Natural Computing: Assigment 2

By Greg Brimble & Paul Georgiou

Task 1

1.1 Fitness Function

We used the following error function when training the neural network:

$$\epsilon = \frac{\sum_{i=1}^{N} \frac{(\text{output}_i - \text{label}_i)^2}{2}}{N}$$

Where:

- i is the index of a particular data point in the training data set,
- N is the total number of data points in the training data set,
- output_i is the calculated value (between 1 and -1) of the particular data point with index, i, and
- label_i is the expected value (either 1 or -1) of the particular data point with index, i.

Although we simply used this error function as a target to minimize, it could, of course, be converted to a fitness function which one would instead try to maximize, by simply using the following:

$$f = 1 - \epsilon$$

1.2 Search Space

We defined our neural network as having:

- four inputs:
  - x
  - y
  - $\sin(x)$
  - $\sin(y)$
- a single hidden layer with six nodes,
- and a single output node.

With each having a bias (11), and the final two layers having a combined 30 weights ($1 * 6 + 6 * 4 = 30$) this creates a search space of 41 dimensions ($11 + 30 = 41$) for our PSO implementation to solve.

1.3 Results

Our adapted Tensorflow Playground (original) can be found here: https://gregbrimble.github.io/playground. The configuration defaults to using PSO to train the neural network, with 25 particles, $\omega = 0.5, \alpha_1 = 2, \text{and} \alpha_2 = 2$.

With this configuration, Figure 1 renders a sample output after 350 iterations.
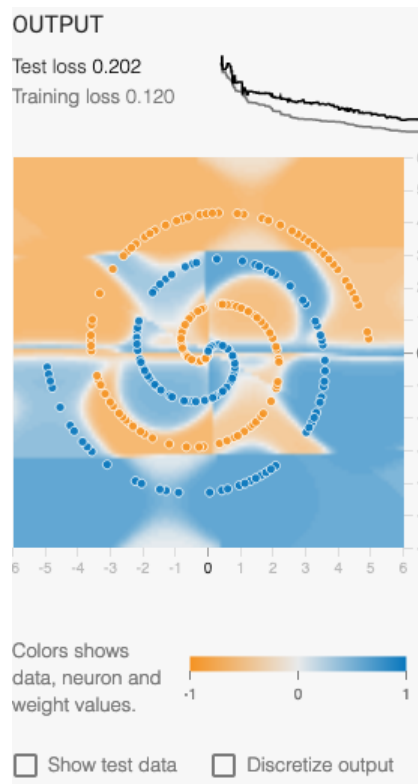
Figure 1: PSO after 350 iterations

Figure 2 clearly demonstrates that no further significant improvements are made to the model beyond approximately 350 iterations.
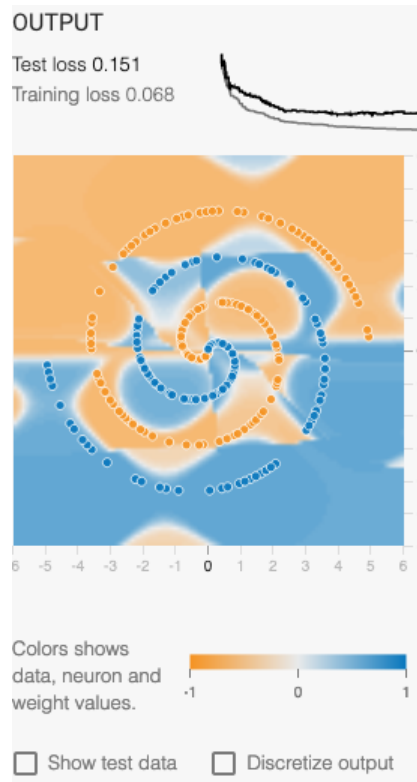
Figure 2: PSO after 1000 iterations

1.4 Comparison against Linear Inputs

Restricting the network to only linear inputs (x and y) significantly curtails the model's ability to fit to the spiral training data. As seen in Figure 3, the minimum training loss is quickly reached, with very little improvement seen with future iterations. The model itself is functionally very simple: practically a 50/50 divide along the spirals' centers.
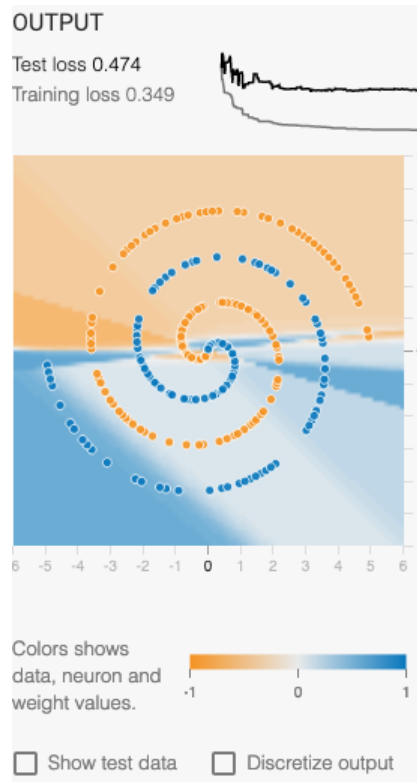
Figure 3: PSO trained with linear features

1.5 Effect of PSO Parameters

Using a higher ω value results in stuttered learning (best seen in the loss sparkline) and a generally poorer model, as demonstrated in Figure 4.
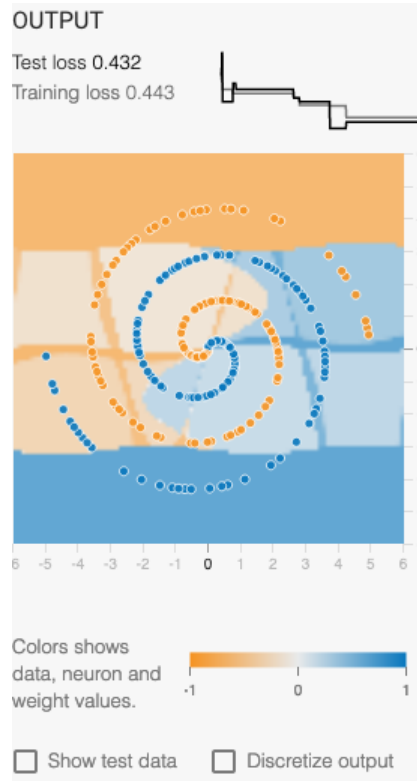
Figure 4: PSO with a higher $\omega = 0.8$

Significantly changing α_1 and α_2 seriously hampered the ability for the model to train. Figure 5 and 6 demonstrate using $\alpha_1 = \alpha_2 = 3$ and $\alpha_1 = \alpha_2 = 1$ respectively.
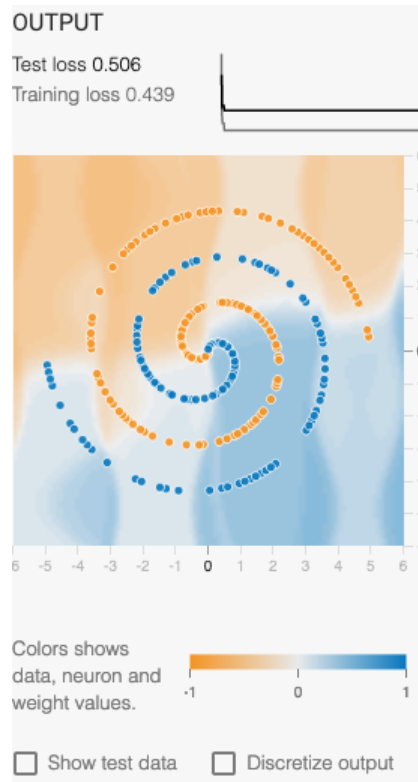
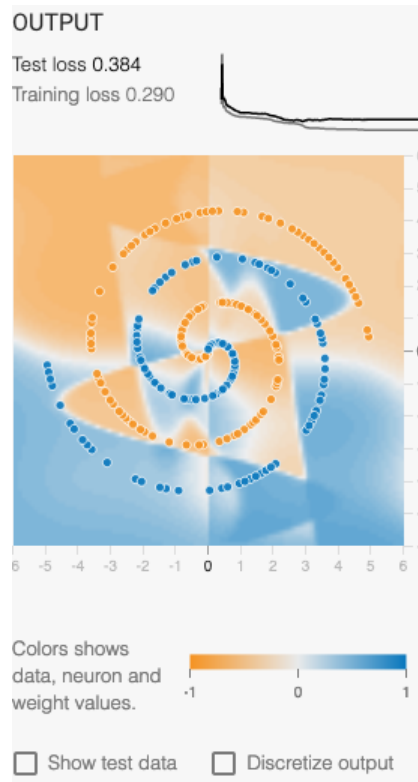Figure 5: PSO with $\alpha_1 = \alpha_2 = 3$

Figure 6: PSO with $\alpha_1 = \alpha_2 = 1$

And finally, increasing the number of particles did improve the training and test loss, but begins to show signs of overfitting to the data (Figure 7).
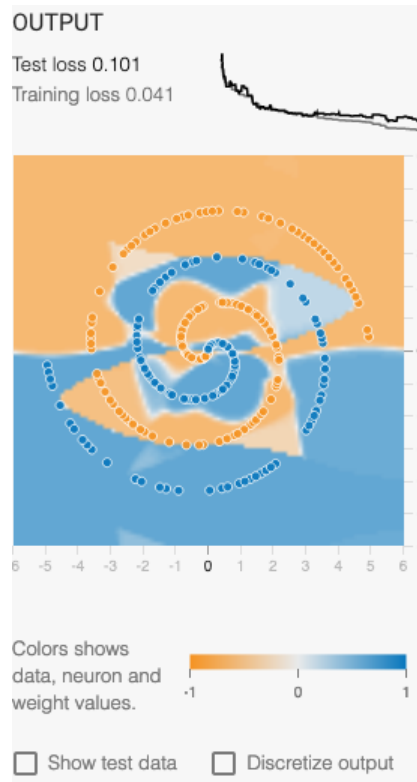
Figure 7: PSO with 100 particles

Task 2

2.1 Evolving the Network Structure

To create a neural network which could be evolved with GP, we first set the maximum number of hidden layers as 8. And

2.2 Further Evolutions

2.3 Operators and Parameters of GA and Their Performance

2.4 Controlling Complexity

Task 3

3.1 Additional Node Functions

3.2 Operators and Parameters of GP and Their Performance

3.3 Comparison with GA

3.4 Comparison with Cartesian Genetic Programs (CGPs)

3.5 Future Work