, mean squared error for test data: 0.021. The result is not good enough, but the best squared error is the lowest result, and chose the number of hidden nodes is 2.

2. Determine the learning parameters, including the learning rate, momentum, initial weight ranges, and any other parameters you used. Describe the rationale of your choice.

Reply: According to Lecture notes, 0.2 is a good start for learning rate. Because the dataset is not that big, the momentum set 0 and do not need a momentum to speed up the process.

Initial weight ranges from -0.5 to 0.5 when set all weights to small random values.

Critical error is 0.001 , because critical error set small ,which can lead to a better performance.

The percent classification accuracy is 101% , which means it will not stop the training process even the training accuracy reached 100% or it get a small enough error(critical error).

3. Determine your network training termination criteria. Describe the rationale of your decision.

Reply: When this get a small enough error, critical error is 0.001. So the percent classification accuracy is 101% which means it will not stop until it get a small enough error (critical error reaches 0.001). The practical result shows that the termination criteria will affect the performance. When using a small enough critical error, the model can get a better accuracy.

4. Report your results (average results of 10 independent experiment runs with different random seeds) on both the training set and the test set. Analyse your results and make your conclusions.

Reply: the report 10 times in the below

Run 1:

Mean squared error for training data: 0.002

Number of incorrect classifications: 0/75

Mean squared error for test data: 0.016

Number of incorrect classifications: 2/75

Run2 :

Mean squared error for training data: 0.002

Number of incorrect classifications: 0/75

Mean squared error for test data: 0.017

Number of incorrect classifications: 2/75

Run 3:

Mean squared error for training data: 0.003

Number of incorrect classifications: 1/75

Mean squared error for test data: 0.021

Number of incorrect classifications: 3/75

Run 4:

Mean squared error for training data: 0.001

Number of incorrect classifications: 0/75

Mean squared error for test data: 0.018

Number of incorrect classifications: 2/75

Run 5:

Mean squared error for training data: 0.001

Number of incorrect classifications: 0/75

Mean squared error for test data: 0.018

Number of incorrect classifications: 2/75

Run 6:

Mean squared error for training data: 0.002

Number of incorrect classifications: 0/75

Mean squared error for test data: 0.016

Number of incorrect classifications: 2/75

Run 7:

Mean squared error for training data: 0.001

Number of incorrect classifications: 0/75

Mean squared error for test data: 0.018

Number of incorrect classifications: 2/75

Run 8:

Mean squared error for training data: 0.001

Number of incorrect classifications: 0/75

Mean squared error for test data: 0.019

Number of incorrect classifications: 2/75

Run 9:

Mean squared error for training data: 0.003

Number of incorrect classifications: 1/75

Mean squared error for test data: 0.015

Number of incorrect classifications: 2/75

Run 10:

Mean squared error for training data: 0.001

Number of incorrect classifications: 0/75

Mean squared error for test data: 0.018

Number of incorrect classifications: 2/75

According to the Mean Squared Error, it is different in each run, and accuracy also is change.

In the each independent experiment, the initial weight is random produced each time. In this case, the average accuracy on training data approximately 99.9%. It seems that weights are perfectly fit on training data. The average accuracy on test data approximately 97.3%. It seems that weights are also good for test data. The dataset is small, more data needed for further detection.

5. (optional/bonus) Compare the performance of this method (neural networks) and the nearest neighbour methods.

Reply : According to the result of Assginment1, when k=3 , the KNN method has the best accuracy which is 96% . For this case by using neural network, when critical error as 0.001, the accuracy reaches 97.3%, which is better than KNN method. In this case, compared with KNN, neural network is a better solution for this case. However, the performance dependent on problem. We can only say that neural network is better than KNN on this problem. It does not mean that neural network is a better method than KNN.

Part 2: Genetic Programming for Symbolic Regression

1. Determine a good terminal set for this task.

Reply : Set X and a random number constant as terminal set for this task.

2. Determine a good function set for this task.

Reply :{+, -, *, %} which are add, subtract, multiply, divide.
Add(+) : Add two results from two child nodes.
Subtract(-): Use the result from left child node subtract the result of right child node.
Multiply(*): Use the result from left child node multiply the result of right child node.
Divide(%): Use the result from left child node divide the result of right child node. Returns 0(or 1) if denominator is 0.

3. Construct a good fitness function and describe it using plain language (and mathematical formula, or other formats you think appropriate).

Reply: The fitness function should be designed to give graded and continuous feedback about how well a program performs on the training set. This case will choose MSE (mean square Error) as fitness function. Because MSE as a common fitness function in machine learning .The less MSE, the better the performance.

If a vector of predictions generated from a sample of n data points on all variables, and is the vector of observed values of the variable being predicted, then the within-sample MSE of the predictor is computed as

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

4. Describe the relevant parameter values and the stopping criteria you used.

Reply: the relevant parameter values show in filr koza.params(ecj/ec/gp/koza/koza.params):
Population = 1024(population is the size for each pool to store candidate formula, 1024 is big enough for this case).
Generations = 51(The performance of first several generation improves fast, however, it does not improve too much after that. So 51 generations is enough).
Tournament size = 7(every time we choose result from pool, we random select 7 candidates and select on with best fitness),.

Max depth = 17(the full size of each tree),

Crossover rate = 80%(80% of next generation are produced by crossover method).

Mutation rate = 10%(10% of next generation are produced by crossover method).

Reproduction rate = 10%(10% of next generation are produced by crossover method).

Ramped half-and-half method (Which is a combination of full and grow method). Because neither full nor grow can provide trees with a wide variety of sizes and shapes.Max depth = 17(program size), grow for subtree mutation is 5 (as many as the minimal depth), and the half builder is 6(minimal depth is 2 by the way).

Stopping criteria:

Generations = 51, when generation reaches 51.

The standardized error reaches 0 or adjusted reaches 1.

When the standardized error reaches 0 or adjusted reaches 1, the program can perfectly fit dataset. It is hard to achieve this sometimes, especially for some complicated problem. Then we will force the program to stop when generation reaches 51.

5. List three different best programs evolved by GP and the fitness value of them (you need to run your GP system several times with different random seeds and report the best programs of the runs).

Program 1:

Fitness: Standardized=1.4062500000933654E-10 Adjusted=0.9999999998593749 Hits=20

Tree 0:

((((x * x) - (x * x)) % (x - x)) + ((x - (x * x)) * (x - (x * x)))) - (((((x - 0.578006) * ((x * (((x - x) % x) * (x - x))) - (x * x))) - ((((x + 0.757628) - ((x + 0.256520) - (x * x))) * ((x + 0.591202) - (x * x))) - (x - 0.660903))) * (((x + 0.426671) - (x * x)) * (x - 0.295678))) * (x – x))

Program 2:

Fitness: Standardized=3.750000000124487E-5 Adjusted=0.999962501406196 Hits=20

Tree 0:

((((x * x) % (x * x)) + (x * x)) - ((x - x) + (x - x))) + ((x * x) * ((((x % x) * (x * x)) + ((x % x) - (x + x))) - (x % x)))

Program 3:

Fitness: Standardized=3.750000000124487E-5 Adjusted=0.999962501406196 Hits=20

Tree 0:

((((x * x) - (x + x)) * ((x * x) + (x - x))) + (((x + x) - x) - ((x - (((x % x) * (x * x)) + ((x - x) + (x - x))))

- (x % x)))) - ((((x - x) % (x % x)) % ((x + x) % (x - x))) + (((x * x) - (x - x)) * ((x * x) - (x * x))))

Program 4:

Fitness: Standardized=1.4062500000933654E-10 Adjusted=0.9999999998593749 Hits=20

Tree 0:

(((x * x) - x) * ((x * x) - x)) + ((x - x) + (x % x))

Program 5:

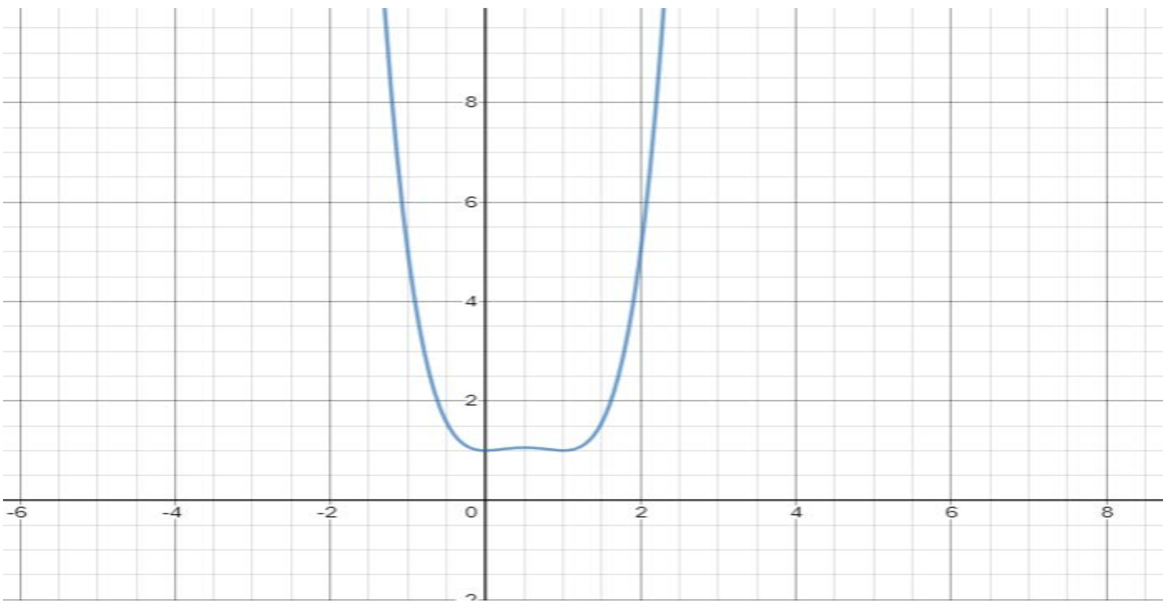
Fitness: Standardized=1.4062500000933654E-10 Adjusted=0.9999999998593749 Hits=20

Tree 0:

(x % x) + ((x - (((x * x) % x) * ((x + x) - (x * x)))) * (((x - 0.480321) * (x - x)) + x))

6. (optional, bonus) Analyse one of the best programs to reveal why it can solve the problem in the task.

Reply:the best programs is programming 4: (((x * x) % (x % x)) + (x % x)) + (((x * x) - (x + x)) * (x * x))) ,and this can transformed to $y = x^4 - 2x^3 + x^2 + 1$.The image as below:



According to put x in the formula ($y = x^4 - 2x^3 + x^2 + 1$), the programs can find output value(y) refer to all input value(x). It is clear that the programs can detect the relationship between the input variables and output variables, even though we don’t know distribution of data set. This is why it can solve the problem in the task.

Part 3: Genetic Programming for Classification

1. Determine a good terminal set for this task.

Reply : Seting X1,X2,X3,X4,X5,X6,X7,X8,X9, A random number constant as terminal sets which refer to 10 numeric value features in dataset. However, not all of these terminal sets shows in the result formula.

2. Determine a good function set for this task.

Reply : {+, -, *, /,exp,log,sqrt,square,sin,cos,tan}
Add(+) : Add two results from two child nodes.
Subtract(-): Use the result from left child node subtract the result of right child node.
Multiply(*): Use the result from left child node multiply the result of right child node.
Divide(%): Use the result from left child node divide the result of right child node. Returns 0(or 1) if denominator is 0.
Exp: apply on nodes have only one child node, do Exponential computing on result of child node.
Log: apply on nodes have only one child node, do Logarithmic computing on the result of child node.
There are 11 attributes. The good function using the 11 attributes.

3. Construct a good fitness function and describe it using plain language (and mathematical formula, or other formats you think appropriate).

Reply: The fitness function should be designed to give graded and continuous feedback about how well a program performs on the training set. This case will choose error rate as fitness function. Error rate is the amount of which classified to wrong class divide the amount of samples. Set hits as the count for right classification. The fitness function is :

Error rate = (total amount of samples – hits)/ total amount of samples.

4. Describe the relevant parameter values and the stopping criteria you used.

Reply : the relevant parameter values show in file koza.params(ecj/ec/gp/koza/koza.params) and tutorial4.params(\ec\app\tutorial4\ tutorial4.params).

Population = 1024(population is the size for each pool to store candidate formula, 1024 is big enough for this case).Generations = 51(The performance of first several generation improves fast, however, it does not improve too much after that. So 51 generations is enough), Tournament size = 7(every time we choose result from pool, we random select 7 candidates and select on with best fitness), Max depth = 17(the full size of each tree),

Crossover rate = 80%(80% of next generation are produced by crossover method),

Mutation rate = 15%(15% of next generation are produced by crossover method),
Reproduction rate = 5%(5% of next generation are produced by crossover method),
Ramped half-and-half method (Which is a combination of full and grow method).
Because neither full nor grow can provide trees with a wide variety of sizes and shapes.
Max depth = 17(program size), grow for subtree mutation is 5 (as many as the minimal depth), and the half builder is 6(minimal depth is 2 by the way).Stopping criteria:
Generations = 51, when generation reaches 51.

5. Describe your main considerations in splitting the original data set into a training set training.txt and a test set test.txt.

Reply : In this case, there are 699 instances in dataset, including 458 benign instances (65.5%) and 241 malignant instances (34.5%). I splitting the data with the ratio 70:30 which is combination of 70% benign instances and 30% of malignant instances as training data. The rest as test data. Because using enough training instances so that a good classifier can be trained, and enough test instances so that the classifier can be well evaluated. So there are 499 training instances and 200 test instances. Like the data in data-test.txt and data-training.txt.

6. Report the classification accuracy (average accuracy over 10 independent experiment runs with different random seeds) on both the training set and the test set.

Reply : The average accuracy of 10 run on training data is 97.29%. The average accuracy of 10 run on test data is 98.52%. the detail data from 10run.txt(/part3/EJC25-3/10run.txt).

run1

Best Individual of Run:

Subpopulation 0:

Evaluated: true

Fitness: Standardized=0.03206412825651302 Adjusted=0.9689320388349514 Hits=483

Tree 0:

(((((cos(x5) + ((square(rlog(x9 * x3)) % x4) - tan(cos(rlog(x2))))) % rlog(tan(sqrt(x2)) - exp(x2))) - tan(cos(cos(exp(square(exp(sin(x5)))))))) + (x6 * x6)) % square(x4 - square(exp(sin(x5))))) - tan(cos(rlog(x2)))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.025 Adjusted=0.9756097560975611 Hits=195

run2

Best Individual of Run:

Subpopulation 0:

Evaluated: true

Fitness: Standardized=0.03006012024048096 Adjusted=0.9708171206225682 Hits=484

Tree 0:

rlog((square(cos(sqrt(x1))) - sin(sqrt(exp(rlog(rlog(cos(sin(x8 + x8) % exp(x3)) + x7) + rlog((x7 * x1) + cos(cos(x8) * x7)))) + x6))) + rlog(x2))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.01 Adjusted=0.9900990099009901 Hits=198

run3

Best Individual of Run:

Subpopulation 0:

Evaluated: true

Fitness: Standardized=0.022044088176352707 Adjusted=0.9784313725490197 Hits=488

Tree 0:

square(rlog(x2) * x6) - ((exp(tan(x9)) + (cos(tan(x6 * x6)) - ((((rlog((sqrt(x4) - (x3 + x6)) * square(x5)) - cos(rlog(x2) * (x8 - rlog((rlog(x2) * rlog(x5)) + x6)))) * square(rlog(x1))) - (((square(rlog(sin(x5 + x1))) / x9) + sin(x5)) / (sqrt(x9) - (sqrt(x9) / x6)))) - (x8 / x8)))) + (sqrt(x8) + rlog(((x6 * (x5 + x1)) * (x6 * ((x6 * ((cos(x2) + ((x2 - (x9 / (x5 + x1))) * x5)) * rlog(x2))) * ((x6 * x6) * x6)))) - rlog(x1))))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.015 Adjusted=0.9852216748768474 Hits=197

run4

Best Individual of Run:

Subpopulation 0:

Evaluated: true

Fitness: Standardized=0.03006012024048096 Adjusted=0.9708171206225682 Hits=484

Tree 0:

square(x2) + (((sqrt(x9) * ((square(rlog((tan(cos(rlog(x5))) / x7) / exp(x6))) + tan(exp(tan(x9)))) - ((x7 / exp(x6)) - x7))) - (exp(cos(rlog(cos(sqrt(x9))))) - cos(exp(rlog((tan(sqrt(x5) * sqrt(x9)) / x7) / (square(cos(x2)) - x7)))))) - exp(tan(tan(x3))))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.015 Adjusted=0.9852216748768474 Hits=197

run5

Best Individual of Run:

Subpopulation 0:

Evaluated: true

Fitness: Standardized=0.02004008016032064 Adjusted=0.9803536345776032 Hits=489

Tree 0:

((((rlog(rlog(square(rlog(x5) * x6) * exp(rlog(cos(x3)))) * ((sin(sin(square(x2))) + ((x9 - tan(x3)) - (exp(cos(cos(tan(x9) + x9))) * sin(sin(x8))))) + sqrt(x7))) - (exp(sin(x3)) * sin(exp(sin(cos(rlog(x5) * rlog(x9 + square(cos(tan(x7)))))))))) + (rlog(sqrt(exp(sin(square(sin(x8)))))) - ((x3 - square(cos(rlog(x4)))) * (x9 + x1))) - (exp(sin(rlog(x9 * sin(x8)))) * sin(exp(sin(x8)))))) * exp(sin(x8))) - tan(x9)) + (((rlog(x2) * rlog(x9 + x1)) - tan(x9)) + (rlog(exp(x6 / x9) * sin(sin(rlog(square((sin(square(x2)) - x2) * sqrt(x2)) * sin(x9)) - (exp(sin(x8)) * cos(x3))))) - (exp(sin(x8)) * exp(sin(x8)))))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.025 Adjusted=0.975609756097561 1 Hits=195

run6

Best Individual of Run:

Subpopulation 0:

Evaluated: true

Fitness: Standardized=0.03206412825651302 Adjusted=0.9689320388349514 Hits=483

Tree 0:

(exp(square(rlog(sqrt(square(x1) + exp(x2))))) / rlog(x3)) - (((((rlog(sqrt(x7 * x8) * tan(x8)) - sin(sin(square(rlog(rlog(x9)))))) + tan(rlog(x7))) + exp(x2)) - (sin(cos(sqrt(x2) * tan(exp(x8)))) - tan(tan(x8)))) - ((exp(square(rlog(sqrt(square(x9 * x6) + exp(x2))))) / rlog(x3)) - (rlog((square(x9 * x6) * (square(square(x9 * x6)) + sin(square(exp(x6))))) * (rlog(x9) / (sqrt(x2) * square(tan(rlog(sqrt(square(x9 * x6) + exp(x2)))) / sqrt(square(x9 * x6) + exp(x2)))))) - rlog(sqrt(x2) * square(exp(tan(sqrt(exp(x6)))) / exp(square(exp(x6) * x6)))))))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.01 Adjusted=0.9900990099009901 Hits=198

run7

Best Individual of Run:

Subpopulation 0:

Evaluated: true

Fitness: Standardized=0.026052104208416832 Adjusted=0.9746093750000001 Hits=486

Tree 0:

sqrt(square(cos(x9 * (rlog(x8) - sin(sin(x6)))))) * (((((rlog(x6) - cos(x9 * rlog(x7))) + (rlog(x3) - sin(x5))) - cos(sin(x9 * (rlog(x8) - sin(sin(tan(rlog(x6)) + sin(sin(x5)))))))) - cos(x9 * (rlog(x8) - sin(sin(tan(rlog(x6)) + sin(rlog(x7))))))) + (rlog(x3) - (sin(cos((rlog(x6) - cos(x9 * (rlog(x8) - square(cos(square(rlog(x6))))))) * x9)) + sin(exp(tan(rlog(x7)))))))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.01 Adjusted=0.9900990099009901 Hits=198

run8

Best Individual of Run:

Subpopulation 0:

Evaluated: true

Fitness: Standardized=0.034068136272545 09 Adjusted=0.9670542635658915 Hits=482

Tree 0:

rlog(cos(exp((((exp(((rlog(x8) - x2) * rlog(x6)) - rlog(cos(x9))) + cos(x5)) * rlog(tan(sin(x5)) - (sqrt(x3) - x6))) * rlog(x6)) - rlog(x6)) + cos(x5)) - x2) - cos(rlog(x6))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.01 Adjusted=0.9900990099009901 Hits=198

run9

Best Individual of Run:

Subpopulation 0:

Evaluated: true

Fitness: Standardized=0.02404809619238477 Adjusted=0.9765166340508806 Hits=487

Tree 0:

((((((x6 - tan(sin(sin(sqrt(x5))) + cos(sqrt(x2)))) - cos(rlog(sin(sqrt(x5)) + cos(sqrt(x2))))) * (sqrt(x8) + x9)) * (cos(x6) + x9)) + sin(sin(rlog(x9)) / ((((x6 - tan(sin(sqrt(x5)) + cos(sqrt(x2)))) - tan(sin(rlog(x9)))) - (sin(sin(sqrt(x5)) + cos(sqrt(x2))) + cos(sqrt(x2)))) / sin(x5)))) + sin(sqrt(x5) / ((((cos(x6) - rlog(x3)) - (sin(sqrt(x5)) + cos(x6 - tan(sin(sin(sqrt(x5))) + cos(sqrt(x2)))))) - ((((x6 - tan(sin(sin(sqrt(x5))) + cos(sqrt(x2)))) - cos(rlog(sin(sqrt(x5)) + cos(sqrt(x2))))) * (sqrt(x8) +

(cos(cos((x6 - tan(sqrt(x2))) + cos(sqrt(x2)))) / exp(sin(rlog(x9)) / (x5 + x7))))) * (cos(x6) + x9))) / sin(x5)))) - tan(sin(x8))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.015 Adjusted=0.9852216748768474 Hits=197

run10

Best Individual of Run:

Subpopulation 0:

Evaluated: true

Fitness: Standardized=0.028056112224448898 Adjusted=0.9727095516569201 Hits=485

Tree 0:

((exp(square(sqrt(sqrt(x8) - (cos(tan(x5)) - square(x2))) - square(x2)) * x6) - exp(x4 - tan(x6))) + ((exp(square(sqrt(x1) - square(x2)) * x6) - exp(exp(tan(tan(x5)) - rlog(tan(x6))) - ((exp(rlog(exp((sqrt(x1) - square(x2)) - rlog(tan(x6))))) + (square(x8) + x6)) - exp(tan(x6) - rlog(tan(cos(square(x2)) - square(x2)))))) + (sqrt(x8) - exp(tan(tan(tan(x5)))))))) - exp(tan(tan(x6))))) + (square(sqrt(x9) - square(x2)) - exp(x6))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.015 Adjusted=0.9852216748768474 Hits=197

7. List three best programs evolved by GP and the fitness value of them.

Reply : Find the three best programs form the file: 10run.txt t(/part3/EJC25-3/10run.txt).

Program1:

Best Individual of Run:

Subpopulation 0:

Evaluated: true

Fitness: Standardized=0.03006012024048096 Adjusted=0.9708171206225682 Hits=484

Tree 0:

rlog((square(cos(sqrt(x1))) - sin(sqrt(exp(rlog(rlog(cos(sin(x8 + x8) % exp(x3)) + x7) + rlog((x7 * x1) + cos(cos(x8) * x7)))) + x6))) + rlog(x2))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.01 Adjusted=0.9900990099009901 Hits=198

Program2:

Best Individual of Run:

Subpopulation 0:

Evaluated: true

Fitness: Standardized=0.02404809619238477 Adjusted=0.9765166340508806 Hits=487

Tree 0:

((((((x6 - tan(sin(sin(sqrt(x5))) + cos(sqrt(x2)))) - cos(rlog(sin(sqrt(x5)) + cos(sqrt(x2))))) * (sqrt(x8) + x9)) * (cos(x6) + x9)) + sin(sin(rlog(x9)) / ((((x6 - tan(sin(sqrt(x5)) + cos(sqrt(x2)))) - tan(sin(rlog(x9)))) - (sin(sin(sqrt(x5)) + cos(sqrt(x2))) + cos(sqrt(x2)))) / sin(x5)))) + sin(sqrt(x5) / (((((cos(x6) - rlog(x3)) - (sin(sqrt(x5)) + cos(x6 - tan(sin(sin(sqrt(x5))) + cos(sqrt(x2)))))) - ((((x6 - tan(sin(sin(sqrt(x5))) + cos(sqrt(x2)))) - cos(rlog(sin(sqrt(x5)) + cos(sqrt(x2))))) * (sqrt(x8) + (cos(cos((x6 - tan(sqrt(x2))) + cos(sqrt(x2)))) / exp(sin(rlog(x9)) / (x5 + x7))))) * (cos(x6) + x9))) / sin(x5)))) - tan(sin(x8))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.015 Adjusted=0.9852216748768474 Hits=197

Program3:

Best Individual of Run:

Subpopulation 0:

Evaluated: true

Fitness: Standardized=0.03206412825651302 Adjusted=0.9689320388349514 Hits=483

Tree 0:

(((((cos(x5) + ((square(rlog(x9 * x3)) % x4) - tan(cos(rlog(x2))))) % rlog(tan(sqrt(x2)) - exp(x2))) - tan(cos(cos(exp(square(exp(sin(x5)))))))) + (x6 * x6)) % square(x4 - square(exp(sin(x5))))) - tan(cos(rlog(x2)))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.025 Adjusted=0.975609756097561 1 Hits=195

8. (optional, bonus) Analyse one of best programs to reveal why it can solve the problem in the task.

Reply : the best ***programs find in the data in*** the file: 10run.txt t(/part3/EJC25-3/10run.txt).

rlog((square(cos(sqrt(x1))) - sin(sqrt(exp(rlog(rlog(cos(sin(x8 + x8) % exp(x3)) + x7) + rlog((x7 * x1) + cos(cos(x8) * x7)))) + x6))) + rlog(x2))

When we input the value of nine features, we can get a output value, if it is bigger than 0, we can classified it as malignant, if it is smaller than 0, we can classified it as benign. With this program, we can detect the relationship between the input variables (all nine features) and output variables (class), So we can classified instance with the output.This is why it can solve the problem in the task.