MLP Coursework 1: Activation Functions

s1771650

Abstract

There are many activation functions have been proposed, one of the most common used in today is ReLU. The report is exploring the variants of ReLU activation function (Leaky ReLU, ELU, and SELU) for hidden units in multi-layer networks, and use MNIST validation dataset accuracy to verify how well the trained model performed. The result shows that ELU has the highest accuracy, thus by using ELU to explore the behavior of deeper networks. Comparing networks with different layers (2-8 layers), different initialisation strategies (Fan-in and Fan-out) and different distribution such as Gaussian distribution and Uniform distribution.

1. Introduction

Multilayer networks are networks with multiple kinds of relaions and it has become one of the most popular social network science. In this coursework is related to training multi-layer network to solve the MNIST digit classification problem. MNIST is a simple computer vesion dataset, which consists of images of handwritten digits. The MNIST data splits into two parts: 50,000 data points of training data, 10,000 points of validation data. In this coursework I will use trainning data to train each model, and observe the behavior by validation accuracy and error.

The experiment is devided into two parts, In the first part I am comparing the behaviour of different activation function (Leaky ReLU, ELU, and SELU) on MNIST task. The second part I will be exploring deep neural network by investing the impact of the depth of the network and different approaches to weight initialisation.

2. Activation functions

In neural networks the neuron really does not know the bound of the value. So we decided to add an activation function to decide whether the neuron should fire or not. The activation function is used to introduce nonlinearity to models and map input nodes and output nodes. The weights on each activation function are optimized via backpropagation. Since Sigmoid is a non-linear funcion, it becomes one of the most widely used activation function today. However, it cannot make significant change because of its gradient is small or has vanished. The Rectified Linear Unit(ReLU) eliminates the vanishing gradient problem. This function was first introduced to a dynamical network by Hahnloser et al. ReLU defined as :

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

where x is the input to a neuron. f(x) is zero when x is less than zero and f(x) is equal to x when x is above or equal to zero. ReLU has the gradient:

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

However, since there are no negative values, the mean of activation is larger than zero, which will act as a bias for the next layer and result in a bias shift. Less bias shift brings the standard gradient closer the natural gradient and speeds up learning. Leaky ReLU, ELU and SELU are the variants of ReLU, they add a function in the negative part to solve this problem. Leaky ReLUs allow a small, non-zero constant when the unit is not active, typically $\alpha = 0.01$. This can alleviates dying ReLU problem. Leaky ReLU defined as :

$$\text{lrelu}(x) = \begin{cases} \alpha & \text{if } x \leq 0 \\ 1 & \text{if } x > 0. \end{cases} \tag{3}$$

Leaky ReLU has the gradient:

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

Leaky ReLu does not ensure a noise-robust deactivation state, Exponential Linear Unit (ELU) saturates to a negative value with smaller arguments. ELU replaces the small linear gradient of Leaky ReLU with a vanishing gradient. LEU defined as :

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

ELU has the gradient:

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

Scaled Exponential Linear Units (SELU) has been proposed for building very deep multilayer perceptions.

SELU defined as :

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

where λ and α are two fixed parameters. For standard scaled inputs (mean 0, standard deviation 1), the values are $\alpha \simeq 1.6733$ and $\lambda \simeq 1.0507$. And SELU has the gradient:

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

3. Experimental comparison of activation functions

For the first experiment, I am going to explore the behavior of Leaky ReLU, ELU and SELU activaion functions on MNIST task by comparing them with RELU and Sigmoid activation functions. I train the model by using the same network architecture, 2 hidden layers and with 100 units per hidden layer. For all experiment, I used a batch size of 50 for a total of 100 epoch. For each epoch, the neurons in the images were chosen randomly in the seed. Since they are the variant of RELU, I expect their performance should be similar to ReLU. As seen Figure 1, I find this to be true. ReLUeliminates the vanishing gradient problem in Sigmoid, thus I can see that the performance of ReLU is better than Sigmoid. Moreover, Sigmoid's output is not zero centered, which makes the gradien has too many different directions and makes optimization harder, as the result it trained slower.

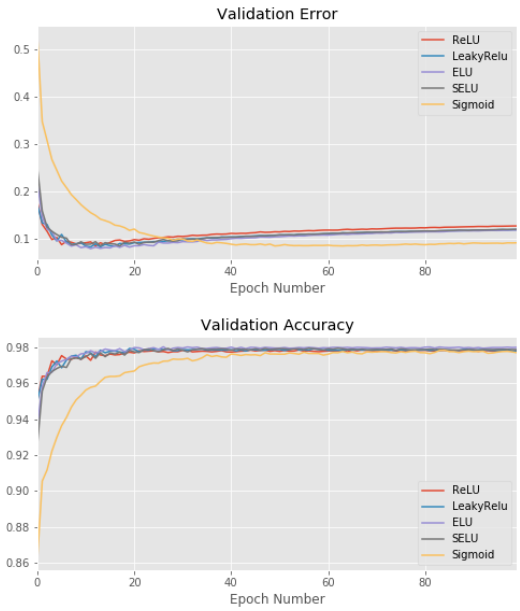

Figure 1. The change in the validation set error and accuracy in training model.

Activation Function	Mean	Standard Deviation
ReLU	0.978391	0.003787
Leaky ReLU	0.97748	0.003923
ELU	0.978401	0.005282
SELU	0.977157	0.006223
Sigmoid	0.969614	0.017422

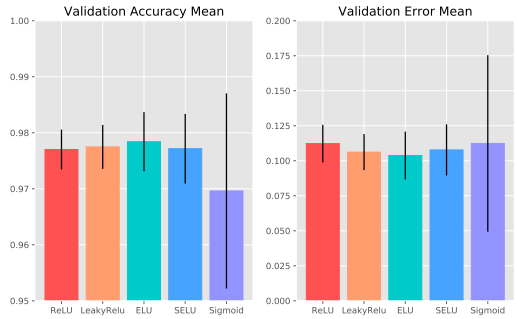

Figure 2. Mean and Standard Deviation of ReLU, Leaky ReLU, ELU, SELU and Sigmoid activaion function

Next, I compare the mean and standard deviation of these functions that show in Figure 2. ReLU rectifies the positive inputs, but negatives inputs give an output of zero. This result in "dying ReLU" problem, which means many neurons end up in a state where they are inactive for most inputs. The variant of ReLU (Leaky ReLU, ELU and SELU) can solve this problem. Therefore, as seen Figure 2, their accuracy is slightly higher and the error is slightly lower than ReLU.

Leaky ReLU adds a small coefficient for negative values, which makes negative activations push mean unit activation closer to zero. ELU replaces the small linear gradient of Leaky ReLU with a vanishing gradient, which makes deactivation state has some robustness to noise. Both of them alleviate dying ReLU problem, and improved performance(higher accuracy, lower error, and faster learning). However, Leaky ReLU does not saturate in the deactivation state. ELUs saturate to a negative value when the argument gets smaller. Saturation means a small derivative which decreases the variation that is propagated to the next layer. Thus in the next experiment, I will choose ELU activation function to explore the behavior of Deep neural networks.

4. Deep neural network experiments

The second part of the experiment is to explore the impact of the depth of the network (number of hidden layers) with respect to accuracy, and a comparison of different approaches to weight initialisation.

The first section, I am using training data to train the model by ELU activation function with 2-8 hidden layers. After training them, Figure 3 shows the verifying result of the change of validation error rate and accuracy rate for each epoch. The model learned fast, soon the error rate drops to under 12% and the accuracy rate rises up to 97%. It is interesting that the error rate fluctuated in the first 35 epoch in 5-8 layers. Finally, each layer remains steadily at approximately 10%, but the higher the number of layers it has, the higher the error rate it got. It can be found that the accuracy rate of the model with 6 layers is the highest, the average is about 98.15%.

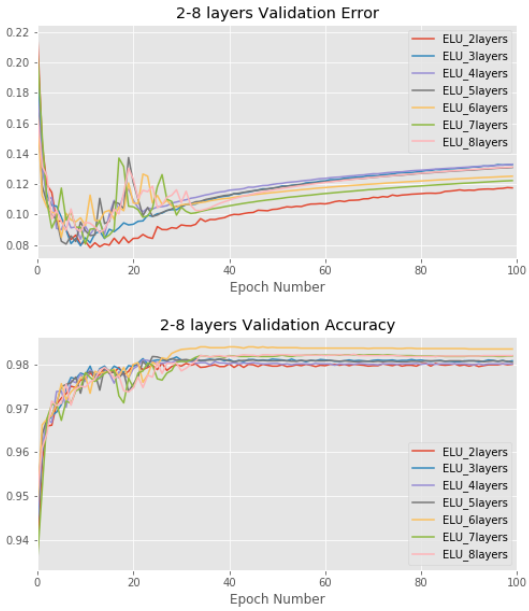

Figure 3. The change in the validation set error and accuracy in training model.

Figure 5.

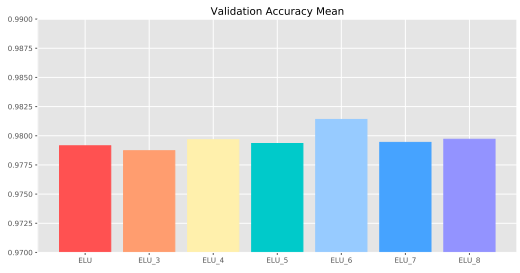

Figure 4. The mean of accuracy of each layer

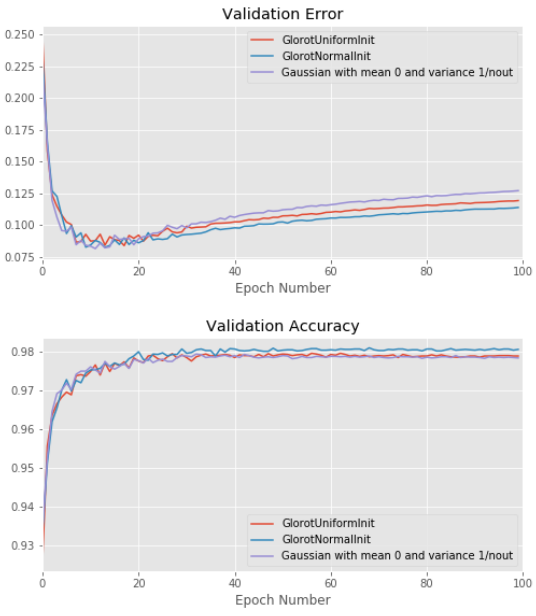

Figure 6.

In the next section, I want to compare the effect of different initialisation strategies, by weight initialisation and different distribution. First, look at different weight initialisation based on "Fan-in", Fan-out" and "Fan-in and Fan-out" in Uniform distribution. Fan-in corresponds to constraining the estimated variance of a unit to be independent of the number of incoming connections (nin); Fan-out constraining the estimated varianceof a unit's gradient to be independent of the number of outgoing connections (nout); "Fan-in and Fan-out" corresponds to Glorot and Bengio's combined initialisation.

In Figure 5, it can be found that although three of them share the similar accuracy, their error rate is different. The label of "GlorotUniform" is Fan-in and Fan-out, which has the smallest error rate. And Fan-out shows the worst behavior. In the beginning, three of them start at approximately 15% of error rate, at the end, Fan-out still has 14% of error rate.

Finally, I am comparing the effect of different distribution, Gaussian Normal distribution and Uniform distribution. In Figure 6, it can be found that the accuracy rate of Gaussian normal distribution is slightly higher than Uniform distribution and the error rate is also lower. Additionally, comparing them with a SELU layer drawing from a Gaussian with mean 0 and variance $1/n_{in}$. The result shows it has the highest error rate.

5. Conclusions

In this report, at the beginning I introduce different activation functions and how they work in neural networks. In the experiment, it can be found that Leaky ReLU, ELU, and SELU perform similarly, they can quickly reach to a high accuracy rate, especially ELU. Moreover, the higher number of layers in the model can perform higher accuracy rate, but there are some overfitting problems. Finally, I found that Glorot and Bengio (2010) random uniform weights initialiser has the lowest error rate.

References

Djork-Arne Clevert, Thomas Unterthiner, Sepp Hochreiter. Fast and accurate deep network learning by exponential lenear units(elus).

Mark Harmon, Diego Klabjan. Activation ensembles for deep neural networks.

(Mark Harmon) (Djork-Arne Clevert)