Handwritten Digit Classification using the MNIST Data Set [1]

Ming Wu wuming1@msu.edu
Zhen Zhang zhangz19@msu.edu

Abstract

In this report we train and test a set of classifiers for pattern analysis in solving handwritten digit recognition problems, using MNIST database. Direction features are extracted for dimensionality reduction. kth Nearest Neighbor, Gaussian Mixture models and Support Vector Machine turn out to be the top candidates for extracted features, where the lowest error rate 1.19% is achieved using 3-NN. Further potential improvement of the classifiers is proposed and discussed.

Keywords: Digit Recognition, MNIST, Linear Classifier, Support Vector Machine, Gaussian Mixture Model, kth Nearest Neighbor

1 Introduction

The problem of handwriting recognition is to interpret intelligible handwritten input automatically, which is of great interest in the pattern recognition research community because of its applicability to many fields towards more convenient input devices and more efficient data organization and processing. As one of the fundament problems in designing practical recognition systems, the recognition of handwritten digits is an active research field. Immediate applications of the digit recognition techniques include postal mail sorting, automatically address reading and mail routing, bank check processing, etc.

As a benchmark for testing classification algorithms, the MNIST dataset has been widely used to design novel handwritten digit recognition systems. There are a great amount of studies based on MNIST dataset reported in the literature, suggesting many different methods. A comprehensive review by LeCun et al.[3] compares the performance of previously proposed classifiers including linear and polynomial classifiers, Nearest Neighbor classifiers, and different neural networks. A best test error rate of about 0.7% is achieved by combining multiple neural network classifiers (Boosted LeNet 4).

One of the major challenges in the recognition of handwritten digits is the within class variance, because people do not always write the same digit in exactly the same way. Many

[1] Technique report of course project for CSE802: *Pattern Classification & Analysis*, MSU, Spring 2010, instructed by Prof. A.K.Jain.

feature extraction approaches have been proposed trying to characterize the shape invariance within a class to improve the discrimination ability[2]. Experiments have shown that by extracting direction features, local structure features or curvature features, the accuracy and efficiency of many classifiers could be improved significantly[2].

In this report we train and test a set of classifiers on the MNIST database for pattern analysis in solving the handwritten digit recognition problem. Direction features are extracted for dimension reduction. Classifiers based on generative models (Multivariate Gaussian, Mixture Gaussian) and discriminative models (SVM,NN) are applied to compare the performance in terms of both accuracy and speed. Potential improvements including combinations of the classifiers and rejecting options, are proposed and discussed.

2 Data Description

The MNIST database contains 60,000 digits ranging from 0 to 9 for training the digit recognition system, and another 10,000 digits as test data. Each digit is normalized and centered in a gray-level image with size 28×28, or with 784 pixel in total as the features. Some examples are shown in Figure 1.

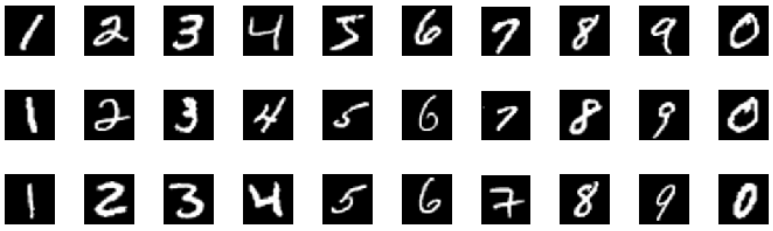

Figure 1: Examples of MNIST data set

3 Feature extraction

3.1 Principle component

Since the original dimension is quite large, the dimensionality reduction becomes necessary. First we extract the principle components from the original data. As shown in Figure 2, the first 50 principle components can interpret approximately 97% of total information (in terms of the total variance retained), which suffices to be representative and informative. We thus choose first 50 principle components as the extracted features.

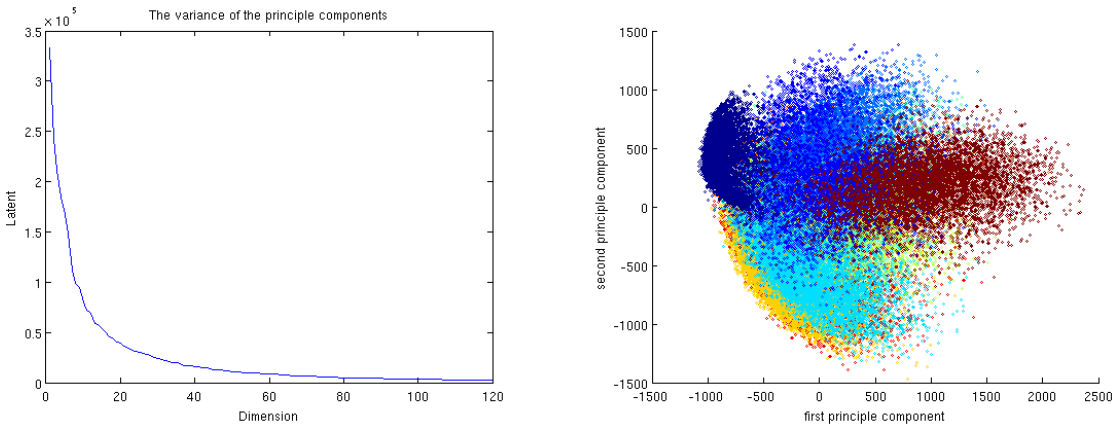

Figure 2: variance of Principle Components (*left*) and scatter plot on PC Coordinate(*right*)

3.2 Direction features

We also extract direction features as described in [6] trying to characterize the within class shape invariance by representing the image with local stroke directions. We compute the gradient components ($\mathrm{A} \rightarrow [gx^*\mathrm{A}, gy^*\mathrm{A}]$) using *Sobel operator*. Then the gradient vectors are decomposed into eight chaincode directions. In each of the resulting 8 (28×28) directional image, 25 measurements are made. The procedure is ilustrated in Figure 3. Finally we obtain the direction feature space with dimension $8 \times 25 = 200$.

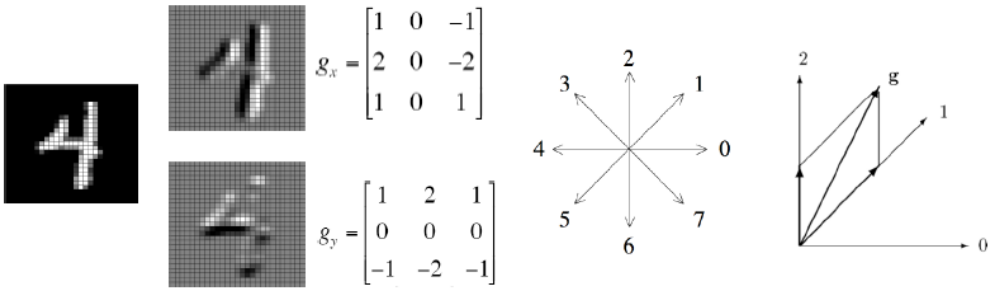

Figure 3: Ilustration of feature extraction

We use 1-nearest neighbor classifier on the three feature spaces (the original 784 pixels, the first 50 principle components, and the direction features). The results are summarized in Table 1. The direction features turn out to give the best performance. Thus we will choose direction feature as the main feature for all classifiers and report results based on it.

\Feature space	Original 784(28×28)	PC (first 50)	Direction features (200)
mean of training error	N/A	2.29%	1.31%
std. of training error	N/A	0.2%	0.11%
test error	3.1%	2.68%	1.46%

Table 1: Error rates for 1-NN using different feature space (training error is obtained by 10-cross validation)

4 Classifiers specification

4.1 kth Nearest Neighbor

As a nonparametrix approach, the kth Nearest Neighbor classifier uses all the training patterns as prototypes.

The classification accuracy is influenced by the number of nearest neighbor k. We thus try different k ($k = 1, 3, 5, 7, 9$) and obtain the test error rate for each classifier. The training error rate is obtained by the 10-fold cross-validation. As shown in Figure 4, the highest accuracy is mostly given by $k = 3$. We hence use 3-NN classifier for the following context.

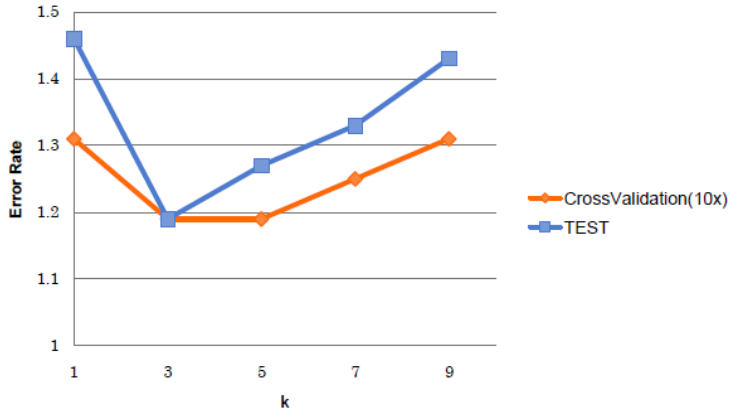

Figure 4: Error rate of k-NN classifier versus different choices of k

4.2 Generative Models

- *Linear Discriminant Function.* We assume a Multivariate Gaussian distribution with equal covariance structure for each class. With a pooled estimate of the covariance, the linear discriminant function has following form [5]:

$$g_i(x) = -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}+_i) + \log P(\omega_i) \tag{1}$$

- *Quadratic Discriminant Function.* Same Multivariate Gaussian assumption but with unequal covariante structure estimated by each class. The quadratic discriminant function can be thus written as:

$$g_i(x) = -\frac{1}{2}\boldsymbol{x}^t\boldsymbol{\Sigma}_i^{-1}\boldsymbol{x} + (\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i)^t\boldsymbol{x} + \omega_{i0} \tag{2}$$

- *Gaussian Mixture Model.* We assume a mixture of Gaussian distributions for each class and we estimate associated parameters using EM algorithm. The number of components in each class is determined by the AIC(Akaike Information Criterion). The class label is assigned by comparing the posterior density.

4.3 Support Vector Machine

We use `libsvm` [1] for training and testing the Support Vector Machine (SVM) classifiers. Based on previous reports and papers [3], our choices of kernel with associated parameters are specified below:

- *Linear Kernal* $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i \cdot \boldsymbol{y}_i$. Reasonable training time and a good performance are achieved for this kernel function and will be reported in the following section.

- *Radial Basis Function Kernel* $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma\|\boldsymbol{x}_i - \boldsymbol{y}_i\|^2), \gamma > 0$. By default, `libsvm` selects $\gamma = 1/\boldsymbol{d}$, where $\boldsymbol{d}$ is the number of features and $\boldsymbol{d} = 200$ when we use the extracted direction features. However we found for this case ($\gamma = 0.005$) the error rate is high (8.05%) and it is time-consuming for training. We therefore change $\gamma = 0.5$ to specify a smaller window size and the performance turns out to be improved.

- *Polynomial Kernel* $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i \cdot \boldsymbol{y}_i + 1)^d$. In contrast to the former report, we have a less efficient performance using this kernel function with a high training cost and low error rate for the extracted features.

4.4 Other classifiers

In addition to the classifers aforementioned, we also tried some other classifiers. However, issues for the implementation or improvement of efficiency arise when we train the multiple classifiers:

- *Neural Network.* It has been highly evaluated by former articles [3] and the Convolutional Neural Network [4] is reported to have the best performance comparing to a series of classifiers. For normal network, the implemention however has a high requirements for memory and training time as commonly encountered. For 50-Principle-Component

we try a 50-12-10 neural network and achieve an test error rate around 9%. However for the direction features with dimension 200, the number of nodes for hidden layer is set to 28 such that the total number of weights becomes roughly 10% of sample size (60,000) to achieve a well-performed system [5]. The 200-28-10 network certainly requires high memory and expensive computations.

- *Logistic Regression models.* We treat each label as response variable and establish a multinomial logit model using the features as covariates. After the model fitted, we assign the labels with highest membership probability. It has remarkably good performance for classical data set like *iris* data. However again due to the high dimensionality and large sample size, it is highly consuming to compute the inverse of a huge matrix when fitting the model with Iteratively Reweighted Least Squares (IRLS) algorithm.

5 Classification results

After an overall estimation and comparison of the performance of the above-mentioned classfier candidates, we report here the results of the following six classifiers working on the extracted direction feature space: LDA (using linear discriminant analysis), QDA (using quadratic discriminant analysis), GMM (Gaussian Mixture Models), SVML (Support Vector Machine with linear function kernel), SVMR (SVM with radial basis function kernel) and k-NN(with $k = 3$). For each classifier we implement the 10-fold cross-validation to obtain the training error rate, and then feed in the test samples to obtain the test error rate. The CPU time for training and testing individual classifier is recorded as a measure of cost through the whole procedure. A comprehensive summary of the six classifers are shown in Table 2 and Figure 5.

	LDA	QDA	GMM	SVM(Linear)	SVM(RBF($\gamma = 0.5$))	KNN($k = 3$)
Training error rate	3.38%	3.77%	1.62%	2.13%	2.05%	1.19%
Test error rate	2.92%	3.59%	1.37%	1.89%	1.73%	1.19%
total CPU time(s)	47.5	77	640.2	430	553.6	3130.6

Table 2: Summary of performances for the selected six classifiers. The total cpu time include both training and testing. If only compare the test time, KNN costs about 0.5 second per test sample (in practical this speed may be improved by spacially organized data structure and specific searching algorithms), while the other 5 classifiers costs less than 0.002 second per test sample.

From the results we conclude that reasonable good performances are achieved for all six classifiers when direction features are used. Furthermore, 3-NN has the lowest error rate of

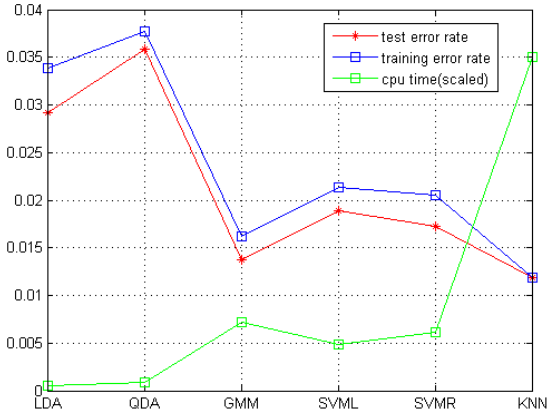

Figure 5: Summary of performances for sixed selected classifiers.

1.19% at the expense of a longer CPU time (3130.6 seconds). On the contrary, linear and Quadratic classifers require shortest time for training and test, however result in a higher error rate (more than 3%). A reasonably sound tradeoff can be achieved by SVM and GMM; the latter has a low error rate (1.37%) that is only second to 3-NN, whereas maintaining a much lower cost comparing to 3-NN.

Predicted\True	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	99.82%	0.29%		0.20%		0.31%	0.97%		0.50%	
2	0.09%	98.64%	0.20%				0.68%	0.10%		0.10%
3			98.91%		0.90%		0.19%	0.31%	0.79%	0.10%
4		0.10%		98.78%				0.31%	0.50%	
5					98.88%	0.31%		0.21%	0.10%	
6	0.09%	0.10%		0.10%	0.11%	99.06%				0.20%
7		0.68%	0.50%				97.37%	0.10%	0.50%	
8		0.10%	0.20%			0.10%	0.10%	98.36%	0.50%	0.20%
9			0.20%	0.92%			0.68%	0.31%	97.03%	
10		0.10%			0.11%	0.21%		0.31%	0.10%	99.39%

Table 3: Confusion matrix for Gaussian Mixture Model classifier

It is also worth of an investigation of the error distribution for each classifier in both the training and testing process. As shown in Figure 7 (notice that label 10 represents digit 0), the histograms of both errors has two-peak pattern, particularly for the last three which have relatively smaller error rate. Heuristically it tells that classifying digits like 3,7 and 9 may be more challeging than digits like 0,1,5; GMM may take advantage of classifying digit 3 than SVM, and 3-NN has the best performance in recognizing 9 than the rest. This gives heuristic hints for the combination of classifiers, meanwhile poses a clear challenges for further improvement of performance since many patterns are commonly misclassified. As shown in Figure 7, some misclassified patterns are diffucult to recognize even for human.

For instance, all classifiers classify “3” at the 5th place in the first row to be “7”.

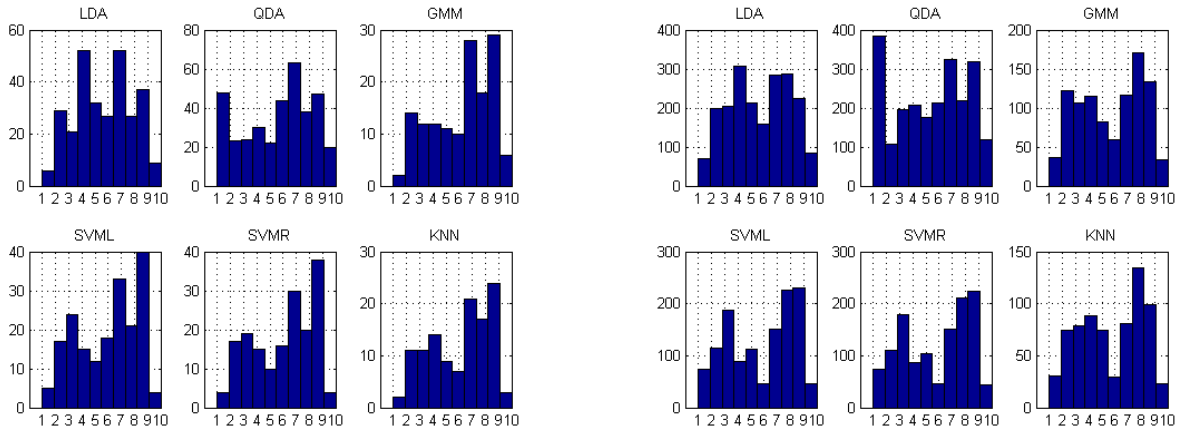

Figure 6: Error Frequency of training (*left*) and test (*right*) data for six classifiers

Figure 7: Examples of ommonly misclassified patterns

6 Potential improvement

To further improve our pattern recognition system we also consider rejection option for individual classifer and combination of the classifiers.

6.1 Rejection option

The error rate of individual classifier can be further reduced if we consider the rejection option, which can also discriminate unexpected patterns like letters or other characters. Since we are using different rejection rules for classifers, we take the Linear Classifier(LDA) to ilustrate, where we assign the label according to the posterior density from estimated Gaussian distribution with equal covariance assumption. However if the maximal density is not as high as expected, the assumption of membership may become problematic. We thus set a threshold of 0.5 for our LDA classifier and it turns out to reject 8 patterns. As shown in Figure 8, these patterns may have high ambiguity. The error rate is reduced to 2.85% from 2.92%.

Figure 8: Rejected patterns for LDA by the rejection rule of threshold = 0.5

6.2 Combination of classifiers

Common techniques for combination of classifiers are considered, such as bagging, boosting and other resampling methods. However a significant improvement can be achieved only if the component classifiers perform better than chance [5], whereas the six classifers we focus on commonly have low error rate. Some other issues like higher cost again arise. Next, we consider weighted sum of several pairs of classifiers, using corresponding training error as the weight. The resulting test error rate, however, always falls between the error rates of the individual classifiers. We conclude that the methods for combining classifiers we implement so far do not significantly improve the recognition system. However it remains a potential way for achieving better results to combine the classifiers.

References

[1] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[2] Hiroshi Sako Hiromichi Fujisawa Cheng-Lin Liu, Kazuki Nakashima. handwritten digit recognition: benchmarking of state-of-the-art techniques. *The journal of the pattern recognition society*, 36:2271–2285, 2003.

[3] Y.LeCun et al. Comparison of learning algorithms for handwriteen digit recognition. *International Conference on Artificial Neural Networks*, pages 53–60, 1995.

[4] John C.Platt Patrice Y.Simard, Dave Steinkraus. Best practices for convolutional neural networks applied to visual document analysis.

[5] David G.Stork Richard O.Duda, Peter E.Hart. *Pattern Classification*. John Wiley & Sons, Inc., 2 edition, 2001.

[6] Changsong Liu Xuewen Wang, Xiaoqing Ding. Gabor filter-based feature extraction for character recognition. *The journal of the pattern recognition society*, 38:369–379, 2005.