MLP Coursework 2: Learning rules, BatchNorm, and ConvNets

s1781323

Abstract

This coursework is the experiment on batch normalization and CNN. At first, we show the result on a baseline model, then test models with different learning rules, batch normalization, CNN, and so on with the explanation of these concepts.

Compared to stochastic gradient descent, the optimizers called RMSProp and Adam converges faster and more property, and it is beneficial for the neural network. Dropout is the method to prevent overfitting. It may worsen the accuracy depend on the usage, but it improves generalization ability. Batch normalization normalizes data in each layer and enhances performance dynamically. CNN increases the performance as well.

The best model in this paper is the combination of batch normalization and CNN. However, it takes the enormous computational cost. This is another critical problem in this field.

1. Introduction

The aim of this coursework is to further explore the classification of images of handwritten digits using neural networks. Recently we see so many article on new techniques and how they bring breakthrough to this industory. In this report, we test several learning rules, batch normalization, and convolution networks and techniques including dropout.

In this report we use EMNIST (Extended MNIST) Balanced dataset, https://www.nist.gov/itl/iad/ image-group/emnist-dataset [Cohen et al., 2017]. EMNIST extends MNIST by including images of handwritten letters (upper and lower case) as well as handwritten digits. Both EMNIST and MNIST are extracted from the same underlying dataset, referred to as NIST Special Database 19. Both use the same conversion process resulting in centred images of dimension 28 X 28. Although there are 62 potential classes for EMNIST (10 digits, 26 lower case letters, and 26 upper case letters). The expected accuracy rates are lower for EMNIST than for MNIST. We use a reduced label set of 47 different labels. We use 100,000 data for training, and 15800 data for validation.

2. Baseline systems

For baseline system, we construct a neuralnetwork with two hidden layers. We use ReLU function as activation, and SGD as learning rule. The hidden units of each layers are 100, and the last output is 47. After the section, we add components or replace learning rules.

The lerning rate is 0.001.

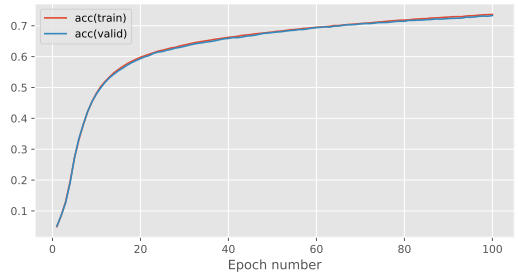

Figure 1. Accuracy of baseline system. Accuracy of train dataset:0.737 Accuracy of valid dataset:0.733

Also we implement three hidden layers because it gives more parameters. As a result the accuracy improves.

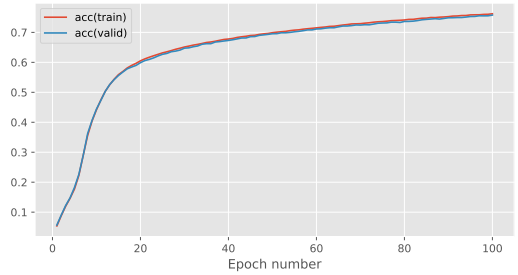

Figure 2. Accuracy of models with three hidden layers. Accuracy of train dataset:0.762 Accuracy of valid dataset:0.758

These result show that the trained model in not overfitted because the accuracy of both train and validation sets are close.

Finally for this section, we applied dropout technique to the last affine layer in the baseline model. The input optimal probability of retention(p) is 0.5. Dropout is the technique to privent overfit. While training, dropout stop some units stochastically and operate forward pass and bachpropagation. This method reduce the parameters while training, so it prevents overfit, however, this may decrease the accuracy of prediction. The result on figure 3 shows that the

difference of accuracies reduces. Also the learning curve goes smooth. However, this method worsen the accuracy. It is hard to find appropriate p and model to use dropout technique.

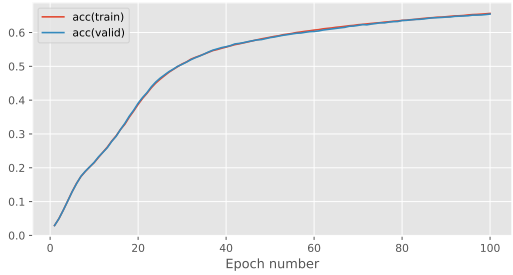

Figure 3. Accuracy of models with three hidden layers. Accuracy of train dataset:0.657 Accuracy of valid dataset:0.655

3. Learning rules

In this section we introduce RMSProp and Adam with gradient descent, and then compare with SGD rule.

RMSProp: RMSProp is presented by Hinton in the Coursera course, and this is a stochastic gradient descent version of RProp, normalised by a moving average of the squared gradient.

$$\begin{aligned} S_i(t) &= \beta S_i(t-1) + (1-\beta) g_i(t)^2 \\ \Delta w_i(t) &= -\frac{\eta}{\sqrt{S_i(t)} + \epsilon} g_i(t) \end{aligned} \tag{1}$$

β is usually 0.9. η is learning rate and 0.001 is appropriate. is the gradient of ith parameter.

Adam: Adam is presented by Kingma and Ba. In addition to storing an exponentially decaying average of past squared gradients like RMSprop, Adam also keeps an exponentially decaying average of past gradients similar to momentum:

$$\begin{aligned} m_i(t) &= \beta_1 m_i(t-1) + (1-\beta) g_i(t) \\ v_i(t) &= \beta_2 v_i(t-1) + (1-\beta_2) g_i(t)^2 \\ \hat{m_i}(t) &= \frac{m_i(t)}{1-\beta_1^t} \\ \hat{v_i}(t) &= \frac{v_i(t)}{1-\beta_2^t} \\ \Delta w_i(t) &= -\frac{\eta}{\sqrt{\hat{v_i}(t)} + \epsilon} \hat{m_i}(t) \end{aligned} \tag{2}$$

β_1 is usually 0.9. β_2 is 0.999 η is learning rate and 0.001 is appropriate. is the gradient of ith parameter.

The learning rule in baseline is Stochastic gradient descent (SGD). However, Dauphin et al. argues that saddle points are usually surrounded by a plateau of the same error, which makes it notoriously hard for SGD to escape as the gradient is close to zero in all dimensions.

	ACC(TRAIN)	ACC(VAL)	ERROR(TRAIN)	ERROR(VAL)
SGD	0.737	0.733	0.919	0.936
RMSPROP	0.944	0.809	0.135	1.37
ADAM	0.943	0.812	0.138	1.20

Table 1. Accuracy and mean square error of models with three different learning rules. Error is mean square error. Model is based on baseline.

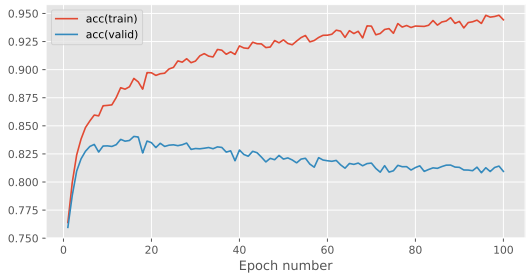

Figure 4. Accuracy of models with RMSProp learning rule. Accuracy of train dataset:0.944 Accuracy of valid dataset:0.809

The results is on figure 4 and 5 and Table 1.

The result shows that these learning rules improves the accuracy of prediction of training and validation dataset. However, the differnece goes up, and this means that overfitting happens. It may be solved by dropout technique.

4. Batch normalisation

In this secion, we present Batch normalization.

Covariate shift in machine learning means that the distribution of input data deviates, and it makes the algorithms hard to predict. In deep neural network, the distribution of input in each activation goes diffenrent, and this is called internal convariate shift. It is known that the whitening technique, which makes the varidation to 1 and average to 0, makes the converge of neural netowrk faster. Before this technique, normalization and standardization is used in the pre-processing of data. Batch normalization is presented by Ioffe and Szegedy in 2015. This method normalize the inputs in each activation. Not only dataset but also all inputs is whitened, therefore, it control internal convariate shift and converge goes faster.

We implement batch to the baseline model with SGD, RMSProp and Adam optimizer. The result is on figure 6 to 8 and table 2.

The accuracy in RMSProp and Adam improves, and it controls overfitting. Although the model with SGD does not change dynamically, batch normalization is beneficial.

Finally for this section, we implement baseline model with dropout, adam and batch normalization. The result in on

Figure 5. Accuracy of models with Adam learning rule. Accuracy of train dataset:0.943 Accuracy of valid dataset:0.812

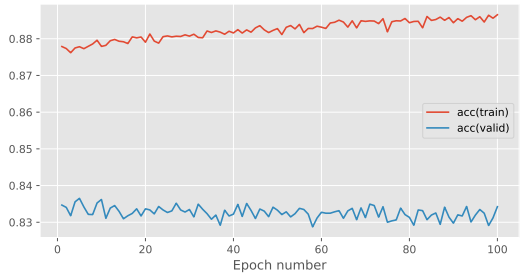

Figure 7. Accuracy of models with Batch and RMSProp. Accuracy of train dataset:0.887 Accuracy of valid dataset:0.834

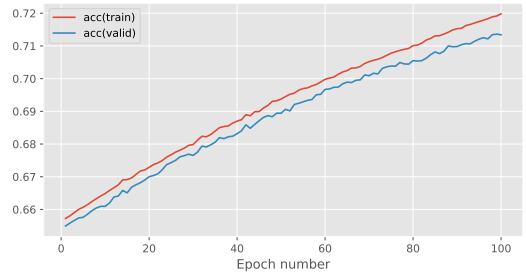

Figure 6. Accuracy of models with Batch and SGD. Accuracy of train dataset:0.720 Accuracy of valid dataset:0.713

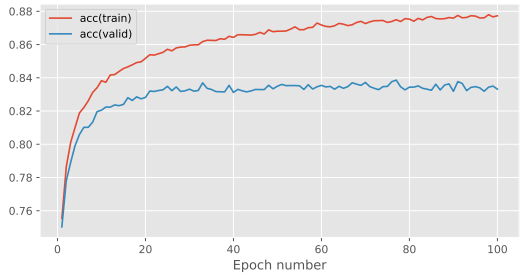

Figure 8. Accuracy of models with Batch and Adam. Accuracy of train dataset:0.877 Accuracy of valid dataset:0.833

Figure 9. Although the acuuracy worsened, the overfitting improves.

5. Convolutional networks

In this section, we present Convolution Neural Network (CNN).

CNN is constructed with convolution layer and pooling layer. Before CNN, we use inputs data as one dimension. However, it ignores the spatial structure of the input images, and it is weak to learn the same features at different places in the input image. Convolution layer has a filter with shared weight. To apply the filter to the image create reduced feature map. Pooling layer is applied after convolution layer. Pooling layer reduces information to appropriate data size. In this experiment, we use max pooling layer. Max pooling takes the maximum value of the units in the region. This layer reduces the cost of computation, controls overfitting, and makes the model robust to precise displacement.

We construct three models. The first model containes one convolution and maxpooling model (Figure 10). The second model containes two convolution and mmaxpooling layers. These are based on baseline model, and the learning rule is SGD, learning rate in 0.001.

The third model is based on the paper Ioffe and Szegedy in 2015. It containes three convolution and max pooling layers, and two batch normalization layers. The activation function is hyperbolic tangent, and the learning rate is 0.03(Algorithm 1).

The epoch size is 50 in this experiment because CNN takes huge computation cost. It takes 48 hours for third model to learn train dataset. The result is on Figure 11 to 13 and table 3.

As a result, CNN improves the accuracy and reduces train error. The combination of batch normalization and CNN improves the performance.

6. Test results

In total, the third model in section 5 marks the highest performance. However, we could not test the model on test set because of time. However, we can say that the CNN model marks great performance, but deep neuralnetwork with batch normalization have a potential to approach thier performance.

7. Conclusions

In this coursework, we could prove that batch normalization, and CNN improves the neural network. These techniques solve various problems, for example, internal convariate shift, location information of inputs, and so on. We realize that the combination of CNN and batch marks impressive performance.

	ACC(TRAIN)	ACC(VAL)	ERROR(TRAIN)	ERROR(VAL)
SGD	0.737	0.733	0.919	
RMSPROP	0.944	0.809	0.135	
ADAM	0.943	0.812	0.138	
SGD + BATCH	0.720	0.713	0.969	
RMSPROP + BATCH	0.878	0.834	0.354	
ADAM + BATCH	0.877	0.833	0.323	

Table 2. Accuracy and mean square error of models with three different learning rules and batch normalization.

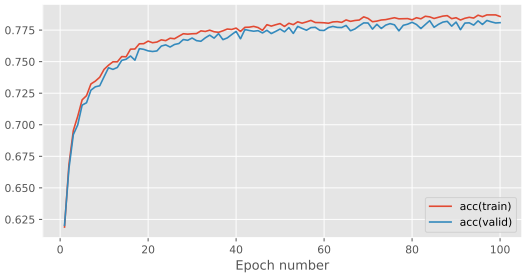

Figure 9. Accuracy of models with Batch and Adam. Accuracy of train dataset:0.786 Accuracy of valid dataset:0.781

The problem is that the computational cost is enormous, especially CNN. We cannot imagine how can solve that problem. However, this problem is the critical topics in this field.

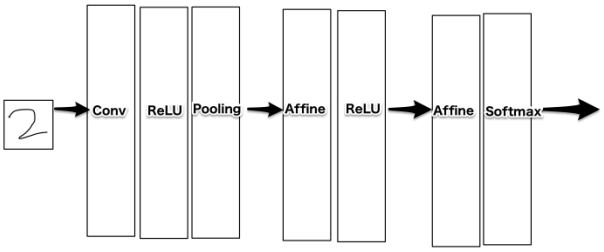

Figure 10. The First Model.

Algorithm 1 The Third Model

ReshapeLayer((1,28,28,))
ConvolutionalLayer(1, 5, 28, 28, 5, 5)
TanhLayer()
MaxPoolingLayer()
ConvolutionalLayer(5,10,12,12,5,5)
TanhLayer()
ConvolutionalLayer(10,20,8,8,5,5)
TanhLayer()
ReshapeLayer((320,))
BatchNormalizationLayer(320)
ReshapeLayer((20, 4, 4,))
MaxPoolingLayer()
ReshapeLayer((80,))
BatchNormalizationLayer(80)
AffineLayer(80, $output_{dim}$, $weights_{init}$, $biases_{init}$)

	ACC(TRAIN)	ACC(VAL)	ERROR(TRAIN)	ERROR(VAL)
FIRST MODEL	0.716	0.710	1.00	1.01
SECOND MODEL	0.743	0.745	0.852	0.847
THIRD MODEL	0.870	0.853	0.373	0.432

Table 3. Accuracy and mean square error of three models with CNN

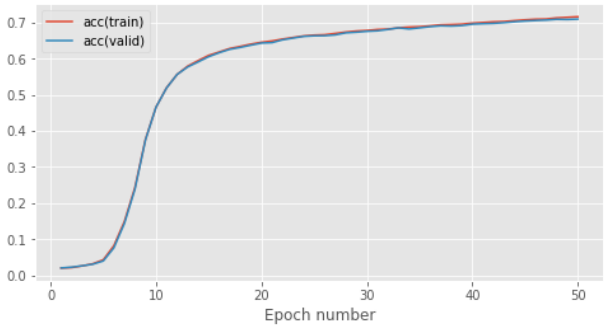

Figure 11. Accuracy of the first model. Accuracy of train dataset:0.716 Accuracy of valid dataset:0.710

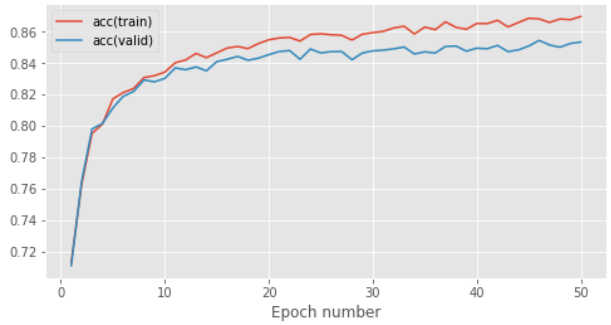

Figure 12. Accuracy of the third model. Accuracy of train dataset:0.743 Accuracy of valid dataset:0.745

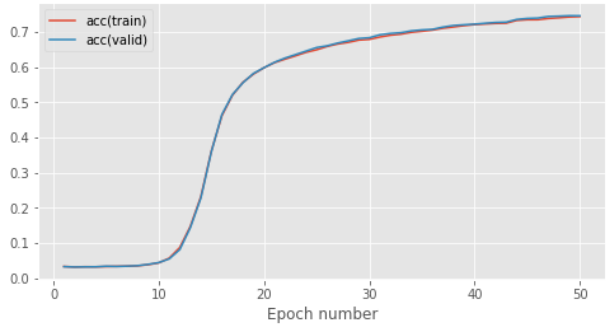

Figure 13. Accuracy of the second model. Accuracy of train dataset:0.870Accuracy of valid dataset:0.853