Computational Intelligence

MIA 2024-25

December- 22- 2024

Neural Networks

Multi-Layer Perceptrons

Marina Bermúdez Granados
marina.bermudez.granados@estudiantat.upc.edu

Asha Hosakote Narayana Swamy
asha.hosakote@estudiantat.upc.edu

Table of contents

1. Introduction to the problem

This project explores the performance of multi-layer perceptrons (MLPs) for image classification using the CalTech 101 Silhouettes dataset, which contains 8,671 images of 101 silhouette categories, such as bonsais, chairs, or elephants. Each image is represented with 784 classification features; their corresponding labels will be encoded using a one-hot scheme encoder. We mainly tested two MLP configurations, including different hidden units, transfer functions, cost functions, and dataset splits.

Each setup was run at least five times to estimate their mean performance, with other hyperparameters (e.g., learning rate, epochs) optimized through initial testing. The results, presented in tables with explanations, analyze how these configurations impact performance. The report also includes a summary of methods, results, and conclusions, providing insights into building effective MLPs for classification tasks.

2. Configurations

We will experiment with the configurations provided within the laboratory exercise. The first configuration has a logarithmic sigmoid function for the hidden and output layers, with the mean squared error as a cost function. The second one changes the output layer to a softmax function with a cross-entropy cost function. We will first focus on fine-tuning the present configurations across different hidden unit quantities before also testing different percentages for training, testing, and validation.

2.1. Fine-tuning both configurations

Our objective in this section is finding the best parameters for both configurations prior to the actual required experiments. We will specifically select the best performance for hidden units 50, 200, and 500. In every case, we will consider learning rates of 0.5, 0.1, 0.01, and 0.001 and any other required parameters for their training functions.

We quickly determined that the best results for the first configuration originated with a variable learning rate gradient descent and resilient backpropagation after briefly testing all possible options within the MATLAB documentation. The gradient descent algorithm also allows the use of momentum, a method utilized in neural networks to accelerate the optimization during training; we tested for values 0.9 and 0.5. As we can see in Table 1 and Fig. 1, the best momentum is 0.9 and the best learning rate is 0.1. However, the selection is not as clear with the training functions. In rapid backpropagation, hidden units 50 and 200 show accuracies around 40% that rapidly decline to 5% with 500 hidden units. Meanwhile, the variable gradient descent starts at 27.33% with 50 hidden units, peaks at 36.65% with 200 hidden units, and falls a bit to 34.76% with 500 units. According to the documentation, rapid backpropagation degrades as the error goal is reduced despite being the fastest algorithm on pattern recognition problems. This instability in training has led us to choose the variable learning rate gradient descent, even if the results are slightly worse.

Hidden Units	Training function	Learning rate	Momentum	Accuracy
50	Variable learning rate Gradient Descent	0.01	0.9	27.33
50	Rapid Backpropagation	0.1	-	40.20
200	Variable learning rate Gradient Descent	0.1	0.9	36.65
200	Rapid Backpropagation	0.1	-	43.89
500	Variable learning rate Gradient Descent	0.1	0.9	34.76
500	Rapid Backpropagation	0.1	-	4.91

Table 1. Best Results Fine-tuning Configuration 1 per Hidden Units

When it comes to the second configuration, there were many candidate training functions that extracted acceptable results. From all the options, we decided to test a conjugate gradient with Powell-Beale restarts and a scaled conjugate gradient. Inspecting the results in Table 2 and Fig. 2, both approaches obtain similar results. There was not much variance between the best and worst sets of parameters. Hence, we decided to prioritize the scaled conjugate gradient due to its speed and more decisive learning rates, picking 0.5 as its final value.

Hidden Units	Training function	Learning rate	Accuracy
50	Conjugate Gradient with Powell-Beale Restarts	0.001	52.73
50	Scaled Conjugate Gradient	0.01	55.31
200	Conjugate Gradient with Powell-Beale Restarts	0.01	58.47
200	Scaled Conjugate Gradient	0.5	58.13
500	Conjugate Gradient with Powell-Beale Restarts	0.1	57.87
500	Scaled Conjugate Gradient	0.5	56.90

Table 2. Best Results Fine-tuning Configuration 2 per Hidden Units

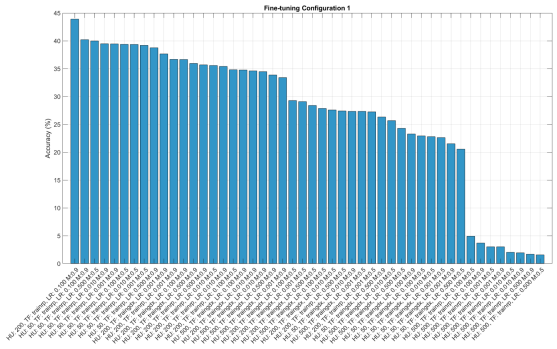

Fig. 1. Fine-tuning Results for Configuration 1

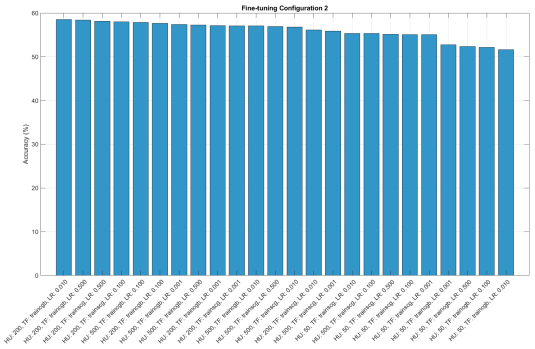

Fig. 2. Fine-tuning Results for Configuration 2

2.2. Experiments with both configurations

Once the training parameters have been decided, we could start preparing the experiments for the laboratory exercise. The first configuration will set its learning rate to 0.1 with a momentum of 0.9 and the variable learning rate gradient descent as its training function. The second one will set its learning rate to 0.5 with the scaled conjugate gradient as the training function.

We have to experiment with the fine-tuned configurations against the previous amounts of hidden units while also varying the training, validation, and testing ratios of the dataset. The options are 80% training with 10% for validation and testing, 40% training and testing with 20% of validation, and 10% of training and validation with 80% reserved for testing. For both configurations, we will perform five runs with 500 training epochs each. The resulting average accuracies and mean squared errors will be captured for later analysis.

3. Results

In this section, we will analyze the results from the main experiments, starting with the first configuration. The results can be inspected in Table 3; the green cells represent the best results while the red ones represent the worst ones. In the case of the 50 and 500 hidden units, the metrics progressively decline as the training ratios decrease. For the 200 hidden units, there is an exceptional peak in performance at the 40% training split. This point is also the best performance from this configuration, with an accuracy of 37.34% and a mean squared error of 1109.90. The worst performance springs in the 10% training split with 50 hidden units, with an accuracy of 22.40% and a mean squared error of 2177.18. When increasing the amount of hidden units instead of the training ratios, there is no clear progression. It seems that the best results tend to occur around the 200 and 500 hidden units.

Config. 1	**50 Hidden Units**		**200 Hidden Units**		**500 Hidden Units**	
	Accuracy	**MSE**	**Accuracy**	**MSE**	**Accuracy**	**MSE**
80/10/10	30.36	1320.65	35.22	1313.07	36.77	1154.02
40/20/40	26.87	1546.48	37.34	1109.90	34.12	1276.83
10/10/80	22.40	2177.18	33.89	1156.70	31.19	1294.07

Table 3. Results of Experiments for Configuration 1

Continuing with the second configuration, the results share some common patterns with the previous one. For instance, we can see that as the training ratios decrease, the metrics decline as well. There is no exceptional peak in this case; all these metrics can be checked in Table 4. The best metrics occur in the 80% training split at 200 hidden units, with an accuracy of 59.24% and a mean squared error of 609.83. Meanwhile, the worst metrics occur in the 10% training split with the 500 hidden units, with an accuracy and mean squared error of 38.56% and 988.65. Similarly, it still seems that the best results happen while applying the 200 hidden units.

Config. 2	50 Hidden Units		200 Hidden Units		500 Hidden Units	
	Accuracy	MSE	Accuracy	MSE	Accuracy	MSE
80/10/10	55.17	737.72	59.24	609.83	56.52	664.82
40/20/40	49.78	768.76	50.01	796.91	49.47	798.39
10/10/80	40.41	1001.01	39.86	1007.72	38.56	988.65

Table 4. Results of Experiments for Configuration 2

4. Conclusions

During the completion of this exercise about multi-layer perceptrons, we have conducted different experiments with previously fine-tuned configurations and analyzed their corresponding results. This laboratory exercise has aided us in our understanding of the practical aspects when implementing neural networks for classification tasks. For instance, we checked experimentally that rapid backpropagation becomes unstable when the error goal is reduced. The first configuration with this function shows an exceptional downgrade, from accuracies around 40% to just 4% when using 500 hidden units. Hence, we decided to select the slightly worse-performing option, variable learning rate gradient descent with the respective hyperparameters. Our experiments should be reliable enough so as to extract proper insights. There were not many issues while fine-tuning the second configuration, immediately finding better prospects for the training functions.

The main experiments studied how the configurations react to different hidden units and training scenarios. We can confidently conclude that utilizing the softmax output layer and the cross-entropy cost function brings far better results than the logarithmic sigmoid layer and mean squared error cost. The output softmax layers tend to be a popular choice for classification tasks, specifically multilabel problems. Similarly, output logarithmic sigmoid layers are used in binary classification problems. Therefore, the performance disparities originate from the design choices for each configuration. Furthermore, the cross-entropy costs are typically used in classification problems, while the mean squared error is used in regression tasks. However, the design choices were only responsible for a part of the results. The percentage dedicated for training also had a notable impact on the metrics. As a general rule, with some mentioned exceptions, the results decreased when using fewer data samples for training. This was to be expected, as the network has a better chance to learn from the data with more instances. We did get surprised when analyzing the hidden units, though. We expected the 500 hidden units to better learn the silhouette features, but the results were usually on par with or slightly worse than the ones using 200 hidden units.

As future improvements, we could experiment with the reaction of the multi-layer perceptron to adding more than one hidden layer. We have only addressed the configurations from the exercise, but most state-of-the-art networks utilize multiple hidden layers. Moreover, we could test other cost and training functions and inspect their impact.