Classification of Textual Data - Comparison and Analysis of Naive Bayes and Logistic Regression

Yuhe Fan, Xinyu Wang, Yicheng Huang

April 3, 2022

1 Abstract

In this project, we implemented a Multilayer Perceptron to classify an image data: Fashion-MNIST. For data processing, we loaded the pictures regarding each pixel as a feature and vectorized the data set. To explore how MLP architectures can affect its performance, we created various models with different layer numbers, active function, dropout regularization, data normalization and evaluated the performance. The result shows that larger layer number has better performance, ReLU has better performance, normalization would improve performance and dropout regularization would increase performance first and then decrease performance. Also ,we compared the performance of MLP and CNN and found that CNN would perform better in this dataset. Finally, we tried to optimize the performance of MLP by changing its architecture and made its performance approached CNN.

2 Introduction

Regarding the implementation, we make MLP as a python class. The constructor takes input as the number of hidden units, the number of layers, the active functions, the number of training epochs, scale factor, optimizer, and ADAM parameters. By adjusting the input argument, we created models with different layer numbers, active function, drop regularization, and data normalization for the experiment. Fashion-MNIST is used for training and testing our models in this project. The dataset is provided by Zalando consisting a total of 60000/10000 training/testing instances associated with 10 categories. Comparing the models applied on this dataset, we found how different MLP architecture would affect the performance.

3 Datasets

Fashion-MNIST is a collection of frontal images of clothing (60000 for training, 10000 for testing) associated with 10 categories. Each instance is a 28*28 grayscale image. We plotted the class distribution and found that the examples are uniformly distributed. We vectorized the data for the data to fit into MLP models. Each instance is a 784 (28*28) NumPy array associated with a category.

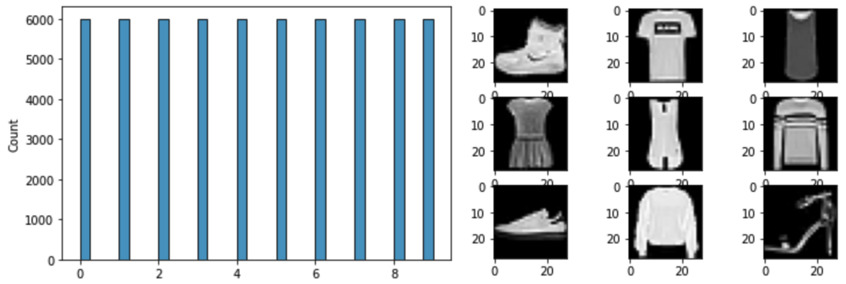

Figure 1: figure distribution and data preview

4 Results

4.1 MLP with different number of layers

To explore how the number of layers affect the model performance, we implemented 3 different models. All of them used the ReLU function, adam gradient descent and softmax as output layer but the number of hidden layers is 0, 1, 2 for different models. The number of units in one hidden layer is fixed to be 128. We trained them with normalized image dataset and tested their performance. The accuracies for 0, 1, 2 hidden layers are 74.87, 81.71 and 82.39 respectively and the loss as a function of epochs is shown as following:

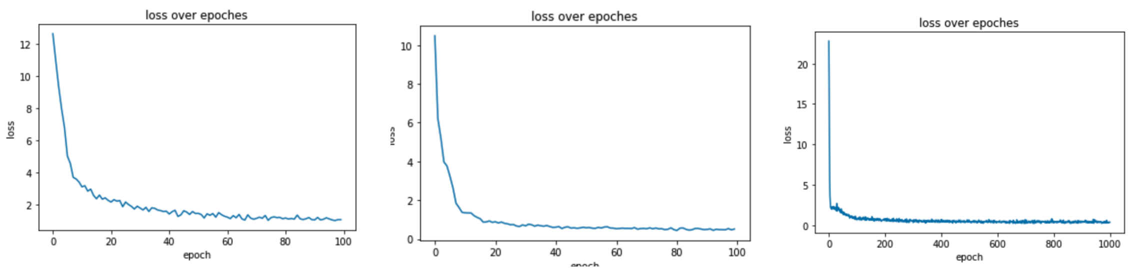

Figure 2: 0 hidden layers (left),1 hidden layer (middle), 2 hidden layer (right)

4.2 MLP with different ative function

To explore how the number of layers affect the model performance, we created two copies of the model with two hidden layers from last experiment and changed the active functions of one copy into tanh and the other into Leaky-ReLU. We trained these model using the same data set used for last experiment and compared their performance. The accuracies of models using ReLU, tanh, Leaky-ReLU are 82.39, 76.6 and 70.49 respectively. The loss as a function of epoches is shown as following:

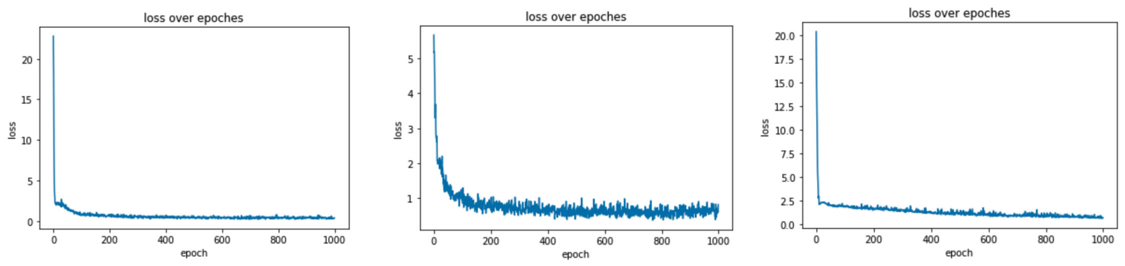

Figure 3: ReLU (left), tanh (middle), Leaky-ReLU (right)

4.3 MLP add/not add dropout regularization

To explore the effect of dropout regularization, we added dropout regularization to the previous 2-layer model with ReLU activations and trained the model with same normalized datasets. The dropout rate was set to 0 (no dropout), 0.1, 0.25, and 0.5. The accuracies of the model with no dropout, 0.1 drop rate, 0.25 drop rate, and 0.5 drop rate are 82.39, 82.01, 82.70, and 76.74 respectively. The losss as a function of epoches is shown as following:

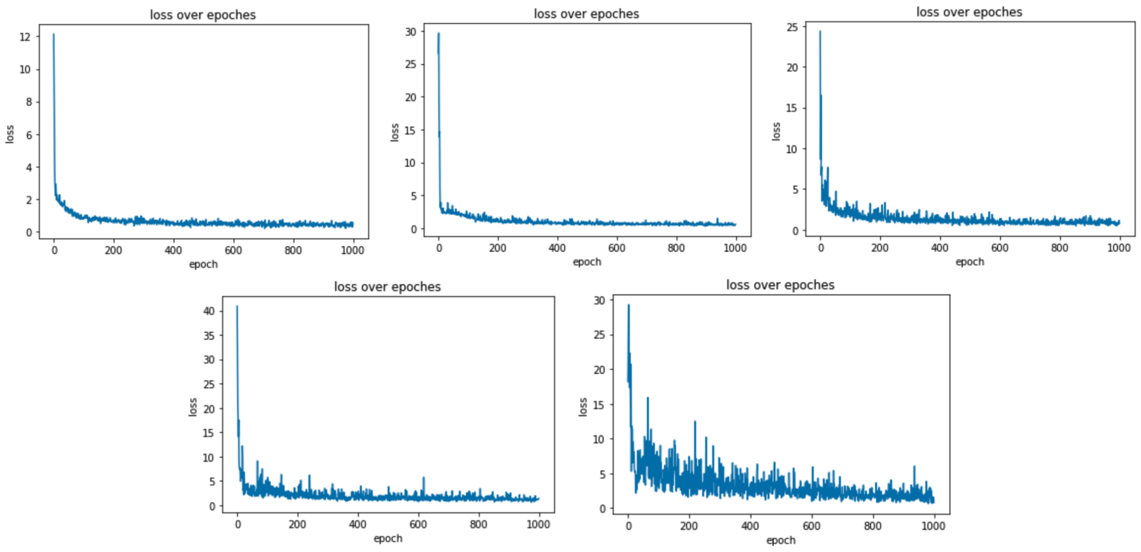

Figure 4: no dropout (top left), 0.2 dropout (top middle), 0.4 dropout (top right), 0.6 dropout (bottom left), 0.8 dropout (bottom right)

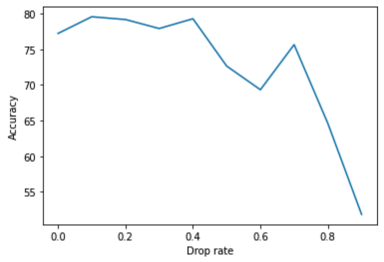

Figure 5: accuracy vs drop rate

4.4 MLP trained with normalized/unnomalized image

To explore how data normalization would affect the model performance, we trained the MLP with 2 hidden layers each having 128 units withReLU activations with both normalized data and and unnormalized data. The accuracy of a model trained with normalized data is 82.39 while the accuracy of a model with unnormalized data is 56.49. The loss as a function of epochs is shown as follows

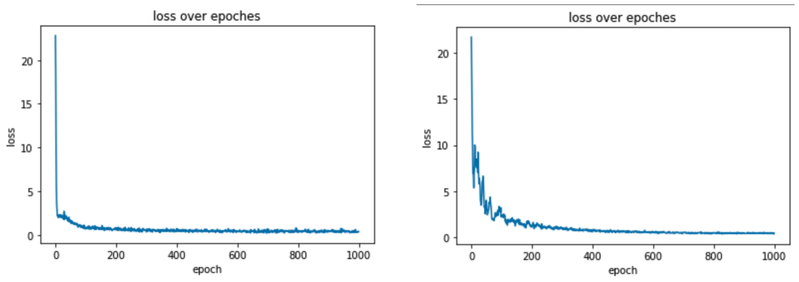

Figure 6: normalized data (left), unnormalized data (right))

4.5 MLP vs. CNN

In this experiment, we used Tensorflow to create a CNN with. 2 convolutional layers and 2 fully connected layers. To increase the model performance, we tried several hyperparameters and the best one has an accuracy of 88.62. Compared with the previous MLP models, This CNN model has higher accuracy.

4.6 improve MLP architecture and compare with CNN

For MLP, in previous experiments we found ReLU activations have better performance, so we decided to improve the performance based on this MLP model. We increased the epoch number and batch size, and the performance increased from 82.39 to 88.14. Accuracy on a testing set of different epoch numbers is calculated to avoid overfitting caused by the increase of epochs. Finally, we got an MLP model with 2 ReLU layer, 128 units per layer, 2000 epochs, 1000 batch size having accuracy 88.14. The accuracy of this model is similar to the CNN created in part 5 but used much more time for training.

5 Discussion

5.1 Optimizer choice

We chose ADAM as our optimizer for the following reasons. ADAM can reduce the oscillations by using the exponential running average of gradients, which would reduce the effect of noise. Moreover, ADAM uses an adaptive learning rate, which would reduce the time (epochs) to approach the global minimum.

5.2 Loss function choice

In this project, MLP is used for multi-classification. Therefore, we used cross-entropy loss as loss function.

5.3 Maximum number of epochs choice

Epoch numbers can be considered as a hyperparameter. AS each epoch can be considered as a single adjustment for the MLP model with a feedforward part and a backpropagation part running on the input batch size of data, and when the batch size is small, the time for executing per epoch is faster. Thus we choose 100 epochs when the batch size is 1000 and choose 1000 epochs when the batch size is 100. In that case, the loss can efficiently converge to a relatively low range and cost less time. More epochs are needed when the model have more hidden layers as they have higher expressiveness and required more adjustment times to reach the low loss.

5.4 Dropout rate

From the plot showing accuracy as a function of drop rate, we can see that as the drop rate increases, the accuracy increase first and then decrease. Thus Dropout may prevent the overfitting by ignoring random neuron units in each epoch and giving more combinations of connecting structures instead of fully connected all units. While dropout may decrease the weight of some important features between layers and leading to decrease the accuracy.

5.5 Normalization

We used normalized data and unnormalized data trained in the same model and found that the normalized data trained model has much better performance. On the other hand, our loss plots show that compared to the model trained by unnormalized data, the loss of the model trained by normalized data converges to a minimum faster and has less oscillation. Both aspects prove that normalization is necessary for data processing.

5.6 MLP vs CNN

In the experiment, we tried a large number of hyper-parameter combinations and increased the epochs (learning time), then finally got an MLP model that had similar accuracy as the CNN model (We did not intentionally try to improve the CNN). Our result shows that MLP and CNN could achieve similar accuracy, but MLP needs more epochs, training time, and more effort to optimize than CNN. Therefor CNN has better performance on classifying the Fashion-MNIST dataset. We think this is because CNN introduced the idea of parameter sharing which would reduce the number of weights parameter and reduce the training time. Parameter sharing also means that CNN would account for spatial information while MLP need vectorized data which means the spacial information would be lost. This also explained why CNN would perform better than MLP in our experiment.

6 Statement of Contributions

All works are distributed equally with group participation and discussion.