COMP 551 - Miniproject 3

Pauline Riviere - Brendon McGuinness - Shubhika Ranjan
Group 63

April 1, 2021

Abstract

The project required us to work on a benchmark image dataset, MNIST handwritten digit dataset, to investigate the performance of a multilayer perceptron (MLP) with different architectures. After data vectorization and normalization, training datasets were used to train a MLP with different depth (no hidden layer, 1 or 2 hidden layers), activation functions (ReLu, sigmoid or tanh) and with or without L1 or L2 regularization. Learning rate and regularization term were tuned using cross-validation. In addition to this we tried implementing data augmentation by shifting the original image in 4 direction by a set pixel giving us 300000 images in training dataset.
We found that the introduction of non-linearity and higher depth increased the performance of the classifier. By tuning hyperparameters and trying different activation function, we could further improve the performance but training time was the major limitation in these experiments.

1 Introduction

The purpose of this project was to understand the implementation of MLP from scratch for the MNIST dataset. The MNIST dataset is a collection of images of hand-written digits with their label (0 to 9). It is divided in a training dataset with 60,000 instances and a test dataset with 10,000 instances. The objective of the classification is to be able to predict the digit. Using an optimized MLP, previous authors were able to achieve more than 95% performance on this dataset [1].
We used the MNIST dataset to compare the performance of a softmax classifier, a MLP with 1 or 2 hidden layers with ReLu activation and a MLP with 2 layers with sigmoid or tanh activation. We also tested the effect of L1 and L2 regularization and tried to improve the performance using data augmentation. Overall, we found that a MLP with 2 hidden layers performed better than a regular softmax classifier or a MLP with 1 hidden layer. We observed a better accuracy with sigmoid activation function but the training time was much longer than with ReLU activation. We compared the performance of normalised and unnormalized dataset on the aforesaid architecture. We noticed that not only the accuracy was better but also the computation time was comparatively very low when using normalised set of data as compared to unnormalized ones. To further improve the performance of our model we introduced the concept of data augmentation in our data preprocessing, but this made the model more computationally expensive.

2 Datasets

The MNIST dataset contains 70,000 images of hand-written digits labelled with their digit from 0 to 9 and divided in a train set (60,000 instances) and test set (10,000 instances). There is an equal distribution of classes in the two datasets as shown in Appendix Fig A.1. As mentioned in the repository, the original black and white images were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field.
After importing the data using keras(the open-source python library), the 28x28 matrices for each images were

flattened to a vector of length 784. Next, we normalized the intensity of the pixels from [0,255] to the [0,1] range by dividing the values by 255. We also transformed the 0 to 9 labels using one-hot encoding, i.e. the (N,1) vector label was transformed into a (N,10) matrix with a 1 at the position of the digit and 0s otherwise.

3 Results

For all our MLP models, we used a softmax layer at the end to perform multiclass classification and we used softmax cross entropy as cost function. We used mini-batch gradient descent to find the best parameters for the models, with a batch size of 100, and the algorithm was stopped when the change in the gradient became too small.

We tuned the learning rate and the regularization term if applicable using 3-fold cross-validation. We selected 3 as number of folds because of the time needed to train the model (around 3 hours) and we tried 3 values for each hyperparameter on an iterative manner. For example, for 2 hidden layers and ReLu, we first tried 0.1, 1 and 10, 1 giving the best accuracy but 10 giving low accuracy. Next, we tried 2,5,7 and 2 gave a better accuracy than 1 but 5 gave a low accuracy. Lastly, we tried 2, 2.5 and 2.8 and 2.5 gave the best accuracy. Given the time required for each training (more than 6 hours for each value with 3-fold cross-validation), it would have been too computationally expensive to try random values to start with.

To plot the test and train performance as a function of training epochs, we used mini-batch gradient descent as well but the algorithm wasn't stopped using the change criteria, only using the number of iterations (number of iterations needed for one epoch = total number of instances/batch size, that is 600 with the MNIST training dataset).

3.1 Effect of non-linearity, network depth and activation function

We evaluated the performance of the models on the test set using different architectures: no hidden layers, a single hidden layers having 128 units and ReLu activation, two hidden layers each having 128 units with ReLu activations, two hidden layers with sigmoid activations, two hidden layers with tanh activation. Results are presented in Table 1 below.

Hidden layers	Activation	Accuracy
None	na	91.9%
One	ReLu	93.3%
Two	ReLu	93.7%
Two	Sigmoid	96.8%
Two	Tanh	90.4%

Table 1: Comparison of accuracy of the above mentioned models according to depth and activation function

We observed a higher accuracy with the introduction of non-linearity and with higher depth:

- The MLP model with 1 hidden layer and ReLu activation (non-linearity) was more effective than the MLP without hidden layer aka softmax classifier (+1.4%)
- The MLP model with 2 hidden layers and ReLu activation was more effective than the MLP with a single hidden layer (+0.4%)

With respect to the activation functions, sigmoid function gave the best accuracy (+3.1% compared to ReLu and +6.4% compared to tanh).

The confusion matrix for the MLP with ReLu and 2 hidden layers is presented in Table 2 below. We observe that the errors came mainly from the MLP confusing 4s and 9s and 8s and 5s.

Digit	0	1	2	3	4	5	6	7	8	9
0	959	0	1	1	0	3	10	2	4	0
1	0	1111	4	2	1	1	3	2	11	0
2	8	7	953	12	7	2	11	10	19	3
3	0	0	17	943	1	17	2	14	13	3
4	1	1	5	1	931	0	12	3	5	23
5	8	1	3	32	8	793	14	4	22	7
6	9	3	4	1	12	11	915	0	3	0
7	3	7	24	5	6	0	0	964	3	16
8	7	5	5	20	10	22	10	13	879	3
9	9	6	1	11	38	8	1	11	6	918

Table 2: Confusion matrix for the MLP with 2 hidden layers and ReLu activation

3.2 Effect of regularization

3.2.1 L2 and L1 regularization

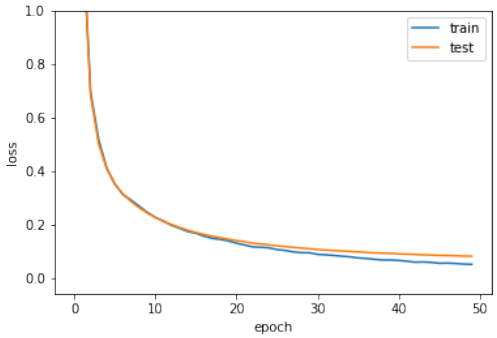
Regularization was used to prevent overfitting the training data. We used both L2 and L1 regularization and the performance for each method is shown in the table below (Table 3) on the two hidden layer ReLu model. As shown, L2 regularization increased performance while L1 regularization decreased performance. A possible explanation is that L1 regularization essentially does feature selecting thus only captures spatially local features on this flattened image dataset (see discussion). In figure A.2, we plotted different strengths of regularization (λ) against performance on samples of the total dataset (N=3000). This was done due to the code being very computationally expensive to run and often taking more than a full day to complete. We used $\lambda = 0.005$ for both L2 and L1 because they were returning the best values on our sampled dataset and ran the code on the full dataset to get the performance values in Table 3 below.

Type of Regularization	Performance
L2	96.5%
L1	11.3%

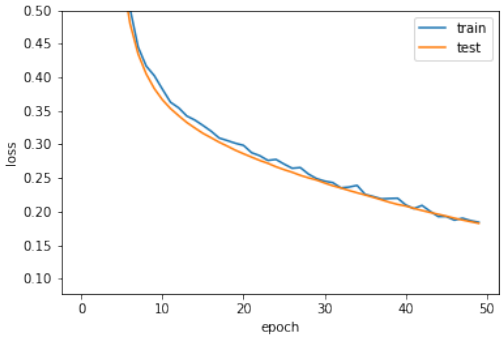
Table 3: Accuracy of the MLP with 2 hidden layers and ReLu activation using regularization

3.3 Test and train performance as a function of training epochs

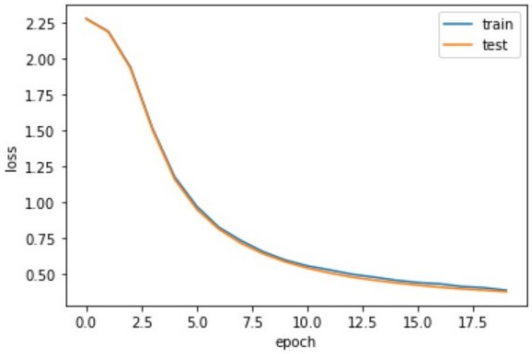
We plotted the performance as a function of training epochs as shown in Figure 1a below. We didn't observe a increase in test error after a certain number of training epochs as expected. This could be due to the fact that we only trained for 20 to 50 epochs depending on the models due to the time needed (around 16 hours for 50 epochs for sigmoid activation) as we see that the cost was still decreasing after 50 epochs. Overfitting could occur for a larger number of epochs.



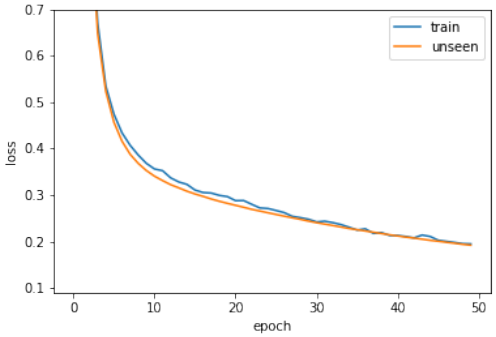
(a) Performance of MLP with 2 hidden layers and sigmoid activation function



(b) Performance of MLP with 2 hidden layers and ReLu activation function



(c) Performance of MLP with 2 hidden layers and tanh activation function



(d) Performance of MLP with 1 hidden layer and relu activation function

Figure 1: Performance of classifiers in terms of their accuracy plotted against fraction of total training set used to fit model.

3.4 Effect of images normalization

As we can notice in the table 4 below, our MLP model works better for the normalized set of data than the unnormalized datset. The normalisation not only eases the learning for the nodes in the layers that follows, but also helps with scaling of the inputs. Normalisation ensures that values that a particular feature would assume are somewhat similar, as the speed of learning is usually proportional to the magnitude of the inputs. If the inputs we provide are of varying scale, the weights connected to few of them will be updated much faster as compared to others.

Different architecture	Performance on unnormalized data	Performance on normalised data
No hidden layer	91.4%	91.9%
1 hidden layer with ReLu function	84.3%	93.3%
2 hidden layer with ReLu function	90.3%	93.7%
2 hidden layer with sigmoid function	94.2%	96.8%
2 hidden layer with tanh function	89.1%	90.4%

Table 4: Performance of our architecture on unnormalised dataset and normalised dataset

3.5 Effect of data augmentation

The data augmentation in this scenario was performed by moving the original images by a few pixels in four selected direction. The resulting training set was of size 300000 images, which increased the accuracy of MLP with 2 hidden layer ReLu activation to 96.7% but at the same time increased the computational cost manifold

times. To train the model here, it took almost 11hours on the virtual environment. But as we can notice from the confusion matrix A.1, the performance comparatively increased for almost all the classes along with 4 and 8 (as compared to original data which had lower accuracy) except for 6 and 9 (which in contrast decreased than before).

4 Discussion and Conclusion

We observed that non-linearity and increasing depth were an efficient way to improve performance. One future direction would be to evaluate the performance of a 3 hidden layers model. Glorot et al. obtained up to 98% accuracy with a 3 hidden layers MLP with ReLu activation [2]. In our work, sigmoid activation function gave the best accuracy. However, the training time was twice as long as with ReLu activation function. For an application in real-world, this trade-off between performance and training time would need to be accounted for. A possible improvement could be to the use of ReLu and sigmoid activation functions in different hidden layers with different combinations.

We added weight decay (L2) regularization into our 2 hidden layers MLP. We found that it increased model performance when an appropriate strength of regularization (λ) term was used. It was a bit difficult to navigate because in our implementation many ranges of λ values would lead to our weights having NaN values. Additionally accounting to how much computationally expensive the 2-layers MLP were, we ran our model on subsamples of the dataset to observe how changing the strength of regularization would influence our models performance. This gave us an optimal value of $\lambda = 0.005$ which gave 96.5% accuracy when trained with the full data set. As an additional experiment we were curious to see how L1 regularization would affect our MLP's perfomance. We found that L1 regularization actually vastly decreased our models performance to about a random guess. Given that L1 regularization pushes many weights to 0 essentially doing feature selection, it may not be appropriate to use on 1-D flattened image data where features are spatially global. On a study on regularization methods used on image data it was shown that L1 regularization works better with datasets which have much more spatially local features whereas L2 regularization works better with data which have much more spatially global features [3]. What flattening an image does is put the following row and the end on the previous row meaning that features that were to an extent spatially local now become global as they as spread out into a single row. Future work could use other forms of regularization such as dropout which in previous work on a single layer MLP has shown to be more effective even than L2 regularization [4].

The normalisation of image dataset ensures that there are both negative and positive values fed to the next layer in order to make training easier. The normalisation is generally preferred to make the gradient calculation easy, which was clearly noticed with respect to the computational time both the training models required. The training on unnormalised data not only took longer but also reduced the efficiency of our model, although not a great extent but the difference could be greater if the dataset was of smaller size.

The employment of data augmentation was to see the effect of sample size. It has been reported in various literatures that data augmentation increases the efficiency of models, since it will be training on somewhat modified dataset. This in turn is supposed to increase the test accuracy. As already mentioned before, the data augmentation increased the accuracy of our model, but increased the computational cost to a very large extent. And since the cross validation could not be performed over this model to determine the various hyperparameters, there is a possibility that the model performance can be further increased. Also, the accuracy for "6" and "9" decreased here. We assume that the sifting of images have caused ambigous learning for these two classes, that decreased their accuracy. The future work in this regard can be various other forms of data transformation along with image shift, that will further increase the data size and maybe its accuracy.

Our model can be futher extended to be used in CNN. This model can also be developed using pytorch which will then enable us to use GPU and decrease its computational cost.

5 Statement of contributions

Pauline performed Task 1, implemented the MLP with 0 and 1 hidden layer and 2 layers with sigmoid activation. Shubhika implemented the MLP with 2 layers and tanh activation and the experimentation on unnormalized data set along with data augmentation. Brendon implemented the MLP with 2 layers and relu activation and regularization. All of us contributed to the writing of the report and approved the final version of the manuscript.

References

[1] Alejandro Baldominos, Yago Saez, and Pedro Isasi. A survey of handwritten character recognition with mnist and emnist. *Applied Sciences*, 9(15), 2019.

[2] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav DudÃk, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[3] Shruti Jaiswal, Ashish Mehta, and GC Nandi. Investigation on the effect of l1 an l2 regularization on image features extracted using restricted boltzmann machine. In *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 1548–1553. IEEE, 2018.

[4] Ekachai Phaisangittisagul. An analysis of the regularization between l2 and dropout in single hidden layer neural network. In *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*, pages 174–179. IEEE, 2016.

6 Appendix

6.1 Figures

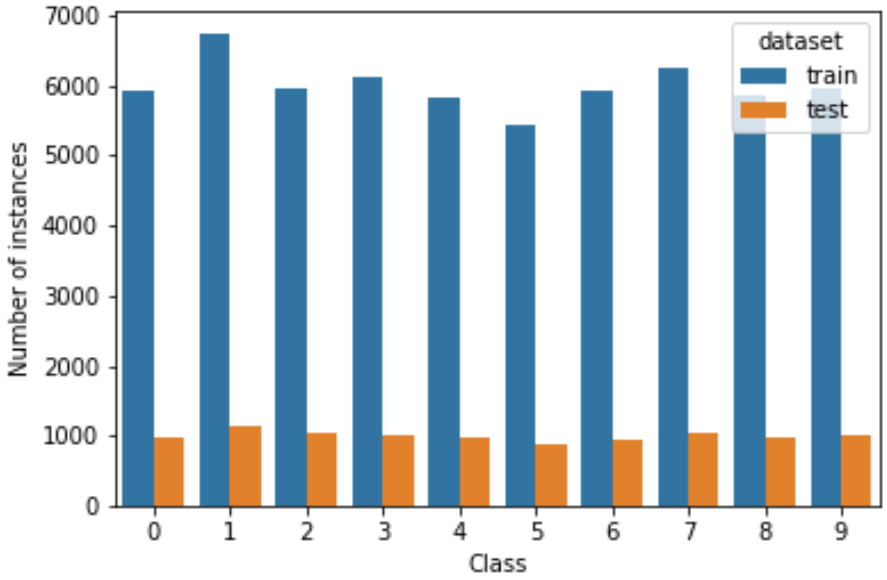


Figure A.1: Distribution of classes over the MNIST train and test datasets

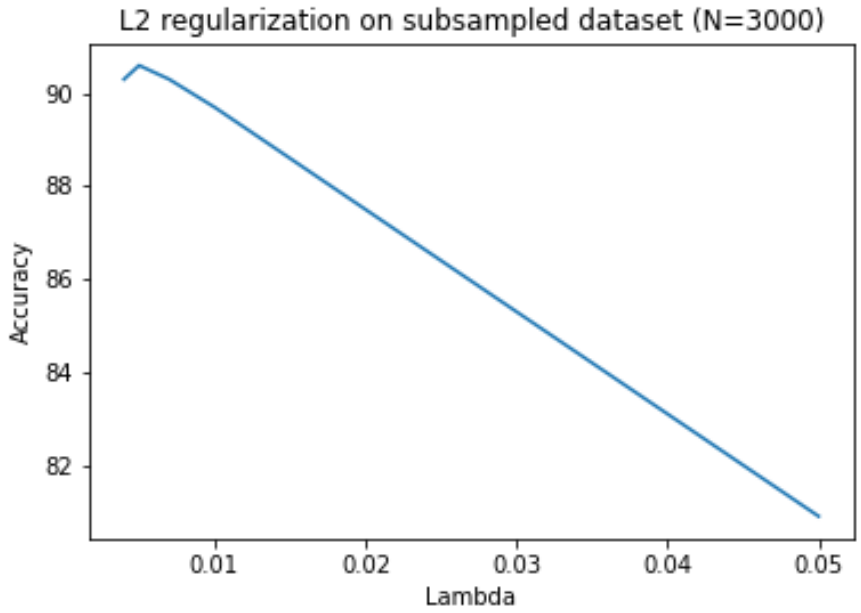


Figure A.2: Performance of 2-layer MLP with L2 regularization and ReLu as its activation function. Performance is plotted against lambda parameter, This was done with a sample of 3000 data points of the dataset due to computational constraints and the time it took to run the code. The first performance point shown was for $\lambda = 0.004$ and lower values than that would lead to NaNs in our model. Additionally higher lambda values would lead to these same errors.

Digit	0	1	2	3	4	5	6	7	8	9
0	964	0	1	1	0	1	7	2	3	0
1	0	1122	0	2	1	0	1	2	7	0
2	6	7	953	12	7	3	12	10	18	3
3	0	1	15	951	1	12	3	13	11	3
4	1	0	4	1	938	0	9	4	5	20
5	8	2	4	32	8	791	14	4	22	7
6	27	3	19	1	19	15	871	0	3	0
7	2	5	21	6	6	0	1	971	4	12
8	5	5	4	10	13	18	5	15	898	1
9	9	9	4	100	130	15	0	53	12	767

Table A.1: Confusion matrix for the MLP with 2 hidden layers and ReLu activation with augmented dataset