MLP Coursework 1

SXXXXXX

Abstract

In this report we study the problem of overfitting, which is the training regime where performance increases on the training set but decrease on validation data. Overfitting prevents our trained model from generalizing well to unseen data. We first analyse the given example and discuss the probable causes of the underlying problem. Then we investigate how the depth and width of a neural network can affect overfitting in a feedforward architecture and observe that increasing width and depth tend to enable further overfitting. Next we discuss how two standard methods, Dropout and Weight Penalty, can mitigate overfitting, then describe their implementation and use them in our experiments to reduce the overfitting on the EMNIST dataset. Based on our results, we ultimately find that both dropout and weight penalty are able to mitigate overfitting.

We further explore alternative loss and activation functions, and evaluate whether their performance relates to problems of overfitting.

Finally, we conclude the report with our observations and related work. Our main findings indicate that preventing overfitting is achievable through regularization, although combining different methods together is not straightforward.

1. Introduction

In this report we focus on a common and important problem while training machine learning models known as overfitting, or overtraining [1], which is the training regime where performances increase on the training set but decrease on unseen data. We first start with analysing the given problem in Figure 1, study it in different architectures and then investigate different strategies to mitigate the problem. In particular, Section 2 identifies and discusses the given problem, and investigates the effect of network width and depth in terms of generalization gap (see Ch. 5 in Goodfellow et al. 2016) and generalization performance. Section 3 introduces two regularization techniques to alleviate overfitting: Dropout (Srivastava et al., 2014) and L1/L2 Weight Penalties (see Section 7.1 in Goodfellow et al. 2016). We first explain them in detail and discuss why they are used for alleviating overfitting. We extend our study to implement a combined L1 and L2 penalty.

In Section 4, we incorporate each of them and their various combinations to a three hidden layer[2] neural network, train it on the EMNIST dataset, which contains 131,600 images of characters and digits, each of size 28x28, from 47 classes. We evaluate them in terms of generalization gap and performance, and discuss the results and effectiveness of the tested regularization strategies. Our results show that both dropout and weight penalty are able to mitigate overfitting. We end Section 4, by testing an alternative activation function, and explore whether the observed performance degradation relates to overfitting or some other problem.

Finally, we conclude our study in Section 5, noting that preventing overfitting is achievable through regularization, although combining different methods together is not straightforward.

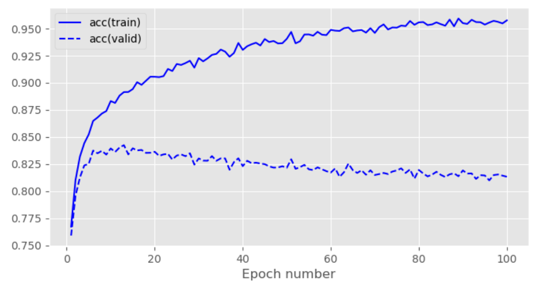
2. Problem identification

Overfitting to training data is a very common and important issue that needs to be dealt with when training neural networks or other machine learning models in general (see Ch. 5 in Goodfellow et al. 2016). A model is said to be overfitting when as the training progresses, its performance on the training data keeps improving, while its is degrading on validation data. Effectively, the model stops learning related patterns for the task and instead starts to memorize specificities of each training sample that are irrelevant to new samples. Overfitting leads to bad generalization performance in unseen data, as performance on validation data is indicative of performance on test data and (to an extent) during deployment.

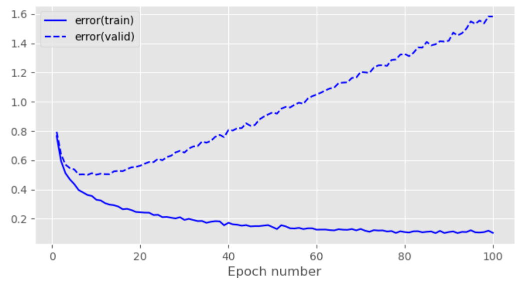
Although it eventually happens to all gradient-based training, it is most often caused by models that are too large with respect to the amount and diversity of training data. The more free parameters the model has, the easier it will be to memorize complex data patterns that only apply to a restricted amount of samples. A prominent symptom of overfitting is the generalization gap, defined as the difference between the validation and training error. A steady increase in this quantity is usually interpreted as the model

[1]Technically, overtraining is the act of training beyond a point at which performance degrades; while overfitting is training to the extent that the learnt function is more complex than required to capture the training data. They correlate, but they are not, strictly speaking, the same thing. To keep things simple, and according to common usage of the terms, we are using them interchangeably in this document, and referring to their joint effect.

[2]We denote all layers as hidden except the final (output) one. This means that depth of a network is equal to the number of its hidden layers + 1.



(a) accuracy by epoch



(b) error by epoch

Figure 1. Training and validation curves in terms of classification accuracy (a) and cross-entropy error (b) on the EMNIST dataset for the baseline model.

# Hidden Units	Val. Acc.	Train Error	Val. Error
32	–.-	-.—	-.—
64	–.-	-.—	-.—
128	–.-	-.—	-.—

Table 1. Validation accuracy (%) and training/validation error (in terms of cross-entropy error) for varying network widths on the EMNIST dataset.

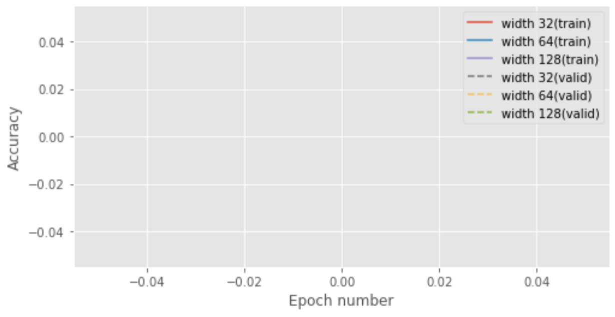
entering the overfitting regime.

Figure 1a and 1b show a prototypical example of overfitting. We see in Figure 1a that **[Question 1 - Explain what these figures contain and how the curves evolve, and spot where overfitting occurs. Reason based on the min/max points and velocities (direction and magnitude of change) of the accuracy and error curves]** .

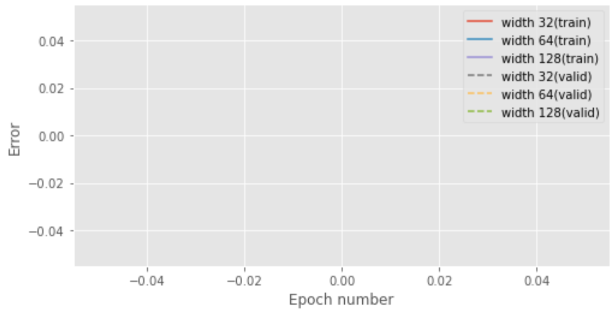
The extent to which our model overfits depends on many factors. For example, the quality and quantity of the training set and the complexity of the model. If we have sufficiently many diverse training samples, or if our model contains few hidden units, it will in general be less prone to overfitting. Any form of regularization will also limit the extent to which the model overfits.

2.1. Network width

[Question Table 1 - Fill in Table 1 with the results from your experiments varying the number of hidden units.] [Question Figure 2 - Replace the images in Figure 2 with figures depicting the accuracy and error, training



(a) accuracy by epoch



(b) error by epoch

Figure 2. Training and validation curves in terms of classification accuracy (a) and cross-entropy error (b) on the EMNIST dataset for different network widths.

and validation curves for your experiments varying the number of hidden units.]

First we investigate the effect of increasing the number of hidden units in a single hidden layer network when training on the EMNIST dataset. The network is trained using the Adam optimizer with a learning rate of 9×10^{-4} and a batch size of 100, for a total of 100 epochs.

The input layer is of size 784, and output layer consists of 47 units. Three different models were trained, with a single hidden layer of 32, 64 and 128 ReLU hidden units respectively. Figure 2 depicts the error and accuracy curves over 100 epochs for the model with varying number of hidden units. Table 1 reports the final accuracy, training error, and validation error. We observe that **[Question 2 - Present your network width experiment results by using the relevant figure and table]** .

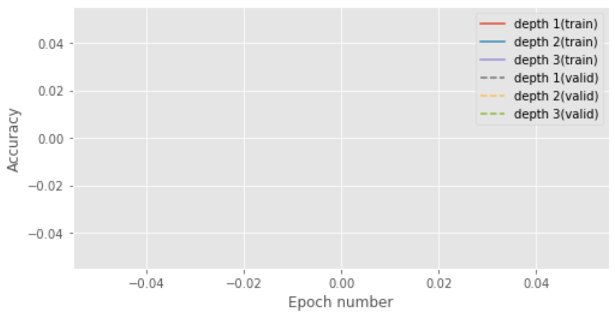
[Question 3 - Discuss whether varying width affects the results in a consistent way, and whether the results are expected and match well with the prior knowledge (by which we mean your expectations as are formed from the relevant Theory and literature)] .

2.2. Network depth

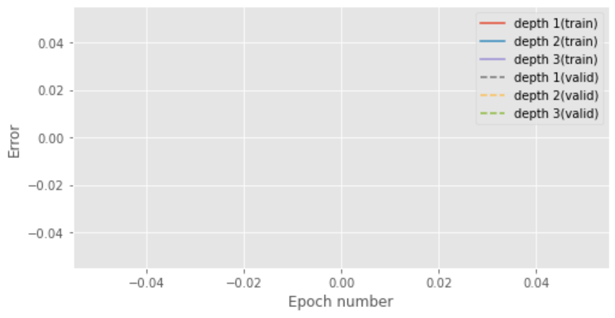
[Question Table 2 - Fill in Table 2 with the results from your experiments varying the number of hidden layers.] [Question Figure 3 - Replace these images with figures depicting the accuracy and error, training and validation curves for your experiments varying the

# Hidden Layers	Val. Acc.	Train Error	Val. Error
1	–.-	-.—	-.—
2	–.-	-.—	-.—
3	–.-	-.—	-.—

Table 2. Validation accuracy (%) and training/validation error (in terms of cross-entropy error) for varying network depths on the EMNIST dataset.



(a) accuracy by epoch



(b) error by epoch

Figure 3. Training and validation curves in terms of classification accuracy (a) and cross-entropy error (b) on the EMNIST dataset for different network depths.

number of hidden layers.]

Next we investigate the effect of varying the number of hidden layers in the network. Table 2 and Figure 3 depict results from training three models with one, two and three hidden layers respectively, each with 128 ReLU hidden units. As with previous experiments, they are trained with the Adam optimizer with a learning rate of 9×10^{-4} and a batch size of 100.

We observe that **[Question 4 - Present your network depth experiment results by using the relevant figure and table]** .

[Question 5 - Discuss whether varying depth affects the results in a consistent way, and whether the results are expected and match well with the prior knowledge (by which we mean your expectations as are formed from the relevant Theory and literature)] .

3. Regularization

In this section, we investigate three regularization methods to alleviate the overfitting problem, specifically dropout layers, the L1 and L2 weight penalties and label smoothing.

3.1. Dropout

Dropout (Srivastava et al., 2014) is a stochastic method that randomly inactivates neurons in a neural network according to an hyperparameter, the inclusion rate (*i.e.* the rate that an unit is included). Dropout is commonly represented by an additional layer inserted between the linear layer and activation function. Its forward pass during training is defined as follows:

$$\text{mask} \sim \text{bernoulli}(p) \quad (1)$$
$$\boldsymbol{y}' = \text{mask} \odot \boldsymbol{y} \quad (2)$$

where $\boldsymbol{y}, \boldsymbol{y}' \in \mathbb{R}^d$ are the output of the linear layer before and after applying dropout, respectively. mask $\in \mathbb{R}^d$ is a mask vector randomly sampled from the Bernoulli distribution with inclusion probability p, and $\odot$ denotes the element-wise multiplication.

At inference time, stochasticity is not desired, so no neurons are dropped. To account for the change in expectations of the output values, we scale them down by the inclusion probability p:

$$\boldsymbol{y}' = \boldsymbol{y} * p \quad (3)$$

As there is no nonlinear calculation involved, the backward propagation is just the element-wise product of the gradients with respect to the layer outputs and mask created in the forward calculation. The backward propagation for dropout is therefore formulated as follows:

$$\frac{\partial \boldsymbol{y}'}{\partial \boldsymbol{y}} = mask \quad (4)$$

Dropout is an easy to implement and highly scalable method. It can be implemented as a layer-based calculation unit, and be placed on any layer of the neural network at will. Dropout can reduce the dependence of hidden units between layers so that the neurons of the next layer will not rely on only few features from of the previous layer. Instead, it forces the network to extract diverse features and evenly distribute information among all features. By randomly dropping some neurons in training, dropout makes use of a subset of the whole architecture, so it can also be viewed as bagging different sub networks and averaging their outputs.

3.2. Weight penalty

L1 and L2 regularization (Ng, 2004) are simple but effective methods to mitigate overfitting to training data. The application of L1 and L2 regularization strategies could be formulated as adding penalty terms with L1 and L2 norm square of weights in the cost function without changing other formulations. The idea behind this regularization method is to penalize the weights by adding a term to the cost function, and explicitly constrain the magnitude of the weights with either the L1 and L2 norms. The optimization problem takes a different form:

$$\text{L1: } \min_{\boldsymbol{w}} E_{\text{data}}(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{w}) + \lambda \|w\|_1 \quad (5)$$

$$\text{L2: } \min_{\boldsymbol{w}} E_{\text{data}}(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{w}) + \lambda \|w\|_2^2 \quad (6)$$

where E_{data} denotes the cross entropy error function, and $\{\boldsymbol{X}, \boldsymbol{y}\}$ denotes the input and target training pairs. λ controls the strength of regularization.

Weight penalty works by constraining the scale of parameters and preventing them to grow too much, avoiding overly sensitive behaviour on unseen data. While L1 and L2 regularization are similar to each other in calculation, they have different effects. Gradient magnitude in L1 regularization does not depend on the weight value and tends to bring small weights to 0, which can be used as a form of feature selection, whereas L2 regularization tends to shrink the weights to a smaller scale uniformly.

3.2.1. Combination of L1 and L2 Regularisation

[Question 6 - Explain how one could use a combination of L1 and L2 regularisation. Discuss any potential benefits of this approach. Present an equation for this combined loss function and explain all relevant variables and hyperparameters.] .

3.3. Label smoothing

Label smoothing regularizes a model based on a softmax with K output values by replacing the hard target 0 labels and 1 labels with $\frac{\alpha}{K-1}$ and $1 - \alpha$ respectively. α is typically set to a small number such as 0.1.

$$\begin{cases} \frac{\alpha}{K-1}, & \text{if} \quad t_k = 0 \\ 1 - \alpha, & \text{if} \quad t_k = 1 \end{cases} \quad (7)$$

The standard cross-entropy error is typically used with these *soft* targets to train the neural network. Hence, implementing label smoothing requires only modifying the targets of training set. This strategy may prevent a neural network to obtain very large weights by discouraging very high output values.

4. Balanced EMNIST Experiments

[Question Table 3 - Fill in Table 3 with the results from your experiments for the missing hyperparameter values for each of L1 regularisation, L2 regularisation, Dropout and label smoothing (use the values shown on the table).]

[Question Figure 4 - Replace these images with figures depicting the Validation Accuracy and Generalisation Gap (difference between validation and training error) for each of the experiment results varying the Dropout inclusion rate, and L1/L2 weight penalty depicted in Table 3 (including any results you have filled in).]

[Question Table 4 - Add a table with the results from your 4 experiments (by which we mean: 4 training runs, NOT 4 sets of experiments) varying your hyperparameters for the combination of L1 and L2 regularisation (based on your implementation).]

[Question Figure 5 - Add figures depicting the Validation Accuracy and Generalisation Gap (difference between validation and training error) for each of your 4 experiments varying your hyperparameters for the combination of L1 and L2 regularisation (based on your implementation) depicted in Table 4.]

[Question Figure 6 - Add figures depicting the Training Accuracy, Error, and Gradient Flow Across Layers for the 1 experiment run with the Custom Activation function (produce these by running the "Problematic Training" cell in the Coursework_1.ipynb notebook).]

Here we evaluate the effectiveness of the given regularization methods for reducing the overfitting on the EMNIST dataset. We build a baseline architecture with three hidden layers, each with 128 neurons, which suffers from overfitting as shown in section 2.

Here we train the network with a lower learning rate of 10^{-4}, as the previous runs were overfitting after only a handful of epochs. Results for the new baseline (c.f. Table 3) confirm that lower learning rate helps, so all further experiments are run using it.

Here, we apply the L1 or L2 regularization with dropout to our baseline and search for good hyperparameters on the validation set. Then, we apply the label smoothing with $\alpha = 0.1$ to our baseline. We summarize all the experimental results in Table 3. For each method except the label smoothing, we plot the relationship between generalization gap and validation accuracy in Figure 4.

First we analyze three methods separately, train each over a set of hyperparameters and compare their best performing results.

[Question 7 - Assume you had a limited experiment budget for exactly 4 experiment runs dedicated to testing the effectiveness of your proposed combination of L1 and L2 Regularization in Section 3.2.1. Argue for which 4 configurations to run based on any previous results or other information you consider relevant. Run those experiments and add the results in Table 4 and Figure 5 (you will then discuss those results alongside all others in the next question below).

(Even if you have not managed to implement your L1 and L2 combination; still argue based on your proposed approach).

] .

[Question 8 - Explain the experimental details (e.g. hyperparameters), discuss the results in terms of their generalisation performance and overfitting. Select and test the best performing model as part of this analysis.]

Model	Hyperparameter value(s)	Validation accuracy	Train Error	Validation Error
Baseline	-	0.837	0.241	0.533
Dropout	0.6	80.7	0.549	0.593
	0.7	–.-	-.—	-.—
	0.85	85.1	0.329	0.434
	0.97	85.4	0.244	0.457
L1 penalty	5e-4	79.5	0.642	0.658
	1e-3	–.-	-.—	-.—
	5e-3	2.41	3.850	3.850
	5e-2	2.20	3.850	3.850
L2 penalty	5e-4	85.1	0.306	0.460
	1e-3	–.-	-.—	-.—
	5e-3	81.3	0.586	0.607
	5e-2	39.2	2.258	2.256
Label smoothing	0.1	–.-	-.—	-.—

Table 3. Results of all hyperparameter search experiments. *italics* indicate the best results per series (Dropout, L1 Regularisation, L2 Regularisation, Label smoothing) and **bold** indicates the best overall.

.

4.1. Custom Activation Function

[Question 9 - A colleague proposed using a custom activation function (which can be identified as CustomActivationLayer() in your Coursework_1.ipynb notebook and the mlp package). Run the prepared experiment in the "Problematic Training" cell in your Coursework_1.ipynb notebook, and display the results in Figure 6. Analyse the results. Is the model overfitting? Explain the cause of the problematic performance of the model?] .

5. Conclusion

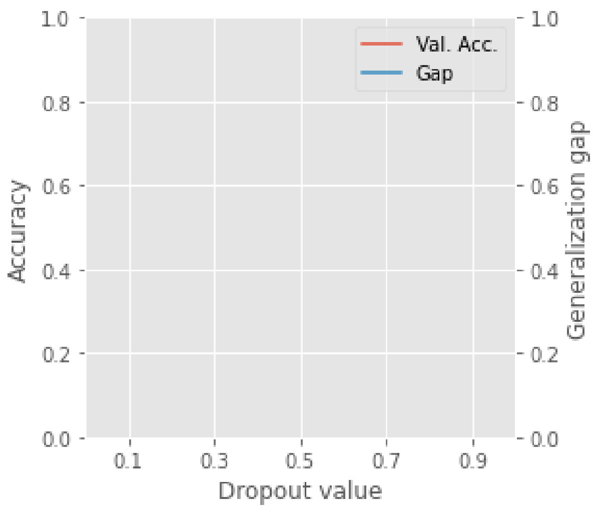
[Question 10 - Briefly draw your conclusions based on the results from the previous sections (what are the take-away messages?), discussing them in the context of the overall literature, and conclude your report with a recommendation for future directions] .

References

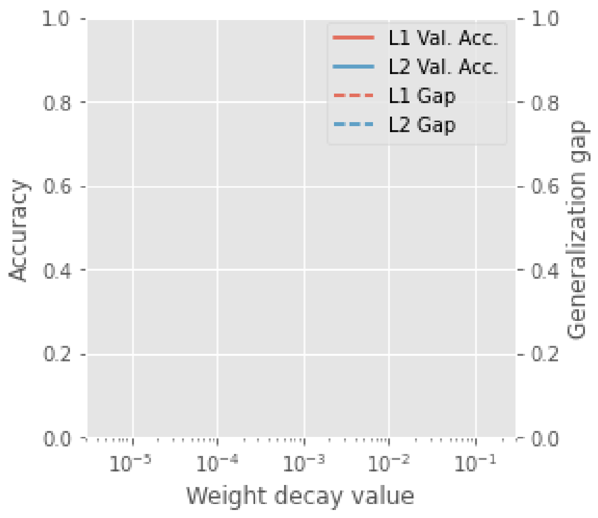
Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Ng, Andrew Y. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 78, 2004.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1): 1929–1958, 2014.



(a) Accuracy and error by inclusion probability.



(b) Accuracy and error by weight penalty.

Figure 4. Accuracy and error by regularisation strength of each method (Dropout and L1/L2 Regularisation).