Individual reflection – Assessment 1

We chose to explore the Wisconsin Breast Cancer Dataset from the UCI ML repository, as introduced in Street et al [1]. The dataset contains 569 datapoints containing features about the cell nuclei found in various subjects with breast tumours that can be classified as either malignant or benign. Our main goal was to explore binary classification methods applied to this dataset.

As a group, we set up a WhatsApp group, made a Git repository and arranged regular check-ins. We decided to all work on one Python notebook for the report using Google Colab connected to our Git repository. This had the advantage of being able to use the same variables and functions in different sections of our code. However, there were some big problems. Firstly, it was hard to keep track of our documentation – although it can be found in the version history, we were not able to easily upload this to the Git repository. Secondly, lots of our work ended up getting deleted due to the way that Colab saves work when multiple people edit the notebook simultaneously. This caused a lot of frustration and in future I would recommend that each group member works on their own separate notebook. I would also set earlier internal deadlines with my group and arrange to meet directly after these, to collectively review our progress. This would ensure that the report is written consistently and allow us to catch bugs early enough to fix them.

We made some big decisions early on. Firstly, we agreed to use the UCI ML repository's Python library to import the data and set the random seed for splitting this into a train-test split. This ensured that our models could be compared fairly by examining their performance on the held-out test data. Next, we created a function that fixed the seed across several Python libraries to ensure that all our work was reproducible, and we rescaled the data, which was required by most of the models to work properly.

We ran into a big issue when I noticed that some models were using the held-out test data for the purposes of training or parameter tuning, leading to inaccurate performance metrics when evaluating the models on that same test data; this is known as 'double dipping'. To avoid this, we used cross-validation techniques to tune or train our models using validation data taken from the training data.

After some discussion and as mentioned in [1], we decided to use sensitivity as our main performance metric. This was because we were viewing our classifiers as diagnostic tools and the higher the sensitivity, the fewer the false negatives – here a false negative would be detrimental, as it corresponds to not diagnosing a subject's cancer. We do however consider other metrics in our analysis too.

As I have some previous experience with coding neural networks, I decided to work on using neural networks (NNs) as binary classifiers. I also wrote the introduction about the dataset and performance metrics in the report and contributed to the overall conclusion.

I experimented with simple feed-forward NNs that had an input size of 30 (to match the dimension of each datapoint), an output size of 2 (corresponding to the binary classes) and varying numbers of hidden layers with varying sizes. They work by performing a matrix multiplication between a given layer and its weight matrix, which is fed to the next layer. This is repeated until the final layer, whose value is passed through a sigmoid function to give an output in the range (0,1). Here we consider 0 as benign and 1 as malignant.

These weight matrices are learned through training the neural network, which involves inputting a training data example, computing a loss function between its output and the true class of the example and then computing gradients to backpropagate through the network, updating the values of each weight. Given sufficient data, the model should converge – that is, learn the best weights for the task of predicting the class of any input data.

The NNs I tried converged well and had excellent performance, with sensitivities of ~95% or above. Adding too many hidden units or more hidden layers caused the models to overfit to their training data, resulting in worse performance on the held-out test data. As such, the simplest model without any hidden layers performed the best on the test data, likely because the simplicity avoided overfitting.

As mentioned in the report, NNs are unnecessarily complicated for such simple binary classification tasks, and really work better on more complex data. It may be interesting to explore whether more complicated architectures or advanced processing techniques that generated artificial features could lead to even higher performance, but I would guess that this is unlikely. A more interesting and modern approach would be to train a convolutional NN to predict diagnosis given the original images of the breast cell nuclei collected in [1] - this could be a great set-up for a modern data science competition.

Beyond this, the general conclusions of our project are that simpler models are better, since they lead to less overfitting, and that the dataset is well suited to binary classification due to its high correlation between features and diagnoses.

References

[1] Street, William Nick et al. “Nuclear feature extraction for breast tumor diagnosis.” Electronic imaging (1993).