MLP Coursework 1 ()

MLP Coursework 1: Activation Functions

s1784849

Abstract

The abstract should be 100–200 words long, providing a concise summary of the contents of your report.

1 Introduction

As neural networks have become become increasingly popular over the last decade(**?**) it has become important to investigate and understand the effects popular activation functions have on the training process, and how different activation functions can be used to achieve state of the art level performance on well understood research problems. Furthermore, the emergence of deep learning(**?**)(**?**) has provided ample motivation for investigating the effects of employing many hidden layers within neural network architectures.

This report provides investigations into two variable aspects of neural networks that effect the potential accuracy and error achieved by these models. Initially, before any investigations are presented, an introduction to the set of activation functions that are employed by neural networks utilised within this report is provided. An experimental comparison of these activation functions is given, along with a sensitivity analysis that provides tuning for the hyper parameter usually labelled as the step size (or learning rate)(**?**). In addition, an investigation into varying the depth of a neural network is presented and follow up conclusions about the effects this has on the potential accuracy attained by the networks is proposed.

The neural networks used for the experiments within this report are trained on the MNIST dataset. The training set is made up of 50,000 examples and the test set (which the validation examples are drawn from) is of length 10,000.

The fundamental research questions this report aims to address are outlined below.

- Two hidden layer network: what is an optimal setting for the hyper parameter known as the step size (or learning rate) on the given activation functions?
- Two hidden layer network: which activation function provides the fastest rate of convergence towards a minimum error and maximum accuracy?
- Two hidden layer network: which activation functions provides the highest ceiling with regards to potential accuracy that can be acquired?
- N-hidden layer network: how does varying the number of hidden layers within our neural network architecture affect the overall model accuracy?

2 Activation functions

The activation functions utilised within this report are leaky rectified linear units, exponential linear units(**?**) and scaled exponential linear units(**?**). Comparisons against one another, in addition to baseline rectified linear units and sigmoid units will be presented in the next section. This section provides an introduction the aforementioned activation functions.

2.1 Sigmoid

Within this report a standard sigmoid function is used to provide a simple baseline to compare other activation functions to. The sigmoid activation function is defined as the following:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

With gradient:

$$\frac{d}{dx}\sigma(x) = \sigma(x) \times (1 - \sigma(x)). \tag{2}$$

2.2 Rectified Linear Unit

The ReLU activation function was proposed as an alternative to the sigmoid function due to it's ability to solve the vanishing gradient problem(**?**). These units are defined as follows:

$$\text{relu}(x) = \max(0, x), \tag{3}$$

With gradient:

$$\frac{d}{dx}\text{relu}(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0. \end{cases} \tag{4}$$

2.3 Leaky Rectified Linear Unit

Although the ReLU activation function helps protect against vanishing gradients during training, it introduces other problems that must be addressed. One such problem is the introduction of dying ReLUâĂŹs, where the function will output zero in the scenario where the weighted sum of the inputs is negative. Once the unit is in this state, itâĂŹs capability for learning ceases as the gradient of any negative input results in zero, which leads to the weight updates equating to zero. LReLUs (or parameterised ReLUs - PReLUs)(**?**) attempt to mitigate this by introducing a non-zero gradient, in the case of LReLU the hyper parameter, alpha, is typically set to 0.01. In the case of PReLU, alpha is a parameter that should be selected or tuned. LReLU is defined as the following:

$$\text{lrelu}(x) = \begin{cases} \alpha x & \text{if } x \leq 0 \\ x & \text{if } x > 0. \end{cases} \tag{5}$$

With gradient:

$$\frac{d}{dx}\text{lrelu}(x) = \begin{cases} \alpha & \text{if } x \leq 0 \\ 1 & \text{if } x > 0. \end{cases} \tag{6}$$

2.4 Exponential Linear Unit

ELUs provide similar functionality to LReLUs and PReLUs, but aim to increase learning speed in order to reach higher classification accuracies(**?**). The ELU activation function is defined as:

$$\text{elu}(x) = \begin{cases} \alpha(e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0. \end{cases} \tag{7}$$

which has the gradient:

$$\frac{d}{dx}\text{elu}(x) = \begin{cases} \text{elu}(x) + \alpha & \text{if } x \leq 0 \\ 1 & \text{if } x > 0. \end{cases} \tag{8}$$

2.5 Scaled Exponential Linear Unit

The SELU(**?**) activation function is a variation on ELU that is defined by:

$$\text{selu}(x) = \lambda \begin{cases} \alpha(e^x - \alpha) & \text{if } x \leq 0 \\ x & \text{if } x > 0. \end{cases} \tag{9}$$

which has the gradient:

$$\frac{d}{dx}\text{selu}(x) = \lambda \begin{cases} \alpha(e^x) & \text{if } x \leq 0 \\ \lambda & \text{if } x > 0. \end{cases} \tag{10}$$

Figure 1: Training and validation error, with training and validation accuracy plotted agaisnt each epoch for step sizes: 0.05, 0.1, 0.2 and 0.5.

3 Experimental comparison of activation functions

The neural network model used within the initial experiment that compares activations functions was trained using 2 hidden layers, each containing 100 units. A soft-max layer was applied to the output which contained 10 units in order to classify digits from the MNIST dataset from 0 to 9. A batch size of 50 and stats interval[1] of 1 was used across all experiments.

Throughout the experiments the sigmoid and ReLU activation functions were used as baselines to compare with the performance of the other activation functions. However, it is shown in figure **??** that any ReLU variant function[2] consistently outperforms a standard sigmoid function[3]. For this reason, the sigmoid baseline is only shown in selected figures and tables presented within this report.

It is shown that although ReLU variant functions consistently outperform a sigmoid baseline, there are occurrences of overfitting the test set when measuring error on the validation set. This is conveyed in figure **??** within the validation error column. The sigmoid line that is plotted (marked in blue) eventually crosses over the ReLU variant function lines, demonstrating that the ReLU variant functions are more sensitive to overtraining. This suggests that employing a technique such as early stoping is particularly important for these activation functions. For this reason measurements in error presented within tables throughout this report will typically be taken within 20 to 40 epochs after the experiment starts, this will be explicitly stated.

In order to find an appropriate value for the step size a sensitivity analysis was performed. The values 0.05, 0.1, 0.2 and 0.3 were used evaluate the performance of the network. In order to gauge a more reliable value for the step size, the experiments were repeated a total of five times[4] and the mean value for error and accuracy was taken across all of these experiments for each epoch. The following seeds were used for each experiment:

- 6102016
- 1237878
- 2673679
- 8978283
- 5627351

A value of 0.2 was chosen for the step size because this value provided the highest ceiling with regards to the level of accuracy that could be reached, see figure **??** and table **??**.

Figure **??** shows the training and validation error, and figure **??** presents the validation accuracy of all four ReLU variant functions.

[1] The interval at which statistics are recorded across the experiment, measured in epochs.

[2] ReLU variant functions reference ReLU, LReLU, ELU and SELU in this report.

[3] By standard sigmoid, this report is referring to the function described within the second section, as a tunable sigmoid may perform better.

[4] Ideally, experiments of this nature should be performed more than five times in order find a more reliable value for any particular hyperparameter, in this instance the experiments were repeated five times due to time constraining factors.

Figure 2: Validation accuracy for various step sizes (sensitivity analysis).

Activation Function	Learning Rate	Validation Error (15 epochs)
ReLU	0.2	0.094
LReLU	0.2	0.090
ELU	0.2	0.088
SELU	0.2	0.101

Table 1: Validation accuracies for a 2 hidden layer neural net using various activation functions on the MNIST dataset.

4 Deep neural network experiments

Through varying the number of hidden layers within our neural network architecture (ranging between three and six hidden layers), it is possible to plot the training accuracy and validation accuracy for each layer increment. The learning rate was kept at a value of 0.2 and the ELU activation function was chosen, as it provided the best performance in the previous experiment. It was found that the accuracy generally levels out to the same point regardless of the number of layers. However, it is shown in figure **??** that the more hidden layers that are used, the longer the time to convergence. The behaviour of a deeper network appears similar to initially increasing the step size.

Figure 3: ReLU, LReLU, ELU and SELU errors plotted against each epoch.

Figure 4: ReLU, LReLU, ELU and SELU accuracy plotted against each epoch.

Figure 5: Relationship between the number of layers within a deep neural network architecture and the training and validation accuracy.

5 Conclusions

Despite the many experiments ran in order to produce this report, the experimental results do not provide a clear narrative. It was expected that the ReLU variant activation functions would outperform the standard sigmoid function, which was indeed the case. However, the performance of each ReLU variant function did not differ much between one another for this particular problem, using this particular data (MNIST character recognition). It was expected that there would be a more noticeable difference within the result set for the performance of each function.

Nevertheless, it was shown that the optimal learning rate for this particular problem lies within the region of 0.1 and 0.2. In addition, the ReLU variant activation functions consistently outperform a standard sigmoid function with regards to speed of convergence, however the ReLU variant functions are particularly sensitive to overtraining, indicating that early stopping is important when implementing these activation functions. Although the variation between ReLU, LReLU, ELU and SELU is almost negligible, LReLU performed the best with respect to fastest convergence, whereas ELU provided the best performance with respect to minimum error and maxi-

mum accuracy.

It was found that additional hidden layers within the outlined network architecture do not provide any additional boost in performance (specifically accuracy) for the particular problem this report aims to address. In addition, it has been shown to slow down the rate of convergence. Whilst it may appear advantageous to use fewer layers (two or three instead of five or six), it is advised to consider the complexity of the function to be approximated before considering whether or not to use additional layers. In this instance, it did not appear necessary, but deep neural networks have provided significant benefits to other problems in the past(**?**)[5].

5.1 Future Work

An investigation into the modification of weight initialisation methods was going to be included within this report, but time constraining factors means that it is likely to appear within future work.

[5]Different (and more complex) network architectures are typically used in other papers and reports, for example the use of convolutional layers, pooling layers, etc.