Grammatical Neuroevolution

BEN WINTER

AIMLAC
UKRI CENTRE FOR DOCTORAL TRAINING
IN ARTIFICIAL INTELLIGENCE, MACHINE
LEARNING AND ADVANCED COMPUTING

1884
PRIFYSGOL
BANGOR
UNIVERSITY

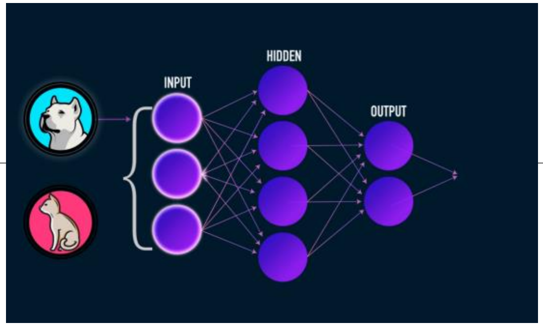
Neural networks

Machine learning technique capable of classifying non-linearly separable data if built ‘deep’.

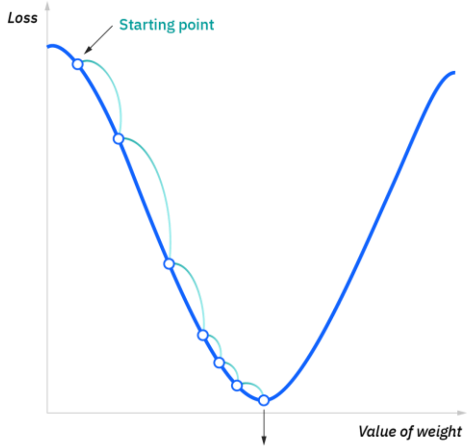
For the most part they are a supervised learning algorithm.

They use a loss function to find a minima by changing the internal weights of the network over a number of epochs.

They use lots of hyperparameters to improve accuracy and provide robustness.



[1] – TowardsDataScience – Visualizing Artificial Neural Networks (ANNs) with just One Line of Code (Adesh Shah)



[2] – IBM – Neural Networks (IBM Cloud Education)

AIMLAC
UKRI CENTRE FOR DOCTORAL TRAINING IN ARTIFICIAL INTELLIGENCE, MACHINE LEARNING AND ADVANCED COMPUTING

PRIFYSGOL BANGOR UNIVERSITY

Evolutionary Algorithms

Inspired from Charles Darwin's 'Theory of evolution'.

Creatures evolve over millennia to be better suited to their environment.

– Natural Selection

– Survival of the Fittest

Metaheuristic optimisation algorithm.

Process

1. Create a population of individuals
2. Assess their fitness
3. Select reproducible individuals
4. Crossover genetic information and mutate
5. Repeat from step 2 until the end of the evolutionary run.

Neat/Neuroevolution

Create population of individuals

An individual represents a neural networks structure.

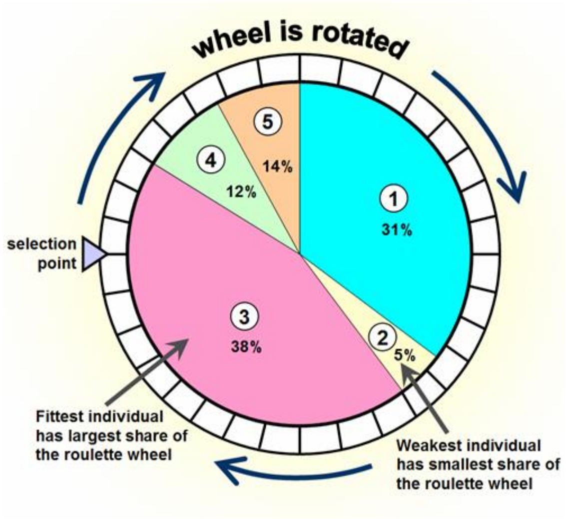
Create a fitness function as the metric you want to judge the individuals by.

The fittest individuals are the individuals in which their genetic information (network structure) produces the most accurate predictions, classifications or regression.

Selection

Roulette wheel selection:

- Allows a chance of variance
- No favouritism



Roulette wheel selection – Newcastle University (2007)

Tournament selection

- Always gain the two fittest individuals from the population

Crossover/Reproduction

Crossover point = 2

	Number of hidden layers	Nodes per hidden layer	Activation function	Number of epochs	Batch size	Accuracy
Parent 1	3	26	Sigmoid	50	4	85%
Parent 2	0	1	Softmax	100	16	80%
Child 1	0	1	Softmax	50	4	40%
Child 2	3	26	Sigmoid	100	16	95%

Child 2 is now the fittest individual.

PRIFYSGOL BANGOR UNIVERSITY

Mutation

- Choose a random point on the genome, in this case (2).
- Swap that gene with a random value of the same type.

						Accuracy
Child 1	0	1	Softmax	50	4	40%
Child 1	0	1	Sigmoid	50	4	70%

Use cases

Stock portfolio creation

[1, 5, 3, 2, 9]

Normalise each gene so the value of all of them = 100

[5, 25, 15, 10, 45]

5% A 25% B 15% C 10% D 45% E

Assess fitness by how much the portfolio returns

If fit, reproduce the individual for future generations.

Travelling salesman problem

[1, 5, 3, 2, 9]

Travel to node 1 first

1 → 5 → 3 → 2 → 9

Assess fitness by the efficiency of the route

If fit, reproduce the individual for future generations.

Grammatical Evolution

Objective is to find an executable program, program fragment, or function, which will achieve a good fitness value for a given objective function.

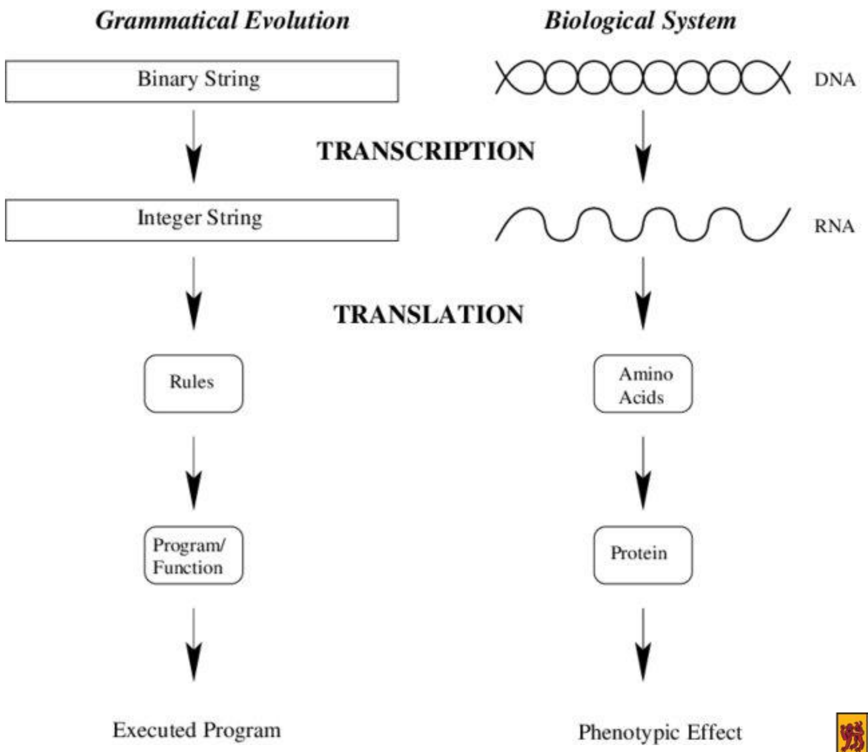
GE uses a Backas-Naur form grammar to evolve solutions.

Search space can be restricted.

The inspiration for this approach comes from a desire to separate the genotype from the phenotype.

Grammatical Evolution

- Constricting the search space, thus avoiding bloat.
- More similar to how the role of DNA in natural evolution works than other EA's.
  - Hidden mutations
- Degeneracy
- Deliberate bias can be added to the search space.



AIMLAC
UKRI CENTRE FOR DOCTORAL TRAINING IN ARTIFICIAL INTELLIGENCE, MACHINE LEARNING AND ADVANCED COMPUTING

PRIFYSGOL BANGOR UNIVERSITY

Grammatical NeuroEvolution (GNE)

GNE is the combination of neural networks and Grammatical Evolution.

GNE is primarily used to optimize a neural networks structure or finely tune hyperparameters through evolution and modularity.

Its recursion and the ability to add deliberate bias to the grammar, makes it ideal when developing a system that you may have a prior for.

Grammatical NeuroEvolution

For a system that wants to find the best number of hidden layers(nl) and nodes per hidden layer(nn).

Genotype: [5, 4, 8, 10, 5]

Grammar:

- <start>:= <expr>
- <expr>:= <var> <op> <expr> | <var> <expr> | <var>
- <var>:= 0 | 1 | 2
- <op> := + | -

Phenotype: [2, 2, 1]

nl: = 2

nn: = 2

Unused gene

<start>

    5 % 1 = 0

<expr>

    4 % 3 = 1

<var>, <expr>

    8 % 3 = 2

2, <expr>

    10 % 3 = 1

2, <var>, <expr>

    5 % 3 = 2

2, 2, <expr>

    5 % 3 = 2

2, 2, <var>

    4 % 3 = 1

2, 2, 1

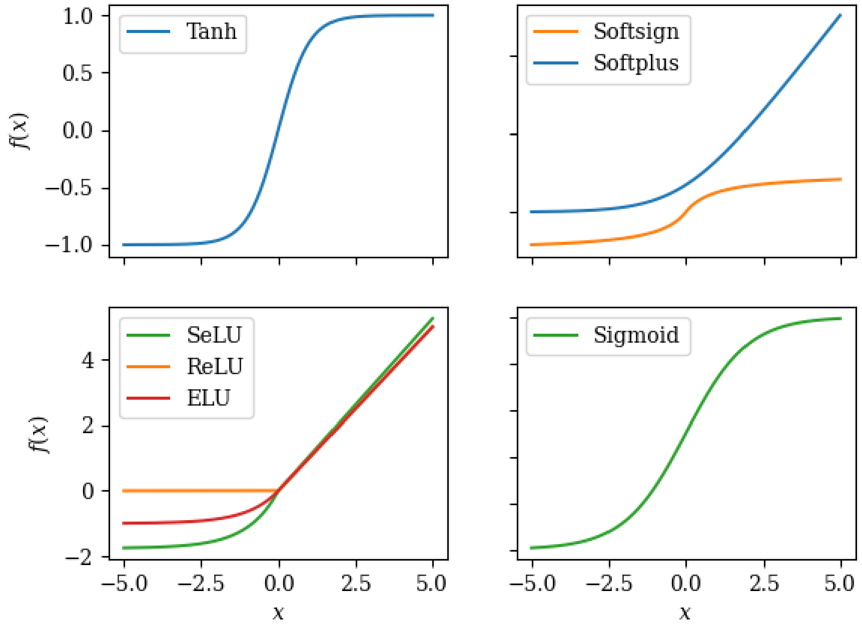
PRIFYSGOL BANGOR UNIVERSITY

Creating custom activation functions

The most used activation functions for the input and hidden layers include: ReLU, SELU, ELU, Tanh.

Recent research has shown that task-specific activation functions can potentially outperform the conventional ones.

We decided to test using grammatical evolution to automatically create activation functions, input them into an existing network architecture and find out if they produced better results than ReLU.



The grammar

Genotype

[93, 36, 86, 0, 92, 43, 83, 88, 61, | **14, 97, 49, 25, 63, 34, 90, 50, 62, 61, 50, 22, 84, 68, 64, 50, 33, 94, 92, 60, 64**]

Grammar

```
start:=            <expr>

expr:=             <pre_op> | <pre_op_non_term> | <pre_op> <op> <expr> | <pre_op_non_term> <op> <expr> |
                   <input> <op> <expr> | <non_terminal> <op> <non_terminal> |
                   (<non_terminal> <op> <non_terminal>) | (<input> <op> <expr>) | (<pre_op> <op> <expr>) |
                   (<pre_op_non_term> <op> <expr>)

op:=               + | / | * | -

pre_op_non_term:=  <sin(non_terminal)> | <cos(non_terminal)> | <tan(non_terminal)> | <abs(non_terminal)> |
                   <min(input, non_terminal)> | <max(input, non_terminal)> | <max(non_terminal)> |
                   <exp(input, non_terminal)> | <sum(non_terminal)> | <tanh(non_terminal)> |
                   <square(non_terminal)> | <sqrt(non_terminal)> | negative(non_terminal)

pre_op:=           <sin(input)> | <cos(input)> | <tan(input)> | <abs(input)> | <min(input, var)> |
                   <max(input)> | <max(input, var)> | <exp(input, var)> | <sum(input)> |  <tanh(input)> |
                   <square(input)> | <sqrt(input)> | negative(input)

non_terminal:=     <var> | <pre_op>

input:=            tensor

var:=              0.1 | 1.0 | 2.0
```

Mapping	**Phenotype**
	<start>
93 % 1 = 0	<expr>
36 % 10 = 6	(<non_terminal> <op> <non_terminal>)
86 % 2 = 0	(var op non_terminal)
0 % 3 = 0	(0.1 op non_terminal)
92 % 4 = 0	(0.1 + non_terminal)
43 % 2 = 1	(0.1 + pre_op)
83 % 13 = 5	(0.1 + max(input,var))
88 % 1 = 0	(0.1+ max(tensor,var))
61 % 3 = 1	**(0.1+max(tensor,1.0))**

Phenotype

(0.1 + max(tensor, 1.0))

Activation functions created

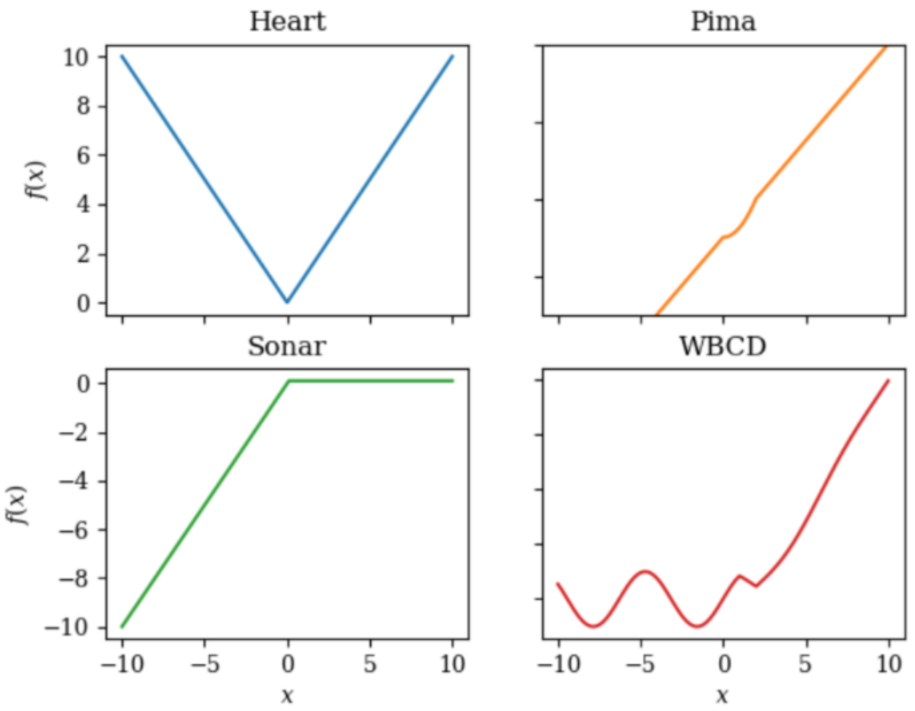


Fig. 2: Activation functions produced from the phenotype found in Table V for each binary classification dataset.

Dataset	Best Phenotype	Training Accuracy	MAE	RMSE	F1-score
Heart	$\lvert x \rvert$	0.686	0.129	0.359	0.882
Pima	$min(x, x^2) - 3.0$	0.663	0.234	0.483	0.750
Sonar	$min(x, 0.1)$	0.776	0.048	0.218	0.957
WBCD	$max(x, 2.0) + \frac{sin(x)}{max(x, 1.0)}$	0.881	0.017	0.131	0.974

TABLE V: Binary classification testing results for our approach using custom activation functions for the input and hidden layers of the neural network averaged over ten runs. The best phenotype produced from all ten evolutionary runs is also shown, where x is the input vector for each activation function.

Improvements on ReLU

Dataset	F1–score with ReLU	F1–score with custom activation function	Percentage Improvement
Heart	0.800	0.882	+10.24%
Pima	0.656	0.750	+14.32%
Sonar	0.933	0.957	+2.57%
WBCD	0.941	0.974	+3.29%

TABLE VI: Comparative table showing the best F1-score testing results on networks with ReLU as its activation function for input and hidden layer neurons vs using an evolved activation function in those neurons.

Parameter description	Parameter
Hidden layers	3
Nodes per hidden layer	8
Optimiser	Adam
Maximum number of epochs	50
Kernel initialiser	Glorot/Xavier uniform [25]
Batch size	8
Output layer activation function	Sigmoid

Architecture used for all NN's

Improvements over generations

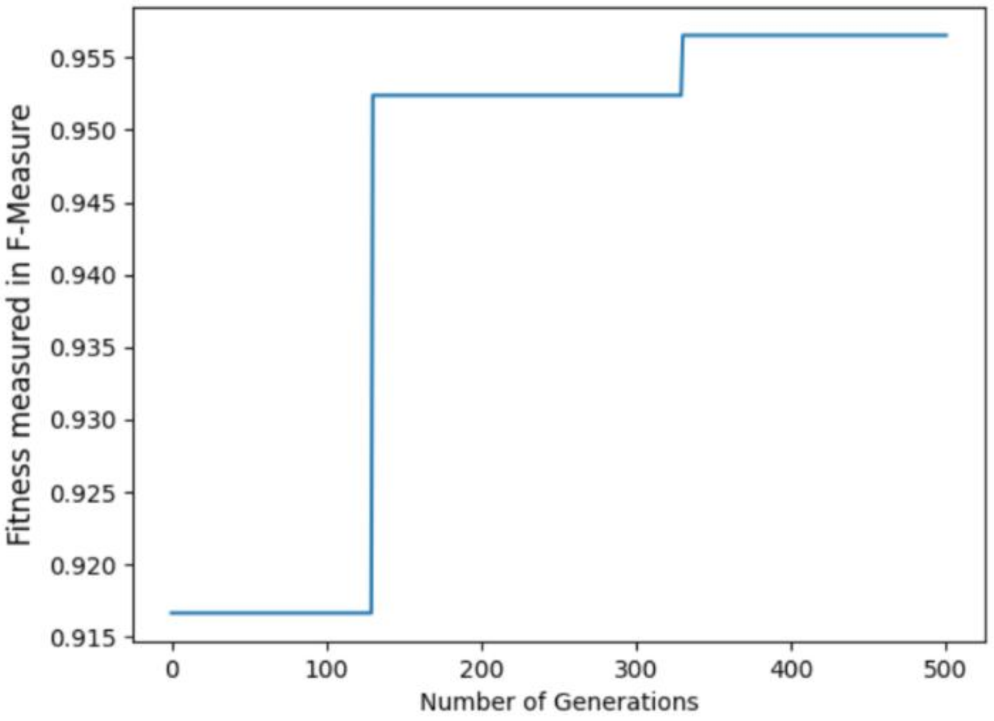
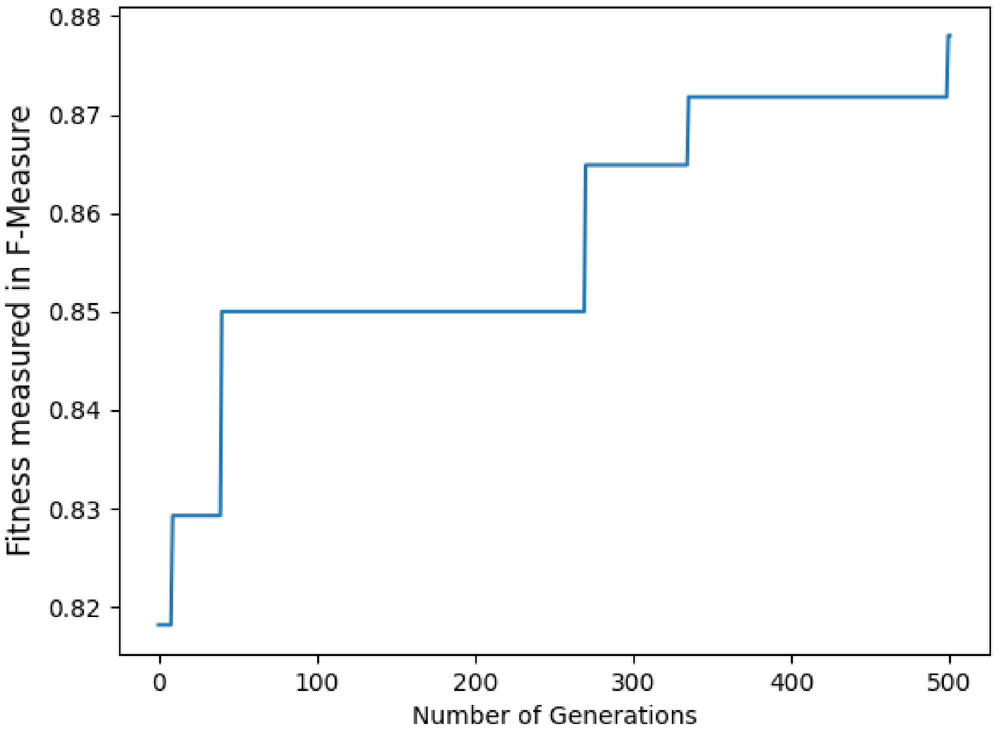


Fig. 3: The evolutionary process, showing the number of improvements over a run for the Sonar dataset.



Heart dataset

AIMLAC
UKRI CENTRE FOR DOCTORAL TRAINING IN ARTIFICIAL INTELLIGENCE, MACHINE LEARNING AND ADVANCED COMPUTING

PRIFYSGOL BANGOR UNIVERSITY

Summary

We discussed what evolutionary algorithms are and some of their uses, namely with GA, GP and GE.

We briefly touched upon neural networks and the difficulty users may face when deciding their architecture.

Lastly, I mentioned our ongoing program that has the ability to create mathematical activation functions for the input layer and hidden layers that outperforms ReLU on all datasets tested so far.

We hypothesise that this method would be most efficient when used in conjunction with a NEAT system whenever a new dataset is to be tested, in order to find the optimum neural network architecture for each parameter of the network.

AIMLAC
UKRI CENTRE FOR DOCTORAL TRAINING IN ARTIFICIAL INTELLIGENCE, MACHINE LEARNING AND ADVANCED COMPUTING

PRIFYSGOL BANGOR UNIVERSITY

Thank you!

Ben Winter – Bangor University

email: eeu60d@bangor.ac.uk

UKRI CENTRE FOR DOCTORAL TRAINING IN ARTIFICIAL INTELLIGENCE, MACHINE LEARNING AND ADVANCED COMPUTING

PRIFYSGOL BANGOR UNIVERSITY