CS3105 - P2 Report

190022658

November 22, 2021

Overview

This practical required us to build a feed-forward Artificial Neural Network (ANN) for a multi-class classification task. This involved -

	Tasks	Done	Notes
Part 1	Missing code filled in and the program works properly	yes	
	Linear classifier setting submitted	yes	
	Results reported	-	
Part 2	Expected performance achieved and chosen hyper-parameter submitted	yes	
	Tuning process and results reported	yes	
Part 3	Data pre-processing implemented	yes	
	Results reported	-	
Part 4	Analysis done and reported	-	

Part 1

- loaded the dataset
- created a static method to split the dataset into training and validation set
- used json-simple library to parse the json file (available in lib/)
- implemented the build method to apply different and multiple activation layers to the ANN
- used the hyper parameters as arguments for build and test

What needs to be set in your *.json* file to make a linear classifier?

In the *linear.json* file, changing the number of hidden layer to 0 was all that was needed to make a linear classifier. Note that the input will be input_dims (7) and output will be output_dims (3).

How is the performance of the trained ANN compared with the linear model?

The accuracy is almost the same. It's as given below -

- accuracy on test set: 0.457
- accuracy on test set: 0.442

Part 2

In part 2, we had to change some hyper-parameters to adjust the accuracy to be atleast 70%. The Part2.json is the final hyper-parameters settings used for this part.

- n_hidden_layers - Increasing the number of hidden layer increased time and decreased accuracy.
- n_nodes_per_hidden_layer - I changed this in multiples of 10, increasing it to more than 20 did increase the accuracy but also increased time. Since the desired accuracy was reached it was not worth the time.
- activation_function - All of the activation functions give the desired accuracy (with all the other options set to what they are)
- learning_rate - the learning rate is to find weight such that the loss is minumum. Increasing it too much would decrease the accuracy and same for decreasing it too much. 0.1 was a good match.

Final accuracy on test set: 0.738

Part 3

Here, we needed to preprocess the data given in data/Part3 before being used for the Machine Learning model. We used data imputation and data standardisation for preprocessing.

The data imputation processing was simple and involved calculating a mean value (for a feature) of non-missing values and replacing the missing value with it.

While implementing this I realised that it would be better if I imputated the value of mean of values which had the same classification. TODO

For data standardisation, we first needed to calculate the mean of each feature and then using that calculate the standard deviation for each feature. This is then used to finally calculate the z-score.

Z = (score - mean) / standard deviation

`https://www.simplypsychology.org/z-score.html`

How did you implement data-preprocessing for the given training and test sets?

Here it is important to note that for standardisation, the mean and standard values (for each feature) for trainset are used in the testset too. This is done because the data will be askew otherwise.

The thought of using the same mean for imputation for both trainset and testset came to mind, but that seemed wrong because the testset should not be changed by the trainset.

Compare performance of the your ANN with and without data-preprocessing.

- **Without preprocessing -** accuracy on test set: 0.331
- **With preprocessing -** accuracy on test set: 0.672

Part 4

For feature importance analysis, I created a new dataset and testset with a feature data missing and then built and trained the ANN using that data. I looped and did this for all features.

This was run for *data/Part1.*

There are bar charts in *charts/* if that is preferred.

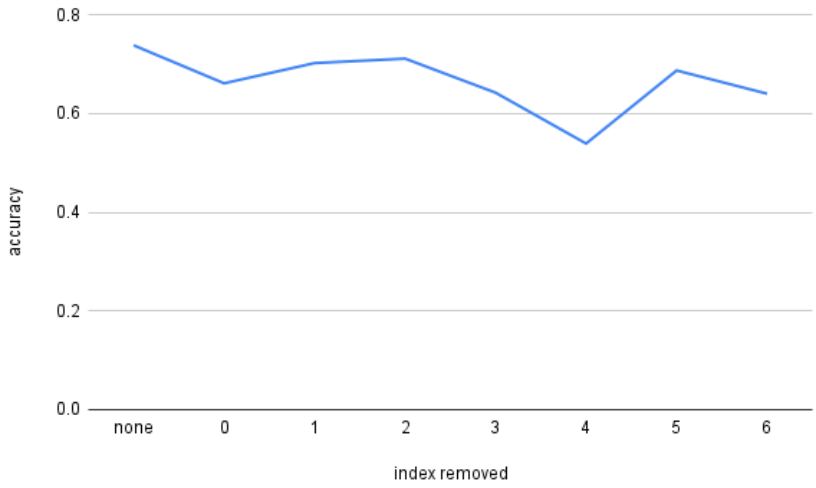

Figure 1: Accuracy vs Index Removed

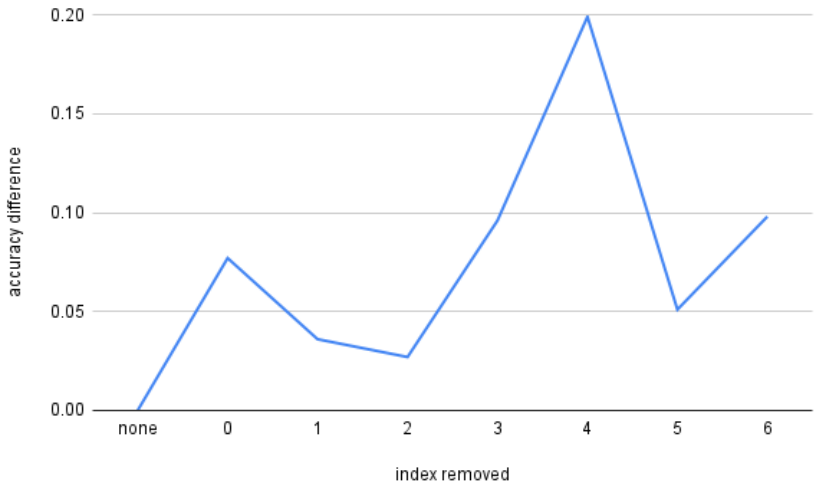

Figure 2: Accuracy Difference vs Index Removed

The charts show that all of the features are important. But there importance can be rated from the least important (2) to the most important (4).