Connor McCurley

EEE 6814 **Homework 1** Fall 2019

Question 1

Question 1 asked us to hand-design a ANN architecture which is capable of creating discrimination regions resembling the mask shown in the handout. Based on the design, it was determined that the discriminator could be created from a two-hidden-layer network shown in Figure 1.

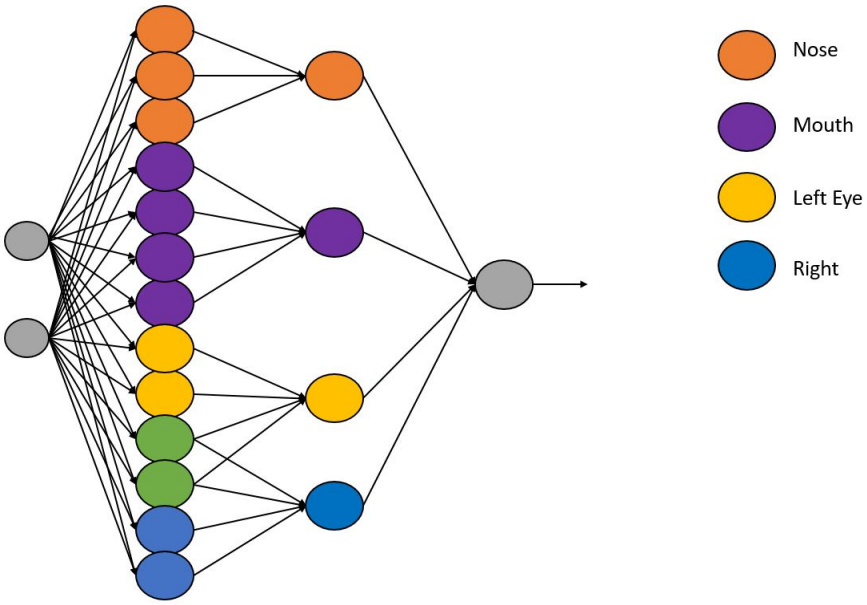


Figure 1: Architecture implemented to solve the mask problem. The network implements a two hidden layer MLP with two input units, 13 units in the first hidden layer, 4 hidden units in the second layer, and 1 output neuron. Sign functions were used for activations in the network.

This architecture defines a two hidden layer MLP with two input features, 13 hidden units in the first layer (3 for the nose, 4 for the mouth, 2 for each eye, and two shared between the eyes), 4 hidden units in the second hidden layer (to determine if a sample is contained in one of the desired discrimination regions), and a single neuron in the output. To select the parameters of the network by hand, I used the following arbitrary weight relationship defined in class:

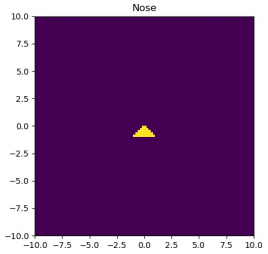
$$x_2 = -\frac{w_1}{w_2}x_1 - \frac{b}{w_2}$$

where x_1 and x_2 are inputs, w_1 and w_2 are corresponding weights, and b is a bias term. From this, I determined the weights for the units in the first hidden layer as:

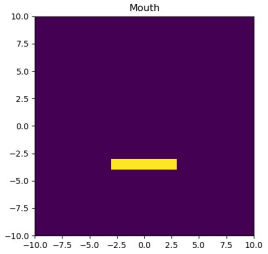
Table 1: Weights in the First Hidden Layer

Mask Region	w_1	w_2	**Bias**
Nose	0	1	1
	1	1	0
	-1	1	0
Mouth	0	1	3
	0	1	4
	1	0	3
	1	0	-3
Left Eye	1	0	5
	1	0	4
Right Eye	1	-5	3
	1	0	-4
Both Eyes	0	1	-5
	0	1	-4

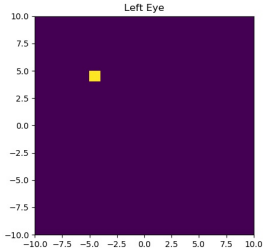
To test the correctness of my parameters, I defined a test dataset and passed it through the network. The outputs of each neuron in the second hidden layer are shown in Figure 2, where yellow represents a value of 1, and blue a value of -1.



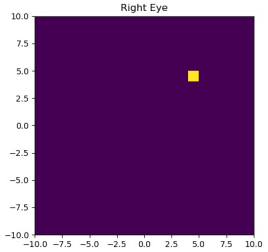
(a) Nose



(b) Mouth



(c) Left Eye



(d) Right Eye

Figure 2: Outputs of the second hidden layer in the "mask network". It can be observed that each unit in this layer determines whether a sample falls within a desired mask region.

From there, I hand-selected a few points both within and outside of each desired region, passed them through the network, and hand-plotted them in the "new parameter space". The process was then repeated as above and the weights in the second hidden layer were found to be:

Table 2: Weights in the Second Hidden Layer

Mask Region	w_1	w_2	w_3	w_4	**Bias**
Nose	2	-0.5	-0.5	0	-2
Mouth	-3	3	0.5	-1	-7
Left Eye	-3	3	0.5	-1	7
Right Eye	-3	3	-2	2	-6

When provided the same test set, this MLP architecture was able to generate the desired "mask" discrimination regions.

Question 1 also posed the question as to whether the same goal could be achieved with a single hidden layer network. I would answer that it could be done if the output layer was truncated and the outputs of the neurons currently in the second hidden layer were be used to create an output vector. An extra neuron, however, would need to be added in the output layer as a "background" discriminator.

Question 2

For question 2, I designed a single hidden layer MLP to solve the Star Problem (Figure 3), and coded backpropagation (code included) to train the network. For this architecture, I chose a single hidden layer MLP with two inputs, 4 hidden units, and a single output neuron, as shown in Figure 4.

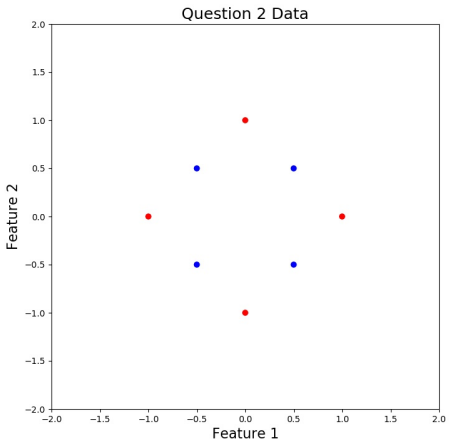


Figure 3: Training data points for the Star Problem. Class labels are denoted by color.

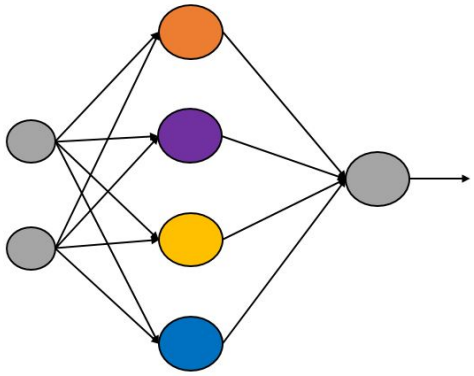


Figure 4: Architecture implemented to solve the Star Problem. The network consists of a single hidden layer MLP with two input units, 13 units in the first hidden layer, 4 hidden units in the second layer, and 1 output neuron. Sigmoid activation functions were used at each neuron in the network.

After my first round of training, I obtained the following results:

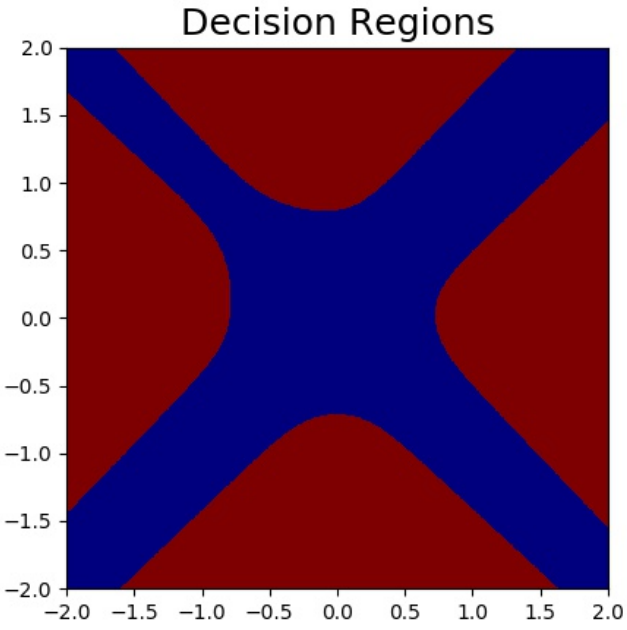


Figure 5: Decision regions learned by the single hidden layer network shown in Figure 4. The regions partially solve the Star Problem, but fail for out-of-sample data points.

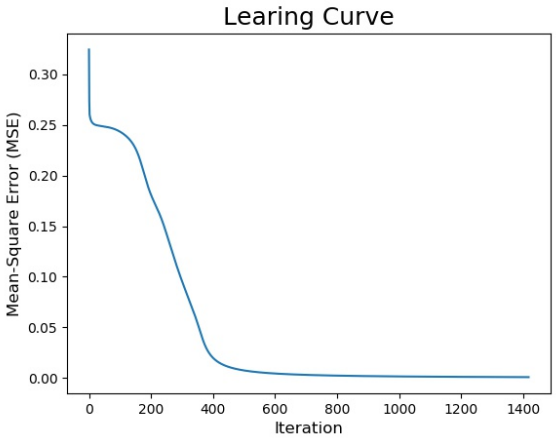


Figure 6: Learning curve for the network shown in Figure 4. This shows that the network's training error falls below 0.002 after approximately 1000 iterations of training with a learning rate of 0.8.

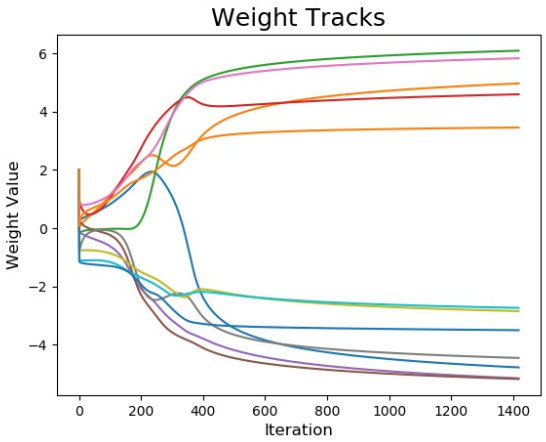


Figure 7: Weight tracks for the network shown in Figure 4. This shows that the network's weights converge after approximately 1000 iterations of training.

Both the weight tracks (which stabilized) and the MSE learning curve demonstrated that the network obtained good performance on the training data. However, when provided out-of-sample data, the model did not generalize to the desired discrimination regions. It occurred to me that, to allow the model to more-easily discover our desired discrimination regions, we could force some of the weights to 0 so that the network would learn only vertical and horizontal lines. After enforcing these constraints, the network learned the following:

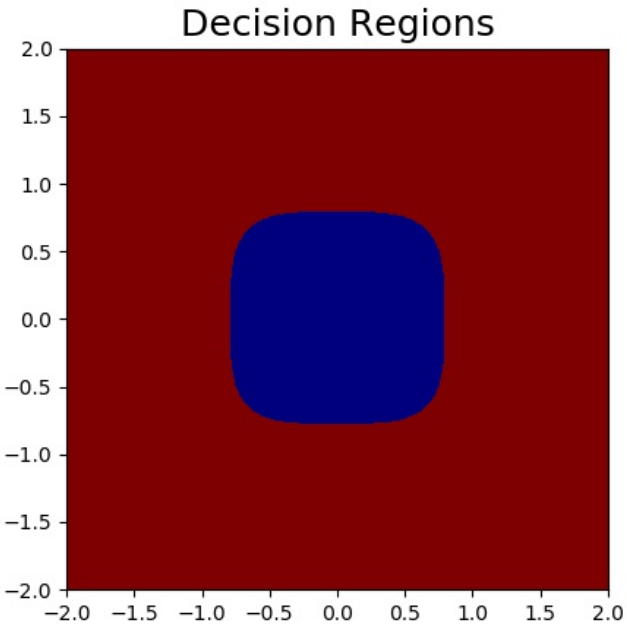


Figure 8: Decision regions learned by the single hidden layer network shown in Figure 4, with constraints to learn only horizontal and vertical decision boundaries. The regions perfectly solve the Star Problem.

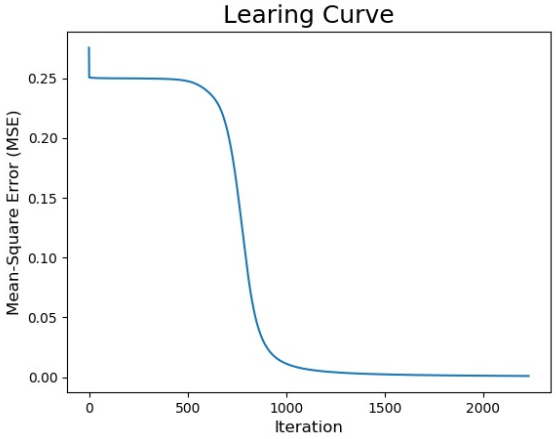


Figure 9: Learning curve for the network shown in Figure 4 with constraints to learn only horizontal and vertical decision boundaries. This shows that the network's training error falls below 0.002 after approximately 1200 iterations of training with a learning rate of 1.2.

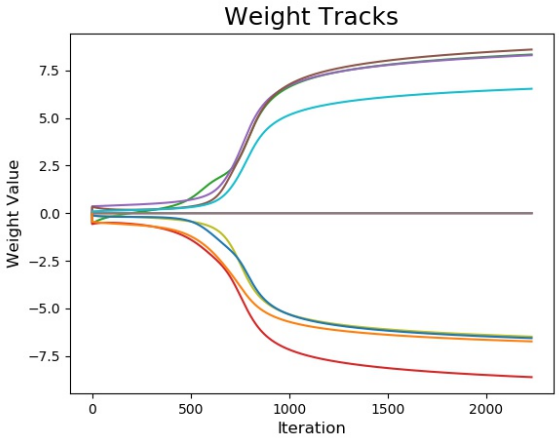


Figure 10: Weight tracks for the network shown in Figure 4 with constraints to learn only horizontal and vertical decision boundaries. This shows that the network's weights converge after approximately 1200 iterations of training.

These results shows that the network was able to perfectly learn the star problem. As a side note, corners of the discrimination region look a bit rounded due to the number of samples in the test grid.

References

[1] Principe, Jose C., Euliano, Niel R., Lefebvre, W. Curt. "Chapter III- Multilayer Perceptrons," in Neural and Adaptive Systems: Fundamentals Through Simulation, 1997