Delft University of Technology

Machine Learning for Electrical Engineering
EE4C12

Image Classification Exploiting Sparsity

Author:
Beichen Wang (B.Wang-14@student.tudelft.nl; 5307171)
Jinyi Zou (J.Zou-5@student.tudelft.nl; 5521424)

October 27, 2022

1 Summary

In this project, we aim to classify images in MNIST dataset using machine learning method. 60000 images of handwritten digits, where each image is of size 28×28, are used to be trained and validation, 10000 images are used to be tested. The image data is firstly extracted from the MNIST dataset and then preprocessed. Afterwards, we choose logistic regression, K-Nearest Neighbors(KNN), and Convolution Neural Network(CNN) to implement image classification. Validation is implemented in training each model to adjust hyperparameter. Next, we compare the performance of those models in test set in regarding to classification accuracy, recall, precision and F1 score. Among these three models, CNN shows the best results, followed by KNN and logistic regression performs the worst.

The report is structured as follows. The next section details the methods and the whole pipeline for the experiments. Section 3.1 gives the argumentation for the reason to choose the models and discussion about the performance. Finally, Section 4 present our conclusion and discussion respectively.

2 Pipeline

In this section, the detailed ML pipeline is described. Figure 1 indicates the workflow of the three model. The detailed description for each model is introduced separately in each subsection.

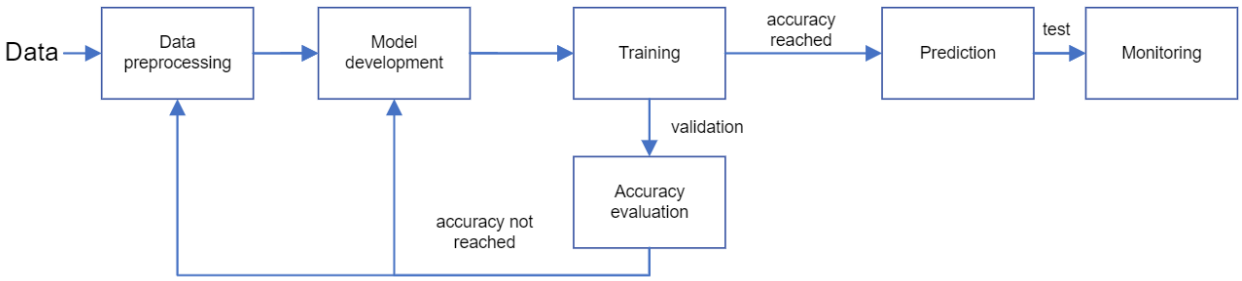

Figure 1: Workflow of image classification based on MNIST

2.1 Logistic Regression

In logistic regression model, the dataset is firstly preprocessed. As the size of each image is 28×28, each image is stored as a vector with length of $784 (= 28 \times 28)$. In the experiment, 60000 data is chosen as the training data and 10000 data is chosen as the testing data. The next step is to define some parameters and functions and construct the model. Bias is firstly set as 0 and weights are set randomly by the code $self.weights = nn.Parameter(torch.randn(784, 10)/math.sqrt(784))$. Afterwards, in each forward propagation, bias and weights are updated to obtain the trained model. Based on grid search, which will be described in Section 3.1, the epoch and learning rate are set as 2000 and 1 in this model. Finally, after the trained model is obtained, we get the accuracy, recall, precision and F1 score with the test set being tested.

2.2 K-Nearest Neighbors

In KNN model, the dataset is firstly preprocessed by normalization. In this and the following CNN experiment, Z-score normalization is implemented, which means that the data are processed by the mean and standard deviation of the dataset in order to make the data follow the standard normal distribution. In the experiment, 60000 data is chosen as the training data and 10000 data is chosen as the testing data. In model construction, we first calculate the distance between points to be classified and points of known class and then select the top K points with the smallest distance from the points to be classified. Finally, an object is classified by a plurality vote of its K nearest neighbors, with the object being assigned to the class most common among its K nearest neighbors. Based on grid search, which will be described in Section 3.1, K is chosen to be 3 in our experiment. Then in training, it returns the category with the highest number of occurrences of the top K points as the

predicted classification of the points to be classified and we get the trained model when the cluster center points become stable. Finally, after the trained model is obtained, we get the accuracy, recall, precision and F1 score with the test set being tested.

2.3 Convolution Neural Network

In CNN model, the dataset is firstly preprocessed by normalization. 60000 data is chosen as the training data and 10000 data is chosen as the testing data. To construct a CNN, an input layer, convolution layer, pooling layer, fully connected layer and output layer should be firstly defined. In our experiment, two convolution layer are chosen. The input channel number is set as 1, the output channel number is set as 10 and the kernel size is set as 5×5 for the first convolution layer. The input channel number is set as 10, the output channel number is set as 20 and the kernel size is set as 5×5 as well for the second convolution layer. For both convolution layers, Rectified Linear Unit (Relu) is used as the activation function. One pooling layer is used for downsampling. A fully connected layer is used, in which the input channel number is set as 320, which equals to the output of last convolutional and pooling layer. And the output channel number is set as 10, which equals to the number of classes in MNIST. Based on grid search, which will be described in Section 3.1, the epoch, batch size and learning rate are set as 10, 128 and 0.1, respectively. Finally, after the trained model is obtained, we get the accuracy, recall, precision and F1 score with the test set being tested.

3 Argumentations

In this section, the process for the model development and validation is introduced, which describes how the hyperparameters in the models are set. Afterwards, the results of each model are shown separately and the comparison is discussed.

3.1 Models Development and Validations

In general, Grid search is used to discover the hyperparameter or a set of hyperparameters that performs best within all three models. In particular, firstly, the number of epochs is determined by pre-experiments. The convergence rate of the model is roughly judged by plotting the loss curve of the model. Based on the loss curve, a sufficient number of epochs is determined and remains constant throughout the subsequent grid search. The reason why the epoch is not searched together with hyperparameters such as learning rate and batch size is that the epoch has little influence on model convergence and parameter update, and the efficiency of grid search within three or more hyperparameters is not high.

The following three tables show how grid search is implemented on three models. As stated, the number of epochs for logistic regression is set to 1000 and for CNN is 10. And for KNN, only the first 1000 testing data are used to grid search because as a non-parametric learning algorithm, KNN spends much more time predicting than training, and 1000 data is sufficient for grid search and selecting the hyperparameter. It is worth mentioning that in the CNN's grid search, different sets of batch size are selected for varied learning rate. Because empirically, smaller learning rate usually require smaller batch size, and vice versa.

Learning Rate	**Accuracy**	**Recall**	**Precision**	**F1**
1	0.9228	0.9217	0.9219	0.9217
0.5	0.9218	0.9208	0.9209	0.9207
0.1	0.9090	0.9078	0.9082	0.9077
0.01	0.8655	0.8632	0.8649	0.8629

Table 1: Grid Search of Logistic Regression

After grid search, the hyperparameter selection of each model is indicated in bold in the table. For logistic regression and CNN, using the hyperparameter or a set of hyperparameters selected and according to the loss curves (plot in Section 3.2), the number of epochs are determined, which is 2000 for logistic regression and 10 for CNN.

K	Accuracy	Recall	Precision	F1
1	0.9622	0.9625	0.9619	0.9622
2	0.9623	0.9624	0.9618	0.9623
3	0.9653	0.9656	0.9657	0.9655
4	0.9644	0.9643	0.9642	0.9642
5	0.9633	0.9636	0.9638	0.9636
6	0.9648	0.9642	0.9643	0.9642
7	0.9634	0.9632	0.9635	0.9633
8	0.9624	0.9626	0.9623	0.9623

Table 2: Grid Search of KNN

(Batch Size, Learning Rate)	Accuracy	Recall	Precision	F1
(32, 0.1)	0.9881	0.9881	0.9880	0.9880
(64, 0.1)	0.9883	0.9883	0.9885	0.9883
(128, 0.1)	0.9904	0.9903	0.9905	0.9904
(256, 0.1)	0.9891	0.9890	0.9891	0.9891
(16, 0.01)	0.9892	0.9892	0.9891	0.9891
(32, 0.01)	0.9851	0.9850	0.9852	0.9850
(64, 0.01)	0.9865	0.9865	0.9865	0.9865
(128, 0.01)	0.9881	0.9880	0.9881	0.9880
(8, 0.001)	0.9859	0.9858	0.9861	0.9859
(16, 0.001)	0.9840	0.9839	0.9841	0.9839
(32, 0.001)	0.9775	0.9773	0.9778	0.9774
(64, 0.001)	0.9713	0.9710	0.9714	0.9711

Table 3: Grid Search of CNN

3.2 Results and Comparisons

In the project, the accuracy, recall, precision and F1 score are chosen to be the criteria to compare the performance of the models.

Accuracy is defined as:

$$Accuracy = \frac{TP + FN}{TP + FP + TN + FN} \tag{1}$$

Recall is defined as:

$$Recall = \frac{TP}{TP + FP} \tag{2}$$

Precision is defined as:

$$Precision = \frac{TP}{TP + FN} \tag{3}$$

F1 is defined as:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4}$$

where TP indicates the tested true positive number, FN indicates the tested false negative number, TN indicates the tested true negative number, and FP indicates the tested false positive number.

3.2.1 Results in logistic regression

Figure 2 indicates the trend of loss function and accuracy in logistic regression. It is noticed that the loss decreases and converges, and the accuracy increases and converges.

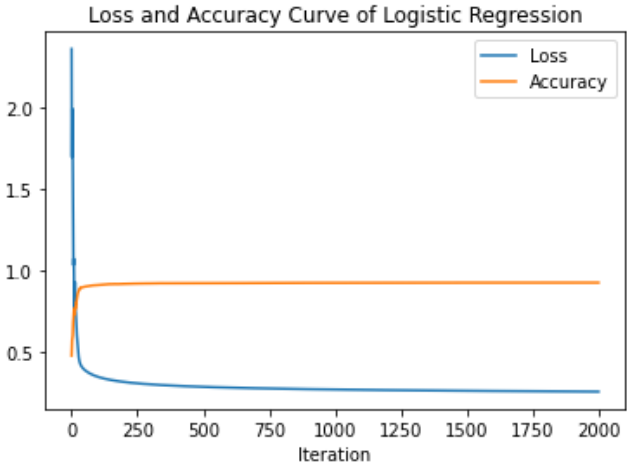

Figure 2: Loss and accuracy vs iteration in logistic regression

Figure 3 shows the accuracy, recall, precision and F1 score in logistic regression.

```
Accuracy: 0.9246000051498413
Recall: 0.9235206805825932
Precision: 0.9236966843709007
F1: 0.9235012622773388
```

Figure 3: Accuracy, Recall, Precision and F1 score in logistic regression

3.2.2 Results in KNN

Figure 4 shows the accuracy, recall, precision and F1 score in KNN.

```
Accuracy: 0.9717
Recall: 0.9713357268999042
Precision: 0.9719031233123564
F1: 0.971528765919764
```

Figure 4: Accuracy, Recall, Precision and F1 score in KNN

3.2.3 Results in CNN

Figure 5 indicates the trend of loss function and accuracy in CNN. It is noticed that the loss decreases and converges to a lower value than that in logistic regression, and the accuracy increases and converges to a higher value than that in logistic regression.

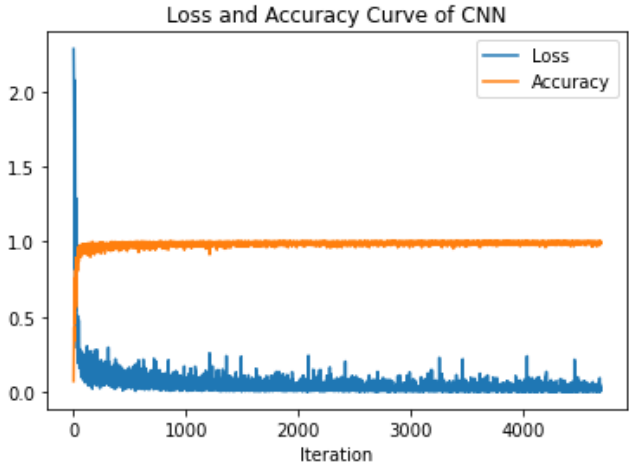

Figure 5: Loss and accuracy vs iteration in CNN

Figure 6 shows the accuracy, recall, precision and F1 score in CNN.

```
Accuracy: 0.9904
Recall: 0.9902974835684455
Precision: 0.9905696364665332
F1: 0.9904138962388028
```

Figure 6: Accuracy, Recall, Precision and F1 score in CNN

3.2.4 Comparison

Accuracy is an intuitive criteria to compare the performance of different models. Precision means the proportion of correctly predicted data in those data predicted as positive examples. Recall means the proportion of correct data predicted in those data that are originally positive examples. Ideally, it is expected that the precision and recall are both high, while, actually, for most of the cases, a higher precision leads to a lower recall, and a higher recall results in a lower accuracy. Considering that, we also choose F1 to compare the performance as it is used to comprehensively evaluate precision and recall.

In our experiments, however, we find that the accuracy, precision, recall and F1 in the model with better performance all higher than those in the model with worse performance. As indicated in Table 4, CNN outperforms KNN and logistic regression with accuracy of 0.9904, recall of 0.9903, precision of 0.9905 and F1 of 0.9904. KNN performs the second best with accuracy of 0.9717, recall of 0.9713, precision of 0.9719 and F1 of 0.9715. Logistic regression performs the worst with accuracy of 0.9246, recall of 0.9235, precision of 0.9237 and F1 of 0.9235.

Model	**Accuracy**	**Recall**	**Precision**	**F1**
Logistic Regression	0.9246	0.9235	0.9237	0.9235
KNN	0.9717	0.9713	0.9719	0.9715
CNN	**0.9904**	**0.9903**	**0.9905**	**0.9904**

Table 4: Results of three models

4 Conclusion

This report presents the whole pipeline to develop machine learning models to implement image classification with MNIST dataset. It is aimed to gain insight into the performance difference of logistic regression, KNN

and CNN and find the model with the best performance.

To conclude, the results of CNN shows that it is the best model among logistic regression, KNN and CNN to implement image classification with MNIST dataset, based on that it has the highest accuracy recall, precision and F1 score.