Classification and Approximation Problems

Karim Samim Karim
Tom Puech

October 16, 2017

Contents

1 Introduction **2**
1.1 Specifications . . . 2

2 Classification problem **3**
2.1 Introduction . . . 3
2.2 Method . . . 3
2.2.1 Normalising the input . . . 3
2.2.2 Deciding the optimal batch size . . . 4
2.2.3 Deciding the optimal number of hidden neurons . . . 4
2.2.4 Deciding the optimal decay parameter . . . 5
2.2.5 Designing a 4-layer network . . . 5
2.3 Discussion . . . 6
2.4 Conclusion . . . 7

3 Approximation problem **8**
3.1 Introduction . . . 8
3.2 Method . . . 8
3.2.1 Mini-batch gradient descent . . . 8
3.2.2 Deciding the optimal learning rate . . . 9
3.2.3 Deciding the optimal number of hidden neurons . . . 9
3.2.4 Designing a 4-layer and 5-layer networks . . . 10
3.3 Discussion . . . 11
3.4 Conclusion . . . 12

1 Introduction

Artificial neural networks is a system based on the human brain that learns how to do tasks. They can be used to learn various tasks in, for example, playing video games, computer vision, speech recognition and many other domains. In this case they will be used for solving an classification problem and an approximation problem.

1.1 Specifications

For this project, a computer with the following specifications has been used:

- Computer:
  - **CPU** Intel® Core™ i7-6700HQ Processor (6M Cache, up to 3.50 GHz)
  - **GPU** nVidia GeForce GTX 1060 (6GB GDDR5 1708 MHz, 1280 CUDA Cores)
  - **RAM** 16GB DDR4
- Frameworks:
  - **Python** 3.6 x64
  - **Theano** 0.9.0
  - **CUDA** 8.6.1

2 Classification problem

In this section, the method used to classify the Landsat satellite dataset will be presented and discussed.

2.1 Introduction

The first problem is to classify the Landsat satellite dataset. It contains multi-spectral values of pixels in 3x3 grids in a satellite image and the class label is associated with the central pixel in each grid. The goal is to predict the class labels in the test data after the neural networks are trained on the training data. To correctly predict the class labels, multilayer feedforward networks are used. They have many layers of perceptrons and can be used to correctly classify the inputs by using a softmax regression layer as an output layer.

2.2 Method

First of all, the training data and the test data are read from two files. Each data sample in the two files consists of a row of 37 values. The first 36 are the input attributes and the last one is the class label.

2.2.1 Normalising the input

After the data was read, a 3-layer feedforward neural network was designed consisting of a hidden-layer of 10 neurons with a logistic activation function and an output softmax layer. The learning rate α was set to 0.01, the decay parameter β was set to 10^{-6}, the batch size was set to 32 and the number of epochs was set to 1000. The number of epochs and the learning rate are unchanged in the classification problem. However, before training the data and testing it, the input needs to be scaled. Two scaling techniques were tested and the best one was chosen:

$$\tilde{x}_i = \frac{x_i - x_{i,min}}{x_{i,max} - x_{i,min}} \tag{1}$$

where $\tilde{x}_i$ is the scaled input, x_i is the input, $x_{i,min}$ is the minimum value in the input and $x_{i,max}$ is the maximum value in the input.

$$\tilde{x}_i = \frac{x_i - \mu_i}{\sigma_i} \tag{2}$$

where $\tilde{x}_i$ is the scaled input, μ_i is the mean of the input x_i and σ_i is the standard deviation of the input. *Equation 1* is used to scale the input such that $x_i \in [0, 1]$ and *equation 2* normalises the input so it has the standard normal distribution $x_i \sim N(0, 1)$. After testing both of them, *equation 2* gave the best results and was therefore always used to scale the inputs. The result can be viewed in *figure 1*.

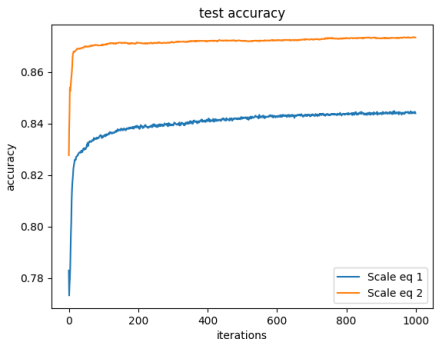

Figure 1: The test accuracy for the different scales.

2.2.2 Deciding the optimal batch size

The optimal batch size was also decided. This was done by evaluating the performances of different batch sizes for the neural network. The batch sizes tested were 4, 8, 16, 32 and 64. To find the most optimal one, the training error and test accuracy were plotted against the number of epochs for all the different batch sizes. The time taken to update certain parameters of the network for the different batch sizes was also done in order to determine the optimal batch size. The optimal batch size that was determined is 32 and the results are shown in *figure 2* and *figure 3*.

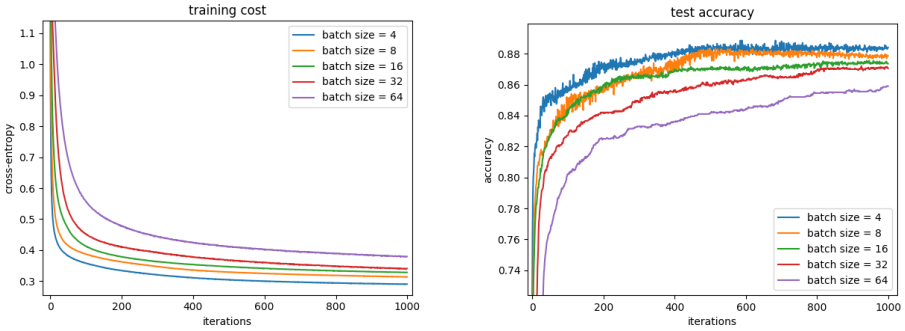

Figure 2: Training cost and accuracy of different batch sizes.

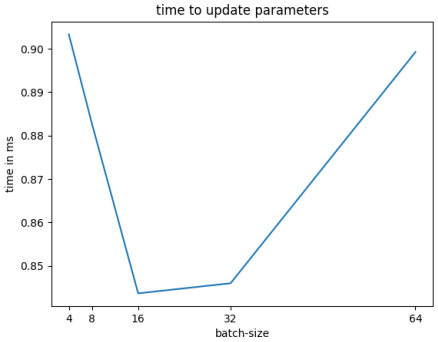

Figure 3: The time to update the parameters for different batch sizes.

2.2.3 Deciding the optimal number of hidden neurons

The optimal number of hidden neurons was also determined. It was decided the way as the batch size; by evaluating the performances of different sizes of the hidden neurons for the neural network. The size of the neurons in the hidden-layer tested were 5, 10, 15, 20 and 25. To find the optimal number of hidden neurons, similar plots were done as when deciding the batch size. Instead the different numbers of hidden neurons were plotted against the epochs and the time taken to update certain parameters. After the optimal number of hidden neurons was decided, it was always used. The optimal size of the hidden neurons determined was 25 and the results are shown in *figure 4* and *figure 5*.

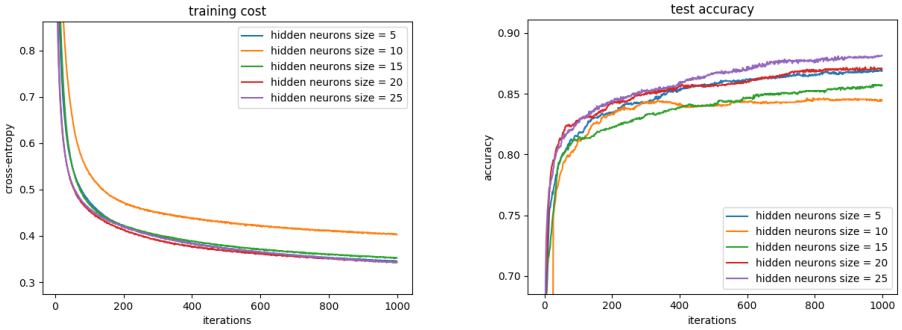

Figure 4: Training cost and accuracy of different hidden-layer neuron sizes

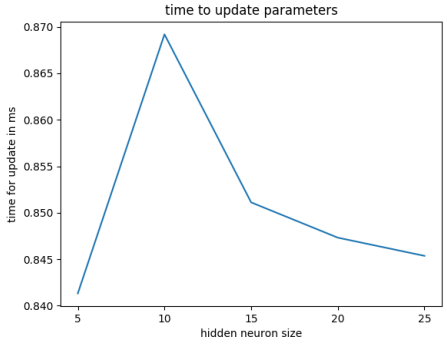

Figure 5: The time to update the parameters for different hidden-layer neuron sizes.

2.2.4 Deciding the optimal decay parameter

Afterwards the optimal decay parameter was decided. The values tested for the decay parameter were 0, 10^{-3}, 10^{-6}, 10^{-9} and 10^{-12}. The training error was plotted against the number of epochs and the test accuracy was plotted against the different decay parameters. Using this information, the optimal decay parameter was decided. The optimal decay parameter determined was 10^{-9} and the results are shown in *figure 6*.

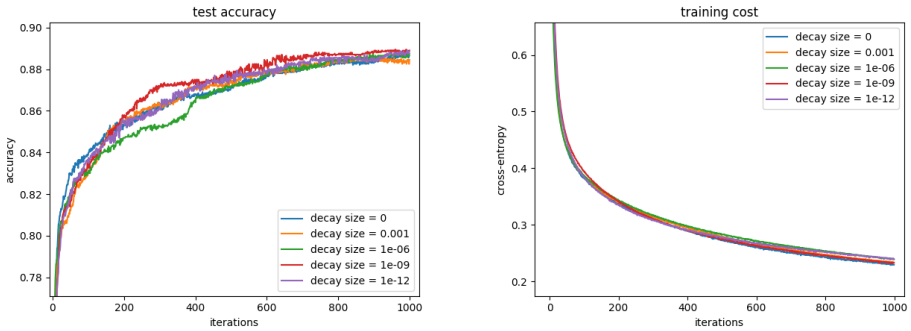

Figure 6: Training cost and accuracy of different decay parameters.

2.2.5 Designing a 4-layer network

The last thing done was designing a 4-layer network with 2 hidden-layers. Each hidden-layer consists of 10 neurons and both of them has a logistic activation function. The batch size was set to 32 and the decay parameter was set to 10^{-6}. Afterwards the training accuracy and

the test accuracy were plotted against the number of epochs. After this a 3-layer network with the same parameters (except that it only has one hidden-layer) and the 4-layer network were compared in terms of performance. The 3-layer neural network has better performance and the results of the 4-layer neural network is shown in *figure 7*.

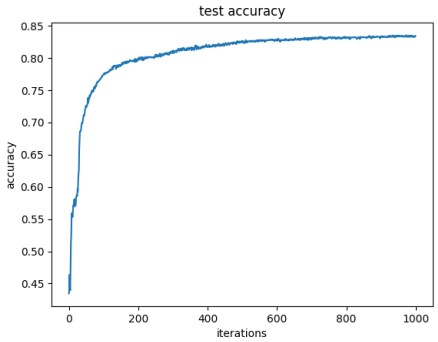

Figure 7: The test accuracy for the 4-layer network.

2.3 Discussion

The inputs are normalised because if they have similar variations, better approximation is achieved. Mainly, there are two approaches to scale the inputs and deciding the optimal one is done empirically. For some inputs *equation 1* is optimal and for others *equation 2* is optimal, in this case it was *equation 2*.

The optimal batch size was also determined in a empiric way, as all the different batch sizes were tested. But when the batch size increases, parallelism and matrix computations are exploited because there are more inputs in each batch but the computations per update increases. Therefore it is possible to predict how the batch size affects the time of the weight update, even though it is done empirically. The optimal batch size that was decided is 32. Its accuracy is not the best of the different batch sizes and the cross-entropy is not the lowest. However, the time to update the parameters is fast compared to the other batch sizes and the accuracy and cross-entropy does not differ too much from the others, which is why it was chosen.

The optimal number of hidden neurons in the hidden-layer was also decided. When the number of hidden neurons increases, the network attempts to remember the training pattern as it tries to lower the training error but it is also possible that the errors in the test data increases. When the number of hidden neurons in the hidden-layer increases the error in the test data decreases but usually it increases after a while. Therefore, the number of hidden neurons is decided empirically but with some knowledge of how increasing the number of neurons affect the data. The optimal hidden neuron size chosen is 25, because it has the best accuracy, the cross-entropy is one of the lowest and the time taken to update the parameters is also the second lowest.

After the optimal number of hidden neurons was determined, the optimal decay parameter was also decided. This was also done empirically, by plotting the graphs explained in the method section and comparing the decay parameters. By doing this the optimal decay parameter was found which is 10^{-9}. This value was chosen because it has the highest accuracy and the second lowest cross-entropy.

After the optimal decay parameter was found, a 4-layer network was designed and compared to a 3-layer network. The parameters used were the same, the only difference is the number of hidden-layers. The performance of the 3-layer network is the optimal one in

this case as seen in the results. This was done also done empirically, as they were both compared based on their performance. The one with the best performance is the 3-layer network because the test accuracy is higher when 3-layers are used.

2.4 Conclusion

Finding the optimal parameters is crucial when designing a neural network and finding the parameters was done empirically. This part of the project was helpful to learn how to find the optimal parameters for a classification problem.

3 Approximation problem

In this section, the method used to predict the housing prices in the California Housing database will be presented and discussed.

3.1 Introduction

The second problem is to use neural networks to predict housing prices in the California Housing database. It contains attributes of the housing complexes in California and one of these is their price. The goal is to predict the price of the houses in the test data after the neural networks are trained on the training data. Multilayer feedforward networks are used in order to predict the housing prices and in this case, the network finds a smooth function that is an approximation of the actual function to predict the housing prices.

3.2 Method

For this dataset only one file is provided, each data sample is a row of 9 values and the last value corresponds to the median price of the house. Thus, the last value has been used as target values and the other ones as inputs. The input features has been scaled and normalised to zero mean and unit standard deviation. A 5-fold cross validation was also used on the training data in order to select the best models. An example of a k-fold cross validation can be seen in *figure 8*.

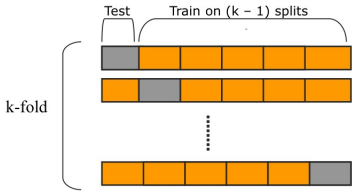

Figure 8: Illustration of k-fold cross validation.

All the test, train and accuracy curves were computed as the average of all the curves obtained through the k-fold cross validation and experiments.

3.2.1 Mini-batch gradient descent

After the data was read, scaled, and a 5-fold cross validation has been implemented, a 3-layer feedforward neural network was designed. This network is composed of a hidden layer of 30 neurons. The learning rate α was fixed to 10^{-4}, the batch size set initially to 32 but later changed to 256 and the number of epochs to 1000. Afterwards the training cost and the test accuracy were plotted against the number of epochs, see *figure 9*.

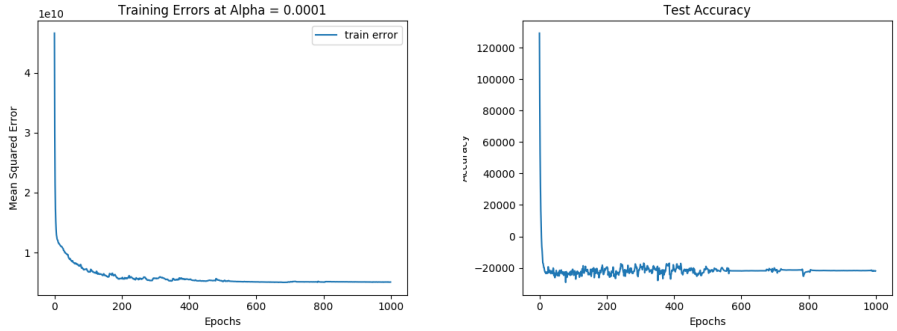

Figure 9: Training cost and test accuracy plotted against the number of epochs.

3.2.2 Deciding the optimal learning rate

In order to decide the optimal learning rate, a 5-fold cross validation was implemented as mentioned in the method section. To reduce the randomness of the algorithm, 10 experiments were done in order to ensure that an optimal parameter can be chosen, see *figure 10*.

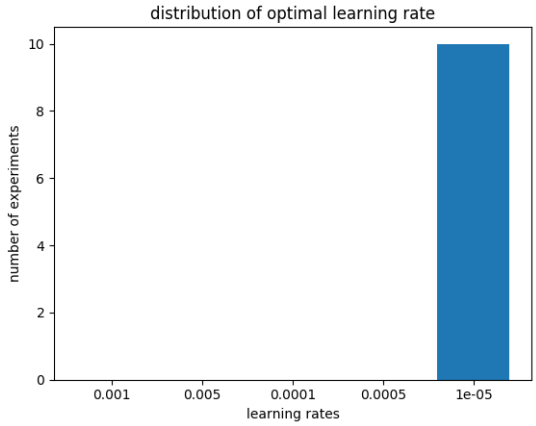

Figure 10: Distribution of optimal learning rate during 10 experiments.

After the 10 experiments, the train and validation errors were plotted against the number of epochs, see *figure 11*.

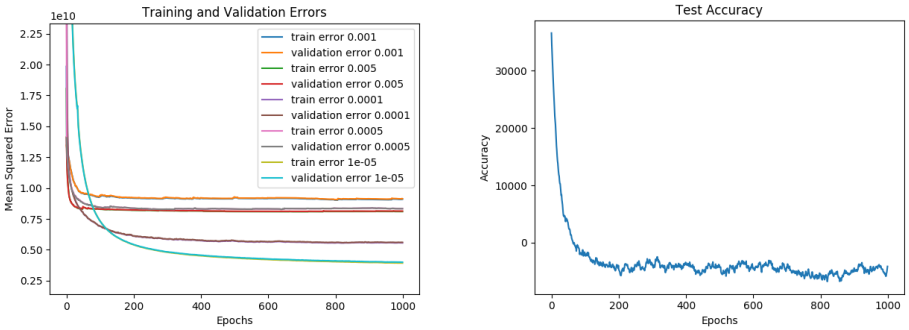

Figure 11: Training and validation errors, test accuracy against epochs.

3.2.3 Deciding the optimal number of hidden neurons

When deciding the optimal number of hidden neurons, a 5-fold cross validation was also used in this case and it was implemented as before. The batch size used here was also 256 and the number of epochs was also 1000. There were 5 different numbers of hidden neurons that were tested which are 20, 30, 40, 50 and 60. In order to decide the optimal one, 10 experiments were done to be sure that the chosen number of hidden neurons is optimal, see *figure 12*.

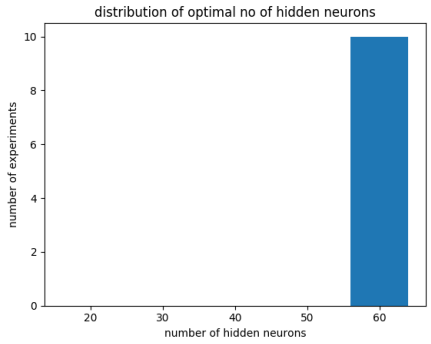

Figure 12: Distribution of optimal no of hidden neurons during 10 experiments.

After that the number of experiments were plotted against the number of hidden neuron which had the minimal test cost. The training error and the test accuracy were plotted against the number of epochs for each learning rate of hidden neurons, see *figure 13* and *14*.

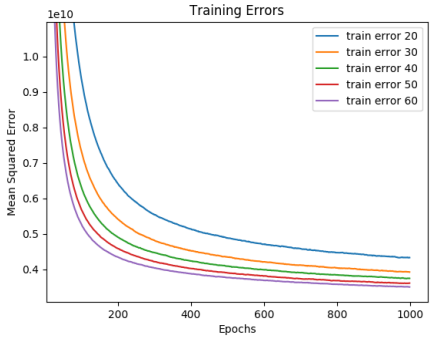

Figure 13: Training errors against epochs.

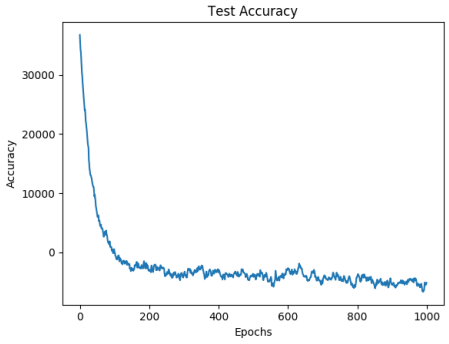

Figure 14: Test accuracy against epochs.

3.2.4 Designing a 4-layer and 5-layer networks

Afterwards a 4-layer network and a 5-layer network were also designed. The data was also normalised and a 5-fold cross validation was also used. The batch size used was 256, the learning rate α was fixed to 10^{-4} and the number of epochs was set to 1000. When the parameters were set and the networks were designed, the 4-layer network and the 5-layer

networks test accuracy were plotted against the number of epochs and compared to the 3-layer network, which had the same parameters. See *figure 15*.

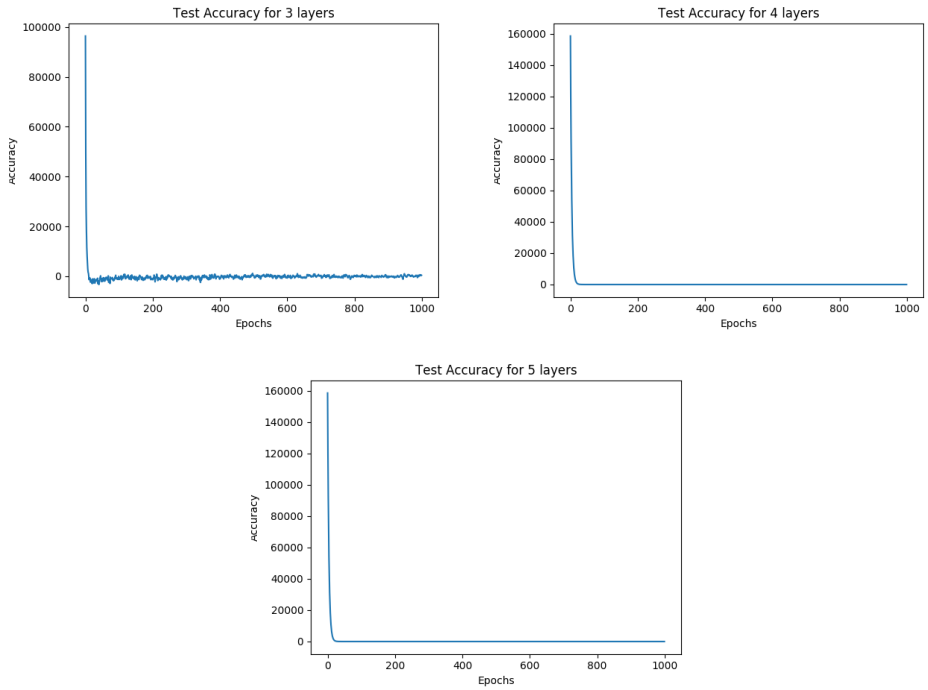

Figure 15: Test accuracy against epochs.

3.3 Discussion

Initially the batch size was set to 32 but the algorithm was too slow, therefore it was changed to 256 which made the algorithm run faster. Afterwards the optimal learning rate was decided. This was done empirically by comparing the training error and the test cost for all of the different learning parameters. This way the optimal learning rate could be determined and by looking at *figures 10* and *11*, the learning rate 10^{-5} was chosen.

After deciding the optimal learning rate, the optimal number of hidden neurons was set. This was determined in a similar way compared to the learning rate, because it was also done empirically by comparing the training error and the test cost for the different numbers of hidden neurons. Therefore the optimal number of hidden neurons was determined, which is 60, see *figure 12, 13* and *14*.

Finally, the optimal number of hidden layers was also determined. Neural networks consisting of 4-layers and 5-layers were designed as mentioned in the results section and compared to a 3-layer network. This was also done empirically as their test accuracy were plotted against the number of epochs to determine the optimal network. The optimal layer chosen was the 4-layer network because it was faster compared to the 5-layer network. However, the 5-layer network had better test accuracy, but it was negligible.

With all the optimal parameters the designed neural network was able to approximate the price of the house with an error less than 100.

3.4 Conclusion

This part of the project was helpful to observe the impact of the parameters on the computing time. The batch size and the number of experiments influences the computing time. On top of that, this part of the project was helpful to learn how to find the optimal parameters for a mapping problem.