Classifying Cancer Cells from Fine Needle Aspirates: Backpropagation versus Linear Programming

L. Stevens

Abstract

In "Machine learning techniques to diagnose breast cancer from image-processed nuclear features of fine needle aspirates" [1] a variant of the multi-surface method (MSM), known as MSM-Tree [2] was used to distinguish malignant from benign cancer cells, given the nuclear features of the cell. A three-layer, fully-connected neural network using backpropagation was used to try and get similar or better results given the same dataset.

Keywords: Breast Cancer, Machine Learning, Linear Programming, Neural Networks, Fine Need Aspirates

Introduction

Wolberg et al. found much success using a variant of the multi-surface method (MSM), known as MSM-Tree [2]. MSM is a classification technique that uses a neural network which uses linear programming instead of backpropagation to train its weights. MSM-Tree is predicated on piecewise-linear separability, and aims to minimise the intersection between the convex hulls of point sets by iteratively looking for a suitable discriminant. Using MSM-Tree, Wolberg et al. concluded that with 95% accuracy, that the true prospective accuracy of their classification sat between 95.5% and 98.5% [1].

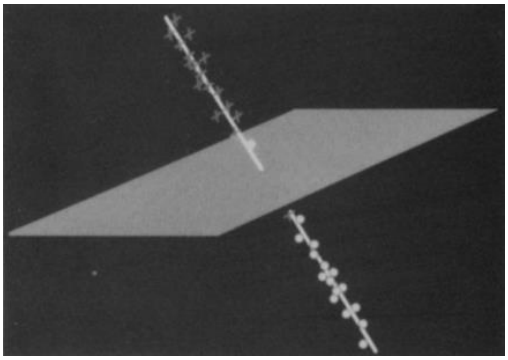


Figure 1 A visualization of the separating plane between each category

Using MSM-Tree to train neural network weights is seen to have many advantages over backpropagation, including:

- Automatic determination of the number of hidden units
- Achieving 100% correctness on the training set if desired
- Faster training
- Elimination of parameters from the training algorithm

[2]

Using a backpropagation-based neural network to classify the same data as used by Wolberg et al. should yield a lower true prospective accuracy than their linear programming-based neural network. Using the WDBC dataset [3] as used by Wolberg et al., the data will be partitioned into 10 randomly selected sets of training and test data. The WDBC dataset has 569 total instances, which will be split into 469 training instances and 100 testing instances. For each of the randomly selected sets of training and test data, the test data will be run through 100 times, recording the accuracy each time. The accuracies will then be averaged, and compared to the average accuracies of the other 9 randomly selected training and testing data sets.

Method

The neural network structure used is a three-layer, fully-connected neural network with a ReLU layer situated between the hidden and output layers. The loss function used was CrossEntropyLoss, which performed the best out of all the loss functions tested (CrossEntropyLoss, BCELoss, BCEWithLogitsLoss, Adam). It is worth noting that since this is a binary classification problem, using a properly configured implementation of either BCELoss or BCEWithLogitsLoss should yield better results. The net is configured with a hidden layer size of 50 neurons, a learning rate of 0.1, and 5000 epochs of training.

Slight modifications were made to the original dataset downloaded from the UCI Machine Learning Repository [4], namely adding correct labels, removing the 'id' column, and renaming the file from 'wdbc.data' to 'wdbc.csv'. This new csv file was then read into a `pd.DataFrame` using `pd.read_csv(path)`, where the malignant (M) and benign (B) output labels were replaced with a 1 or 0 respectively, so that the data would be compatible with the neural network. The `pd.DataFrame` was then converted into an `np.Array`

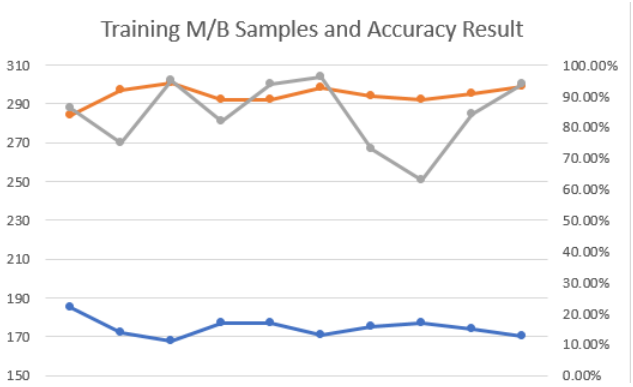
and partitioned into training and testing sets. Before partitioning, the `np.Array` was shuffled to randomise the contents of the training and testing sets. This partitioning process is indiscriminate of the labels of selected entries, which allows either set to have a disproportionate amount of malignant and benign entries. The amount of malignant and benign entries in each set is printed to `std_out` at runtime. Each partitioned `np.Array` is then converted to a `torch.tensor`, before being wrapped in a torch.Variable. Both the training input set and the testing input set are normalized before training begins.

The neural network is then trained for the specified amount of training epochs, which by default is 5000. In each epoch, the training set is fed through the neural network, and the loss is calculated by feeding the predicted and actual labels into the CrossEntropyLoss function. The loss is printed to `std_out` every 200 epochs. Before the end of each epoch, the networks gradients are zeroed and the net performs its backwards pass. Finally, we use stochastic gradient descent to optimise the training process. A confusion matrix is then generated for the training, with the results printed to `std_out`.

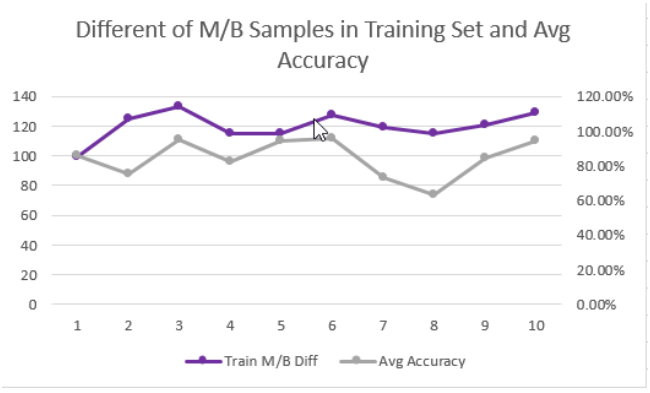
Finally, the test set is fed through the neural network and the accuracy is calculated by comparing the labels predicted by the network to the actual labels. This is completed 100 times, with the result being the average accuracy of 100 testing cycles.

Results

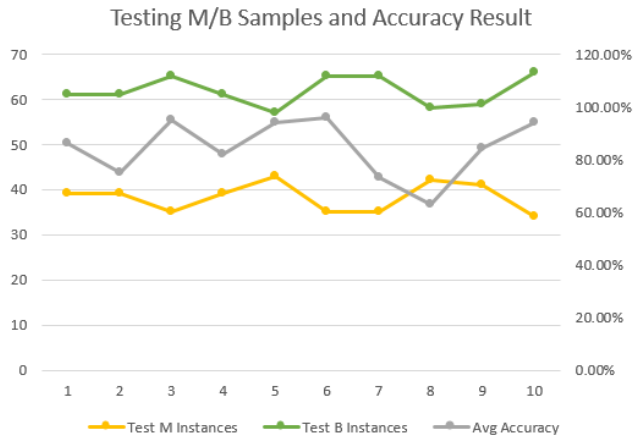
Test Batch No	Avg. Accuracy
1	86%
2	75%
3	95%
4	82%
5	94%
6	96%
7	73%
8	63%
9	84%
10	94%
Average	82.4%



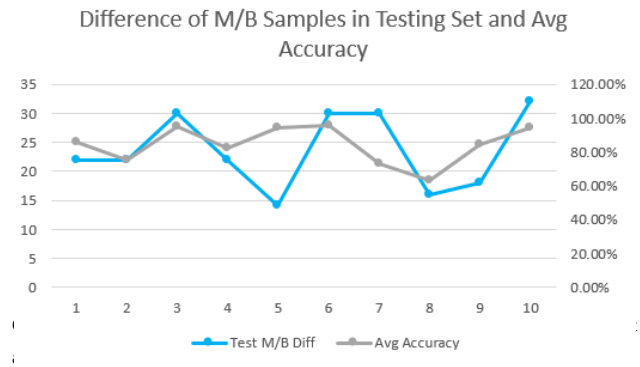
Graph 1. Comparing the amount of malignant and benign samples in the training set and the average accuracy of the testing.



Graph 3. Comparing the difference in malignant and benign samples in the training set and the average accuracy



Graph 2.. Comparing the amount of malignant and benign samples in the training set and the average accuracy of the testing.



		Predicted	
		B	M
Actual	B	277	0
	M	46.2	132.4

Graph 5. Average Testing Confusion Matrix

The erratic set of results seems to imply that the model does not function as intended. Some interesting observations from the recorded data include:

- The average confusion matrix shows that the model never falsely predicted a benign cancer cell to be malignant.
- Though the total average accuracy of 82.4% is not awful, the standard deviation of average accuracy is quite high. It is difficult to reason with the data collected why this is the case.
- There seems like there could be a correlation between the difference in malignant and benign samples in the training set, and the average accuracy of predictions.

Conclusion and Future Work

It is hard to compare the results of Wolberg et al. to the results of this test because of the tainted outcomes of the testing. To further this work, several changes must first be made to the model:

- The neural network should be optimized as well as possible for binary classification. This means implementing BCELoss or BCEWithLogitsLoss correctly.
- The average confusion matrix seems to imply the issue is with classifying malignant cells. This may be due to the fact that the dataset contains over 20% more benign instances than malignant.
- Wolberg et al. used a specific set of data points which make it easiest to classify each class (Mean Texture, Worst Area and Worst Smoothness) [1], whereas every data point was used by the backpropagation neural network.

1 W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994) pp. 163-171.

2 Bennett. K.P. Decision Tree Construction via Linear Programming. Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society Conference (1992) pp. 97-101

3 http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data

4 http://archive.ics.uci.edu/ml/index.php