Further exploring the classification of images of handwritten digits on the EMNIST dataset (Learning rules, BatchNorm, and ConvNets)

Man Wu

Abstract

In this report, I investigate and further explore the performance the classification of images of handwritten digits using Deep Neural Network (DNN) and Convolutional Neural Network (CNN) trained on the EMNIST Balanced dataset. I firstly implement the experiments on the DNN. After many comparisons, I find that when the activation function is Relu and the learning rate is 0.001, the model performs best on the dataset. In addition, add the batch normalization layer and dropout layer can improve the performance of the model. At the second stage, I mainly focus on the investigation of the Convolutional Neural Network. I construct and train different networks containing one and two convolutional layers and max-pooling layers. After a very long time running, the experiment has the results that the two-layer CNN has better performances than one-layer CNN no matter the accuracy rate or the error rate. Finally, I use the test set to access the accuracy of the DNN and CNN and find that the accuracy rate of the CNN is higher than the DNN, especially the error rate of CNN is much lower than DNN.

1. Introduction

As deep learning is very popular in recent years, many researchers have done a great deal of research on it. However, implementing a deep learning network is not an easy task because it has many different conditions that need to be considered such as learning rate, the number of layer, some other hyperparameters and so on. In addition, given the dataset and special situations, we need to consider which network should be used, the Deep Neural Network (DNN) or Convolutional Neural Network (CNN). Therefore, it is a meaningful thing to explore deep learning by ourselves. In this project, I mainly do some experiments and further explore the classification of images of handwritten digits using neural networks (DNN and CNN). I try to find a model with best structure and hyperparameter based on the dataset. The dataset used in these experiments is the EMNIST, which is an extended version of the MNIST database. The training data, validation data and test data all have the batch size of 100.

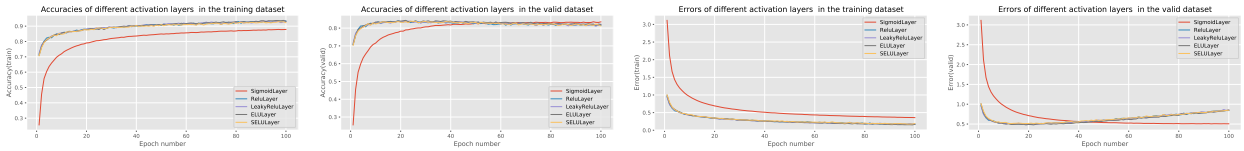
2. Baseline systems

In order to find the best baseline for EMNIST Balanced dataset, I try to perform the baseline experiments from two aspects. Firstly, I tried several activation functions (Sigmoid, Relu , LeakyRelu , ELU and SELU) to find out which one is the most suitable. And in the first experiment, I plot a figure to compare these activation functions, which is shown in Figure 1. It is obvious to see that Relu, LeakyRelu, ELU and SELU have nearly the same performance on the dataset and all perform better than Sigmoid. Also I find that the Relu function converges a little faster than other functions. So I used Relu as the activation function for this experiment. Secondly, I adjust the value of learning rate to 0.01 and want to confirm whether it is better than 0.1. The results showed in Figure 2. Comparing the results, I find when the learning rate is 0.1, it causes overfitting at the point that the epoch number is 22. However, this situation did not happen when learning rate is 0.01. Thus, I set the learning rate to 0.01 in the baseline experiment. In addition, the data performs well when the number of hidden layer is two, so I did not change the depth of the hidden layer all through the baseline experiments.

3. Learning rules

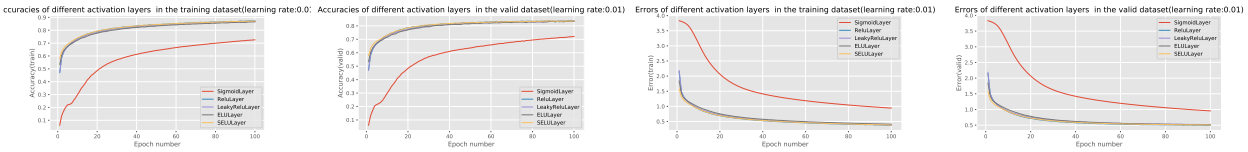
RMSProp (Tieleman & Hinton, 2012) and Adam(Kingma & Ba, 2014) are learning rules, which are both used to improve the accuracy of the training data. In this section, I implement the experiment to compare the performance of RMSProp and Adam to the original learning rule which is the Gradient Descend learning rule. I try to find the best learning rule for the following experiment. According to baseline section, I set the learning rate to 0.01 and the activation function is Relu.

The results are showed in Figure 3. As we can see from the figure, the Gradient Descend performs much better than RMSProp and Adam. When looking at the Adam learning rule, the accuracy dropped dramatically in the first 20 epoch both on the training data and validation data. And its error-rate curve is very turbulent. When focusing on the RMSProp learning rule, it performs worse than Adam and its accuracy and error rate curves are approximate straight lines. In addition, it has very low accuracy rate and very high error rate, which can be said that the performance is very bad. The situation is very strange and I doubt that whether the value of the learning rate is set incorrectly. So I decided to change the learning rate to 0.001 and try again.



(a) Accuracies in the training dataset (b) Accuracies in the valid dataset (c) Errors in the training dataset (d) Errors in the valid dataset

Figure 1. The accuracies and errors of different activation layers in the training and valid dataset. Please use high resolution for better view.



(a) Accuracies in the training dataset (b) Accuracies in the valid dataset (c) Errors in the training dataset (d) Errors in the valid dataset

Figure 2. The accuracies and errors of different activation layers (learning rate = 0.01) in the training and valid dataset. Please use high resolution for better view.

The new results are showed in Figure 4. As expected, the previous result is really a question of the learning rate. In Figure 4, Adam and RMSProp perform better than Gradient Descend. However, I find that the error-rate curves of Adam and RMSProp on the validation dataset are overfitting after the epoch of 20. So I add a dropout layer with dropout ratio set to 0.5 to prevent overfitting (Krizhevsky, Sutskever, & Hinton, 2012). The results are satisfactory and âĂIJdropoutâĂİ effectively solves the problem of overfitting. From the results which are showed in Figure 5, we can see that Adam performs a little better than RMSProp, which has a little higher accuracy rate and lower error rate on validation dataset. Overall, I need to update the learning rate to 0.001 and the learning rule to Adam of baseline. In addition, I need to add the dropout layer.

4. Batch normalisation

In order to investigate the impact of the using of batch normalization, I need to create a model with batch normalization layer, which is used to compare with my previous model. The two models are based on my baseline experiments (learning rate: 0.001, learning rule: Adam, add dropout layer). The results are showed in Figure 6. As seen in the figure, their performances are nearly the same and I cannot judge which is better. However, I find the batch normalization model has lower error rate on the validation set and the curve converges a little faster than the model of no batch normalization. My observation is confirmed in Table 1. Thus batch normalization is a good way to improve accuracy and decrease error of the experiment results.

5. Convolutional networks

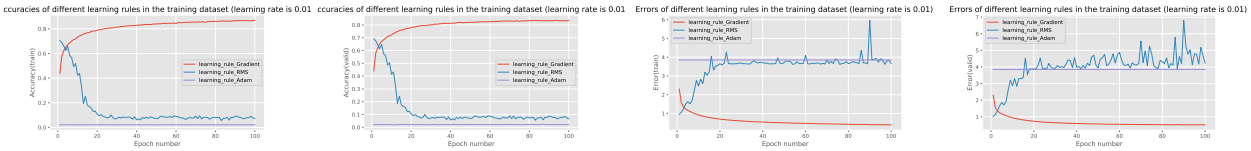
In the CNN experiments, I use the baseline of DNN mentioned in the previous sections. Due to incorrect parameter settings, I need to run CNN multiple times and spend a lot of time on each running. I forgot to curve the comparison line of the two methods (one layer and two layer) and only remember the overall trend of the lines.The curve of the one-layer CNN nearly like a straight line, and from 1 epoch to 100 epoch the accuracy and error rate of the validation dataset nearly the same. However, the curve of the two-layer CNN shows a rising trend from 1 epoch to 100 epoch on the accuracy rate of the validation dataset and a declining trend on the error rate of the validation dataset. Fortunately, I recorded their final running results, which are showed in Table 2. From the table, we can see that the two-layer CNN has better performance (higher accuracy and lower error) than one-layer CNN. I can assume that as the number of layers increases, the data trained by CNN will have much higher accuracy and lower error rates.

6. Test results

The results of the test set trained on batch-normalized DNN and two-layer CNN individually are showed in Table 3. From the table, we can obviously see that CNN has a distinct advantage over DNN in image recognition with higher accuracy rate and lower error rate. And DNN has a very bad performance on the handwriting recognition database. Maybe the DNN is not suitable for the database.

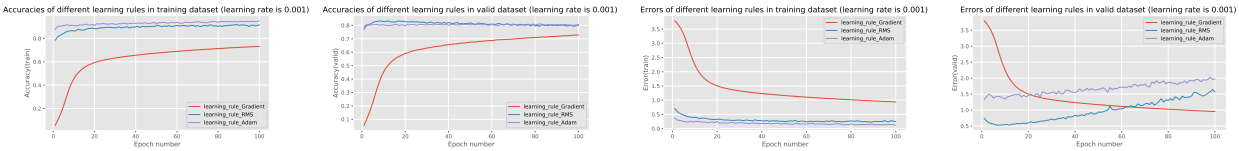
7. Conclusions

In the task of investigating and exploring the performance the classification of images of handwritten digits using Deep Neural Network (DNN) and Convolutional Neural



(a) Accuracies in the training dataset (b) Accuracies in the valid dataset (c) Errors in the training dataset (d) Errors in the valid dataset

Figure 3. The accuracies and errors of different learning rules (learning rate = 0.01) in the training and valid dataset. Please use high resolution for better view.



(a) Accuracies in the training dataset (b) Accuracies in the valid dataset (c) Errors in the training dataset (d) Errors in the valid dataset

Figure 4. The accuracies and errors of different learning rules (learning rate = 0.001) in the training and valid dataset. Please use high resolution for better view.

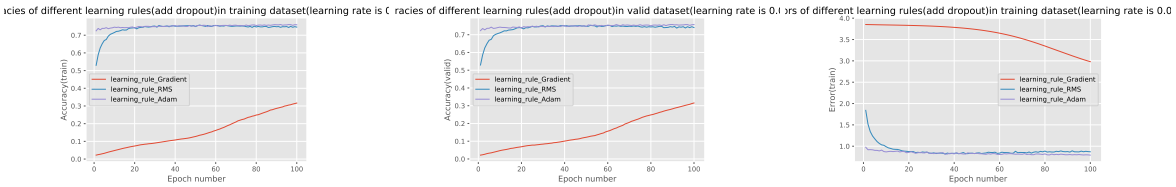
Network (CNN) trained on the EMNIST Balanced dataset, I have gained a lot and get some conclusions by myself. During experiments on the DNN, I find that when the activation function is Relu and the learning rate is 0.001, the model performs best on the dataset as well as adding the batch normalization layer and dropout layer. During experiments on the CNN, I find that two-layer CNN performs better than one-layer CNN. And I can assume that with the number of layers increases, the data trained by CNN will have much higher accuracy and lower error rates. If there is a chance in the future, I will try to train the EMNIST Balanced dataset with large amount of CNN layers, I believe the accuracy will be higher and higher. Both DNN and CNN have their own advantages, but I think CNN has more advantages in the field of image recognition, especially handwritten digits recognition.

References

Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, *4*(2), 26–31.



(a) Accuracies in the training dataset (b) Accuracies in the valid dataset (c) Errors in the training dataset

(d) Errors in the valid dataset

Figure 5. The accuracies and errors of different learning rules add dropoutLayer (learning rate = 0.001) in the training and valid dataset. Please use high resolution for better view.

(a) Accuracies in the training dataset (b) Accuracies in the valid dataset (c) Errors in the training dataset (d) Errors in the valid dataset

Figure 6. The accuracies and errors of different BatchNormalizations in the training and valid dataset. Please use high resolution for better view.

Table 1. The accuracies and errors of No Batch Normalization and Batch Normalization

Dataset	No Batch Normalization		Batch Normalization	
	Accuracy(%)	Error(%)	Accuracy(%)	Error(%)
Training Dataset	76.9	74.6	76.5	74.6
Valid Dataset	76.3	76.3	75.9	75.5

Table 2. The accuracies and errors of different CNN layers

Dataset	one-layer CNN		two-layer CNN	
	Accuracy(%)	Error(%)	Accuracy(%)	Error(%)
Training Dataset	86.2	42.9	87.2	40.4
Valid Dataset	84.6	47.9	85.4	43.6

Table 3. The accuracies and errors of DNN and CNN on test dataset

Dataset	DNN		CNN	
	Accuracy(%)	Error(%)	Accuracy(%)	Error(%)
Test Dataset	72.0	91.7	80.2	50.7