Task 2

Task 1 Model Analysis

The final results show that the model begins to train correctly though they are not consistent across multiple runs (due to the random initialization of the weights). The results vary in accuracy between 55% to 70%. This would suggest that there have not been enough iterations of the model, the learning rate or batch size is too high or more likely a combination of these three factors.

The following figures show the same model (with the same parameters) across multiple runs.

Figures 1 and 2 show the inconsistency of the loss curves across multiple runs. Figure 1 shows an example of under fitting, the model does not have enough capacity for the complexity of the data set. Figure 2 shows the same model on another run, where the model has trained more generally. However, it also appears to show underfitting. Shown by the intersection in the loss curve. This may just be an early intersection in the log curve that could be removed with further iterations. Both figures would suggest there are too few epoch iterations to generalize the model or the batch size is too large for the given dataset. I would suggest the latter would play a greater part; the training dataset is only 250 samples therefore a batch of 128 samples will lead to the model underfitting as will only update 2 times per epoch. I would be in favor of reducing the batch size to 64 or 32 or perhaps increase the percent of the dataset used in training.

Figure 1

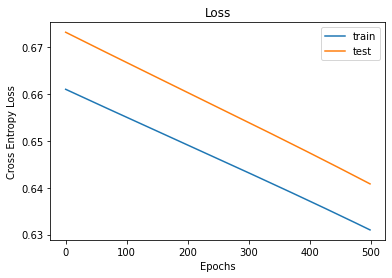
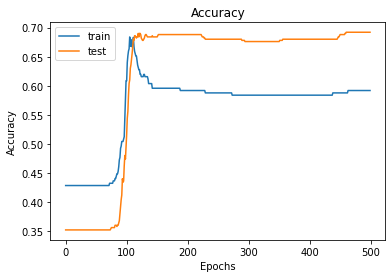
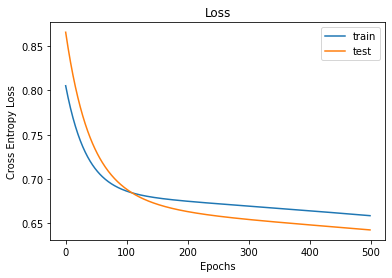


Figure 2

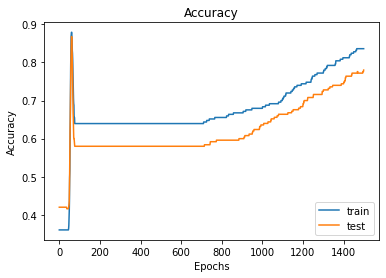
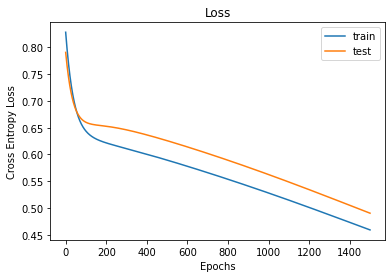


In conclusion the model appears very inconsistent across different training runs. There are not enough training epochs to fully train the model leading to a wide range of accuracy scores and loss curves. The model is too dependent on the random initialization of the neuron weights and would require much more iteration using a smaller batch size and learning rate to lead to a consistent generalization of the problem. Also, the dataset is biased (301/500) 60% positive samples to 40% negative samples. This can lead to misleading accuracy results.

Task 2 Model Improvements

Given the seemingly inconsistent nature of the results obtained in Task 1 and final accuracies ranging from 55% to 70% I was firstly interested in eliminating this variation across runs by increasing the number of epochs as I believed the model was too dependent on the random weight initialization. Figure 3 seems to confirm these assumptions and that the intersection discussed previously was a small deviation in a larger curve realized over a larger number of epochs. Figure 3 shows the same parameters as Task 1 with only the number of epochs changing to 1500. This also makes more sense as the training loss is now steadily less than the testing loss along with the training accuracy remaining higher than test.

Figure 3



I decided to firstly vary the number of epochs to see when the model reaches a stable state.

Figure 4 shows a table of the averaged, final accuracy and loss across 3 runs for a range of epoch values with no other parameters changed from Task 1.

Figure 4

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Epochs** | 1000 | | | 2000 | | | 3000 | | | 4000 | | |
| **Accuracy** | 0.56 | 0.66 | 0.60 | 0.88 | 0.84 | 0.77 | 0.90 | 0.93 | 0.93 | 0.93 | 0.88 | 0.94 |
| **Loss** | 0.65 | 0.61 | 0.60 | 0.44 | 0.45 | 0.51 | 0.36 | 0.32 | 0.33 | 0.24 | 0.32 | 0.26 |
| **Mean Acc.** | 0.60 | | | 0.83 | | | 0.92 | | | 0.92 | | |
| **Mean Loss** | 0.62 | | | 0.47 | | | 0.34 | | | 0.27 | | |

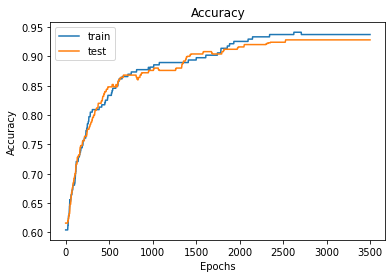
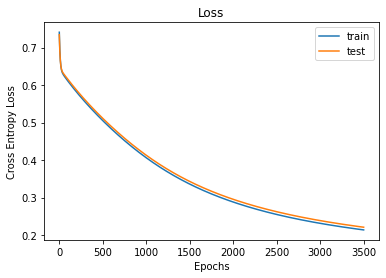
As shown the in the table the final accuracies and losses plateau between 3000 and 4000 epochs. I decided start changing other parameters. I also experimented with reducing the batch size as the dataset is small. This will hopefully provide a higher update frequency to the neurons without excessive iteration and perhaps allow for a reduction in the number of epochs (reduced training time).

Through decreasing the batch size to 64 (4 parameter updates per epoch) (32 created gradient instability) and reducing the epochs to 3000 I achieved an accuracy of 0.92 and a loss of 0.24. I also reduced the training time by 1/3 vs 4000 epochs.

Next, I tried increasing the number of neurons in the second layer to 30. The ‘Mixcancer’ dataset has 30 features therefore by using 30 neurons the network can learn from all combinations in the previous (input) layer. This ‘fully connected’ approach is more computationally expensive than before and has increased the training time by 10%. I finally reduced the learning rate to 0.006.

The results range in accuracies from 0.93-0.97 and losses from 0.15-0.20 with a final F1 score of 0.96. Figure 5 shows the final ANN accuracy and loss graphs along with a parameter table.

Figure 5



|  |  |
| --- | --- |
| **Parameter** | **Value** |
| Epoch | 3000 |
| Learning rate | 0.006 |
| Batch size | 64 |
| Layer 2 neurons | 30 |

I also experimented with adaptive learning rate reducing it on each epoch iteration. However, this had minimal benefits. I have left this commented out in the Jupyter Notebook.

I also attempted to change both layer activation functions independently however this greatly reduced the accuracy and loss regardless of neuron count.