**Vibhav Jha BE 700 HW2 Fall 2021**

**Part 1. Implementing the Perceptron Algorithm on Data from HW1**

In implementing the perceptron algorithm, I chose to add a relu type activation function to bind the output values to 0 and 1. Using a learning rate of 0.5 I was able to achieve a highest accuracy of 0.8966 after 500 iterations. Increasing the learning rate to 6 and the number of iterations to 600 the accuracy increased to 0.9195.

For learning rate 6 the genes with the highest weights (at least greater than 100), or of the most positive impact to cancer prognosis were 41044\_at (18), 32434\_at (17), 36559\_g\_at (6), 38221\_at (23), 33741\_at (28), 40131\_at (5), and 35291\_at (2). The genes with the lowest weights (less than -100) or most negative impact towards classifying cancer prognosis were 32607\_at (22), 34682\_at (19), 39028\_at (10), 37439\_at (30), 41872\_at (27), 33582\_s\_at (4). Moving forward to improve the classification accuracy, it may be beneficial to remove all genes aside from the highest weighted.

Code for this part is in the file: HW2\_Perceptron.m and coded in MATLAB R2020b.

**Part 2. Implementing the Perceptron Algorithm on Synthetic Data**

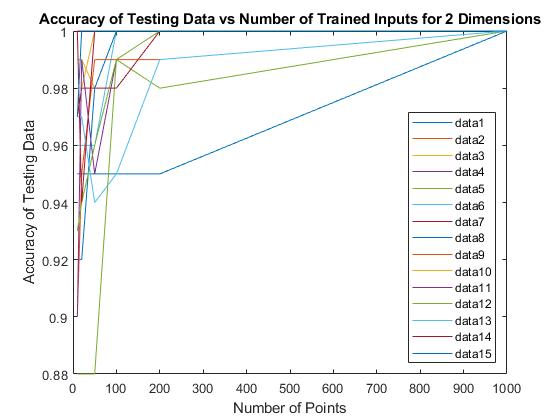
Using the same framework coded above, the perceptron algorithm was applied to purely random data. Weights were initialized to 0 and for the initial data analysis the learning rate was set to 0.5. Clear trends were found in which when testing with 100 inputs, for 10 dimensions it is beneficial to have 100 or more training values. In table one, for 10 dimensions accuracy started quite low using only 10 inputs, but when the size of the training and testing sets were equal or greater the accuracies were 0.94 or greater. This was also shown in 2 and 3D however, the difference is much more subtle. For less dimensions cutting the training set to 50 produced good results, similar to accuracies only achieved by greater than 200 training inputs for 10 dimensions. Variances for these accuracies were overall quite low, and the highest variances (shown in table 2) came from a low number of inputs within the 10D set. Figures 1 through 3 show the learning curves of multiple runs and number of inputs. In two-dimensional data (figure 1) the accuracies. seem rather chaotic, but have lower overall variances. In three-dimensional data, the same is seen, however with higher variances than when compared with two-dimensional data. For ten-dimensional data, only after 100 points does the accuracy begin to approach 1. From this, it is clear that high dimensional data needs a training set about equal to the size of the testing set in order to approach accuracies of 1.

**Table 1**. Average testing set (N = 100) accuracies for N = 10,20,50,100,200,1000 trained points and each dimension over 15 single perceptron runs of randomized inputs and initial hyperplanes. Weights were initialized at 0 each time and the learning rate was 0.5.

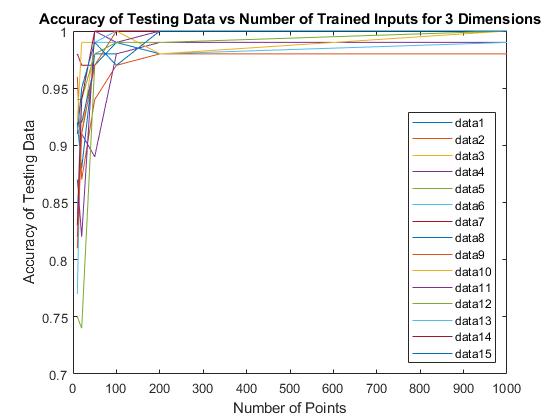
|  |  |  |  |
| --- | --- | --- | --- |
| **Dimensions** | | | |
|  | **10** | **3** | **2** |
| **10** | 0.738 | 0.876667 | 0.954667 |
| **20** | 0.831333 | 0.91 | 0.963333 |
| **50** | 0.879533 | 0.974667 | 0.971333 |
| **100** | 0.942667 | 0.988667 | 0.989333 |
| **200** | 0.967333 | 0.991333 | 0.994 |
| **1000** | 0.993333 | 0.996667 | 1 |

**Table 2.** Variances in testing set accuracies for N trained points and each dimension over 15 single perceptron runs of randomized inputs and initial hyperplanes. Weights were initialized at 0 each time and the learning rate was 0.5.

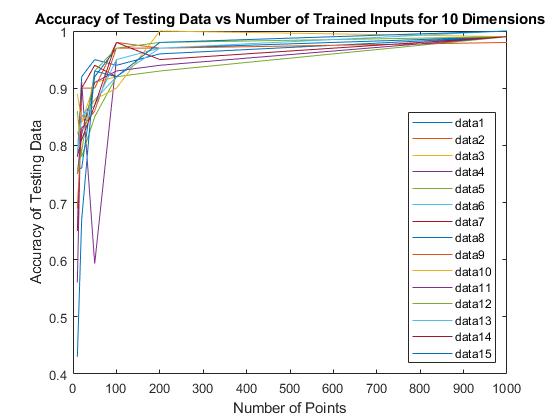
|  |  |  |  |
| --- | --- | --- | --- |
| **Dimensions** | | | |
|  | **10** | **3** | **2** |
| **10** | 0.012789 | 0.004542 | 0.001386 |
| **20** | 0.004025 | 0.003747 | 0.001021 |
| **50** | 0.006744 | 0.000732 | 0.000948 |
| **100** | 0.000646 | 9.16E-05 | 0.000256 |
| **200** | 0.000273 | 6.49E-05 | 0.00016 |
| **1000** | 3.56E-05 | 3.56E-05 | 0 |



**Figure 1.** Testing Accuracy versus number of trained inputs for two dimensions of 15 separate runs. Weights were initialized at 0 each time and the learning rate was 0.5. Here the accuracies do not follow a smooth curve, but are all above 0.88 and points between each N seem to have little variation.



**Figure 2.** Testing Accuracy versus number of trained inputs for three dimensions of 15 separate runs. Weights were initialized at 0 each time and the learning rate was 0.5.



**Figure 3.** Testing Accuracy versus number of trained inputs for 10 dimensions of 15 separate runs. Weights were initialized at 0 each time and the learning rate was 0.5.