Combining Bimodal Distribution Removal and Neural Networks for Solving Classification Problems

Linhan Yang

Research School of Computer Science, Australian National University
u6014930@anu.edu.au

Abstract. What proposed in this paper is a combination of Bimodal Distribution Removal (BDR) and Neural Networks for solving classification problems. This paper uses Statlog (German Credit Data) data set from the UCI website to classify people described by a set of attributes as good or bad credit risks. Compared with results around 1990s using distance-based classification methods which got accuracy about 73%, this result is better and has accuracy of average 76.7%. As an extension, BDR is applied on MNIST image dataset with Convolutional Neural Network and gets the average test accuracy 97.45%.

Keywords: Artificial Neural Network, Convolutional Neural Network, Bimodal Distribution Removal, Deep Learning, Classification, Image Recognition, Machine Learning, Data Mining

1 Introduction

Classification is one of the most important machine learning tasks and the most common real world problem. And the Neural Network is one of the most popular and widely used techniques for solving classification problems. The idea is basically divided the whole dataset into training set and testing set, and use training set to train model by pairing the input with the expected output, then use testing set to estimate how well the model is trained. However, training sets are usually noisy in the context of real world. Slade and Gedeon (1993) indicates that most data cleaning methods perform well on artificially noise data but less well on real world data and proposes that applying Bimodal Distribution Removal method (BDR) can clean real world noise well and also avoid overfitting. In order to display with the most outstanding influence of BDR, Statlog (German Credit Data) data set (Dheeru et al. 2017) is chosen. This real world data set contains data used to evaluate credit applications in Germany and is noisy. A version with continuous attributes is used, in which contains 1000 instances and 24 features and has no missing values. The BDR is implemented with a customized Neural Network. The Neural Network has three layers, and is fed with 24 dimensions inputs corresponding with each feature and predicts outputs with two dimensions. Which dimension's value is bigger indicates which credit state it predicts. When the model completes, new customer's credit state can be predicted, which provides good sights for businesses to make appropriate decisions before starting a transaction. Additionally, the MNIST dataset (LeCun et al. 1998) is also choose to judge the effectiveness and scalability of BDR. It is a famous dataset of handwritten digits that range from 0 to 9. It contains 60000 images for training and 10000 images for testing. Compared with the German credit one, it is much larger and has image data. Deep learning architecture of Convolutional Neural Network (CNN) is built for learning. Ciresan (2011) indicates that deep neural models imitate the nature of human brain and are the most promising architecture for image recognition. The task is for CNN to recognize those handwritten digits and classify new images.

Data pre-processing techniques such as normalization are implemented. Cross-entropy is used for computing errors and Stochastic Gradient Descent is utilized for minimizing errors. The visualization of training such as plot of changing of variance and histogram of errors helps to achieve better BDR performance. The performance of the network is measured by a confusion matrix.

2 Method

2.1 Data Preparation

The first step is loading and preprocessing data. This paper first uses Statlog (German Credit Data) data set from the UCI website. The version with continuous attributes is used, where categorical variables in the original dataset is already interpreted to numeric values with the support of domain knowledge. Having loaded dataset, unsurprisingly values for different features are in different scales, which indicates the implementation of data normalization. This ensures standardised feature values and thus improve the ability to learn. This project simply transforms data to have zero mean and unit variance. For the ith sample, the approach of normalization can be written as:

$$z_i = \frac{x_i - \bar{\delta}}{\sigma},$$

where $\boldsymbol{\delta}$ is the sample mean and $\boldsymbol{\sigma}$ is the sample standard deviation. Then normalized data is split into two classes corresponding with two output neurons for the Neural Network to perform classification, with zero representing

customer in good credit and one representing customer in bad credit. Next, the whole dataset is randomly split into 80% as training set and 20% as testing set.

Moreover, for the MNIST dataset normalization technique is also utilized, while it normalizes to the range 0 to 1 by simply dividing by 255 each pixel. What worth mentioning is that most images are full-color image with all 3 RGB channels. So it is necessary to reduce depth to 1 and explicitly declare it by reshaping images to grayscale and to the same size. What's more, it is problematic that target values of MNIST are single numbers. Therefore, each of them is preprocessed into 10 distinct class labels. This project takes 50% of MNIST training data and keep the testing data. This is because the large dataset and sufficient batch learning enable high accuracy to be achieved after just few epochs. In order to for present the significant effectiveness of BDR during epochs, subset of training data are taken and this paper focuses on with less data how accurate can the model achieve with the assistant of BDR.

2.2 Model Training

When data is ready, the next step is feeding them into Neural Network to train model. For the German Credit dataset, this project builds a Neural Network with two hidden layers. The setting of Neural Network utilizes the rules of thumb. According to Karsoliya (2012), the number of hidden neurons should be in the range between the size of the input layer and the size of the output layer, and the number of hidden neurons should be 2/3 of the input layer size pluzing the size of the output layer. Therefore, it has 24 input neurons corresponding to each different features, 18 hidden neurons in the first hidden layer, 15 hidden neurons in the second hidden layer, 2 output neurons representing good or bad credit and uses relu as activation function. To train the Neural Network, in other words, to optimize its parameters including weights and bias, is to minimize the error. Basically, error is computed at each epoch and passed backwards. Each sample of training weights and biases are nudged in terms of the most rapid decrease of error. This project applies Stochastic Gradient Descent (SGD) with the learning rate equals 0.12 to do the optimization. The above training work is continuously proceeded in 1000 epochs. Specifically, propagation of inputs is going through the Neuron Network and outputs are generated using the randomly initialized parameters. Then errors are computed using cross-entropy. It gives two distributions over discrete variable x, where *q(x)* is the estimate for true distribution *p(x)* is given by:

$$H(p, q) = -\sum_{\forall x} p(x) log(q(x))$$

After that, propagation of output activation is going back through the network using the training target in order to update weights and biases. So gradient of error function is computed and SGD is used for finding the most effective fixtation. For each training pattern, the Neural Network learns and grows. And when finialiasing learning all patterns, next epoch of learning starts, which ensures enough information of training set is learned by the Neural Network.

In terms of the MNIST, CNN is implemented for image recognition. Apart from input and output layers, it has a combination of convolution layers, max-pooling layers, fully connected layers. The reason why no dropout layers is because BDR provides a reasonable point for stop training and avoid overfitting. More specifically, the input layer has 20 filters with the size of 5 by 5 and uses ReLU as activation function. Each filter performs a different convolution on the input to the layer and as a result a feature map is generated. Then the output of this layer is feeding into the second convolution layer. This layer has 50 filters as Karpathy (2016) points out that initial layers represents generic features while the deeper layers more complex features and so that increment of the number of filters in each layer can improve learning. After that, max-pooling layer reduces the parameters in the model by sliding a 2 by 2 pooling filter across the previous layer and taking the max among 4 values. The next layers are two dense layers. The first one has an output size of 128 while that of the second one using Softmax equals 10, which corresponds to all classes of digits. It is noticeable that the weights from the convolution layers are flattened before passing them to the fully connected dense layer. The model is then compiled using same optimizer SGD with the learning rate equals 0.01 and loss function as the previous one. During training, each batch is equally distributed by choosing batch size to 30.

2.3 BDR

As discussed earlier, BDR method is robust for removing outliers in dataset and provides a natural point to halt training. According to Slade and Gedeon (1993), the variance of errors in training set is quite large very early in training. And with the learning process going on, this variance drops sharply. When the variance is relatively low (below a threshold - T1), BDR begins. Certain patterns are removed from the training set until the variance decreases to a lower value (another smaller threshold - T2). Briefly, the method can be described as follows:

1. Variance of errors of all patterns in training set is calculated every 50 epoches. If variance of errors in training set is smaller than T1 but larger than T2, start BDR method. And if variance is smaller than T2, then stop training.
2. Compute mean error of all patterns in training set $\bar{\delta}_{ts}$, select patterns with error larger than $\bar{\delta}_{ts}$, which forms a sub-training set.
3. Calculate the mean error of all patterns in the sub-training set $\bar{\delta}_{ss}$, and calculate standard deviation of all patterns in the sub-training set σ_{ss}.
4. Patterns with errors satisfying the condition will be removed from training set.
$$error \geq \bar{\delta}_{ss} + \alpha\sigma_{ss} \qquad (0 < \alpha < 1)$$
5. Back to step 1.

Figure 1: Description of Bimodal Distribution Removal

This allows removing outliers and training to proceed together, and provides an appropriate point to halt training. Therefore, Neural Network can learn more efficiently. In the implementation, outlier detection is executed every 50 epochs, and samples that have high errors are dropped from training dataframe. In order to accurately select patterns with high errors, a mask is used. Its size is same as the training set in one specific epoch. Firstly, it stores indices of those patterns with errors larger than the mean error. And then indices of patterns that fail to satisfy the condition are removed. The removal of training set are performed based on this mask. In the next round of removal, a new mask is created.

However, a key problem is to determine the values of two thresholds and the coefficient of standard deviation $\boldsymbol{\alpha}$. These values can be affected by the chosen dataset, the modeling of Neural Network and also the chosen method for computing error. In this project with a particular German Credit dataset, experiments are conducted to determine appropriate values. In terms of T1 which indicates the start of BDR, a plot of histogram of errors in training set is printed for analyzing every 50 epochs. Each time when the histogram is plotted, the variance of errors in that epoch is also known. At the beginning, errors are normally distributed as predictions are randomly made without training. Then with learning proceeds, two peaks of error frequency forms and patterns with high errors become obvious. Based on this evidence on this particular dataset, variance is selected as T1 where outliers are separated from other patterns. This provides a reliable range of ideal T1 values. In terms of T2 which indicates to stop training, this value can be typically set to 0.01 as after enought removal the variance will decrease sharply. And for $\boldsymbol{\alpha}$, it will affect removal size. On the one hand, if $\boldsymbol{\alpha}$ is large, less patterns in the training set will satisfy the condition, which may result in less effective outlier removal. But those patterns are more likely to be outliers because their errors are relatively high. And on the other hand, if $\boldsymbol{\alpha}$ is small, training set may be over-reduced, which will get a loss of information in interesting patterns. Therefore, $\boldsymbol{\alpha}$ should be an appropriate value depending on the number of the instances in training set and effectiveness of removal. This project tests with different values of $\boldsymbol{\alpha}$, evaluates performance with the size of reduced training set as well as the averaged accuracy of prediction and eventually determines the value of $\boldsymbol{\alpha}$.

2.4 Evaluation

To determine and evaluate the performance of trained Neural Network, measure of confusion matrix is utilized. In a confusion matrix, counts of numbers of correct or incorrect predictions are displayed according to actual class and predicted class. Calculating a confusion matrix provides a better idea of what is accurate and what types of errors it is making in the built model. This project utilizes two confusion matrices to evaluate the result, one for training set and another for testing set.

3 Results and Discussion

3.1 Visualization of Data

As discussed above, German Credit dataset is split randomly into training set and testing set, respectively 80% and 20%. In this dataset there are 1000 instances, the splitting result is illustrated below:

```
-1.2539  0.2568 -0.5032  ...   -0.4997 -1.3042  0.0000
 1.1315  0.2568 -0.5032  ...   -0.4997  0.7660  0.0000
 1.1315 -0.2407  1.3433  ...   -0.4997  0.7660  0.0000
           ...             ⋱              ...
-1.2539 -0.7383 -0.5032  ...   -0.4997  0.7660  0.0000
-1.2539 -0.2407  1.3433  ...   -0.4997  0.7660  1.0000
-1.2539 -0.4895 -0.5032  ...   -0.4997  0.7660  0.0000
[torch.FloatTensor of size 822x25]
```

Figure 2.1: Overview of training data

```
 0.3363 -0.2407 -0.5032  ...   -0.4997  0.7660  1.0000
-0.4588  1.9983  1.3433  ...   -0.4997  0.7660  0.0000
-1.2539 -1.2359  1.3433  ...   -0.4997 -1.3042  0.0000
           ...             ⋱              ...
-0.4588 -0.2407 -0.5032  ...   -0.4997  0.7660  0.0000
-0.4588 -0.9871 -1.4264  ...   -0.4997 -1.3042  1.0000
 1.1315  0.2568 -0.5032  ...   -0.4997  0.7660  0.0000
[torch.FloatTensor of size 178x25]
```

Figure 2.2: Overview of testing data

The number of rows represents number of patterns sampling from the original dataset, and columns include 24 features and one target column.

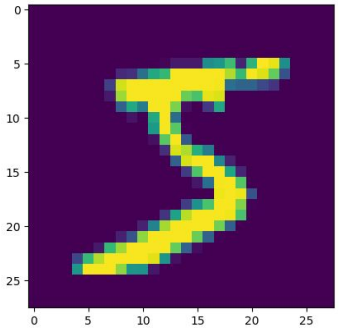

Figure 2.3: First image sample in MNIST

For MNIST dataset, first sample is plotted for visualization (See Figure 2.3). Basically it can be recognized as number 5. And as illustrated before, the image is 28 pixels by 28 pixels.

3.2 Effectiveness and scalability of BDR

This section will compare simple Neural Network learning and combination of BDR and Neural Network on the same German Credit dataset, and display a visualization of averaged analysis results to evaluate the effectiveness of BDR. The experiments records results of 10 runs on each method and compute an averaged result. Note that two methods utilize same training dataset and testing dataset, same Neural Network module and same measure for evaluation. And also note that simple Neural Network learning trains with the same epochs as the one with BDR because if use BDR training will stop earlier. The results are displayed in the table 1 below:

Run	Train_Size	Test_Size	Total_epochs	Test_Accurancy	BDR_Test_Accurany	Averaged_Accurancy
1	799	201	1051	70.59%	74.63%	Averaged accuracy of simple Neural Network is 71.9%. Averaged accuracy of Neural Network combining BDR is 76.7%.
2	805	195	951	69.96%	73.89%	
3	799	201	901	75.01%	79.03%	
4	804	196	751	73.05%	78.66%	
5	774	226	801	72.32%	74.51%	
6	817	183	1251	68.87%	71.87%	
7	787	213	851	72.45%	80.19%	
8	790	210	851	75.25%	79.51%	
9	811	189	951	70.52%	76.97%	
10	803	197	701	71.77%	77.49%	

Table 1: Results of 10 runs on two methods

The results using BDR are slightly better than using only simple Neural Network. We can conclude that BDR is effective for improving prediction accuracy of Neural Network, which means outliers are effectively removed from the training set.

While effectiveness of BDR can be evaluated by the performance of implementing it on simple Neural Network, the scalability of BDR can be evaluated with a image dataset using CNN. The excellent result of MNIST proves this (See Figure 3 below). This sample run achieves testing accuracy 97.49%. All numbers in matrix look nice except that model classifies several digits to 9 as the last column has relatively larger values of misclassification. Although training data is reduced, the result after BDR is still outstanding and it enables model to use less data to learn the best results in shorter running time.

```
Testing Accuracy: 97.49 %
Confusion matrix for testing:
[[ 965    0    0    1    0    4    5    1    3    1]
 [   0 1128    0    1    0    0    2    0    4    0]
 [   1    3 1005    1    2    2    1    5   10    2]
 [   0    0    4  982    0    7    0    5    7    5]
 [   1    0    3    0  933    0   11    4    4   26]
 [   3    0    1    8    0  869    7    0    2    2]
 [   3    2    2    1    1    2  945    0    2    0]
 [   0    6    9    0    0    0    0 1000    2   11]
 [   2    0    2    8    1    5    9    3  940    4]
 [   1    5    0    2    4    4    0    8    3  982]]
```

Figure 3: Evaluation of CNN with BDR on MNIST

3.3 Exploration of BDR

In order to explore deeper into BDR and visualize how it works, this part focus on one sample run using German Credit dataset. Figure 3.1 below shows the evaluation of the sample run using confusion matrix. Obviously, left bottom and top

right corner are all 0s in the confusion matrix for training. This means that the accuracy for training is already 100%. And for the confusion matrix for testing, the accuracy is still impressing, which achieves 80.14%. Within this high-achieving run, Figure 3.2 shows how variance changes during training. The horizontal axis is scaled epochs (reduce its value to 10%), and the vertical axis is variance of errors in the training set. As illustrated in Figure 3.2 below, when training just starts, the variance is very small. This is because initially the Neural Network has no patterns to learn and predicted values are calculated using random weights and biases, which makes errors normally distributed. Figure 3.3 is what frequencies of errors look like before training. The horizontal axis is error while the vertical axis is the frequency. This occurs the same for Figure 3.4. According to Figure 3.3, though each error value has a certain frequency, the variance is small. And at the early epoch (before around epoch 200), the variance increase sharply because the Neural Network keeps learning and errors are gradually close to two peaks distribution, which is illustrated by Figure 3.4. But note that in the process of forming such distribution, the variance will keep increasing. When the prototype of two peaks has formed (corresponds to point A in Figure 3.2), the variance starts to drop. The reason is as the Neuron Network keeps learning, overall errors become smaller and lower errors of patterns take up more. With the training goes on, bimodal is approximately formed at point B, whose error distribution is visualized in Figure 3.4. At this stage, obviously outliers are separated from the training set and is easy to remove. Therefore, the first removal of BDR starts at point B. Effectively, the variance suddenly drops as outlier patterns are removed. Point C is the second removal. But this time the decrease of variance is not dramatic so it implies this time not much patterns are removed. The procedure of BDR will end if variance is just below T2. Interestingly, BDR with large α performs more stable while a small α may produce an excellent result and finish earlier. Also it is noticeable that for the MNIST dataset, as training set is large and has sufficient information for model to learn, usually after a few epochs high training accuracy can be achieved. So not surprisingly two error peaks are formed in the earlier epochs and BDR then starts.

```
Confusion matrix for training:

 453    0
   0  122
[torch.FloatTensor of size 2x2]

Testing Accuracy: 80.41 %

Confusion matrix for testing:

 121   15
  23   35
[torch.FloatTensor of size 2x2]
```

Figure 3.1: Evaluation of sample run

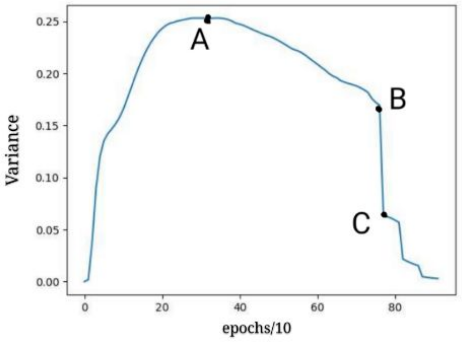

Figure 3.2: How variance changes during training

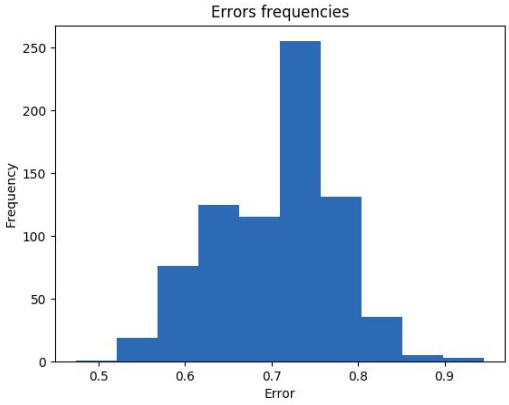

Figure 3.3: Error frequencies at epoch 0

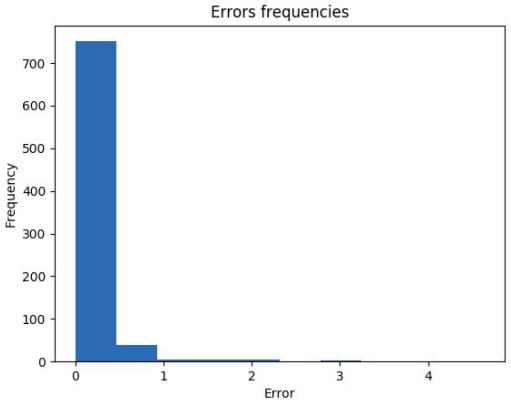

Figure 3.4: Error frequencies at epoch 351

3.4 Comparison with a Distance-based Class Classification Methods

Oya et al. (1990) conducts a research using a distance-based classification methods to predict German Credit. The result is shown in Figure 4 on the left. The fifth row describes the accuracy of predicting German credit using different methods. Among these result, the best one is the CCM one, which is 73.4 ± 0.8%. The result using BDR in Neural Network is average 76.7%, which is better than the distance-based classification methods. This is because the German Credit data is a real world dataset and not surprisingly they are noisy. This biases will have an influence on the

Data Set	CCM	k-NN	Parzen	Best Known
Cancer	97.0 ± 0.2	97.3 ± 0.3	96.4 ± 0.3	96.2 [23]
Iris	95.0 ± 1.0	94.7 ± 0.8	91.8 ± 1.0	98.0 [36]
Housing	84.4 ± 0.6	83.4 ± 1.1	83.1 ± 1.1	83.2 [23]
Diabetes	74.1 ± 0.6	74.6 ± 0.8	73.4 ± 0.9	76.0 [33] 78.0 [6]
Credit Card (A)	86.4 ± 0.6	86.5 ± 0.6	86.5 ± 0.6	85.5 [8] 87.0 [6]
German Credit	73.4 ± 0.8	73.1 ± 0.7	71.7 ± 0.7	
Labor	88.4 ± 4.1	83.5 ± 4.6	69.9 ± 5.6	90.0 [5]
Soybean	88.9 ± 0.5	89.3 ± 0.6	86.7 ± 0.6	87.0 [22]
Voting/All	94.1 ± 0.6	92.2 ± 0.6	89.3 ± 0.9	95.6 [19]
Voting/Red	89.1 ± 0.7	88.9 ± 1.0	88.2 ± 0.7	89.4 [19]

Figure 4: Distance-based classifiers

distance-based classifiers and eventually affect the performance of the model. However, this paper utilizes BDR to remove outliers and feed those clean data into the Neural Network for learning, thus better performances and accuracies are gained.

4 Conclusion and Future Work

In conclusion, this paper studies the classification problem of German credit data by a combination of BDR and Artificial Neural Networks as well as a exploration of how the BDR method works. It obtains a better prediction ability than the results obtained by Oya et al. (1990) and gets the accuracy of average 76.7%. And as an extension, combination of BDR and CNN is experimented on a larger image dataset MNIST. And average accuracy 97.35% is achieved with smaller training set and shorter running time.

The results still need to be improved in the future. For dataset with less instances, removing reasonable number of patterns from dataset rather than reducing dataset too much remains a question that worth discussing. And for dataset with irrelevant features it is hard for BDR to remove as it is observation-oriented. So combining BDR of outlier removal as well as feature removal is a implementable thought. And the work of how to effectively get more scattered split of interesting patterns and outliers need to be continued. Moreover, in terms of determine two thresholds of variance to start BDR and terminate training, more intelligent methods should be developed as dataset and modeling of Neural Network vary from different problems.

References

Ciresan, D. C., Meier, U., Masci, J., Maria Gambardella, L., & Schmidhuber, J. (2011, July). Flexible, high performance convolutional neural networks for image classification. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence* (Vol. 22, No. 1, p. 1237).

Dua, D. and Karra Taniskidou, E. (2017). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Ekin, O., Hammer, P. L., Kogan, A., & Winter, P. (1999). Distance-based classification methods.Infor, 7(3), 337-352. Retrieved from https://search-proquest-com.virtual.anu.edu.au/docview/228438303?accountid=8330.

Karpathy, A. (2016). Cs231n convolutional neural networks for visual recognition. *Neural networks, 1.*

Saurabh Karsoliya, “Approximating Number of Hidden layer neurons in Multiple Hidden Layer BPNN Architecture,” *International Journal of Engineering Trends and Technology- Volume 3 Issue 6- 2012.*

Slade, P & Gedeon, TD “Bimodal Distribution Removal,” *Proc. IWANN Int. Conf. on Neural Networks,* Barcelona, 1993.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, 86(11):2278-2324, November 1998.