A4 - ARTIFICIAL NEURAL NETWORKS - TICKETING ROUTING AGENT

20 Decemberr 2019

190020774
University of St. Andrews
CS5011 Artificial Intelligence Practice

Contents

1 Introduction

1.1 Abstract

Artificial neural networks (ANNs) can be used to make predictions on previously unseen data related to a particular model. There are a number of methods to train these networks in order to make accurate predictions, one of which being backpropagation.
In this assignment, we were tasked with creating and training an ANN designed for an IT help desk in which tickets are logged. Based on the nature of the issues provided in the ticket, this is then assigned to a particular team best equipped to resolve the situation.

1.2 Parts Implemented

For this assignment, I have implemented the basic and intermediate agents.

1.3 Compiling and Running Instructions

If running the following programs on the lab machines, provided you have not installed the required dependencies already, you will have to type the following commands to ensure the program works correctly:

```
pip3 install --user scikit-learn
pip3 install --user pandas
pip3 install --user numpy
pip3 install --user numpy==1.17.0
```

To run the provided programs, navigate to the /src directory and type the following commands based on which agent you require:

```
python3 A4main.py Bas
```

For the basic agent.

```
python3 A4main.py Int
```

For the intermediate agent.

2 Design, Implementation and Evaluation

2.1 PEAS Model

The PEAS model for the neural network is as follows:

- Performance Measure: Number of correct predictions made by the network
- Environment: The structure of the network itself, including but not limited to: Number of input units, number of hidden layers hidden neurons
- Actuators: The method used for learning. In this network this is the backpropagation algorithm
- Sensors: Being able to receive inputs. In this case the network must be able to read in and encode a CSV file.

2.2 Problem Definition

The problem definition for the basic agent is fairly straightforward, as a single network must be instantiated, trained and saved to a file.
For the intermediate agent, a text-based interface was required to be created, so that users could pass in their own information into the network and receive a prediction. This agent must also be able to handle incomplete datasets in order to make an early prediction.

2.3 System Architecture

The agents in this assignment were developed in Python rather than Java, which had been used previously. As such, the system is composed of a main script, from which depending on the user's choice either the basic or intermediate agent classes will be instantiated, and the relevant functions will be called.

2.3.1 Basic Agent

The basic agent consists of a single function `trainNetwork(self)` which reads in the training data from the CSV file, and encodes the values appropriately. Since the inputs to the network are yes / no questions, this can be achieved using a binary encoding: 1 for yes, 0 for no. We do not need to make any changes to the outputs since the network is able to handle string values.
Once the encoding has been performed, the network is created with its relevant hyper-parameters and number of hidden neurons. Choices for these are discussed in the performance and analysis section. The network is then trained with the training data, stopping

when the maximum number of iterations has been reached or when the no-change tolerance limit has been reached. Once training has been completed, the network is then saved to a file.

2.3.2 Intermediate Agent

For the intermediate agent, a command line tool was created allowing the user to provide their own values for each input tag. Furthermore, after a minimum of 4 inputs the user is then asked if they would like to make an early prediction. As making an early prediction does not require all the input tags to be specified, the remaining inputs need to be defined such that the network is able to handle this input. One method of achieving this is by filling the remaining tags with some kind of average of the existing data. As inputs should be either 0 or 1, it would not make sense to use the mean or median, as this would produce non-integer values. Therefore, if an early prediction is requested, the remaining values are replaced with the mode of the provided values.

Once the inputs have been taken from the user, the neural network is instantiated by loading the file created by the basic agent. The inputs are passed into this network and a prediction is then made. The user is then asked if the network has made the correct prediction. If it has, then the ticket routing is complete and the agent is ready to receive another ticket. If not, then the user is asked to provide the correct response team. These inputs and the correct response team are then appended to the training table, and the network is then retrained and updated based on this extra information. Once training has completed, the agent will then ask the user for another ticket.

2.4 Performance and Analysis

2.4.1 Constructing the Neural Network

The neural network in this assignment is a feedforward network with one hidden layer. From this, we need to determine the appropriate number of hidden units and relevant hyperparameters (learning rate, momentum) in order to minimise the loss.

Since there are 9 inputs into the network, a base point of 9 hidden neurons was first tested with learning rate and momentum both set to 0.1. The loss curve of this network was then compared to the loss given by networks with different numbers of hidden neurons, whilst keeping all other parameters constant. These results are shown below.

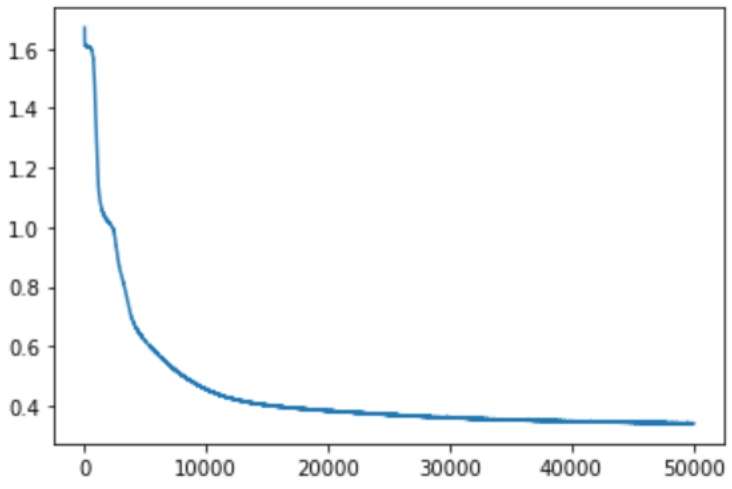

Figure 1: 9 Hidden Neurons

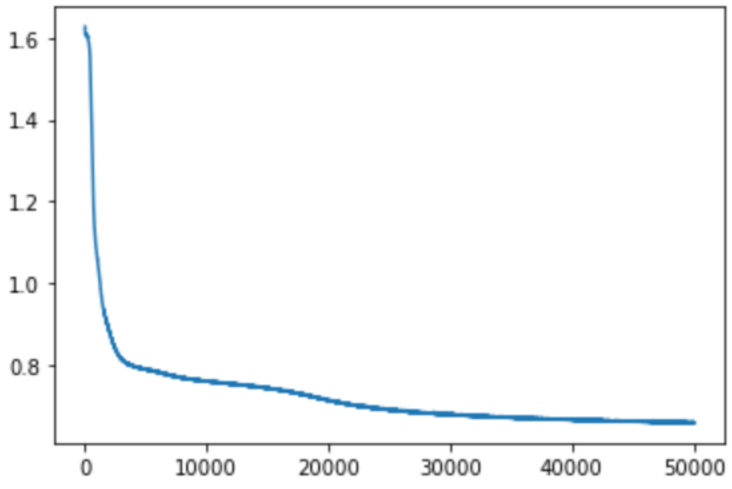

Figure 2: 10 Hidden Neurons

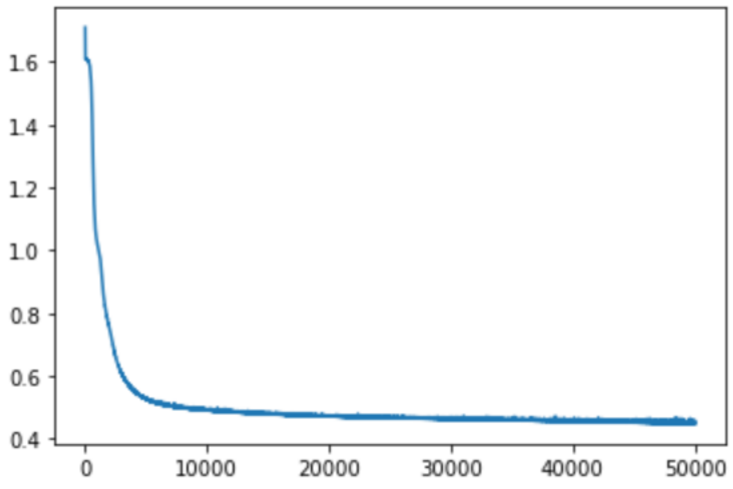

Figure 3: 8 Hidden Neurons

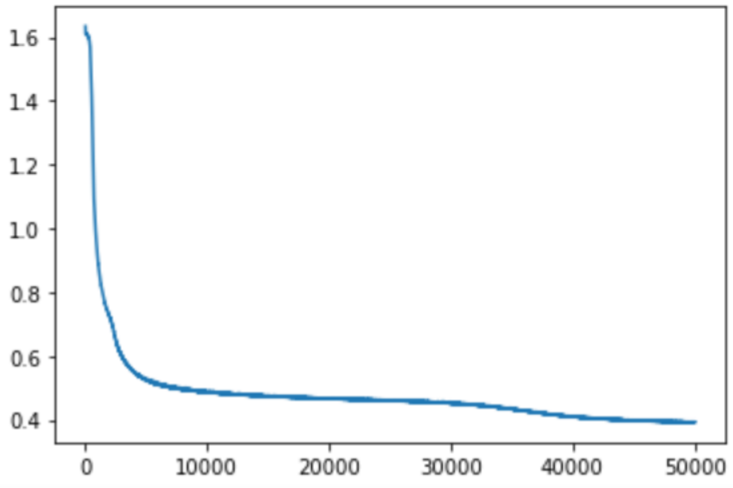

Figure 4: 7 Hidden Neurons

From these results, we can see than the number of hidden neurons which result in the lowest loss is 7. The relevant hyperparameters then need to be decided. Both learning rate and momentum should be in the range [0,1]. Keeping momentum at 0.1 as in the previous cases, loss curves were found for learning rates of [0.1, 0.3, 0.5, 0.7, 0.9]. These are shown below.

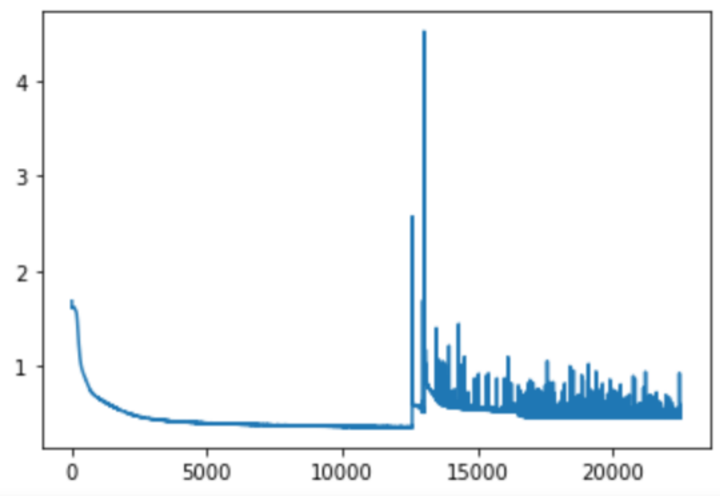

Figure 5: 7 Hidden Neurons, Learning rate = 0.3, Momentum = 0.1

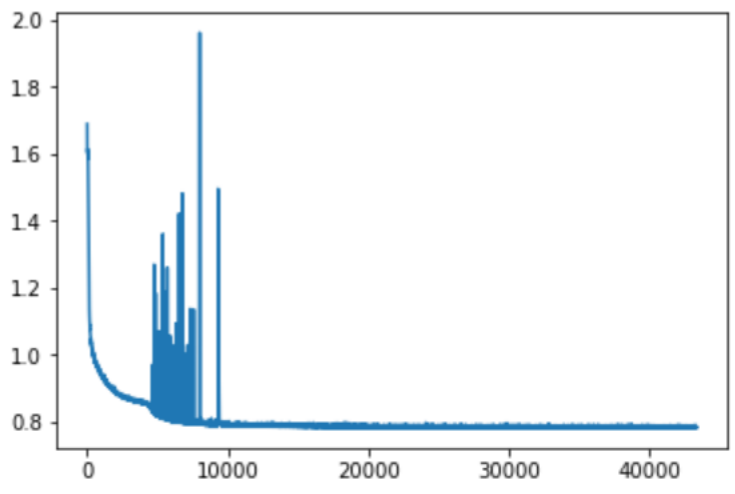

Figure 6: 7 Hidden Neurons, Learning rate = 0.5, Momentum = 0.1

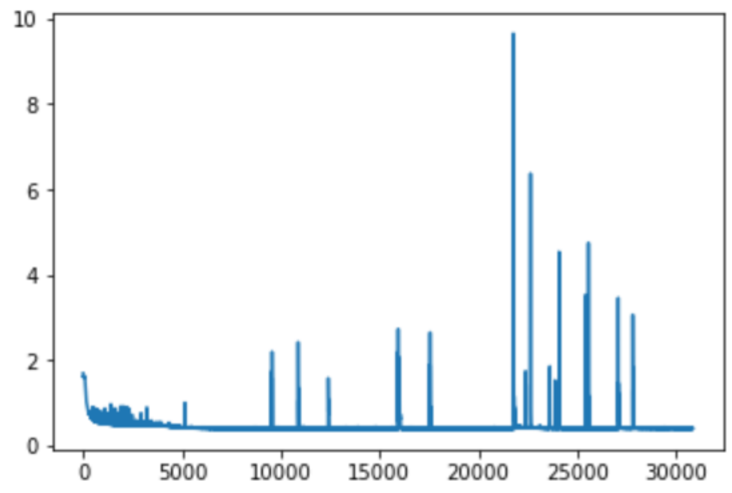

Figure 7: 7 Hidden Neurons, Learning rate = 0.7, Momentum = 0.1

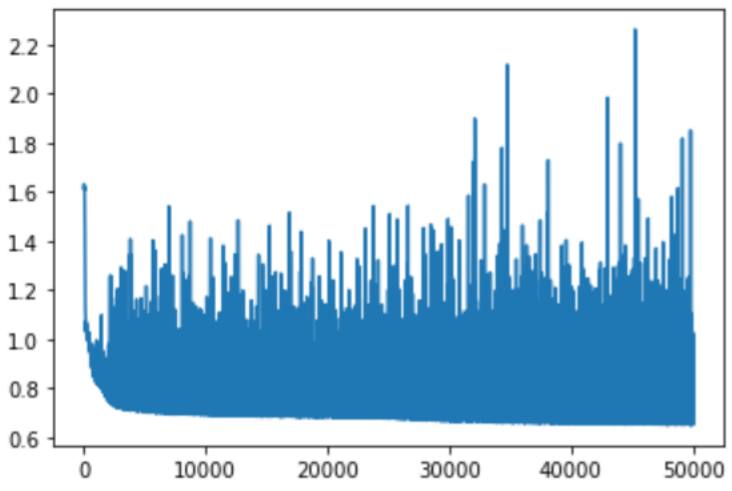

Figure 8: 7 Hidden Neurons, Learning rate = 0.9, Momentum = 0.1

Keeping learning rate at 0.1, the most appropriate momentum value was then found. Curves for the momentum values of [0.09, 0.07, 0.05, 0.03, 0.01] are shown below.

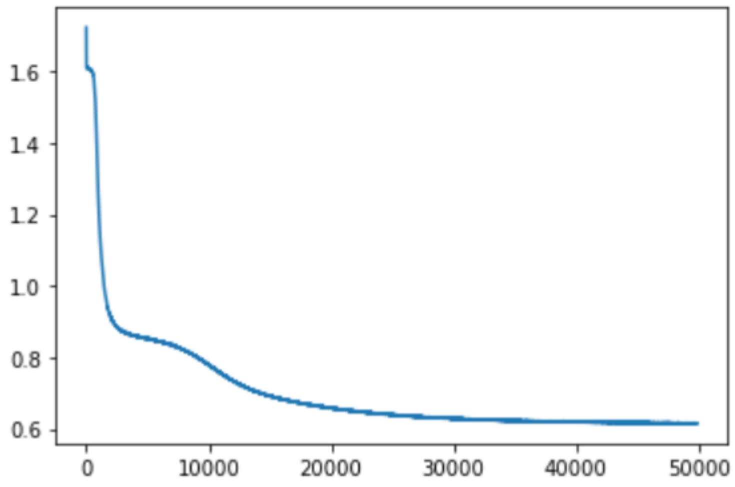

Figure 9: 7 Hidden Neurons, Learning rate = 0.1, Momentum = 0.09

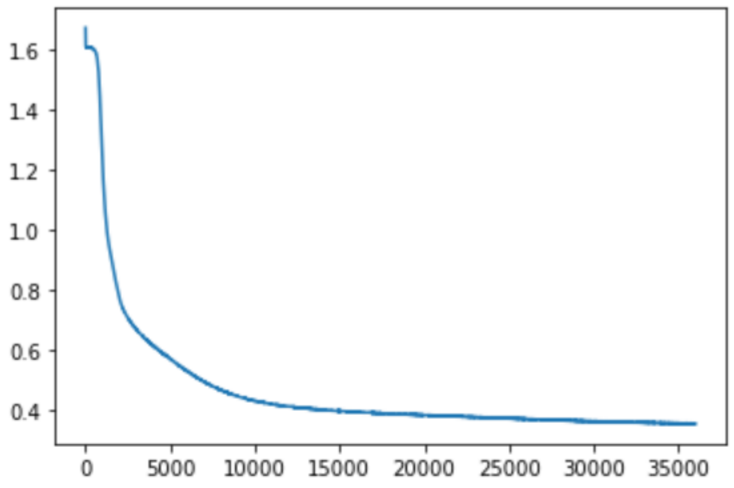

Figure 10: 7 Hidden Neurons, Learning rate = 0.1, Momentum = 0.07

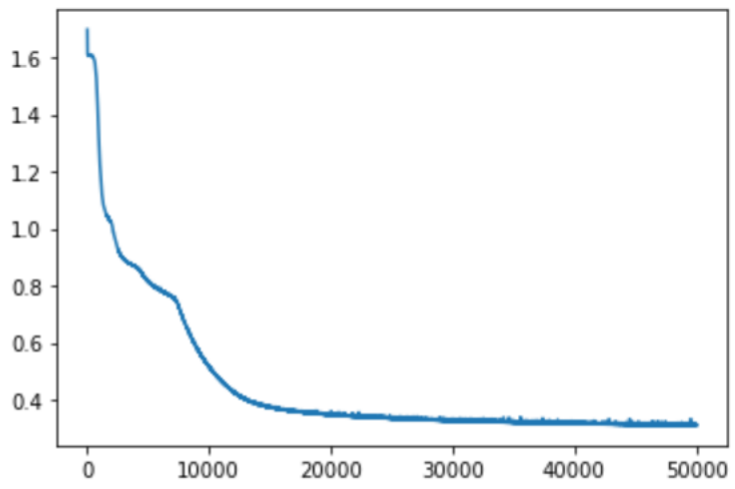

Figure 11: 7 Hidden Neurons, Learning rate = 0.1, Momentum = 0.05

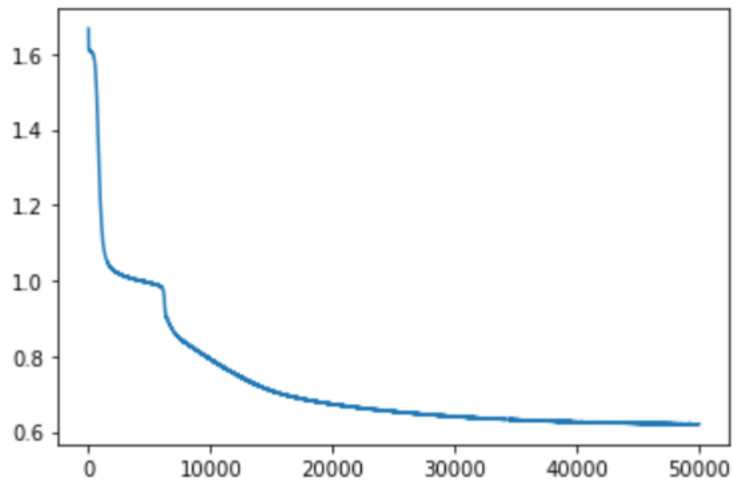

Figure 12: 7 Hidden Neurons, Learning rate = 0.1, Momentum = 0.03

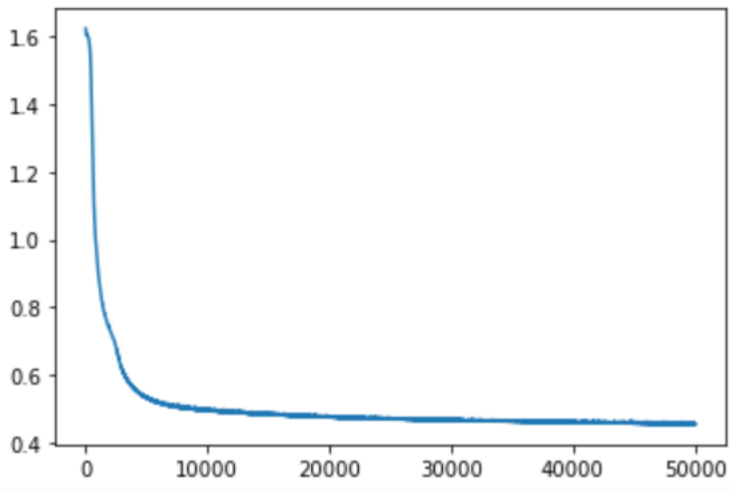

Figure 13: 7 Hidden Neurons, Learning rate = 0.1, Momentum = 0.01

From this, we can see that the most appropriate neural network for this problem has 7 hidden layers, a learning rate of 0.1, and a momentum of 0.05.