Third Project - Neuroevolution using NEAT/HyperNEAT

Iker Garcia and Gonzalo Piérola

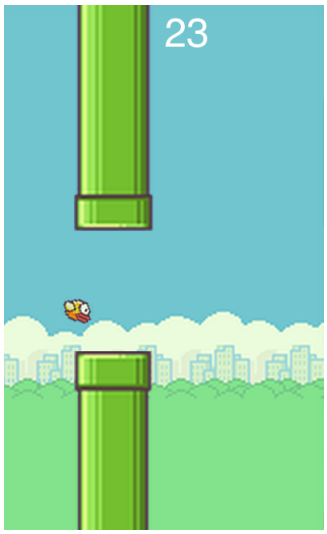
Abstract

This report introduces a project about how to use neuroevolutionary approaches to generate neuronal networks architecture. For this task it is used the NEAT library which evolves the neuronal network using a genetic algorithm. This neuronal network is used to learn how to play the flappy bird game. Finally the network results are compared with a second neuronal network with predefined architecture which is also trained to learn how to play the game.

1 Description of the problem

The goal of the project is to apply NEAT to evolve the network architecture of DNNs for improving their performance on a given classification task. The chosen task is the game Flappy Bird, the goal is to determine if for a given input the bird must jump or no to avoid hitting the pipes.

Flappy Bird is a 2D game where a bird must avoid crashing into pipes. There are always two pipes, one in the top of the map and one in the bottom, both are placed in the same coordinates in the X axis leaving a gap between them. The bird must go through that hole, if It is able to avoid the pipes another two pipes will appear. The heights of the pipes is random so the gap will be at a different heights every time that two new pipes are generated. If the bird crash into the wall or if It touches the top or the bottom of the map it dies, to avoid this to happen the bird can perform two actions. The first one is flap, when the bird flaps It will move upwards a short distance. The second one is do nothing, if the bird does not flap It will descend (the more time It does not flap, the faster It will descend).



Flappy Bird Game

2 Description of our approach

The project is divided into 2 parts: Solve the problem with a DNN and then Solve the same problem using NEAT to seach for an architecture that improve the performance of the original network.

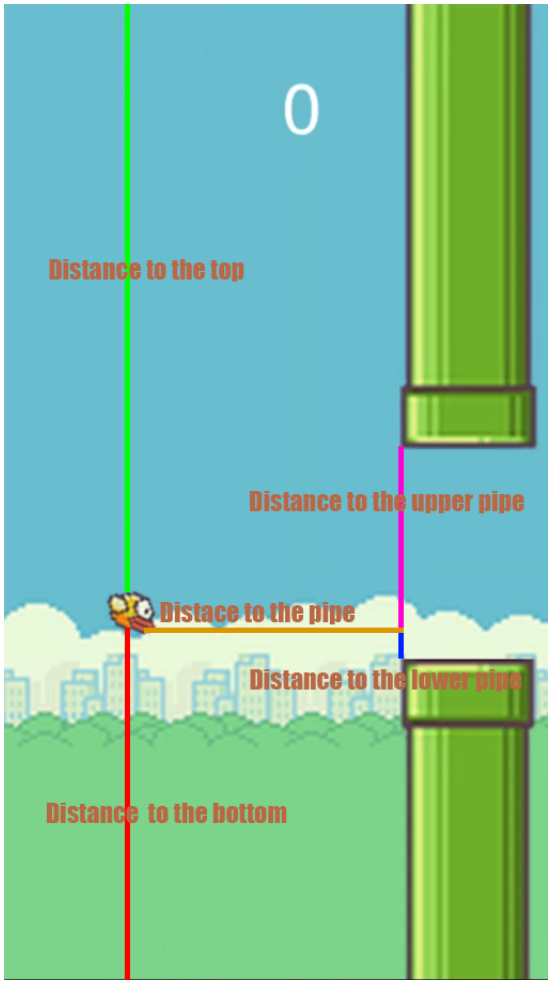
2.1 Solving the problem with a DNN

We choose a Multilayer Perceptron to solve the problem because it fits very well with the problem.

2.1.1 Input

The input of the MLP will be a vector that contains 5 elements.

- Distance in the X axis from the bird to the next pipe.
- Distance in the Y axis form the bird to the lowest point of the pipe in the top.
- Distance in the Y axis from the bird to the highest point of the pipe in the top
- Distance in the Y axis from the bird to the top of the map.
- Distance in the Y axis from the bird to the bottom of the map.



Input of the model

2.1.2 Output

The MLP has 2 output units, one represents that the bird must jump and the second one represents that the bird must do nothing. It is possible to represent the output with a single unit and If it activates th bird flaps and if It does not activate the bird does not flap, but we find easier to the implement the output with 2 neurons.

2.1.3 Training data

We have generated a dataset that contains approximately 60.000 inputs with the correct action to perform using a bird that can play the game without crashing. The dataset is contained in the file "flapyData.csv" and It is available on github [1] We use 80% of the dataset for training and 20 for testing.

2.1.4 MLP architecture

We implemented the MLP as as follows:

- Layers: 5 (input), 10, 4, 2 (output).
- Activation function: Relu
- Max. Iterations: 1000
- Tol. 0.001

2.1.5 Results

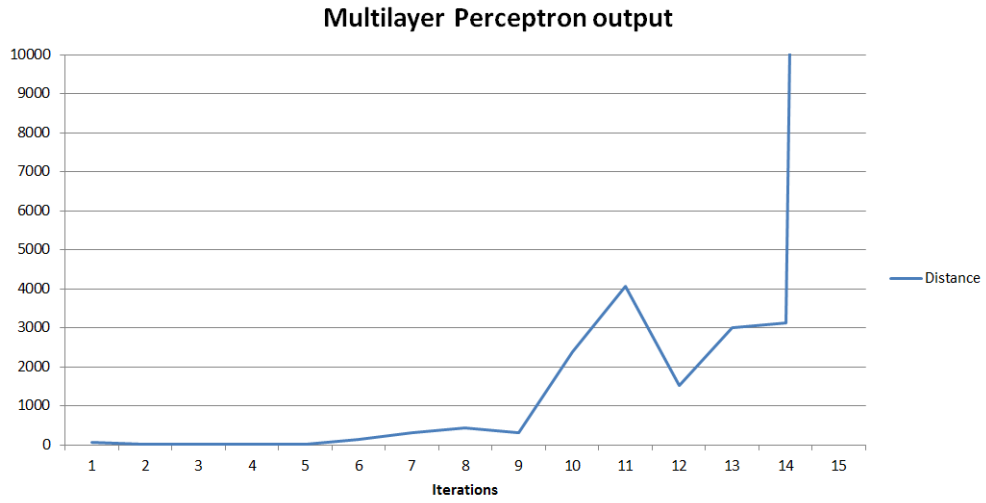
After approximately 30 iterations the MLP achieves a really high accuracy, It is almost perfect (We trained with 80% of the dataset and we tested with the remaining 20%.)

MLP predictions:

	precision	recall	f1-score	support
0	1.00	0.97	0.98	758
1	1.00	1.00	1.00	11413
avg / total	1.00	1.00	1.00	12171

Results of the MLP

This means that the bird can play the game for virtually infinite time.



Number of Iterations vs Distance

2.2 Solving the problem with NEAT

2.2.1 Input

As in the previous implementation, the input data is an array containing 5 elements, which gives information of the bird and the next walls position. The elements of this vector are the sames that the ones in the MLP solution.

2.2.2 Output

As in the MLP we have two output neurons, one represents that the bird must flap the other represents that the bird must do nothing.

2.2.3 Fitness function

The fitness of each genome is the distance that the bird has been able to fly without crashing. So to evaluate each genome we play using the neural network that It produces until the bird crashes or until It reaches a distance longer than 110.000 and we measure the distance It has arrived.

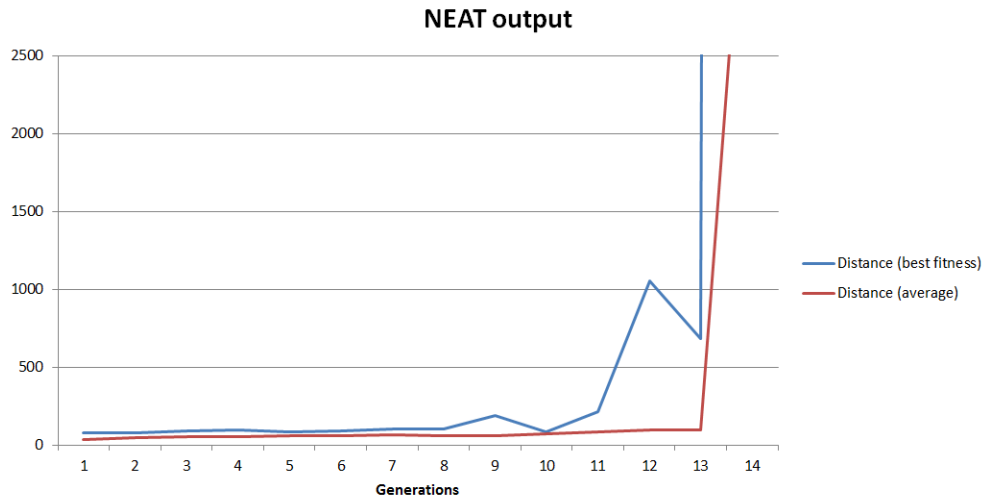
2.2.4 NEAT

We used NEAT to generate the network and Its weights. All the configuration is available in the "FlapyBirdNEAT" file. But this are some of the critical parameters:

- Fitness criterion: max
- Fitness threshold: 100.000
- Population size: 50
- Activation default: sigmoid
- weight mutate rate: 0.8

2.2.5 Results

As we can see here our model is able to learn and improve with generations until It reaches the specified threshold.



Number of Iterations vs Distance

As with the MLP, the best genome is able to achieve a infinite distance without crashing.

3 Implementation

All the project has been implemented in Python 3. For the Flappy Bird implementation we have used mainly the pygame library, but we have also needed sys and random libraries.

We used a very simple Flappy Bird implementation developed by GameDevLapse [2]

For the first part of the implementation we have used the NEAT library to generate the neuronal network architecture.

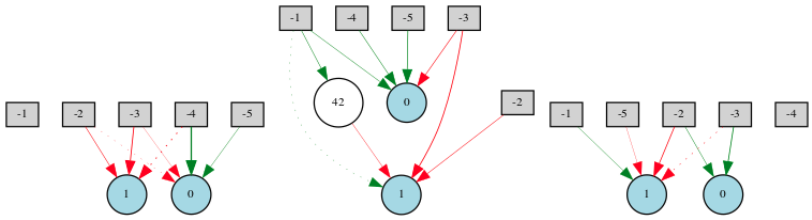
In the second part of the implementation we have used pandas and numpy libraries, first to generate the data set and then to read it. Then we have used the sklearn train_split_tests to divide the data set in two groups one for training and other for test. Then we have used the MLPClassifier also from sklearn to define the multilayer perceptron. Finally we have also used metrics from sklearn to evaluate the classifier results with the test data.

All the implementation is written and explained in this jupyter notebook: `Flappy.ipynb`

4 Results

In this section we will discuss if the neuroevolution approach has improve the results of the MLP aproach. As we have seen in the previous sections, both approaches have been able to learn how to play Flappy Bird, both are able to play for infinite distance without crashing, or at least the time that they are able to fly without crashing is bigger than our patience. So with this results we could come to the conclusion that both approaches are equally good, nevertheless if we analyze our MLP and the network generated by NEAT we can find big differences.

The MLP is a network with two layers of hidden units with 10 and 4 neurons. NEAT produces different network every time that we use it, but here are three examples:



As we can see NEAT is able to produce networks that produce similar results to the MLP but much less complex. So, in this project we have not been able to improve the accuracy of the MLP because It was almost perfect, but we have been able to use NEAT to reduce the complexity of our model. We went from 20 neurons and 98 weights in the MLP to 7-8 neurons and 6-10 weights, with is a huge complexity reduction.

5 Conclusions

In this project we have probe that NEAT can improve the performance of a DNN. We found that NEAT can be useful to reduce the complexity of our models. NEAT can helps to the reduce the number of hidden layers of our models but It can also reduce the number of connections between neurons. Analyzing the NNs generated by NEAT we found some interesting things, some of the input units does not have any connection, and we can see that the connection between some input neurons and the output is stronger than the connection between other inputs and the output. This means that NEAT can also be used as a dimensionality reduction or feature selection algorithm. After a little research we found that there are many projects that have explore the usage of HyperNeat as feature learner [3] [4].

We can conclude that the NEAT is a efficient method for generate neuronal networks that learns how to play video games, at least the simple ones like the Flappy Bird, for example, there are other implementations with this library that learn to play some Mario Bros levels[5]. This is because the

way NEAT works, evolving the network in order to increase the value of a fitness function, this is a unsupervised learning model, in which the network is developed and trained without any data from outside. So this model is very useful in this type of problems in which there is not a reference data set for training. During the project, specially at the beginning we found Neat really useful. In this project for training the MLP we needed a dataset containing a lot of possible inputs and the best possible output. However for the NEAT approach all we needed was test each genome and see how far they could fly. This can be very useful for a lot of problems were creating a dataset for training or defining a loss function for the MLP (or other DNNs) is hard or even impossible.

References

[1] Iker García (ikergarcia1996) and Gonzalo Piérola (Guamedo). Github repository: Flappy bird neat vs mlp [https://github.com/ikergarcia1996/Flappy-Bird-NEAT-vs-DEEP], 2017.

[2] Nicola Lamacchia (nicolalamacchia) Fortas Abdeldjalil (Fcmam5) and Eric G. (gileri). Youtube: Create flappy bird in python [time lapse] [https://youtu.be/h2Uhla6nLDU], 2015.

[3] Joshua Harguess Phillip Verbancsics. Generative neuroevolution for deep learning [https://arxiv.org/pdf/1312.5355.pdf], 2013.

[4] Tomáš Kocmánek. Hyperneat and novelty search for image recognition [http://kocmi.tk/photos/DiplomaThesis.pdf], 2015.

[5] SethBling. Mari/o - machine learning for video games [https://www.youtube.com/watch?v=qv6UVOQ0F44], 2015.

[6] Srdjan Susnic. Machine learning algorithm for flappy bird using neural network and genetic algorithm [https://www.askforgametask.com/tutorial/machine-learning-algorithm-flappy-bird/], 2017.