MLP Coursework 2: Learning rules, BatchNorm, and ConvNets

s1449640

Abstract

In this coursework, I've analysed the performance of a typical deep neural network trained on EMNIST Balanced, which includes handwritten letters of upper and lowercase, as well as handwritten digits. Initially, I've looked at the performance of a baseline model, tweaking the learning rate, number of layers, activation functions, which allowed me to receive an accuracy of about 85% on the validation set. Then, I implemented two other learning rules, RMSProps and ADAM, which are widely seen as having overall better performance in both the industry and literature. In my tests, however, they did not improve the performance at all, with a possible cause being my lack of using regularisation and not having selected ideal hyperparameters. Next, I tested Batch Normalisation, which kept the model at performance levels very similar to the baseline. Finally, I implemented Convolutional Neural Networks, which offered much smoother training, but significantly degraded performance. A possible reason for this is that I should have used more fully connected layers at the end, however that would have significantly increased the training time.

1. Introduction

This coursework is based on the EMNIST Balanced dataset, with some upper and lower case letters merged, such that the total number of labels is 47. Since this is a much larger dataset than the usual MNIST one that we worked with previously, it is interesting to analyse the tools available for optimising the algorithms performance, including different learning rules, and different architectures. It is difficult to tell whether the baseline algorithms will perform better than a more complex one at the outset.

2. Baseline systems

After several tests, my baseline performance was actually the best, using three hidden layers with ELU activation functions, a learning rate of 0.002, and no regularization. I found that changing the activation function didn't make much of a difference, which was consistent with the previous coursework findings, however tweaking the learning rate had by far the biggest impact.

It is probably the case that an even smaller learning rate would have been better, since it seems to me that the results are converging too fast, after only 10 epochs.

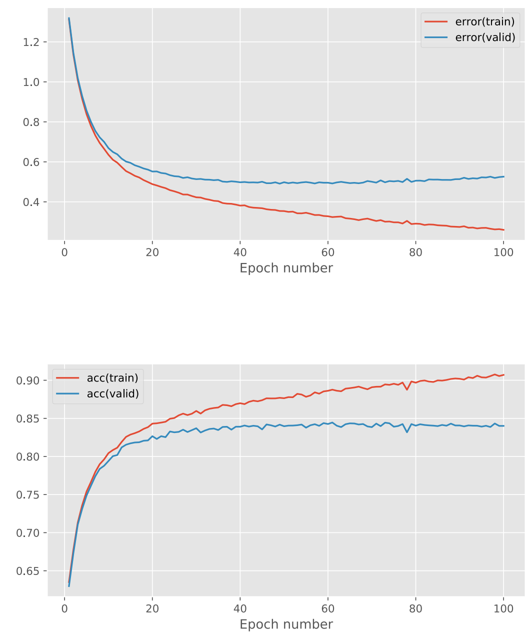


Figure 1. Baseline Performance

3. Learning rules

RMSProp showed no real improvement to the performance on the validation set. It intuitively works because we want to dampen large values, and if the weight matrix is large then dw^2 is large, and then we divide by a large number. Meanwhile, the opposite is true if the weights are small.

3.1. RMSProp

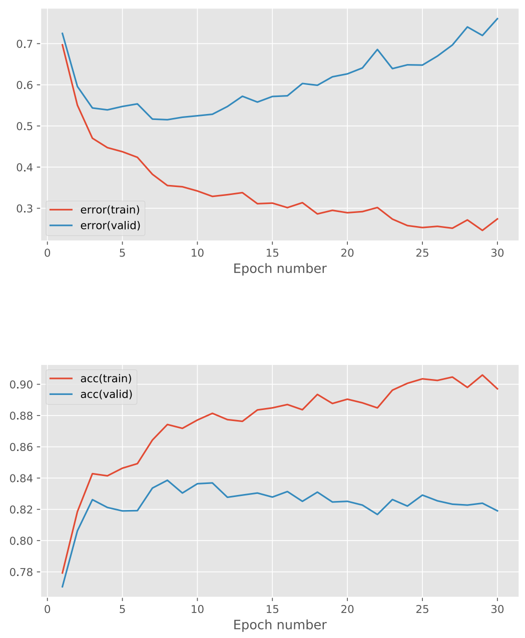


Figure 2. RMSProp Learning Rule

3.2. ADAM

ADAM combines RMSProp and Momentum leaning rules. While in the literature it is usually thought of as offering almost guaranteed performance gains, I found almost no difference compared to RMSProp or the baseline.

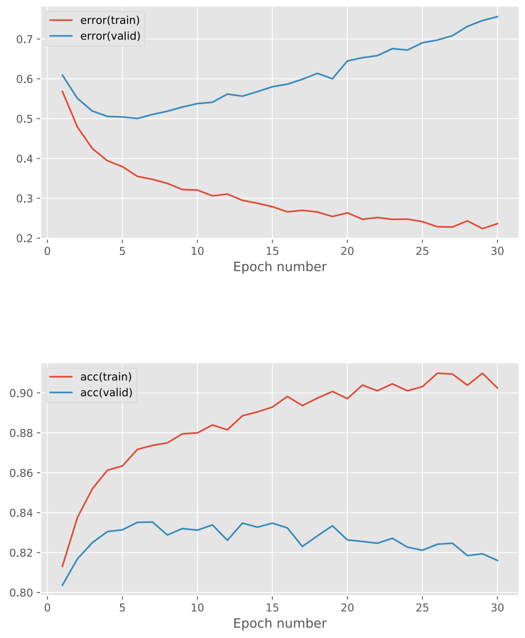


Figure 3. Adam Learning Rule

4. Batch normalisation

Batch Normalisation can be thought of as input normalisation (as used in linear regression models) applied to Neural Networks.

Given some intermediate values in the Neural Network, it calculated the mean and variance of the values and normalises the input matrix by first substracting the mean then dividing by the variance. This ensures that the mean is centred on 0 with a variance of 1, usually used to make learning algorithms converge faster. In my tests, it was noisier than the baseline and performed surprisingly worse.

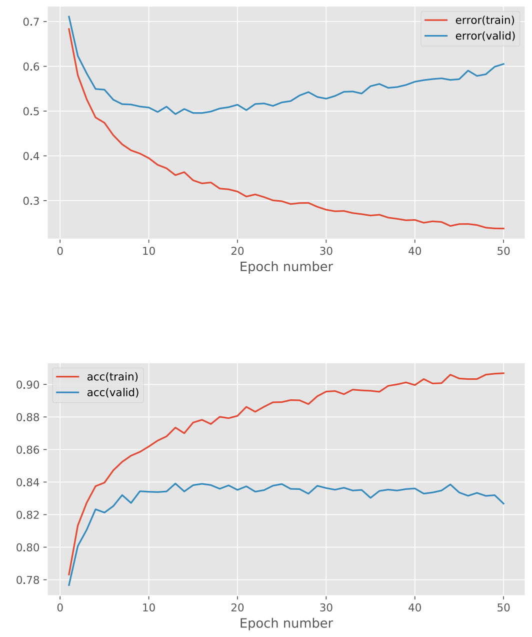
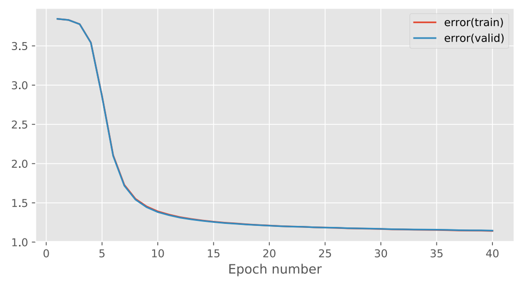


Figure 4. Batch Normalisation Performance

5. Convolutional networks

My convolutional neural networks trained several orders of magnitude slower than traditional fully connected networks, while offering seemingly no performance benefit. A possible reason for this is that I have not added any fully connected layers before the Softmax at the end, or that my learning rate was much faster than it needed to be.

5.1. One Convolutional Layer



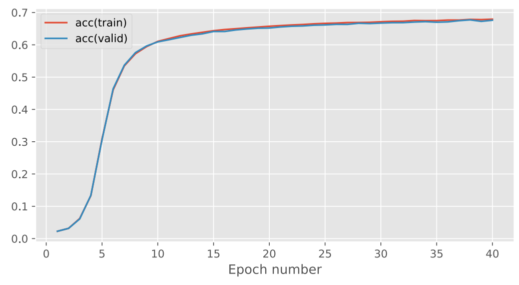
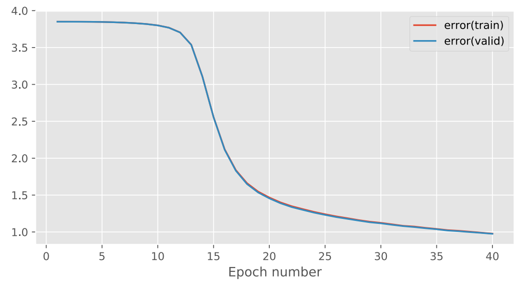


Figure 5. One Convolutional Layer

The difference between one and two conv layers is almost non-existent.



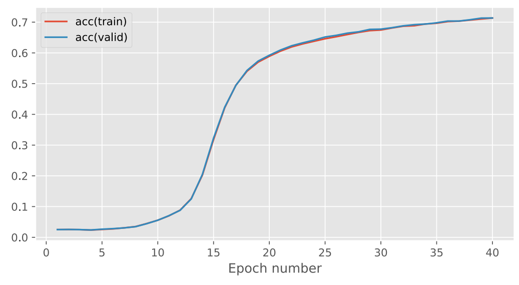


Figure 6. Two Convolutional Layers

6. Test results

The best performance could be seen on the baseline. One possible reason for this is that I have made several tests on the baseline to tweak the value such that it looked optimal, while other tests might have needed different hyperparameters.

7. Conclusions

On this learning task, traditional neural networks performed better and they trained several orders of magnitude faster.