Results

***TEST 1***

numHidden 2

numGenes 21

minGene -3.0

maxGene 3.0

popSize 1000

maxEvaluations 20000

mutateRate 0.01

mutateChange 0.05

seed 1552314780712

random java.util.Random@1f17ae12

neuralNetworkClass class coursework.Examp1leEvolutionaryAlgorithm

Training Set Training

Fitness 0.12685081756437944

0.12685081756437944

Fitness on Test 0.2863117606133829

Tournament select with tournament size 20 and Arithmetic Recombination Crossover

***TEST 2***

numHidden 2

numGenes 21

minGene -3.0

maxGene 3.0

popSize 1000

maxEvaluations 20000

mutateRate 0.01

mutateChange 0.05

seed 1552317357506

random java.util.Random@1f17ae12

neuralNetworkClass class coursework.ExampleEvolutionaryAlgorithm

Training Set Training

Fitness 0.10925899704529288

0.10925899704529288

Fitness on Test 0.2534671607971799

*Roulette select with Uniform Crossover*

***TEST 3***

numHidden 2

numGenes 21

minGene -3.0

maxGene 3.0

popSize 1000

maxEvaluations 20000

mutateRate 0.01

mutateChange 0.05

seed 1552317815304

random java.util.Random@54a9ec12

neuralNetworkClass class coursework.ExampleEvolutionaryAlgorithm

Training Set Training

Fitness 0.10850586736525256

0 Out of 8 Landed Safely

The average fitness was 0.10850586736525256

*Tournament(10) select with Uniform Crossover(2 children)*

***TEST 4***

numHidden 2

numGenes 21

minGene -3.0

maxGene 3.0

popSize 1000

maxEvaluations 100000

mutateRate 0.01

mutateChange 0.05

seed 1552318176392

random java.util.Random@5b87f79d

neuralNetworkClass class coursework.ExampleEvolutionaryAlgorithm

Training Set Training

Fitness 0.06667759150795138

*Tournament(10) select with Uniform Crossover(2 children)*

***TEST 5***

numHidden 2

numGenes 21

minGene -3.0

maxGene 3.0

popSize 200

maxEvaluations 20000

mutateRate 0.01

mutateChange 0.05

seed 1552912154050

random java.util.Random@7eda2dbb

neuralNetworkClass class coursework.OwnEvolutionaryAlgorithm

Training Set Training

Fitness 0.07009206083495512

0.07009206083495512

Fitness on Test 0.14321661339792563

My own algorithm: get 50% of the good chromosomes of the current population, produce children with them until we reach number of correct population, and replace new population with old population