Investigating the Multi Layer Perceptron

This paper describes the implementation of a multi-layer perceptron and gives the results of training it on some test problems.

The key feature that I wanted to include in the MLP was flexibility on its structure, so the code I produced allows the user to define:

- Number of layers.
- Number of inputs and number of outputs for each layer.
- Activation functions.

Training approach

Layer structure

When choosing the initial weights for a layer with i inputs and j outputs I generated j random one-dimensional vectors of magnitude 1, each containing i elements, and then combined them into the required i by j matrix for that layer.

Training data

For each train, the data is randomly split into two equal groups – training data and validation data. Training data is then used to update weights during the training process, and the validation data is used in the stopping criteria.

Activation functions

For all MLPs used I applied the sigmoid logistic function.

Stopping criteria

I found the most challenging aspect of the development to be the stopping criteria for the training. The stopping criteria I ended up with were:

- All training data points and all validation data points have been correctly classified; OR
- The validation data error has not reduced in the last X epochs (set to 100 as default); OR
- The number of epochs exceeds a maximum Z (set as 20,000 as default).

To test the training and visualize the results I also created classes for generating data points and mapping boundaries between classes (in two dimensions only).

Approach to testing

To make sure the MLP was working as I expected I ran some tests using 2-dimensional data so that the results could easily be visualised. My primary interest was to see how the use of multiple layers (as opposed to the single layer of a simple perceptron) and appropriate layer sizes (i.e., number of inputs and outputs) made more complex classification possible.

My understanding from reading lecture notes was that:

- A single layer could only create individual linear boundaries.
- Two layers are required to combine one or more linear boundaries into a single boundary made up of different line segments.

- At least three layers are needed to create several different “shapes” as boundaries that are not connected to each other.
- The size of the hidden layers in an MLP of 3+ layers governs how many different line segments are available to be used in a complex boundary.

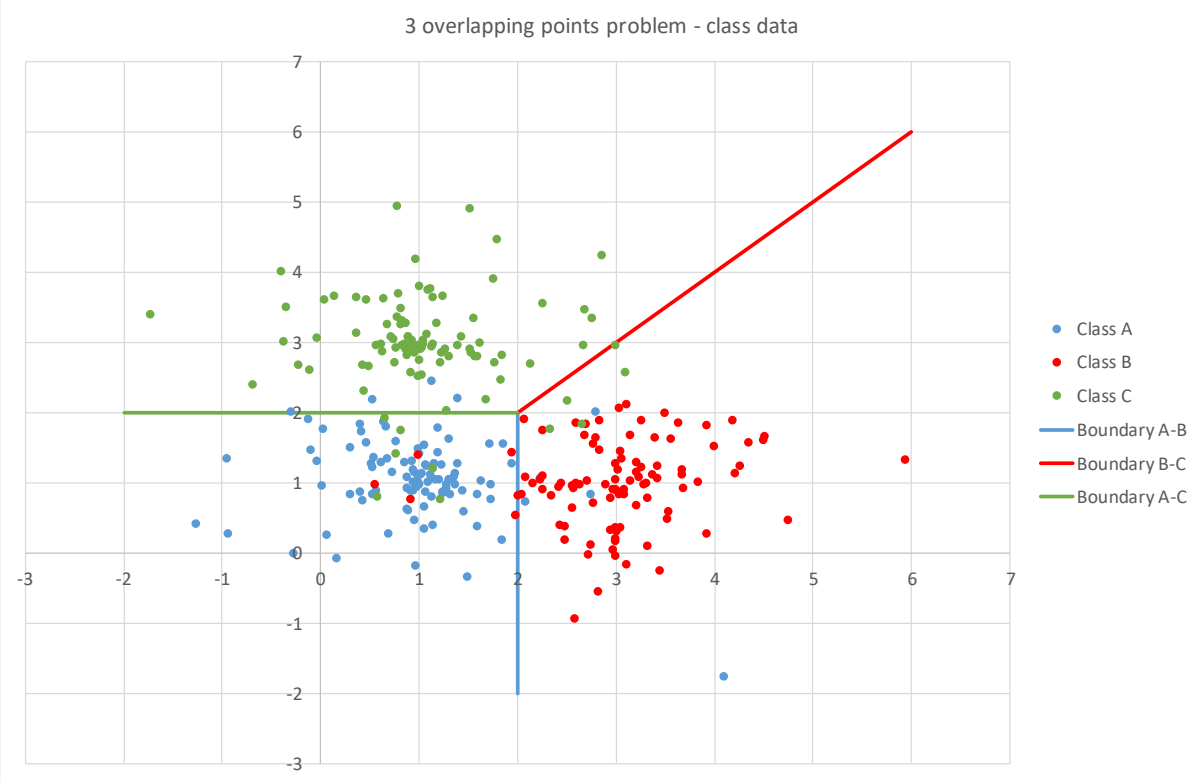
To test out my understanding, I created two problems that a single layer perceptron could not cope with:

- 3 overlapping “points” – where data from different classes overlap with each other.
- 2 ring problem – where data is in a “shape” that is separable, but not linearly separable.

These two scenarios are now described in turn.

3 overlapping points scenario

This scenario consisted of three separate classes (A, B and C) where the data overlap with each other. The data for each class was created by generating random data points with Gaussian distribution centred on the three mean points of (1, 1), (3, 1) and (1, 3). This class data is illustrated below.



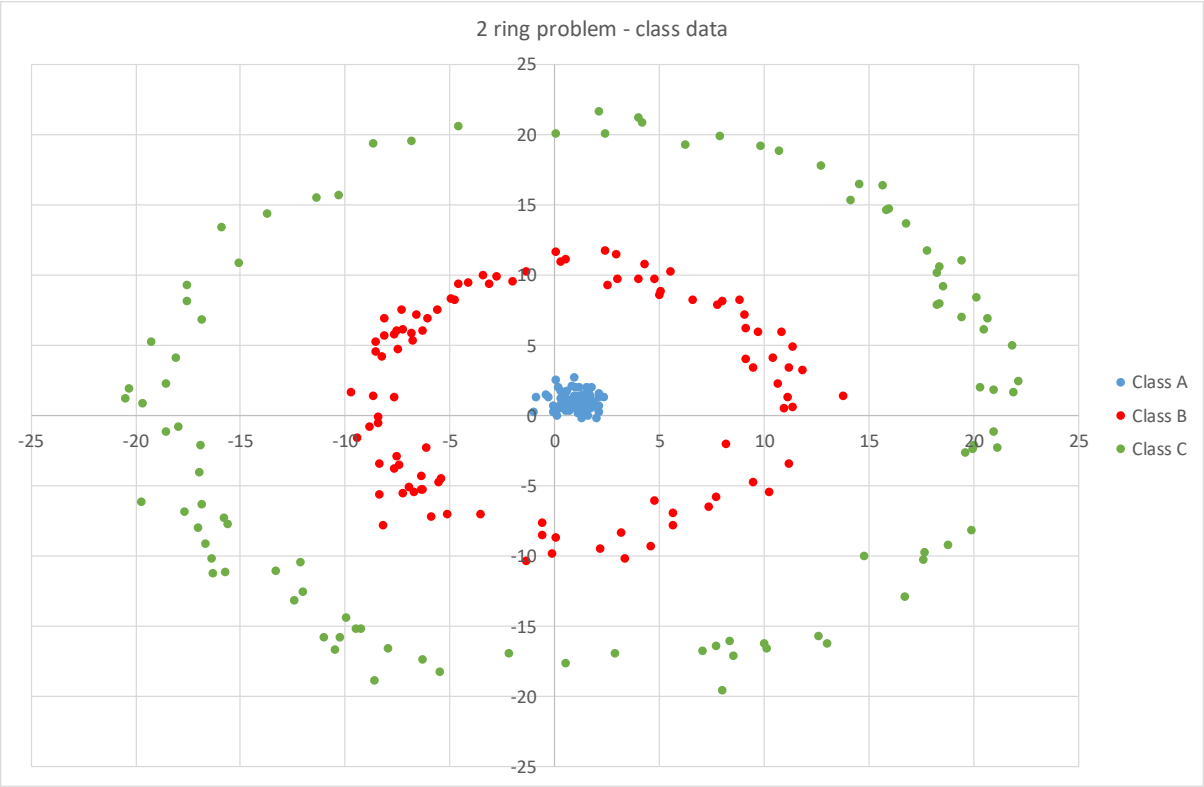
The “correct” classification answer is shown on the chart as the three boundaries. It is almost certainly impossible for a learner to generate these boundaries as there are insufficient data points for the random “bias” introduced by the Gaussian distribution to be averaged out. Nevertheless, a learner (including an MLP) should be capable of generating boundaries reasonably close to the correct answer.

The purpose of using this data was to demonstrate that increasing a multi-layer perceptron’s number and size of layers would have little useful effect on a problem where the classes are not separable. Although an MLP could get to a point where the data is correctly classified, clearly this would just lead to overfitting.

2 ring scenario

This scenario consisted of three separate classes (A, B and C) where the data do not overlap. Class A data points were generated using a Gaussian distribution centred on the point (1,1). Classes B and C were generated as rings of data points, also centred on (1,1) but placed randomly at radii with a Gaussian distribution with means of 5 (class B) and 10 (class C).

(The random distribution of data is not necessary to create the desired scenarios of rings that do not overlap. I just envisaged that the capability to generate such random data would be useful in the future and so with it available I used it for this purpose.)



Tests and results

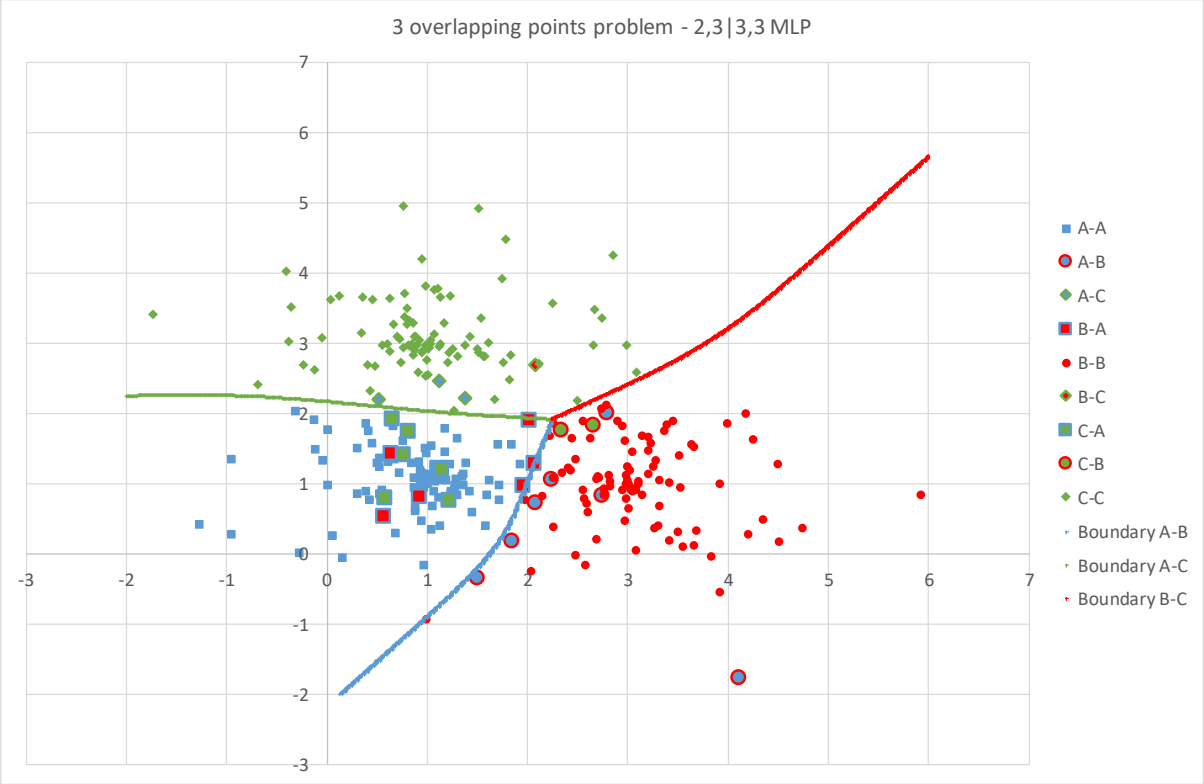
Tests and results for each of the two scenarios are given below.

3 overlapping points scenario

The “correct” answer for this scenario should not correctly classify all data. Since the class data overlap, correctly classifying all data is overfitting. A single layer MLP would give a reasonably good outcome, but a two-layer MLP might possibly (though only by chance) give a better outcome as it could generate boundaries that “bend”. The chart below shows the results for a 2-layer MLP. This MLP is described as a 2,3|3,3 MLP, which signifies that:

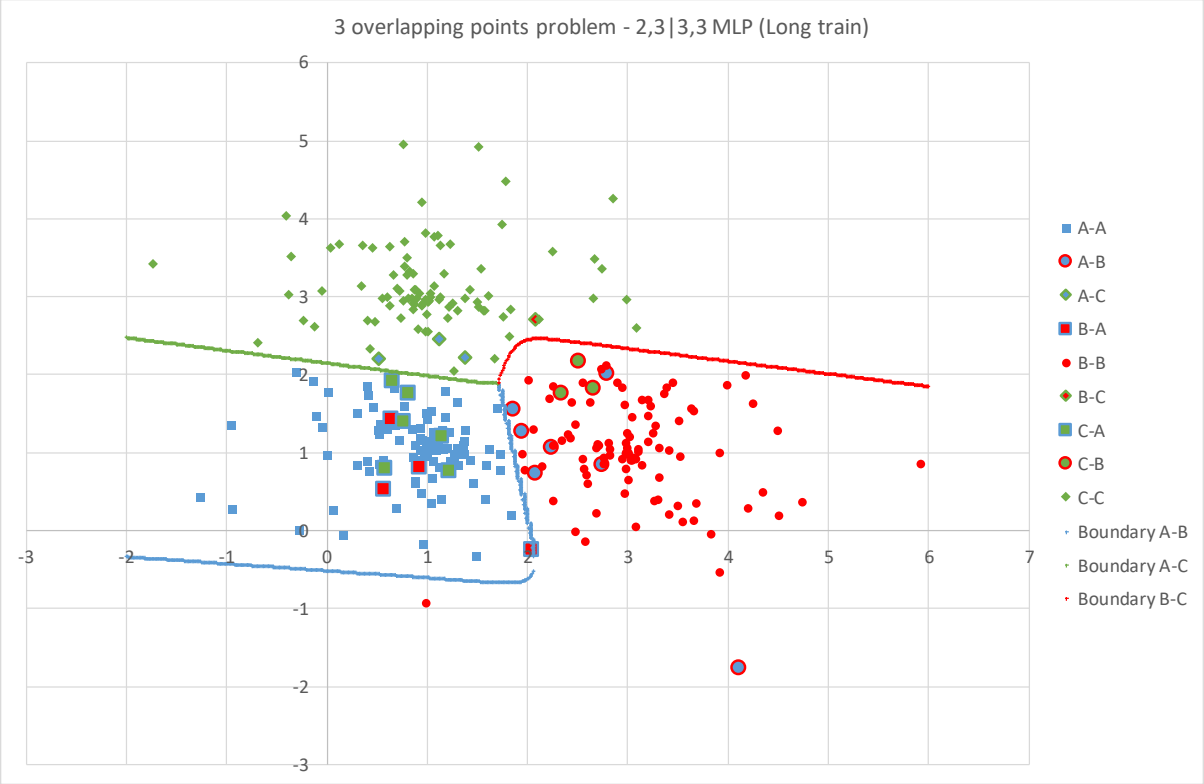
- The first layer (2,3) has 2 inputs and 3 outputs.
- The second layer (3,3) has 3 inputs and 3 outputs.

In addition, both layers have a bias. (This notation will be used throughout the paper to describe an MLP’s layer structure. All layers have a bias.)



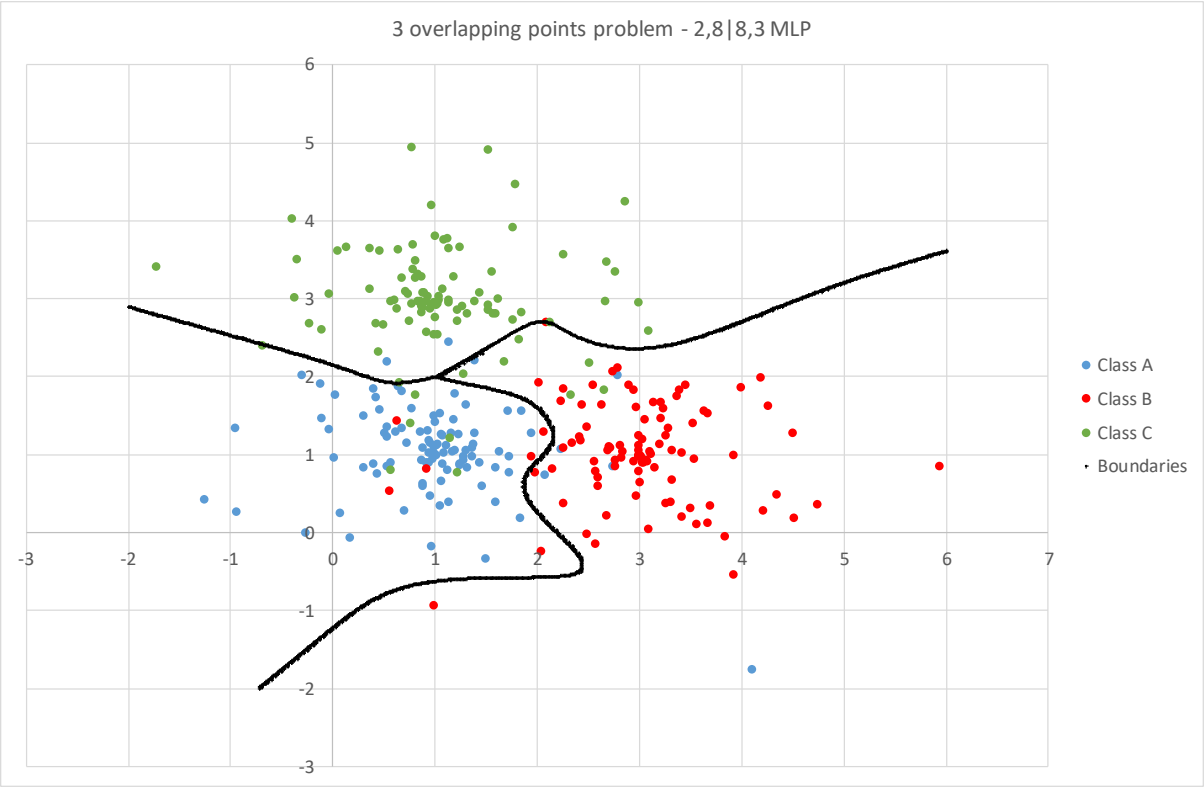
As expected, the MLP has learnt three boundaries, and because the MLP has two layers, each boundary is made up of separate straight line segments. Although the boundaries on the chart might appear to be smoothly curved rather than connected straight segments, this is a result of the method used to map the boundaries, which is approximate to certain tolerances.

I then decided to loosen the stopping criteria to see how “creative” the MLP could get in terms of creating complex boundaries. I removed the criterion that the train should stop if the validation data error did not reduce in the last X epochs. This meant the two remaining criteria were to stop if a) all points were correctly classified, or b) epochs reached the maximum of 20,000. I kept the limit of 20,000 only so that a train did not take absurdly long. The result is shown below.



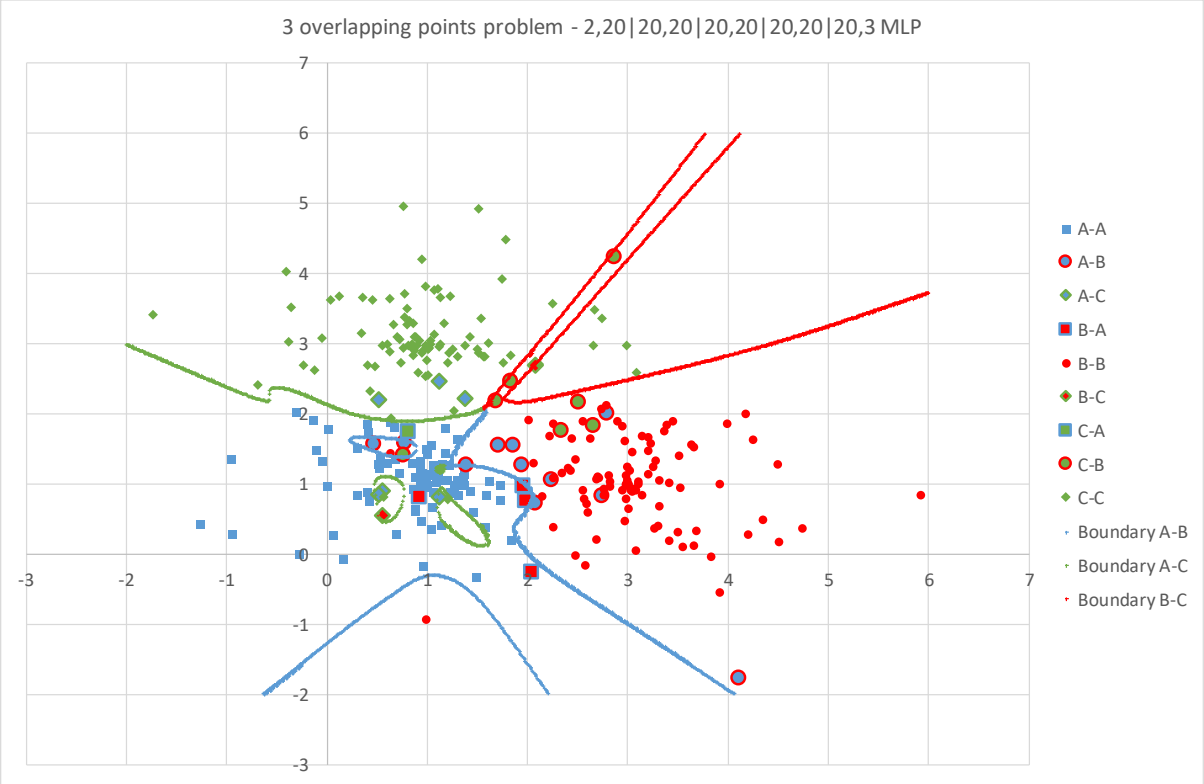
The result is more extreme “bends” in two of the boundaries. This is not a “better” result as if anything it is simply overfitting more than the previous train. However, it does appear to illustrate the limitation of this MLP’s structure. This MLP appears limited to boundaries made up of combinations of only 3 possible line segments, presumably because there are only 3 outputs to the first layer. Both the red and blue boundary contain a section that is parallel to the green boundary, so this represents one of the possible line segments. Each of the red and blue boundaries also contain another line segment, bring the total to 3.

Increasing the “hidden dimensions” to a 2,8|8,3 MLP produces the result below.



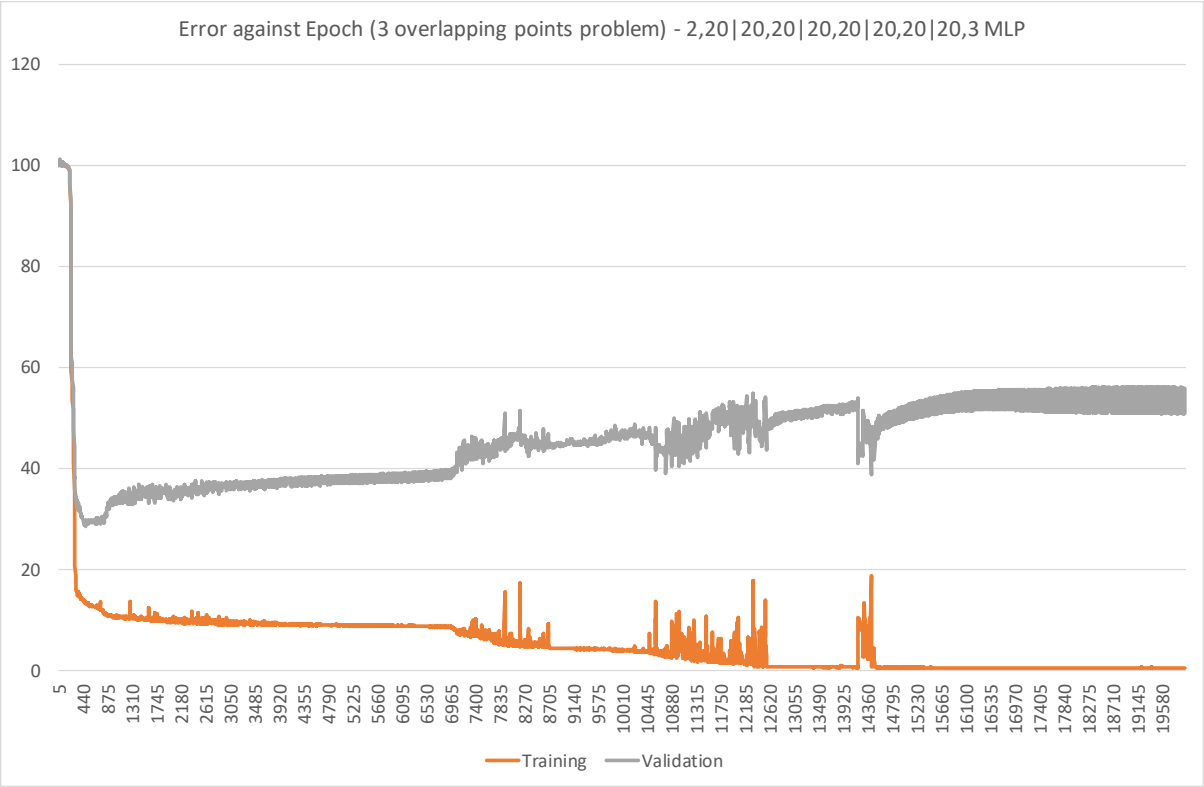
This MLP is clearly able to create more complex multi-line boundaries. It is more difficult to decide how many unique line segments are being used because of the approximations in the boundary mapping, but I believe 5 are clearly discernible, and it seems plausible that there are another three in the apparently “curved” sections, which would bring the total to 8.

To illustrate the effects of increasing the number and size of layers, I then trained a 2,20|20,20|20,20|20,20|20,20|20,3 MLP on the same problem, again using the "long train" approach, with the results shown below.



Although this is obviously not a useful training approach as it gives drastic overfitting, it does illustrate the complex boundaries possible.

The chart below illustrates the extent of the overfitting.



Had the stopping criterion been retained that checked when validation data did not reduce, training would have stopped within a few hundred epochs. Although the training data error continues to reduce after this point, the validation data error increases.

2 ring scenario

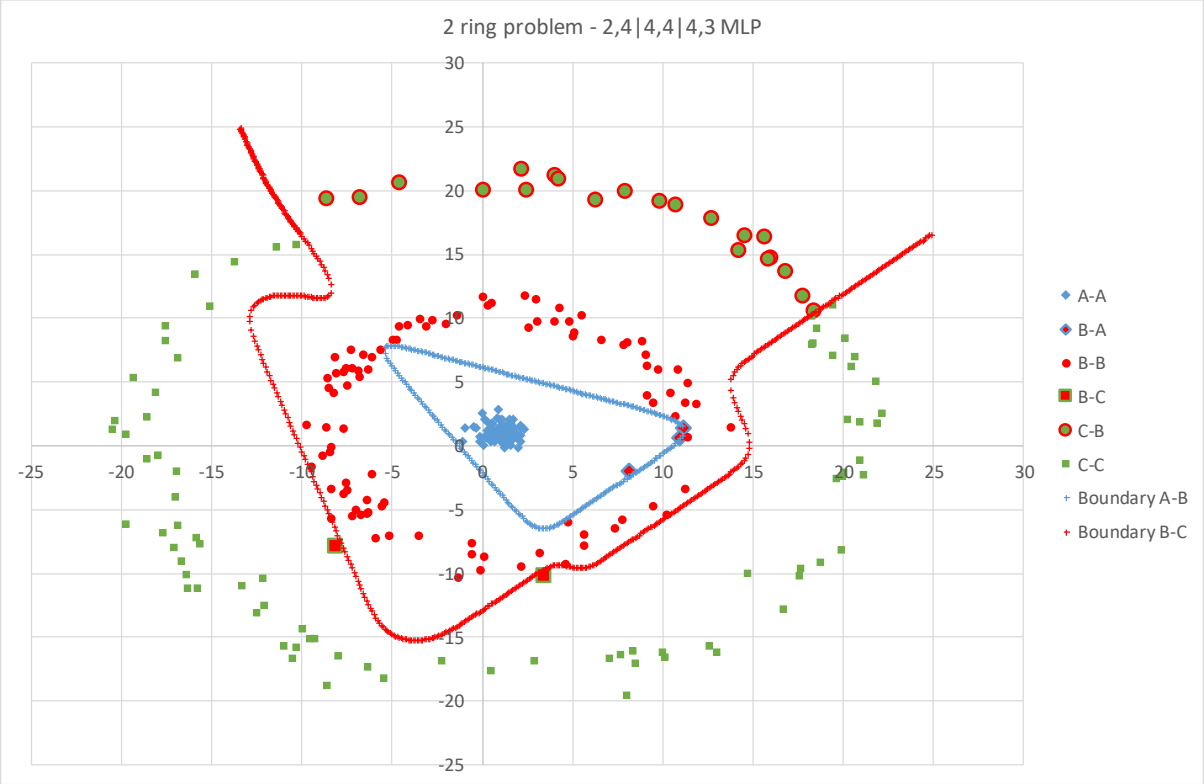
The 2 ring scenario should be solvable by a suitably constructed MLP. There are two issues to consider in the design:

- There must be 3 or more layers so that complex independent shapes can be formed for a single boundary between classes.
- There must be sufficient dimensions in the hidden layer that a ring can be made from the line segments without having to include misclassified data. In other words, although a closed shape can be formed with only three line segments (a triangle) it would be impossible to construct a triangle that remains within the gaps between the two rings of classes A and B, or of B and C.

To illustrate this, I tested the following MLPs:

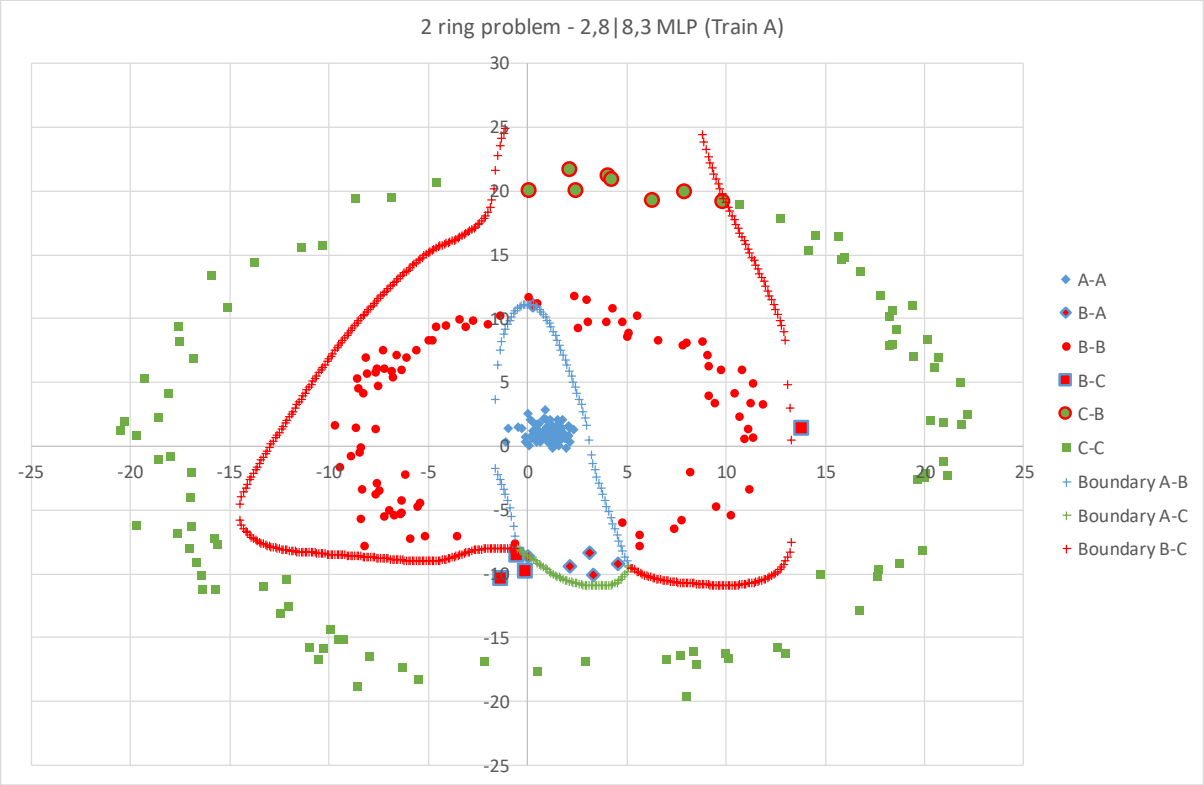
- 2,4|4,4|4,3 MLP – although this should be able to create complex shapes, I expected that with only 4 line segments available it would not be able to form closed shapes that correctly fitted the data.
- 2,8|8,3 MLP – although this would be able to use more line segments, the MLP would not be able to create independent shapes.
- 2,8|8,8|8,3 MLP – this version should create a boundary that correctly classifies all data.

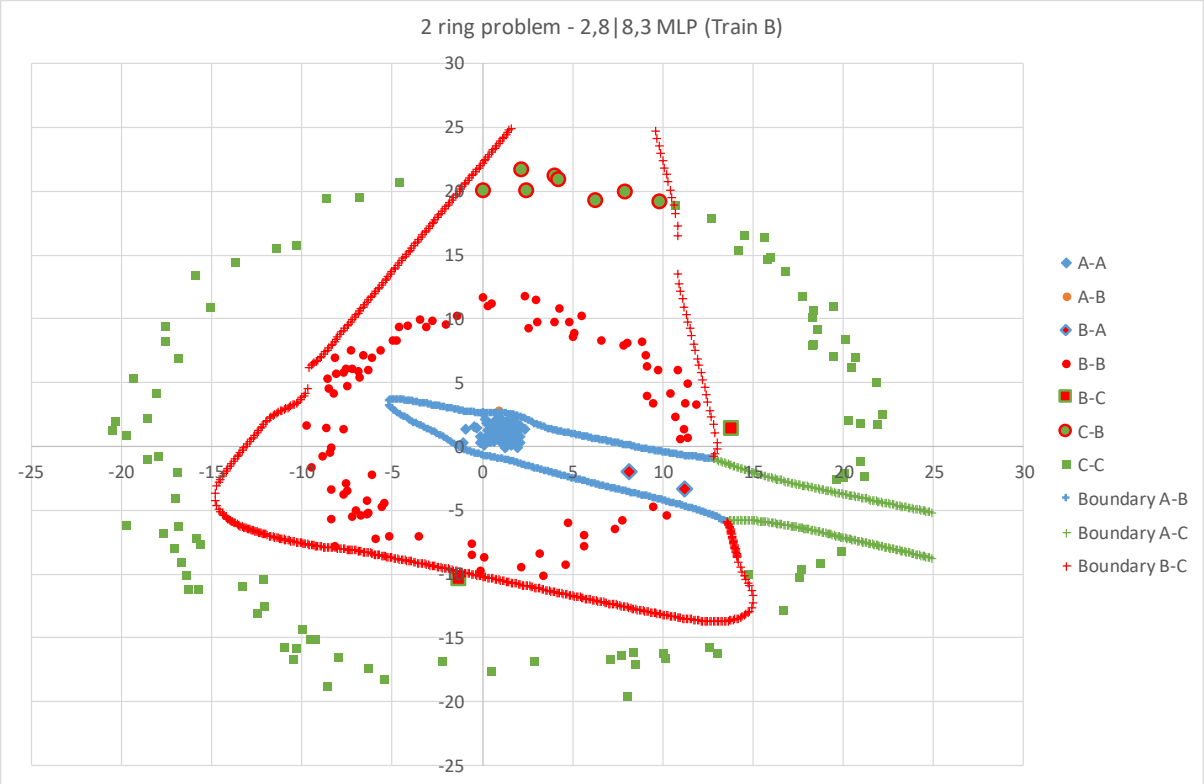
As shown below, as expected, with only 4 line segments to play with, the 2,4|4,4|4,3 MLP fails to create closed shapes that correctly classify data. Although the triangle formed is a closed shape, it still misclassifies 2 points.



It is surprising that the B-C boundary is not also a closed shape. I would have expected that lower training and validation errors would result if a line segment parallel to the top line of the blue triangle were used to form the top element of the B-C boundary. While it might not be possible to construct such a shape that correctly classified all points in B and C, it is hard to see how it could be worse than the result shown above where a 90 degree arc of the class C ring is misclassified.

Below are the results from two different trains of the 2,8|8,3 MLP. As expected, this MLP fails because it is unable to create separate shapes.





Finally, as shown below, the 2,8|8,8|8,3 MLP is able to create a “correct” solution.