MLP Coursework 1

s1651343

Abstract

This report mainly focus on the exploration of three different regularization methods, L1 weight penalty, L2 weight penalty and Dropout method, to alleviate the overfitting problem of a neural network. By comparing validation performance of each or combination of these methods, with different parameter settings, one best model is selected and a test performance on the selected model is demonstrated. Furthermore, a research review on Dropout method is summarised in last part of this report.

1. Introduction

Multi-layer neural networks have several non-linear hidden layers such as the activation functions and these involved nonlinear relationships between inputs and outputs usually cause sampling noise, when training the model, and co-adaptation occur(Nitish Srivastava, 2014), so the network will have an overfitting problem.

In this report, we identified a severe overfitting problem of a simple neural network for the EMNIST dataset, and analyse the reason of this problem in section 2. By varying the number of layers and hidden units, we set up a baseline that is a model without any regularization for comparison with the following improved models. Then in section 3 we introduced three common regularization approaches, L1 weight penalty, L2 weight penalty and Dropout weight penalty; and we theoretically analysed how these methods can be used to mitigate this problem. Moreover, in the next section 4, we conducted a series experiments to find a best configuration of these methods. A best model was selected based on validation performance, having around 86% accuracy in validation set. The test performance of this best model was also reported in section 4.

The dataset used for this report is the EMNST dataset which contains inputs image of dimension 28×28 and targets of 47 classes, and also equal number of samples per class.

In our experiment, all training, validation and test set has same batch size of 100; training set has 1000 batches per epoch, and both validation and test set has 158 batches per epoch. All models in this report is trained over 100 epochs, using a stochastic gradient descent method with Adam learning rule and default learning rate 0.001 if not specified.

2. Problem identification

Figure 1 and Figure 2 shows error and accuracy curves for a simple model on EMNIST dataset, which has one hidden layer with 100 ReLU units. We can clearly observe an obvious overfitting, as the error of validation set grows significantly from epoch 20 to epoch 100, while the error of training set is still in a decreasing trend. The reason of this overfitting problem might be a too simple model used, or large magnitude in weights, or the unit in each layer learn too much from other units so strong co-adaptation occurs. One thinking of the solution might be suppressing weights that is too large in magnitude for non-important features, so the model can learn less about them. Another way might be increase the complexity of model, such as adding more units or more layers.

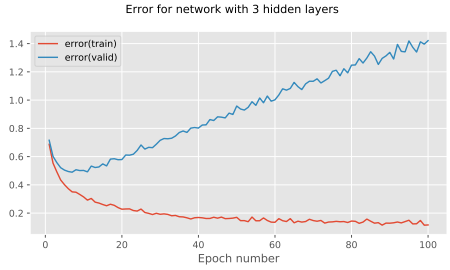

Figure 1. Error curve on the training and validation set of EMNIST dataset.

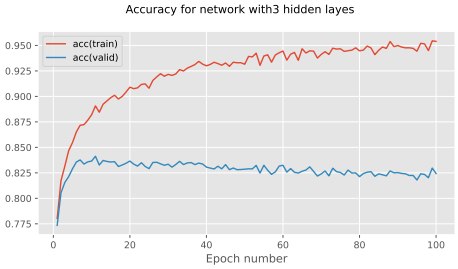

Figure 2. Accuracy curve on the training and validation set of EMNIST dataset.

Therefore, in order to check whether the performance of model would have relationship to the complexity of the model, we changed the number of hidden units in each layer and the number of hidden layers. As shown in Figure 3 (the solid line represents tracing for training set and dash line represents tracing for validation set), the training errors for all epochs are smaller and accuracy increases as the number of units increases. However, the validation performance is more complicated. The models of 32 and 64 hidden units, which has larger training error, have smaller training error and similar accuracy as 128-units model.

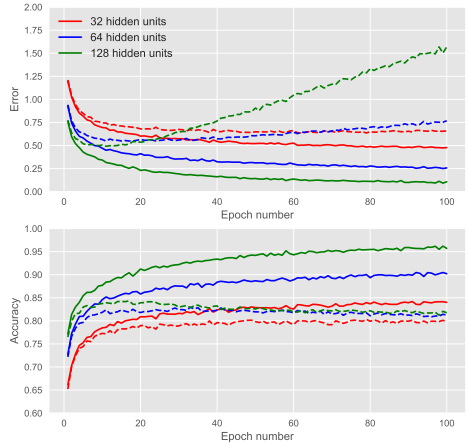

Figure 3. Error and Accuracy curve on the training and validation set of 1-hidden-layer models with 32, 64 and 128 hidden units respectively.

This is because these two models are even simpler than the previous model and the 128-units model, they do not overfit on training set, so both models generalised well on validation set. Even though there is a overfitting problem on the 128-units model, the accuracy of validation set is merely slightly higher than the rest three models, which means only increasing number of units has little effect on solving overfitting problem. The next consideration turns to be using more hidden layers to try to alleviate that overfitting. We tried models with 1, 2 and 3 hidden layers and 128 units per layer separately. Both error and accuracy plots (Figure 4) show obvious overlapping of curves for each model, which suggests more layers also cannot improve performance significantly, and overfitting problem still exits. Only a slightly less overfitting appears in 3-hidden-layers model.

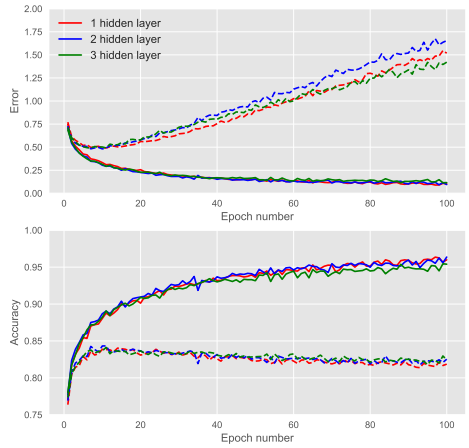

Figure 4. Error and Accuracy curve on the training and validation set of 1/2/3-hidden-layer models with 128 hidden units respectively.

Hence, although the overfitting problem is unresolved, the accuracy is not lower than Figure 2 when the model is more complicate, so we will use the 3-hidden-layers model with 128 hidden units per layer as our baseline for the experimental exploration of regularization methods in section 4.

3. Dropout and Weight Penalty

In order to prevent the severe overfitting problem described above, many methods have been proposed, such as early stopping, data augmentation, model combination and so fourth. Yet, we will only discuss three widely used approaches, Weight Penalty(L1/L2) and Dropout Method, due to limitation of one report, and compare their validation performance.

3.1. Weight Penalty

Weight Penalty is an idea of punishing weights that will not consistently evolve in a direction during training, by making these weights decay to zero (Bilen, 2020).

$$E^n = \underbrace{E^n_{train}}_{\text{data error}} + \underbrace{\beta E_W}_{\text{regularization}} \tag{1}$$

Equation 1 shows the mathematical definition of Weight Penalty, where E^n means the error function at n-th batch and β is the constant to control the strength of regularization E_W. Higher strength represent a better generalisation of the model on unseen data. L1 Weight Penalty is defined when $E_W = \sum_i |w_i|$, shown in algorithm 1, and L2 Weight Penalty is $E_W = \frac{1}{2}\sum_i w_i^2$, shown in algorithm 3. Because we are using gradient-based optimiser, at each batch n, the weight matrix will be updated by the following rule:

$$W^n \longleftarrow W^{n-1} - \epsilon \Delta E^n \tag{2}$$

Thus, we need to compute gradient w.r.t. weights, using algorithm 2 and 4. By add extra regularization term, both L1 and L2 will suppresses the random-walking weights from being large in magnitude, so the network will not learn too much about the noisy features. Therefore, for our overfitting problem, since we want to suppress the influence of those noisy training data, theoretically, both L1 and L2 should mitigate the problem. In fact, experiments conducted also proved this assumption, which will be described in detail in section 4. The difference is that L1 will push weights of trivial features to zero at a constant rate, which leads to sparse weight matrix; in contrast, L2 will only push them to almost zero so the weight matrix is still dense, so L1 will shrink faster if $|w_i|$ is small.

Algorithm 1 L1 regularization term

Input: weights W and β:
$E_w = \sum_i \sum_j |W_{ij}|$
Output: βE_W

Algorithm 2 Gradient L1 regularization term

Input: weights W and β:
if $W_{ij} > 0$ **then**
  $\frac{\partial E_W}{\partial W_{ij}} = 1$
else if $W_{ij} < 0$ **then**
  $\frac{\partial E_W}{\partial W_{ij}} = -1$
else
  $\frac{\partial E_W}{\partial W_{ij}} = 0$
end if
Output: $\beta \Delta E_W$

Algorithm 3 L2 regularization term

Input: weights W and β:
$E_w = \sum_i \sum_j W_{ij}^2$
Output: βE_W

3.2. Dropout Method

The basic idea of Dropout method is to randomly deactivate some units at each layer during training, so we obtain a 'thinned' network. The training then equivalent to 2^n different 'thinned' networks (Nitish Srivastava, 2014), and we use a scaled version of this network at test time. The reason why dropout works for resolving overfitting problem is that it is actually cut the connection between some units in the network, so the retained units will not be influenced too much by other units' noise.

The formal procedures of forward and backward propagation are shown in Algorithm 5 and Algorithm 6. In the forward propagation of Dropout, we randomly mask some units at training and scaled back these units when we are checking validation and test performance. Also in the code file, we considered the situation that all batches of an epoch can use same mask to deactivate units, by setting a condition share_across_batch to **TRUE**.

The difference between L1/L2 weight penalty methods and Dropout method is that the later is actually achieve better validation performance on a number of different tasks because it introduce more stochasticity in to the model and be proved is indeed equivalent to L2 weight penalty when we marginalised the noise (Nitish Srivastava, 2014).

4. Balanced EMNIST Experiments

4.1. L1 Weight Penalty

In this section, we will demonstration step by step of the exploration of best model. First of all, we tried the 3-hidden-layers model and L1 regularization with different penalty coefficients. Starting with the a fairly small coefficient, 1e-05, we can see there is still a significant overfitting as the error of validation set grows from 20 epochs to 100 epochs (Figure 5). However, when we double the coefficient to 5e-05, overfitting problem is mitigate considerably, and accuracy increase to around 85% which is our best performance using L1 penalty. We can apply an even stronger regularization with larger coefficient such as 1e-04, but the model will begin to slightly underfit as the red dash line of error showing a minor declining trend, so we stop trying more larger coefficients for L1 regularization.

Algorithm 4 Gradient L2 regularization term

Input: weights W and β:
$\frac{\partial E_W}{\partial W_{ij}} = W_{ij}$
Output: $\beta \Delta E_W$

Algorithm 5 Dropout layer forward propagation

Input: inclusion probability p and input data X:
generate random number θ_{ij} for all units X_{ij}
if Drop some units **then**
  if $\theta_{ij} > p$ **then**
    generate mask $m_{ij} = 0$
  else
    $m_{ij} = 1$
    $X_{ij} = X_{ij} * m_{ij}$
  end if
else
  $X_{ij} = X_{ij} * p$
end if
Output: masked X

4.2. L2 Weight Penalty

For the same model but with L2 regularization method, we tried four setting of coefficients: 1e-03 ,5e-04, 1e-04, 5e-05. As observed in Figure 6, both 1e-04 and 5e-05 shows a obvious overfitting, while 1e-03 has a subtle sign of underfitting as the error of 1e-03 and 5e-04 has almost same validation error at 100 epoch but 1e-03 has larger training error. In fact, we also tried a even smaller coefficient 1e-05, but since the larger coefficients have already shown a significant overfitting, there is no necessity to report that. Our best performance of L2 penalty also has about 85%.

4.3. Dropout Method

Experiments also conducted for models using Dropout method with different inclusion probabilities. Firstly, we tried the model setting suggested by (Nitish Srivastava, 2014), 0.5 for hidden layers and 0.8 for inputs, but the model does not converge at all, since our training dataset is exponentially smaller in size and our model only have 128*3 units. Therefore, we decided only apply Dropout on the hidden units in all hidden layer with inclusion probabilities 0.9, 0.8, 0.7, 0.6 and 0.5 (Figure 7). One significant

Algorithm 6 Dropout layer backword propagation

Input: inclusion probability p, mask generated in forward propagation m_{ij} and gradient w.r.t. output of this layer $\Delta_{X_{ij}} E^n$:
$\Delta_{X_{ij}} E^n = m_{ij} \times \Delta_{X_{ij}} E^n$
Output: masked gradient $\Delta_{X_{ij}} E^n$

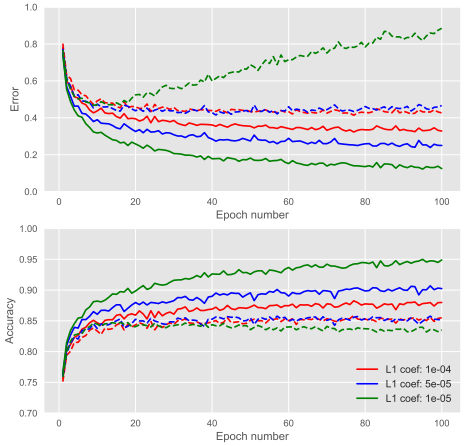

Figure 5. Error and Accuracy curve on the training and validation set of models with L1 penalty coefficient 1e-04, 5e-05, 1e-05 respectively.

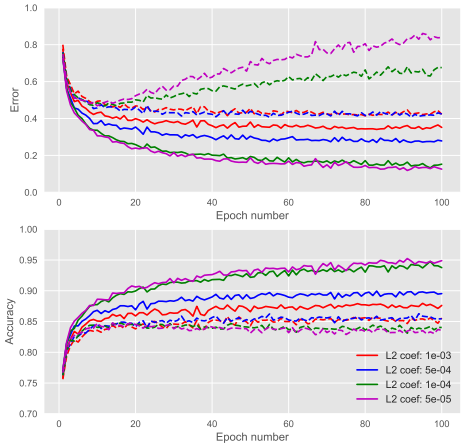

Figure 6. Error and Accuracy curve on the training and validation set of models with L2 penalty coefficient 1e-03 ,5e-04, 1e-04, 5e-05 respectively.

difference from the weight penalty is that Dropout error start with a very large value but rapidly declined (cannot shown in figure due to a more suitable setting of y-axes for displaying other cases). This is because when inclusion probability is smaller, the fewer units are activated to train the model, which leads to a similar situation as the 32-units model discussed above whose error also start decreasing at a relatively large value.

Also, even for the slightly overfit case, inclusion probability = 0.9, the difference between validation error and training error is only about 0.25 which is smaller than the overfit case of L1 and L2. Moreover, the strong overlapping of plots for different setting of models shows effective improvement on overfitting by applying Dropout method. Therefore, we believe Dropout method has a better generalisation performance in most cases, though the best validation accuracy (inclusion prob = 0.9) is still around 85%.

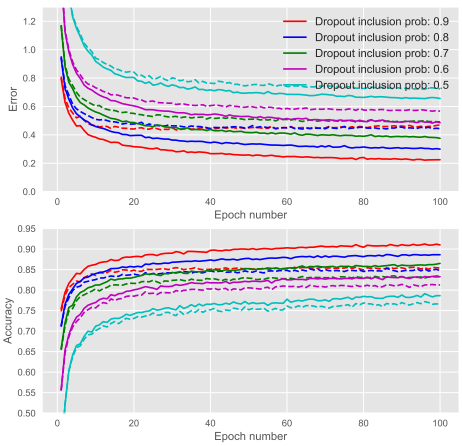

Figure 7. Error and Accuracy curve on the training and validation set of models with inclusion probability 0.9, 0.8, 0.7, 0.6 and 0.5 respectively.

Configurations	Final Valid Acc
Dropout + L1 5e-05	85.53%
Dropout + L1 1e-05	85.94%
Dropout + L2 5e-04	85.29%
Dropout + L2 1e-04	86.01%
Dropout + L2 1e-04 + lr.=3e-04	86.06%

Table 1. Different combination's validation accuracy at epoch 100. Dropout inclusion probability used in each configuration is 0.9. 'lr.' is short for learning rate.

4.4. Other combinations

Based on the results obtained above, we decide to try several combinations of our best Dropout methods with different settings of L1/L2. We did not the best performed L1 and L2 because combining with Dropout, the model might be underfitting, so slightly overfit cased are applied with Dropout. The Table 1 shows that L1 and L2 actually can reach similar performance, we chose Dropout0.9+L2(1e-04) for further changing only because it is used more commonly in application. On top of that, we noticed there is a quite wiggly shape of curved in this chosen combination (Figure 8), which might be resulted by a slightly large learning rate. Thus, we tried different learning rate for this combination and an better convergence occurred when using a third of the default learning rate (shown in Figure 9). Even though the training set is a little underfitting, the validation accuracy converges well and steady after 80 epochs, and the final accuracy is still around 86%.

Therefore, we use this set as our best model and obtained an obvious improvement in test performance compared with the 3-hidden-layers model with 128 unit shown in 10.

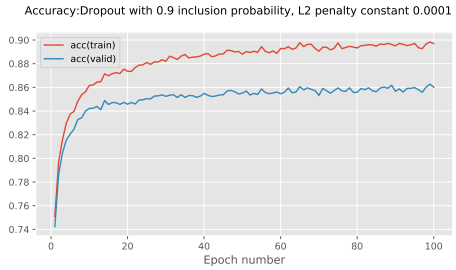

Figure 8. Accuracy curve on the training and validation set of models with 0.9 inclusion probability of Dropout and 1e-04 of L2 penalty.

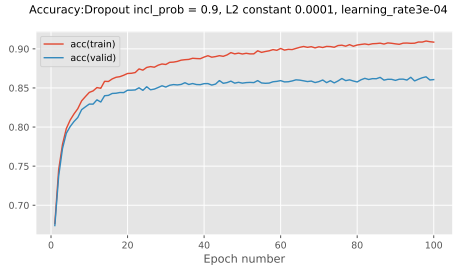

Figure 9. Accuracy curve on the training and validation set of models with 0.9 inclusion probability of Dropout and 1e-04 of L2 penalty. Learning rate is changed to 1e-04

5. Literature Review: *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*

This paper proposed a state-of-art method, Dropout, to avoid the overfitting problem faced by many deep neural networks. It starts with an introduction clarifying the reason of overfitting and the necessity of this kind of method to avoid such problem. Then in section 2, it mentioned that the developing of this method is motivated by the mechanism of sexual reproduction and thinking about successful conspiracies. It also listed and briefly commented on other related work published. Section 4 presented a mathematical detailed description of this method and section 5 mentioned and explained how some approaches can be used to improve performance of dropout, such as max-norm. Also, dropout is proved useful for unsupervised pretraining data when learning rate is small, even though, at first, there were some worries that it will erase the pretraining information.

Staring from section 6, more experimental analysis were conducted. Exhaustive experimental results, ranging from image data to speech data as well as text data, were summarised and proved that dropout can increase performance for all data set, including ImageNet, MNIST and so on. In some tasks, it can decrease error by over 10%, such as in CIAFR-100. Comparing with Bayesian neural network, dropout lost on small data set, but in application, dropout will be less time consuming and already has a good enough performance which beat all other methods in code quality except Bayesian neural network. Moreover, this paper also examined the effect of dropout on neural networks in several aspects: features, sparsity, rate, data size and test time procedure. Dropout was also shown how to be extended to Restricted Boltzman Machines in section 8, and section 9 described deterministic versions of dropout which is results of marginalising noise.

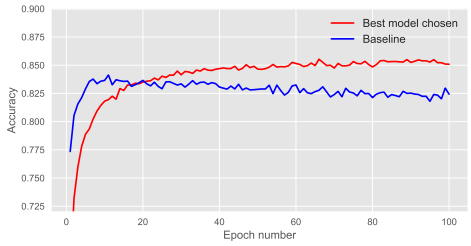

Figure 10. Accuracy curve of baseline and the best chosen model.

Dropout, in this paper, was fully demonstrated as an amazingly efficient technique to solve the overfitting problem and also shown can be applied on a various of cases. However, there is still a little problem of dropout. Even though it is faster than Bayesian neural network, it still more time-consuming to train than a standard neural networks, around 2-3 times longer. The whole idea of dropout seems like finding a balance between speed and accuracy.

6. Conclusions

In this report, we identify a severe overfitting problem in section 2 and proposed some basic idea on mitigate this problem. Then, by trying several simpler variation of the first model, we set a baseline for further work. Thereafter, in section 3, we theoretically introduced three well-known approaches for regularization, and summarised the included algorithms in pseudo-code. In section 4, we step-by-step demonstrated the logic of our exploration for the best configuration of method and selected a best model to check its performance on unseen test data. A significant improvement in our model was shown. In the last section 5 of this report, we closely reviewed the paper of dropout method and have a deeper understanding of how this method work and its strength and limitation. As proposed in (Nitish Srivastava, 2014), the performance of dropout is expected to be better than our results, due to the much simpler model we used. Therefore, a further work can be the experiment on more-units neural network with dropout.

References

Bilen, Hakan. Machine learning practical, lecture 4, 2020.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky Ilya Sutskever Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, pp. 1929–1958, 2014.