Handwritten Digit Classification using PCA, Logistic Regression and Deep Neural Networks

Ziao Shi, Huanqing Wang, Xingyue Fang

December 2022

1 Overview

The problem our project is trying to address is to correctly identify the handwritten digits in the MNIST dataset.

In this project, we will use this data set to compare performance of various classification techniques. The methods we will use are: Logistic Regression, Random Forest, Principal Component Analysis and Convolutional Neural Network.

We split the entire data set into three parts: a training set, a validation set and a test set. For each method, we use the training set and validation set to train model and perform model selections (in particular tuning hyperparameters). After that, we select the best model for each method and use the test set to compare performance of different models.

2 Background and Introduction

2.1 Background of the MNIST Data Set

Handwritten digit recognition is a problem with abundant applications. Its applications include postal mail sorting, bank check processing, etc.

The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The MNIST database contains 60,000 training images and 10,000 testing images.

The set of images in the MNIST database was created in 1998 as a combination of two of NIST's databases: Special Database 1 and Special Database 3. Special Database 1 and Special Database 3 consist of digits written by high school students and employees of the United States Census Bureau, respectively.

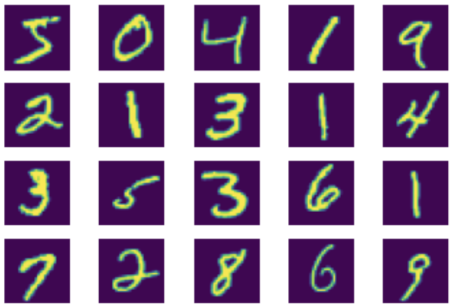

Figure 1: Preview of the First 25 Digits

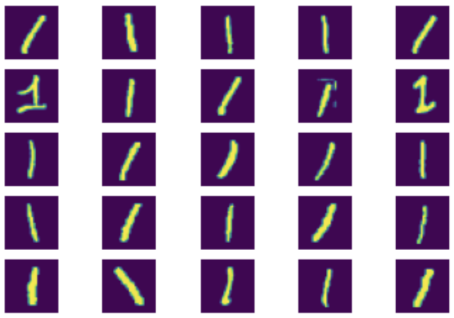

Figure 2: Preview of the First 25 Instances of Digit 1

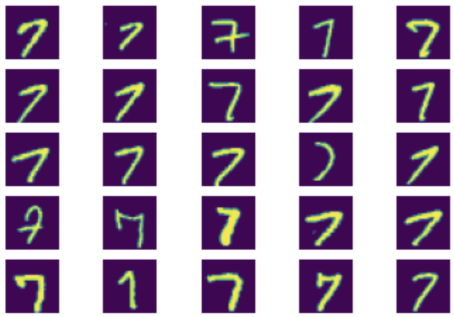

Figure 3: Preview of the First 25 Instances of Digit 7

The training inputs of the following algorithms are the vector of pixels of the digit images along with the true labels. We use cross validation to perform model selection and then apply the model to the testing data, which only contains the vector of pixels of the images.

2.2 Principal Component Analysis

Often, data with many features is really generated by a process with only a handful of parameters. Principal component analysis (PCA) is extensionally used for taking a high-dimensional data set and embedding it in a lower dimension. PCA defines the best d-dimensional subspace as the subspace with the larges variance in the data. The rationale is that, higher spread of data along a direction implies more information from data.

The images in MNIST were converted into the same 28x28 pixel format. This means each image has 784 dimensions to analysis. PCA will be used to reduce dimensions, meanwhile keeping a low mis-classification rate. For a 0-mean variable X, its population variance can be calculated as $\frac{1}{n}X^TX$. But in practice, if X is ill-conditioned, then X^TX will be even more ill-conditioned. Therefore, we will use SVD instead: if $X = U\Sigma V^T$, then $X^TX = V\Sigma^2V^T$. From the formula above, singular vectors in the matrix V are the eigenvectors of X^TX, which are call the principal components of X. If the singular values Σ_{ii} are in descending order, then the i^{th} singular vector in V is the i^{th} principal component.

On the other hand, PCA only transforms data without doing any predictions. We introduce two methods: random forest and logistic regression, to classify the hand-written digits along with PCA.

2.3 Random Forest and PCA

Since both of the input pixels and digits are discrete, the most starightforward method would be a decision tree. Namely, a tree is built by splitting the input data set into subsets—which constitute the successor children. The splitting is based on pixels at different image positions. This process is repeated on each derived subset in a recursive manner called recursive partitioning.

However, trees that are grown very deep tend to learn highly irregular patterns: this in general leads to over-fit. To overcome this issue, we use the random forest method. To be precise, a tree using random forest fits a number of decision trees on various sub-samples of the data set and uses averaging to improve the predictive accuracy and control over-fitting.

2.4 Logistics Regression and PCA

Multinomial logistic regression is a linear model for multi-class classification. The conditional probabilities of the outcome class $c \in C$ are modeled using the softmax function:

$$P(y = c|x, \beta_c, b_c) = \frac{\exp(\beta_c x + b_c)}{\Sigma_{d=1}^{C} \exp(\beta_d x + b_d)}$$

The parameters of a logistic regression are most commonly estimated by maximum-likelihood estimation (MLE). Again, the input of logistic regression will be the PCA-transformed pixel values.

2.5 Deep Neural Networks

Convolutional Neural Network (CNN) is a class of neural networks that is heavily used to analyze visual imagery. Here, we use a CNN to capture the structural information from the image. CNN makes use of the shared-weight architecture of the convolution kernels to construct feature maps for the image input. The main idea of CNN is to take advantage of the hierarchical pattern in image data to do pattern recognition, i.e. each pixel is closely related to its neighbors, and therefore the network can hierarchically grasp regional patterns and eventually provide a result with high accuracy.

3 Algorithm Performance

3.1 Random Forest and PCA

First of all, we select the number of trees used in random forest:

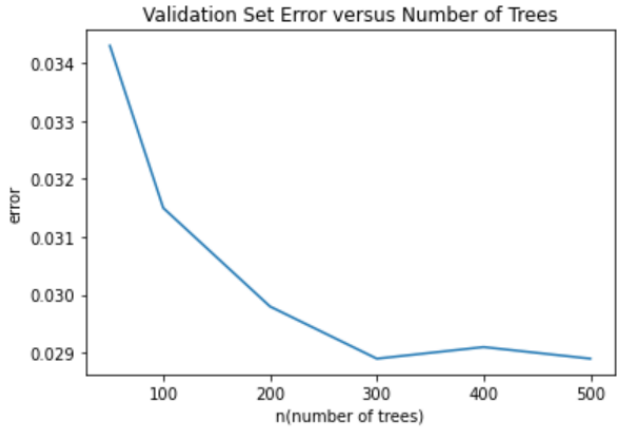

Figure 4: Validation Set Error versus Number of Trees

From the graph above, using 300 trees yields with the lowest error rate. Without using PCA, the error rate is 2.89% and the confusion matrix is

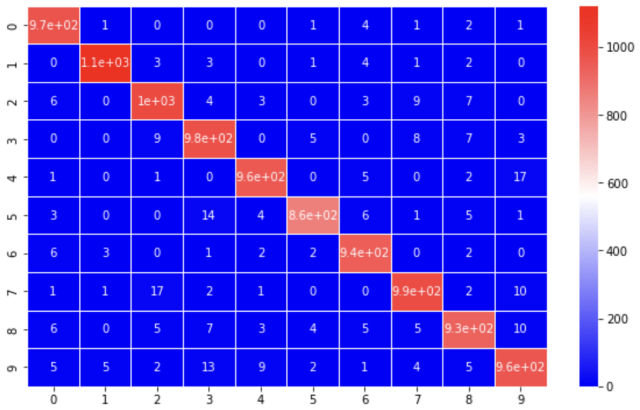

Figure 5: Confusion Matrix of Random Forest without PCA

Using 300 trees in random forest, we find the number of principal components that yields the lowest misclassification rate.

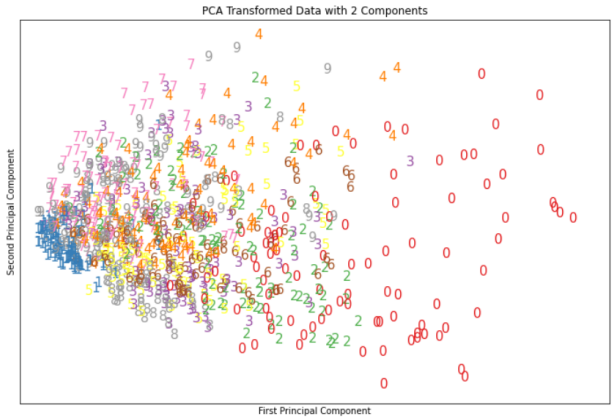

Figure 6: PCA Transformed Data with 2 Components

First, we visualized the pixel data transformed by PCA with 2 components as above. We can easily see that this transformation did not separate out different labels successfully because the number of components is not enough.

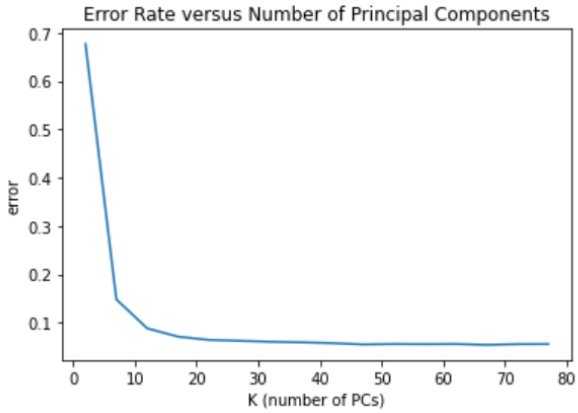

Figure 7: Error Rate versus Number of Principal Components

We chose the number of principal components to be 67, which is the argmin of the error rate. This model's performance on the validation set is as follow:

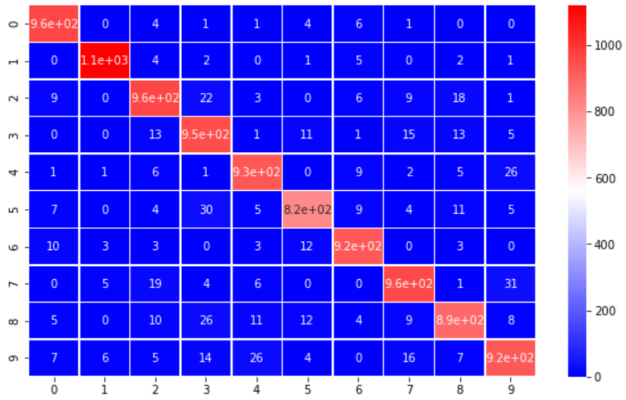

Figure 8: Confusion Matrix of Random Forest with PCA

Compared with data without PCA, misclassification rate increases from 2.89% to 5.55%, which is acceptable.

3.2 Logistic Regression

We use the 'sklearn.linear' model implementation of multiclass logistic regression. This implementation of logistic regression assumes a regularization term and $C = \frac{1}{\lambda}$. A small regularization term corresponds to very large C. We try a few C's with a wide range of orders of magnitude to pick the optimal C:

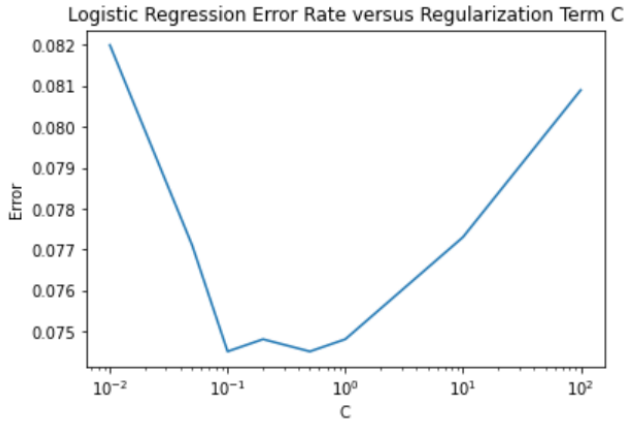

Figure 9: Logistic Regression Error Rate versus Regularization Term C

Here, we pick $C = 0.1$ to train the final logistic regression model on the training and validation set together. Then, we retrain with the training and validation set together. The Logistic Regression model has an error rate of 7.09% on the test set.

Also, we would determine how well PCA works in terms of classification by using the PC's computed to train a logistic regression classifier to predict the class label given those principal components.

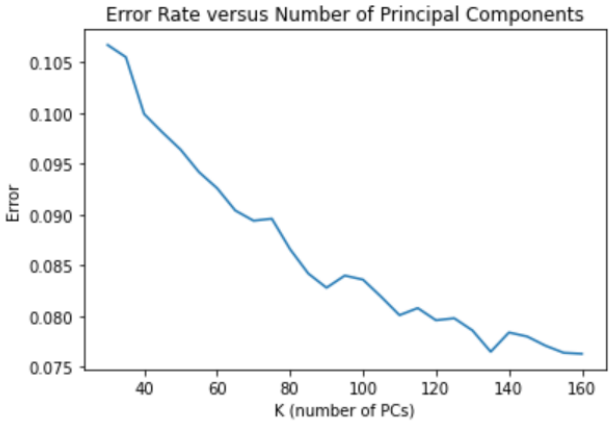

Figure 10: Error Rate versus Number of Principal Components

Here, we pick $K = 135, C = 0.1$ to train the final logistic regression model with PCA projection on the training and validation set together. The Logistic Regression with PCA model has an error rate of 12.07% on the test set, with is quite a big increase.

3.3 Neural Networks

The network contains two successive convolutional layers, two pooling layers that come after the convolutional layers and two dropout layers after the pooling layers. With ReLU activation function, Adam optimizer and cross entropy loss, we run some epochs to train the network on the training set and get the error rate on both training set and validation set.

Below is a graph of misclassification rate against the number of epochs (the number times that the learning algorithm works through the entire training dataset) for CNN:

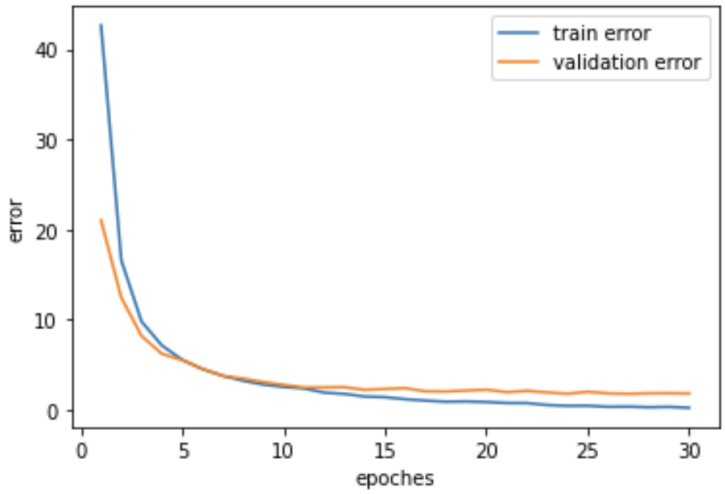

Figure 11: Misclassification rate versus number of epochs

We can see that as the number of epoch increases, the loss decreases. The final misclassification rates on the train set (0.17%) and validation set (1.76%) are fairly small, suggesting that CNN provides high accuracy for recognizing hand written digits. The model also performs quite well on the test set, with an error rate of 1.70%.

4 Method Comparison

Random forest method with PCA has misclassification rate of 4.85% on the test set, where we used 300 trees and 67 principal components. The running time is around 90 seconds.

Logistic regression method taking $C = 0.1$ without PCA has misclassification rate 7.09% and running time 61 sec. Logistic regression method taking $C = 0.1$ with 135 principal components has misclassification rate 12.07% and running time 48 sec. Reducing number of components indeed speeds up the running time, with the sacrifice of accuracy.

Convolutional Neural Network takes 11.44 sec to train the network, with an error rate of 1.84% on the test set. This method runs quite fast(as the network is not too deep) but still reaches the highest accuracy among the methods we use in the project.

5 Conclusions and Future Work

As shown in the results above, even a simple CNN provides very accurate result, this is also why CNN is now one of the most widely used image recognition method. PCA, Random Forest and Logistic Regression also provides descent result as two classic classification methods.

If we were given more time, we can certainly improve our results in the following way:

1. for Random Forest, we can spend more time on tuning the parameters to improve the accuracy.

2. for CNN, we can try to add more layers to the model and run more epochs to get an even higher accuracy rate.

3. We can try to ensemble different methods together to get a stronger classifier.