MLP Coursework 1: Activation Functions

s1778365

Abstract

Training a Deep Neuron Network (DNN) is a sufficiently difficult task, it is really important to have a great activation function and to initialise hyperparameters properly. In this report, we are going to compare 5 different activation functions which are Sigmoid Unit, ReLU (Rectified Linear Unit), LReLU (Leaky Rectified Linear Unit), ELU (Exponential Linear Unit) and SELU (Scaled Exponential Linear Unit) based on the famous dataset MNIST (Modified National Institute of Standards and Technology database). And then, the activation function SELU will be used to explore the compact of the number of hidden layers and the effect of different initialisation strategies as well.

1. Introduction

There were a lot of optimizations have been introduced in order to improve the performance of deep neuron networks during last ten years. These optimizations include getting a better activation function, error function, initialisation strategy, weights updating algorithm and so on, all of which are of critical importance for training a DNN effectively and precisely. And this report will give you a clearer understanding of how a DNN is trained and how the activation function, number of hidden layers and initialisation strategy influence the behavior of DNN.

We will use MNIST dataset to conduct all the experiments. The data in MNIST is divided into two components, training data set (50000) and validation data set (10000). In this report, we will compare five activation functions (Sigmoid, ReLU, LReLU, ELU and SELU) by implementing DNN with 2 hidden layers and each hidden layer has 100 units. The learning rate for this experiment will be set to 0.1. And then, SELU activation function will be used to implement DNN with different numbers of hidden layers varying from 2 to 8. The learning rate will be set to 0.5. At last, different initialisation methodologies including Glorot and Bengio's combined initialisation and Gaussian distribution initialisation will be evaluated using a DNN with 8 hidden layers. And we will modify the learning rate to 0.03. The modification of the learning rate aims at avoid overflow during training process.

2. Activation functions

Apart from Sigmoid Unit, there are 4 more activation functions that we will use in our experiments, which are ReLU, LReLU, ELU and SELU. The mathematic expression of these activation functions is demonstrated blow. ReLU:

$$\text{relu}(x) = \max(0, x), \tag{1}$$

which has the gradient:

$$\frac{d}{dx}\text{relu}(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0. \end{cases} \tag{2}$$

LReLU:

$$\text{lrelu}(x) = \begin{cases} ax & \text{if } x \leq 0 \\ x & \text{if } x > 0. \end{cases} \tag{3}$$

which has the gradient:

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

ELU:

$$\text{elu}(x) = \begin{cases} a(exp(x) - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0. \end{cases} \tag{5}$$

which has the gradient:

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

SELU:

$$\text{selu}(x) = \lambda \begin{cases} a(exp(x) - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0. \end{cases} \tag{7}$$

which has the gradient:

$$\text{selu}(x) = \lambda \begin{cases} a(exp(x) - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0. \end{cases} \tag{8}$$

During these experiments, the program may have some overflow errors as exponential function output increases dramatically when input increases. Therefore, it is extremely important to calculate the exponential outputs of negative inputs only when we are implementing activation functions.

3. Experimental comparison of activation functions

The selection of the activation functions for DNN is critically important, a proper activation function will make the training process more effective and convergent. Although sigmoid unit has been successfully applied in DNN, but the truth is that it is only linear at a small range and will easily get saturation when the inputs have large magnitudes. Therefore, RELU was used later to overcome this problem and was proved to be more effective in many applications. However, the explorations on activation functions are still very meaningful and in this experiment, LReLU, ELU and SELU were used to make comparisons with sigmoid unit and ReLU.

At the first stage, these five activation functions were used to implement DNN containing 2 hidden layers with 100 units at each layer. In addition, in order to control the variance, all the other hyperparameters was kept unchanged and only one type of activation functions was used at every DNN.

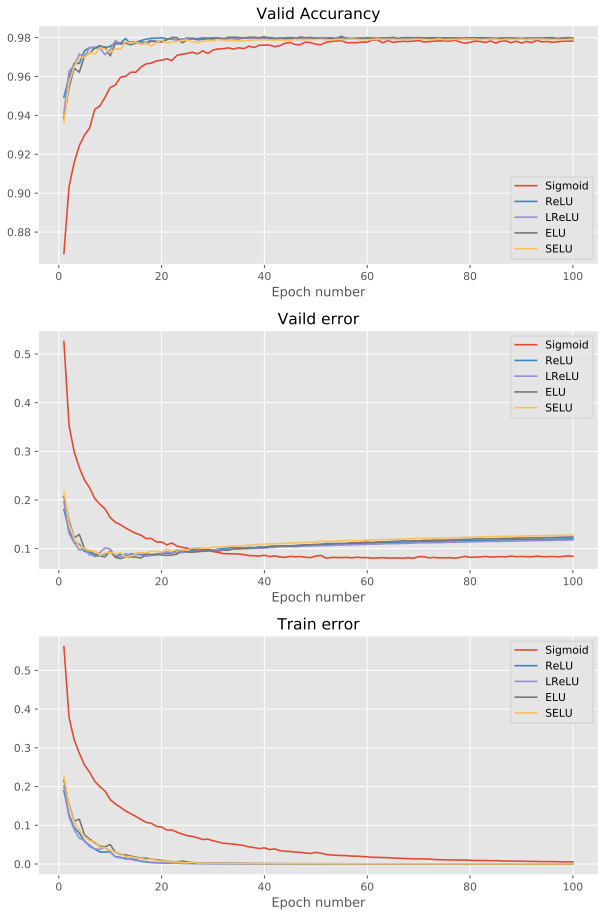

Figure 1. Classification accurancy, validation error of validation dataset, and training error of training dataset. Five activation functions were compared with each other.

From the figure 1, it is clearly shown that all the activation function has almost the same performance in classification. However, the DNN implemented with sigmoid units was more difficult to train, it took around 60 epochs to achieve the final accuracy while other activation functions only consumed 20 epochs. As the learning rate was exactly the same for all the DNN, it is fair to conclude that DNN implemented by sigmoid units learn slower than other activation functions. The reason is mentioned above, sigmoid unit is easy to get saturation when inputs have large magnitude. As a result, even some weights are extremely far from the ideal ones, they can only 'move' to the answer step by step. At the same time, the figure of validation errors illustrates that the DNN implemented with ReLU, LReLU, ELU and SELU were slightly overfitting. The error curves start to raise after 20 epochs. However, it is not able to figure out whether the DNN implemented with sigmoid unit was overfitting or not, so I have conducted an extra training on this DNN for 300 epochs. As demonstrated in figure 2, it seems that the DNN starts to be overfitted slowly after 100 epoch. And this trend was also caused by the saturation characteristics of the sigmoid unit. After the DNN was trained accurately, a lot of the weights have large magnitude, so the weights were changing slowly.

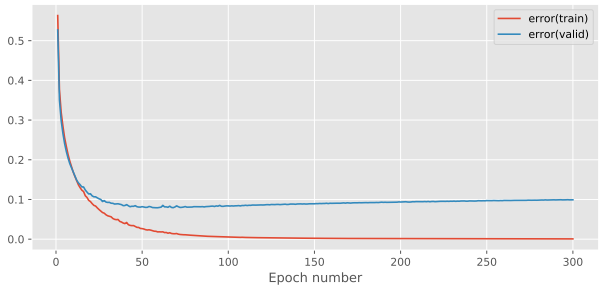

Figure 2. Validation error of a DNN using sigmoid units. The network was trained for 200 epochs

When comparing the activation functions LReLU, ELU and SELU, it is illustrated in the figure 1 that they almost had the same and excellent performance. But it is noticeable that their training errors were more fluctuant during the training process. Among these three activation functions, it seems that SELU had best performance as it has a more stable (convergent) training error curve. In fact, SELU have the characteristics to control the mean, deampen and increase the variance when necessary(G. Klambauer & Hochreiter, 2017). So it is totally reasonable to use SELU to explore following issues.

At the second stage, comparisons were conducted among the DNN implemented with two of the activation functions from LReLU, ELU and SELU, while other conditions remained the same. From figure 3 we can see that all of the models had outstanding performance, there were just slightly differences among different models. The truth is that the MINST dataset is just a simple dataset, so it have not sufficient data to enable those models to reveal greatly

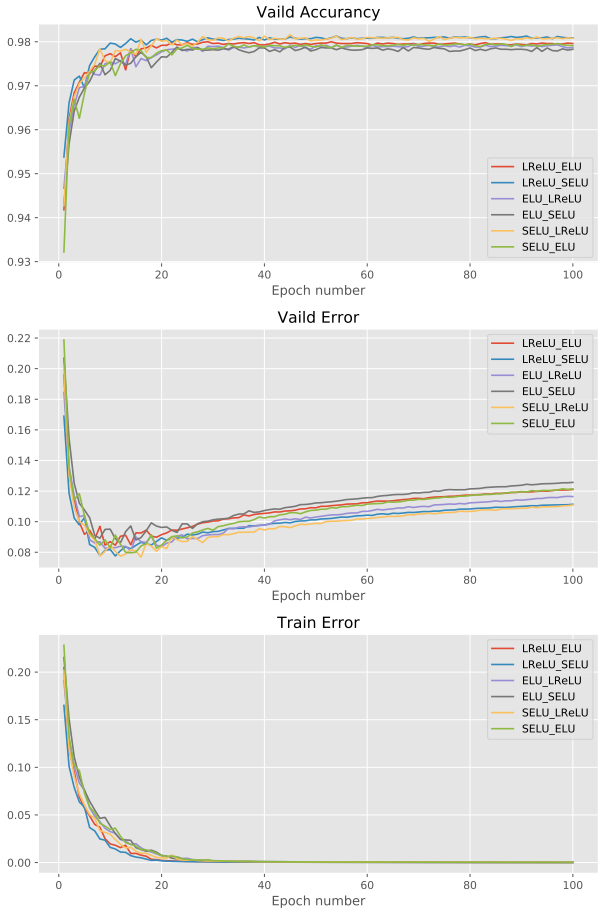

Figure 3. Classification Accurancy and error of validation dataset, and training dataset. In this experiment, five activation functions were compared.

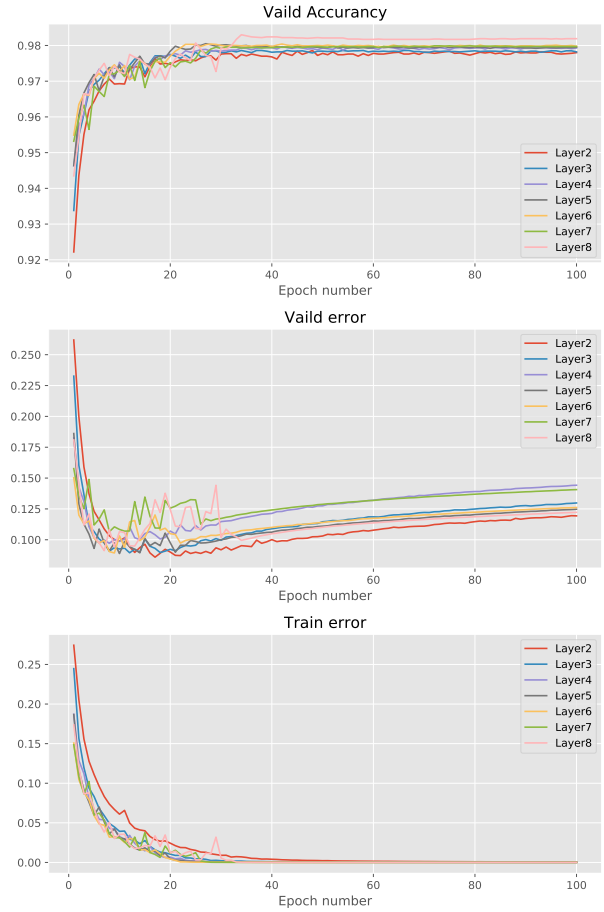

Figure 4. Classification accurancy, validation error of validation dataset, and training error of training dataset. Differnt depth of DNN weere trained.

different performance. But we can still found out that the DNN implemented with SELU and LReLU actually had the highest validation accuracies which are slightly larger than 98

4. Deep neural network experiments

4.1. The impact of the depth of the network

It is generally agreed that the more hidden layers a DNN has, the more flexible it is. In this section, we will compare DNN with different numbers of hidden layers (from 2 to 8) and figure out what is the impact of the depth of the network. And only the activation function SELU was used to implement the DNN.

As illustrated in figure 4, the validation accuracy was improved slightly when the depth of the network increased. And the DNN with 8 layers had the highest classification accuracy. But the tradeoff is that it was more difficult to be trained accurately. From the training error curves, we can also realise that the more hidden layers the DNN has, the more fluctuant will its training process be, which is the second disadvantage. The reason why the networks became more and more unstable during training process is that the flexibility of the networks have been increased, and thus, the networks were more sensitive to the noise from the inputs. As a result, the weights were sometimes modified to cover the noise as well. Therefore, it took more time for the weights to be trained precisely. However, the accuracy of the networks was improved at the same time as the networks were able to catch more features.

4.2. The impact of the different intialisation strategies

Proper initialisation is of vital importance to have a convergent and effective DNN. In practical, Glorot and Bengio's combined initialisation was proven to work quiet successfully in most cases. But it is worthy to explore some other more effective strategies. Considering the DNN with 8 hidden layers have the highest validation accuracy, this network was used to evaluate different intialisatoin strategies. In this section, four kinds of strategies will be compared, which are shown in table 1.

From figure 5 we can draw the conclusion that Xavier initialisation and Fanout initialisation are better than the other two methodologies. However, to be honest, their perfor-

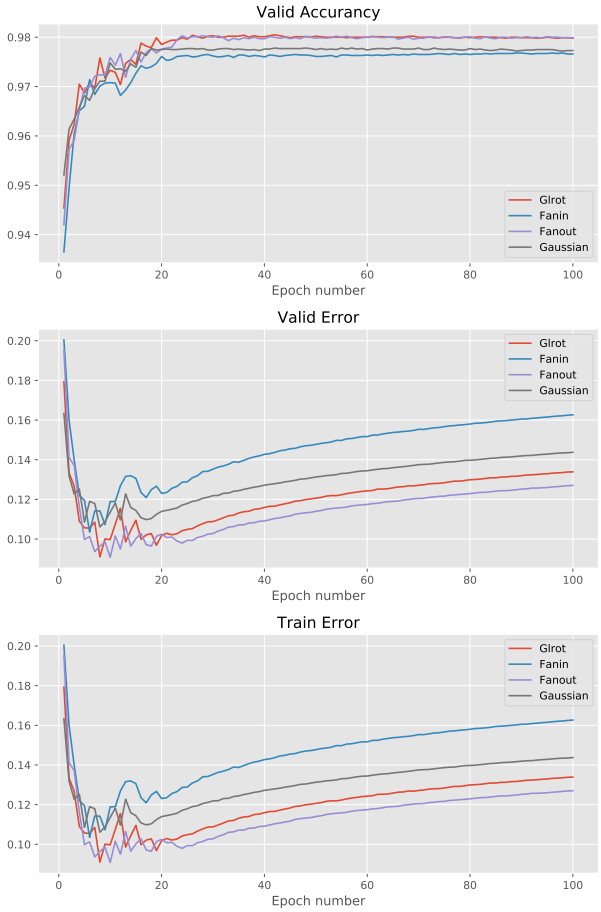

Figure 5. Classification accurancy, validation error of validation dataset, and training error of training dataset. The same network with 8 hidden layers was trained under different initialisation strategies.

mances were actually similar and were all outstanding. The initialisation strategies seems no to influence the final outputs of this experiments greatly. This result may be caused by the simplicity of the training dataset. As the dataset is so straightforward (simply some gray values), the network were always convergent with most of the random initialisation.

5. Conclusions

We have compared different activation functions, we have also explored the impact of the number of hidden layers and initialisation strategies. Although it is not so clear but we can still found out that SELU has better performance as it seems to be more stable during training process. In addition, it is clearly show that increasing the depth of the networks will result in more fluctuant training process and consume more time but the accuracy will improve slightly. Although we have come out with the conclusion that the Glrot and Gaussian initialisation have better performance, the importance of the initialisation was not able to show in this application as the dataset is oversimplified. In the future exploration, we can take advantage of a more complicated dataset to enlarge the differences between different initialisation strategies.

STRATEGY	DISTRIBUTIOIN
FANIN	$\mathrm{U}(-\sqrt{3/n_i}, \sqrt{3/n_i})$
FANOUT	$\mathrm{U}(-\sqrt{3/n_o}, \sqrt{3/n_o})$
FANIN	$\mathrm{U}(-\sqrt{6/(n_i+n_o)}, \sqrt{6/(n_i+n_o)})$
GAUSSIAN	$\mathrm{U}(-\sqrt{1/n_i}, \sqrt{1/n_i})$

Table 1. Initialisation strategies

References

G. Klambauer, T. Unterthiner, A. Mayr and Hochreiter, S. Self-normalizing neural networks. *In Advances in Neural Information Processing Systems (NIPS)*, 2017.