A neural network classification method for handwritten digit recognition

Xiang Li

Research School of Computer Science, The Australian National University, Canberra
{Xiang Li}@u6342240@anu.edu.au

Abstract. Handwritten recognition has been an ignited research area of machine learning because of possible improvement for human-machine learning interface. I have built a classifier model for handwritten digit recognition on neural network with PyTorch. The dataset includes the vectors of pen-based written number 0-9. The final testing accuracy of classification can reach to 95.91%, which is a little lower than some the results of other authors but higher the other results with different method. The different classifiers, selections, thresholds and the reduction and clean of neurons and patterns impact the result of accuracy.

Keywords: Recognition, pre-processing, dataset, neural network, neuron selection, pattern reduction, loss, heuristic

1 Introduction

The author is confident with his writing; however, his writing is so poor that it is often recognized as other meaning in mistake. Sometimes he even cannot identify the writing of himself. It inspires him to complete a research for handwritten digit recognition. The computing process of writing recognition is a significant approach for those confident writers to become aware how poor they write and lead them to improve the handwriting. It is easy for computers to read the hand writing image or get the dynamic pen movement, but the task is difficult to identify what does the writer really writes. Neural network is an effective method in the area of machine learning to learn the training datasets with some images or vectors and classify the testing datasets.

1.1 Background for handwriting

Handwriting recognition (HWR) is the ability of a computer system to receive and interpret intelligible handwritten input from sources such as paper documents, photographs, touch-screens and other devices. The image of the written text may be read offline or online. For instance, apple pencil for iPad Pro and iPad 2018 allows us to write notes in iPads, it uses the handwriting recognition to distinguish what people are writing, the process is online for iOS to recognize the dynamic writing with pen. Another instance is that taking photographs by using the application Google translate in iPad and recognize what letters are in the photo, the process is offline as the image is static.

1.2 Background for dataset

The raw data is a digit database collection from 44 people for writing the numbers from 0 to 9. After normalization, resampling and converting to bitmaps, blurring and downsampling, the static image has been formatted to the 8-points and one number vector data (Alimoglu and Alpaydin, 1996). Thus, the input vector size is 2*8 +1 = 17 integers, the first 16 integers are the x and y coordinates of 8 points in the order of {x1, y1, … x8, y8} and the last integer is the classification code from 0 to 9. Both the training data and the testing data have the same vector size.

1.3 Method for handwriting recognition

My programming method is PyTorch 0.3.1 based on Python 3.6 with Jupyter Notebook. The classification method is a 4-layer NN.

2 Method

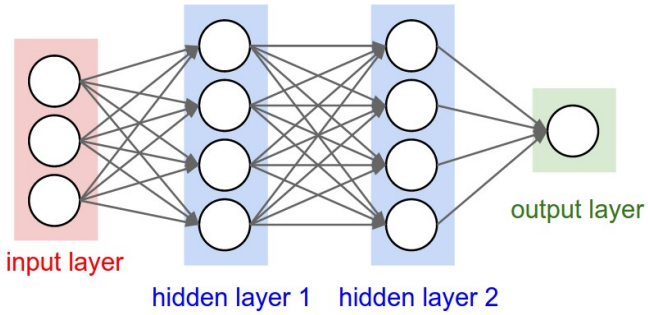
2.1 Data preprocess

I use the data preprocess method, feature scaling. It is a method used to standardize the range of independent variables or features as data normalization. The process is changing x to x′ in the first 16 columns of the data vector into floats from 0 to 1. The prediction column cannot be normalized as the prediction may differ if the number of comparison column changes.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

2.2 Model

The model is built up into 4-layer neural network. It includes an input layer, an output layer and two layers of hidden neurons, the picture below is an example of 4-layer neural network in Google Image.



2.3 Validation set

I set up a validation set from 20% training data in order to prevent overfitting by early stopping training when the prediction accuracy on the validation dataset decreases, as this is a sign of overfitting to the training dataset. The validation process is supposed to start after training at least 20% of training sets as in the early training process, the testing accuracy fluctuates as some data is unfavorable to the training and the number of early samples is too small to prevent the harmful data.

2.4 Threshold

A simple technique of modifying the threshold at the output of neural network to modify the neural network result to minimize the false positive results while keeping correct results as high as possible. (Milne, Gedeon and Skidmore, 1995) The threshold function in PyTorch is torch.nn.Threshold() to add threshold in the last hidden neuron layer.

2.5 Feature selection

After setting all the parameters of the neural network, I use evolutionary algorithm to select the appropriate features for training and remove those features which may impact the training model negatively. It is obvious that all sorts of potential data attributes, or features, are relatively unhelpful for classification. Furthermore, using all of these irrelevant attributes, mixed in with the powerful predictors, may actually have a negative effect on the resulting model. Feature selection is the process of narrowing down a subset of features, or attributes, to be used in the predictive modeling process. Feature selection is a powerful defense against overfitting, increasing model generalizability.

2.6 Neuron reduction

Some of the hidden neural units are excess. Gedeon and Harris in 1995 claimed that those units perform no real function in the final product, and may be supportive of significant units, may solve for other functions, or may literally be unused. In all, they are unnecessary for the post-learning utilization of the network and can be deleted. They provided the method to distinguish those excess units by calculating the angle between pairs of vectors in patterns. When the angle less than approximately 15° are considered sufficiently similar and one of vectors is removed. Alternatively, the angle more than approximately 165° can be seen to be effectively complementary, and both of the vectors can be removed.

2.7 Just reducing training sets

According to Gedeon, Wong and Harris in 1995, frequency distributions of the errors for all patterns in the training set show that very early in training, the distribution of the pattern errors is approximately normal with a large variance. The number of reduced training dataset is 20% as I have set 20% of the training dataset to validation set, cutting off half the rest of data may result in lack of training dataset for classification. The method Error Sign Testing(EST), is used to clean the dataset by comparing the data with loss larger than the average of the whole training dataset. Slade and Gedeon in 1993 state that there are low-error peak and high-error peak in a training model. The data in low-error peak have the loss lower than average of the whole dataset and data in high-error peak have the loss higher than average of the whole dataset. I will split the training data in low-error peak or high-error peak into two datasets by calculating and comparing every single data's loss. At last the two training accuracy results will be compared and discussed.

2.8 Cleaning training sets — Heuristic pattern reduction

Heuristic pattern reduction is introduced by Gedeon and Bowden in 1992. The process of this approach is firstly selecting those data which losses are higher than the average loss of the whole dataset. Then select those data which losses are higher than the average loss of the new dataset and remove them from the whole dataset by recording the index. It is supposed to be a significant method to improve the training model as it double checks the loss of one data and remove the real high-loss data.

3 Results and Discussion

The default optimizer is torch.optim.Adamax () and the default activation function is torch.nn.function.leaky_relu().
The default number of each layer of hidden neurons is 3.
To avoid the data selection repetition, I use a separate dataset as test set different from training set.

3.1 Result of first training

After building up the neuron network, the initial testing accuracy is 82.82% after the first training.

```
Testing Accuracy: 82.82 %
```

3.2. Result after data preprocess

47	100	27	81	57	37	26	0	0	23	56	53
0	89	27	100	42	75	29	45	15	15	37	0
0	100	7	92	5	68	19	45	86	34	100	45
0	67	49	83	100	100	81	80	60	60	40	40
100	100	88	99	49	74	17	47	0	16	37	0
0	100	3	72	26	35	85	35	100	71	73	97
0	39	2	62	11	5	63	0	100	43	89	99
13	89	12	50	72	38	56	0	4	17	0	61
48	96	62	65	88	27	21	0	21	33	79	67
100	100	72	99	36	78	34	54	79	47	64	13
91	74	54	100	0	87	23	59	81	67	100	39
0	85	38	100	81	88	87	50	84	12	58	0
35	76	57	100	100	92	68	66	81	38	82	9
50	84	66	100	75	75	51	51	100	42	97	13
99	80	63	100	25	76	79	68	100	62	97	23
24	66	43	100	59	65	34	28	0	1	16	11

Figure 1.1

0.47	1.00	0.27	0.81	0.57	0.37	0.26	0.00	0.00	0.23	0.56	0.53
0.00	0.57	0.31	0.68	0.72	0.90	1.00	1.00	0.76	0.75	0.50	0.51
0.00	1.00	0.07	0.92	0.05	0.68	0.19	0.45	0.86	0.34	1.00	0.45
0.00	0.67	0.49	0.83	1.00	1.00	0.81	0.80	0.60	0.60	0.40	0.40
1.00	1.00	0.88	0.99	0.49	0.74	0.17	0.47	0.00	0.16	0.37	0.00
0.00	1.00	0.03	0.72	0.26	0.35	0.85	0.35	1.00	0.71	0.73	0.97
0.00	0.39	0.02	0.62	0.11	0.05	0.63	0.00	1.00	0.43	0.89	0.99
0.13	0.89	0.12	0.50	0.72	0.38	0.56	0.00	0.04	0.17	0.00	0.61
0.57	1.00	0.22	0.72	0.00	0.31	0.25	0.00	0.75	0.13	1.00	0.50
0.74	0.87	0.31	1.00	0.00	0.69	0.62	0.64	1.00	0.79	1.00	0.38
0.48	0.96	0.62	0.65	0.88	0.27	0.21	0.00	0.21	0.33	0.79	0.67
1.00	1.00	0.72	0.99	0.36	0.78	0.34	0.54	0.79	0.47	0.64	0.13
0.91	0.74	0.54	1.00	0.00	0.87	0.23	0.59	0.81	0.67	1.00	0.39
0.00	0.85	0.38	1.00	0.81	0.88	0.87	0.50	0.84	0.12	0.58	0.00
0.50	0.84	0.66	1.00	0.75	0.75	0.51	0.51	1.00	0.42	0.97	0.13
0.99	0.80	0.63	1.00	0.25	0.76	0.79	0.68	1.00	0.62	0.97	0.23
0.24	0.66	0.43	1.00	0.59	0.65	0.34	0.28	0.00	0.01	0.16	0.11

Figure 1.2

The two figures are some numbers before preprocess and after preprocess.

After data preprocess, the testing accuracy raises a little to 83.28%.

3.3 Threshold

I change the threshold from 0 to 0.6, as threshold=0.2 and threshold=0.3 have the highest testing accuracy, the testing accuracy is in figure 2:

threshold	test accuracy
0	83.28%
0.1	80.93%
0.2	84.42%
0.25	82.82%
0.3	83.70%
0.4	80.96%
0.5	79.62%
0.6	72.36%

Figure 2

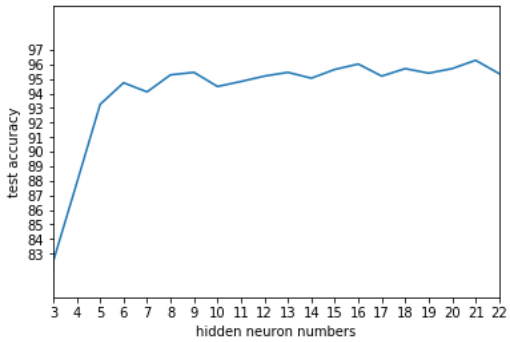
In spite the effect of random seed, the testing accuracy has 2 peaks threshold=0.2 and 0.3. When the threshold value increases from 0, some of small numbers are changed to 0 so that they cannot affect the neural network as the related neurons stop when a value is 0. However, when the threshold is larger than the best value between 0.2 and 0.3, the beneficial data are forced to be 0 because of the restriction of threshold. Thus, the threshold limits the prediction of neural networks.

3.4 Validation Set

I set up a validation set from 20% training data randomly and set break number as 0.05 at each epoch after finishing 30% epochs. When the former accuracy minus present accuracy is higher than break number, the epoch loop breaks and print "the training model is overfitting". Otherwise, it prints "the training model is not overfitting". However, the dataset is not overfitting and it always prints "the training model is not overfitting" and never breaks out of the training loop. Apparently, the validation set is not beneficial to the testing accuracy but just restricts the neural network not overfitting.

3.5 Neuron selection

I built a model from number 3 to number 22 as the hidden neuron units for the neural network model. The figure below shows the testing accuracy within different numbers of hidden neurons.




```
[16, 16]
The training model of dataset is not overfitting
Testing Accuracy: 96.03 %
```

Figure 3

Figure 3 illustrates that the testing accuracy increases significantly within the hidden neuron numbers between 3 and 6. The accuracy fluctuates from 7 to 22 and the highest testing accuracy is 96.03% when the hidden neuron numbers are 16.

3.6 Feature Selection

The function of evolutionary algorithm selects 6 features(individuals) shown in figure 4. However, as the features are 8 pairs of x and y coordinates on a graph, if one feature is selected, the couple features it belongs to will also be selected. Thus, I keep columns 0,1,2,3,8,9,14 and 15. In other words, those are only 4 pairs of coordinates kept in the new training and testing dataset. The result will be discussed and compared in the final comparison.

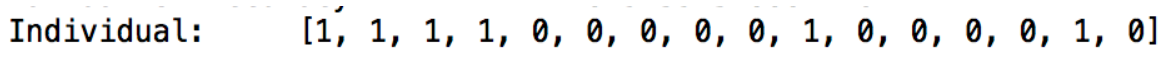


Figure 4

3.7 Neuron reduction

I write a function to show all the angles between different pattern vectors in all layers of hidden neurons. Current layer of hidden neurons minus 1 when the angle is over 165° and current layer of hidden neurons minus 2 when the angle is less than 15°. This function is excess in this dataset as all angles are between 15° and 165°. However, this function may be useful in other datasets.

3.8 Reducing and cleaning dataset

The figure of reduced training dataset is 20%, and I will calculate and print four results: The first one is without any reducing and cleaning method, the second and third are using over_loss function, under_loss function to remove the peak of loss and check the impact of loss and the forth one is using heuristic_pattern_reduction function to clean the dataset by removing those data with higher losses. Additionally, the four results of not reduced training dataset will also be calculated and printed.

3.9 Final result and Comparison

I use random seeds to ensure the dataset keep the same as I remove the data randomly.
The testing accuracy of my training model reaches 95.91% eventually. The table is below in figure 5:

DATA(with 20 % reduction)	testing accuracy	DATA	testing accuracy
over average loss	**93.85%**	over average loss	**94.97%**
under average loss	**95.40%**	under average loss	**95.00%**
heuristic pattern reduction	**95.91%**	heuristic pattern reduction	**95.54%**
without any operation	**94.97%**	without any operation	**95.00%**
over average loss(ea)	**88.62%**		
under average loss(ea)	**88.74%**		
heuristic pattern reduction(ea)	**89.31%**		
without any operation(ea)	**91.31%**		

Figure 5

From figure 5 we know that with if the features of dataset are selected by the evolutionary algorithm, the testing accuracy will drop to under 92%. I think the reason is that the dataset is transformed by digit pictures, all the features contribute to the training and testing. If some coordinates are removed, the picture is not complete, and the integrity is destructed. Thus, testing accuracy declines.

To compare over average loss, under average loss and without any operation, the testing accuracy is the lowest in over average loss and highest in under average loss. Thus, the spilt to two datasets illustrate that the loss of data impact negatively the training model.

To compare the data with 20% reduction and without 20% reduction, obviously the highest figure appears in heuristic pattern reduction in the data with 20% reduction within the figure 95.91%. If we reduce the data, we can prevent the training model from some data with high loss. It proves the same conclusion of Gedeon and Bowden (1992) that both reducing dataset and cleaning dataset will contribute to the improvement of training together. The image of final result is shown in figure 6 and the picture of error distribution (loss distribution) of the whole training set after heuristic pattern reduction is shown in figure 7.

```
In epoch 1100 ,data changes from 4775 to 3668, delete 1107

The training model of dataset is not overfitting

deleteed 4775 3668 1107
For heuristic pattern reduction, Testing Accuracy: 95.91 %
```

Figure 6

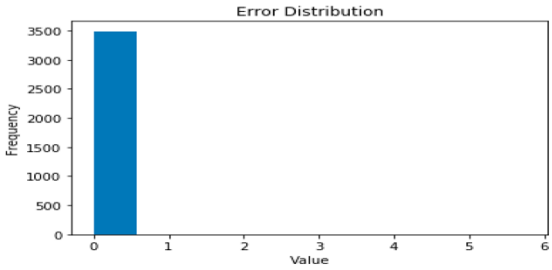


Figure 7

Another training model had the table below in figure 8 (Alimoglu and Alpaydin, 1996):

METHOD	WR-DEP	WR-INDEP	FREE PARS	EPOCHS
Dyn. 5-NN	99.73, 0.00	97.91, 0.00	3748i	1, 0.00
Dyn. MLP, h=16	98.26, 0.31	95.26, 0.37	280f	15.20, 3.29
Dyn. Rec., h=16	97.93, 0.36	94.23, 0.71	474f	35.30, 4.79
Sta. 5-NN	96.26, 0.00	93.34, 0.00	3748f	1, 0.00
Sta. MLP., h=10	93.54, 0.41	92.25, 0.31	760f	17.40, 3.63

Figure 8

Between figure 5 and figure 8, it is indicated that the comparison between static handwriting model among independent writers and my model illustrates that my testing accuracy is higher than Alimoglu and Alpaydin, however, comparison between dynamic handwriting model and my model illustrates that my testing accuracy is lower than Alimoglu and Alpaydin, Additionally, the writers' dependency is unknown, but the all the figures are over 92%. For my model, if we use all the features, the testing accuracy is all over 93%. Thus, I think that my neural network model has the similar effect on the same dataset compared to the result of Alimoglu and Alpaydin in 1996.

4 Conclusion and Future Work

In conclusion, the pen-based handwriting recognition is an inspiring task to deal with. By improving the method, reducing and cleaning the dataset, the neural network model fits the dataset and the testing accuracy reaches approximately 95.9%. However, a dataset with larger capacity can replace the current dataset because the effect of improving method is not significant though solving a difficult work. Furthermore, the efficiency of cleaning dataset---heuristic pattern reduction should be checked in another dataset. Moreover, if the dataset includes English letters, the training method should be improved as there are more kinds of outputs. The selection of different datasets may result in different rate of testing accuracy with the same method. In the future, I aim to choose another dataset to build a deep learning model like CNN or RNN and finish pruning.

References

Alimoglu, F., Alpaydin, E. "Methods of Combining Multiple Classifiers Based on Different Representations for Pen-based Handwriting Recognition," Proceedings of the Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium (TAINN 96), Istanbul (1996)

Fujita, O, "Statistical estimation of the number of hidden units for feedforward neural networks," Neural Networks, Atsugi (1998)

Gedeon, T.D., & Bowden, T.G. (1992). Heuristic pattern reduction. In International Joint Conference on Neural Networks,Vol. 2, pp. 449-453 (1992)

Gedeon, T.D., & Harris, D. Network reduction techniques. In Proceedings International Conference on Neural Networks Methodologies and Applications Vol. 1, pp. 119-126 (1991)

Gedeon, T.D., Wong, P.M., & Harris, D. Balancing bias and variance: Network topology and pattern set reduction techniques. In International Workshop on Artificial Neural Networks (pp. 551-558). Springer, Berlin, Heidelberg (1995)

Milne, L.K., Gedeon, T.D. and Skidmore, A.K. "Classifying Dry Sclerophyll Forest from Augmented Satellite Data: Comparing Neural Network, Decision Tree & Maximum Likelihood," Proceedings Australian Conference on Neural Networks, pp. 160-163, Sydney (1995)

Slade, P., & Gedeon, T. D. (1993, June). Bimodal distribution removal. In International Workshop on Artificial Neural Networks (pp. 249-254). Springer, Berlin, Heidelberg.