Assignment 1: Perceptrons and Neural Networks

Warm Up

1. The perform function in MATLAB uses the mean squared error function, which takes the squares of each of the errors (the difference between each element of the set of inputs to its respective element in the set of target values), and averages the results.

2. The error of the trained network net1 is 0.0479.

3. By using

   ```
   perc= perceptron();
   perform(perc,target,y)
   ```

   MATLAB computes the error of this newly trained perceptron as 0.1712.

A Perceptron for Mammals and Birds

4. After training the perceptron `perc_birds` on the 40 elements of the input, each with 16 features, and using the perform function, MATLAB computes that there no categorization error.

5. As there is no error, the perceptron correctly differentiates between all the birds and mammals included in the training examples.

6. The perceptron only did 3 iterations over the training data.

7. Using the test data as an input, the perceptron has a mean-squared-error of 0.1667.

8. The perceptron failed to correctly identify

   - beaver
   - raccoon
   - dolphin
   - platypus
   - bat

   By looking at the weights the perceptron has learned to operate on the McRae functions, we can examine why it has made some of these mistakes. For the beaver, raccoon, dolphin and platypus, it is mainly a case of too few defining features. Both the raccoon and beaver had several people assign the 'has a tail' feature, which was weighted in favour of being categorised a bird, and their only other assigned features ('is brown' and 'is small' respectively) were weighted zero by the perceptron.

   A somewhat similar problem occurred with the dolphin and the platypus, both of which only had one assigned characteristic, making it difficult for the perceptron to categorise them. In the case of the bat, it is clear to see that it shares many of the characteristics typically assigned to birds- it flies, has wings, and is small- this is a case where a linear separation is difficult if not impossible.

A Neural Network for Mammals and Birds

9. After training the neural network `net1` on the training data, it reports a classification error of 0.0177, which is a higher error than the perceptron, which had no error on the training data.

   The results of applying the neural network on the test data, however, are far better to the results we found with the perceptron- an error value

of 0.0598 and only two incorrectly categorized animals- the whale and the bat.

10. By doing 10 iterations of the training and performance of `net1`, we can see a clear pattern emerge in the graphs created. Although the initial error value of net1 on the training data varies, the final error after doing 1000 iterations on the training data lies in the range $[0.010, 0.025]$, as the neural network analyses the test data, its performance tends improve rapidly at first before reaching a horizontal tangent at it's final error, although in some cases it has a local maximum value where the network creates more errors before improving.

    The error value on the test data varies quite a lot with a range of approximately $[0.0100, 0.1350]$, and with between 2 and 5 incorrectly categorised elements.

11. By adjusting the learning rate, the number of hidden layers and their respective sizes, and the stopping criterion of the training for the neural network net1, we can find the optimum values of each to minimise error in both the training and test sets.

    Increasing the learning rate has a significant impact on the training error value, however it neglects to vastly improve test error values, and 2 or 3 miscategorisations are made with each iteration. By adding the line `net1.trainParam.lr =1;` we set the learning rate to 1. The training error rate is reduced to approximately 1×10^{-5}, and the testing error value averaging 0.07.

    Adding and removing hidden layers has a startling effect on both the training and test data error values. Setting 2 hidden layers of 1 unit each, for example, often increases both error values to 0.25, with up to 14 incorrectly categorised elements in each set.

    At best, adding more hidden layers and units within those networks kept the error values at the same level as before, but at worst the added layers made the errors unpredictable, and caused a large number of mistakes to be made. By testing different values, I found that the best compromise was using the values:

    ```
    numHidLayers = 1;
    hidLayerSpec = [3];
    net1 = create_simple_network(hidLayerSpec);
    ```

which left the training error untouched, and caused the testing error value to stabilise and approch 0.070, with both platypus and bat emerging as incorrectly categorised in every iteration.

By adding a stopping criterion of anything greater than 1×10^{-5}, we would stop the neural network from continuing the training after a smaller amount of iterations, and thereby increase the overall error value on both the training and test data sets. It is therefore in our best interest to leave this figure as low as possible. In this case, I chose to simply use: `net1.trainParam.goal = 0.00001;`