Summary and Comparison of Different Methods Used in Handwritten Digit Recognition

Jinlin Fan*

Department of Statistics, School of Economics

Xiamen University

Abstract

Handwritten digit recognition has become a hot topic which is concerned with methods from divergent fields, such as pattern recognition, machine learning, Graphic problems and Topology and so on, most of which can be included in whether Memory-based methods or Learning-based methods. An efficient method can make a breakthrough in relevant industries. In this article, we focus on MNIST database. Different methods will be considered and combined, such as PCA-based method, Neutral Network, SVM, and method based on distances. We also do some try, with the help of LDA/QDA, to solve a obvious problem arisen in classification of 4 and 9. Finally we compare different method from aspects of accuracy and the amount of time spent. We conclude that different methods should be considered in cases whether you prefer accuracy, time-saving, or interpretation of your result, but learning-based method usually get both accuracy (92.2%) and time-saving.

Key Words: Comparison; Memory/Learning-based; Principal component analysis; Neutral Network; LDA/QDA; Accuracy and Time-saving.

1 Introduction

The recognition of handwritten digit is a classic problem in pattern recognition, perhaps from around the 1980s. Because the handwritten has do not have the tantamount size, orientation, position on a piece of paper and thickness, finding a efficient and accurate algorithm or classifier to handle the task really matters, which will be beneficial to lots of applications like scanning and sorting envelopes and processing checks[1]. There are papers proposing two types of algorithms: **memory based** and **learning based**[2]. Memory based methods try to store the training set first and classify an new unknown digit by comparing it with the stored digits, while learning based methods aim at learn the intrinsic pattern and set up a function to complete classification (A kind of procedure changes the classification task into a prediction task).

*Grant Thanks for Professor Jingyuan Liu, Wei zhong

In this article, we use samples of digits from **MNIST**'s dataset from website-Kaggle.com, which contains 42,000 training samples(with labels) and 10,000 testing samples. While in this article, we will only a subset of this dataset, virtually we sample 3000 samples as training set and sample again in the rest of original data to get the testing set with size of 1000. And every image has 28×28 pixels and grayscale (0-255), which consist high dimension vectors.

Before the analysis, we can check the dataset with images in figure 1.

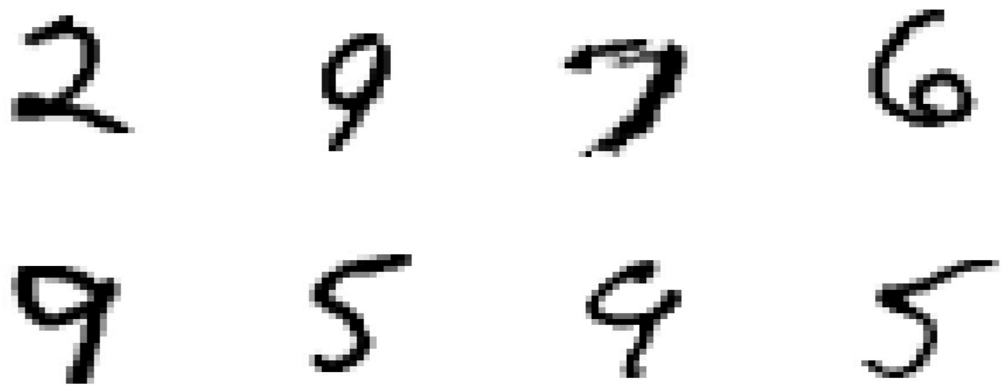

Figure 1: A sketchy graph of several samples

Although, there are many mature approaches belonging that two types we referred before.In this article, we try some combinations of them. Because memory based steps can help us catch the main feature of written digits, and learn-based methods can give us a accurate function, maybe very convoluted, for which they are not clear to interpret. Obviously, memory-based algorithms can show us better geometric interpretation.

In following sections, several algorithms and methods will be represented one by one. We will try PCA-based methods, **Neutral Network** based on global PCA and 1NN method based on local PCA.

And we will also discuss a method based on **Tangent Distance**. Tangent Distance, the Euclidean distance between two tangent planes, is a invariant distance which is much more robust, kind of objective, than *Euclidean* distance when we take transformations like moving, scaling and rotations into consideration.

What's more, a classification problem–"4 or 9" arises in the results of different methods. We should take account of how to do to solve it.

At the end of every section, we will give a summary of the results, comparison and some comments of that section.

All of the calculation are done by R language. The R codes are included in Appendix.

2 Different combinations of methods

2.1 Neutral Network based on Global PCA

As we talk about dimension reduction or approximation of high dimension data, we actually want to find functions f and g:

$$f : \mathbb{R}^n \to \mathbb{R}^m;\ g : \mathbb{R}^m \to \mathbb{R}^n$$

For a give data matrix $\mathbf{X_{l\times n}}$= $[X_1, X_2, .., X_n]$, we want the **Frobenius norm** ($\|\cdot\|$) to be minimized:

$$\underset{f,g}{argmin}\, \|\mathbf{X} - f\cdot\, g(\mathbf{X})\|_\mathbf{2}$$

PCA aims at finding the optimal linear combination to minimize the error of reconstruction supported by following theorems.

Theorem 2.1. [3] *If* $\mathbf{X_{l\times n}}$ *is a standardized data matrix with n dimensions, and the optimal linear combination* $\mathbf{A_{n\times m}} = [\mathbf{a_1}, \mathbf{a_2}, ...\mathbf{a_m}]$ *can satisfy*

$$\underset{A}{argmax}\, \mathbf{A^T}(\mathbf{X^T X})\mathbf{A}$$

subject to

$$\mathbf{A^T A} = \mathbf{I}$$

then $\mathbf{a_i} = \mathbf{e_i}, i = 1, 2, ..m,$ $\mathbf{e_i}$ *is the ith eigenvector of the covariance matrix* Σ *of* $\mathbf{X}$

Theorem 2.2. [4] *Let the data* $\mathbf{X}$ *be a* $n \times p$ *matrix with 0-mean columns, where n and p are the number of observations and the number of variables, respectively. Let the SVD(Singular Value Decomposition) of* $\mathbf{X}$ *be*

$$\mathbf{X} = \mathbf{UDV^T}$$

$\mathbf{Z} = \mathbf{UD}$ *are the principal components, and the columns of V are the corresponding loadings of the PCs.*

Theorem 2.3. [5] *Let* $\mathbf{A}$ *be a read* $m \times n$ *matrix with rank r. The matrix approximation problem*

$$\min_{rank(Z)=k} \|\mathbf{A} - \mathbf{Z}\|_\mathbf{2}$$

has the solution

$$\mathbf{Z} = \mathbf{U_k \Sigma_k V_k^T}$$

where $k < r$*, and* $\mathbf{U_k}, \mathbf{\Sigma_k}, \mathbf{V_k}$ *follows the SVD decomposition of* $\mathbf{A}$ *with first k columns.*

Actually the result above can help us transform the dataset into a lower dimension space, which will obviously save our time when use high-computational-cost method like NN.

In this network, many simple "neurons" are put together, and each neuron can put out a value respective to its unique input value. Usually, a sigmoid function is set in a single neuron:

$$f(z) = \frac{1}{1 + exp(-z)}$$

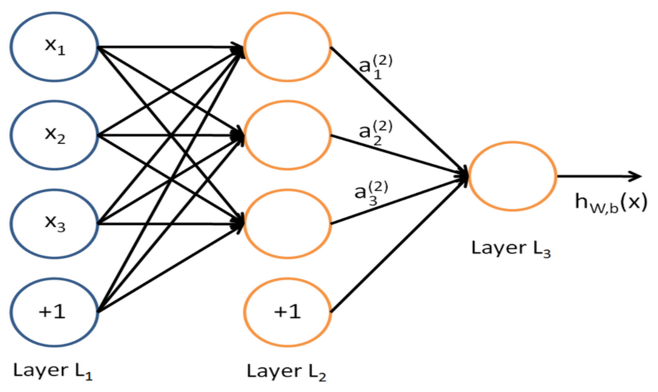

Figure 2: BP Neutral Network[6]

it has a range [0,1], then its derivative is calculated by $f(z)(1-f(z))$. In this figure, we can check that each circle is a single neuron and the circle with "+1", actually is a constant, or a intercept. The rightmost layer is output layer, which can give us final result, while the leftmost is input layer. The middle layer is called hidden of latent layer, because its value cannot be detected.

In this cases, suppose that we have parameters[6]: $(\boldsymbol{w}, a) = (\boldsymbol{w_1}, b_1, \boldsymbol{w_2}, b_2)$, in which $w_{ij}^{(k)}$ is the coefficient connect the ith neuron in kth layer and jth neuron in (k+1) layer. And let a_i^k be the input to ith neuron in kth layre and $h_{w,b}(x)$ be the final output, we can have a simple formula:

$$\begin{aligned}
a_1^{(2)} &= f(w_{11}^{(1)}x_1 + w_{22}^{(2)}x_2 + w_{13}^{(1)}x_3 + b_1^{(1)}) \\
a_2^{(2)} &= f(w_{21}^{(1)}x_1 + w_{22}^{(1)}x_2 + w_{23}^{(1)}x_3 + b_2^{(1)}) \\
a_3^{(2)} &= f(w_{31}^{(1)}x_1 + w_{32}^{(1)}x_2 + w_{33}^{(1)}x_3 + b_3^{(1)}) \\
h_{w,b}(x) = a_1^{(3)} &= f(w_{11}^{(2)}x_1 + w_{12}^{(2)}x_2 + w_{13}^{(2)}x_3 + b_1^{(2)})
\end{aligned}$$

Thus, **because NN contains the linear combination of single neurons and non-linear sigmoid function, even though written digits have many uncertain shapes and sizes, NN method can hold the complicated nonlinear relationship well in some aspects.**

So given the formula, our goal is to estimate these parameters conditioning on: *minimizing the classification error ϵ, and accuracy rate θ*

$$\epsilon = \frac{N_{mistaken}}{N_{Total}};\ \theta = 1 - \epsilon$$

by cross-validation method in training dataset.

2.1.1 Experimental result

Try this procedure on our dataset. Firstly, according to the screeplot we choose 20 principal components (Figure 3).

20 principal components are enough, because their cumulative **explained variance** is **95.09%**. And then we get lower-dimension training set and testing set by project original vectors on this new subspace. Go on estimating the NN with backpropagation algorithm, then we finally get our model and prediction, the details are showed in Table.1.

We find that the error is $\boldsymbol{\epsilon = 0.078}$, accuracy rate is $\boldsymbol{\theta = 0.922}$, someway, we can regard such a classifier as a well-performed classifier.

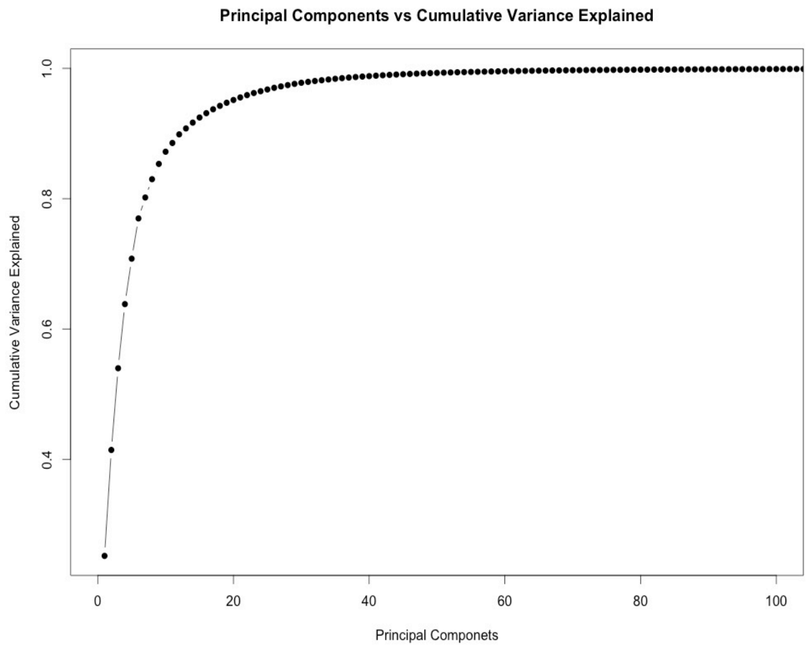

Figure 3: Screeplot of 3000 training digits

ori vs pre	0	1	2	3	4	5	6	7	8	9
0	100	0	0	0	0	2	1	1	2	1
1	0	100	1	0	0	0	0	0	2	1
2	2	0	107	2	0	1	0	0	3	1
3	0	1	3	92	0	3	1	0	1	0
4	0	0	1	0	69	0	0	1	0	7
5	0	0	0	2	0	88	0	0	2	0
6	0	0	0	1	0	3	91	0	0	0
7	0	0	0	2	2	0	0	95	0	4
8	0	1	0	2	0	0	2	0	82	0
9	1	3	1	3	8	0	0	3	0	98

Table 1: Comparison between Original Labels and Predicted Labels by NN-PCA

Also, we can find that some digits are easy to be classified into a wrong group, such as 4 and 9, 3 and 8. We can show some wrong images in Figure 4.

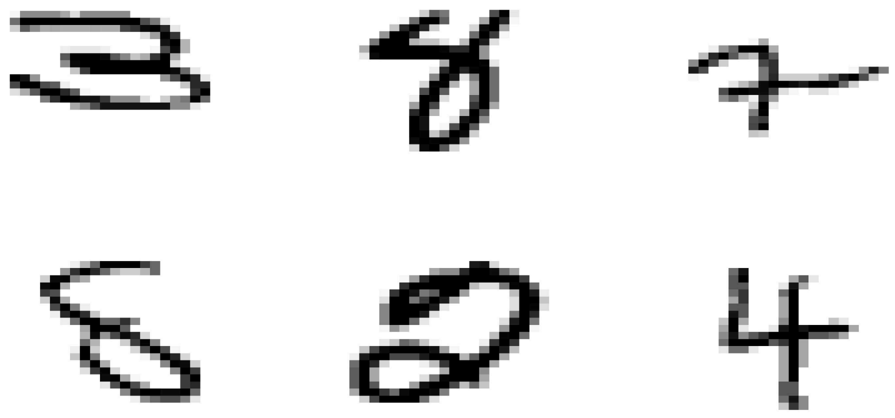

Figure 4: Images of Wrongly Classified Digits

Respectively,they are 3, 8, 7, 8, 2, 4 and wrongly classified as 6, 3, 4, 6, 0, 9. It is clear to see 4 and 9 actually have some similar shapes, and for the third images, the writer may be careless to leave a redundant mark on this 7. From this result, **this classifier shows that if something wrong happens, it may attribute to nonstandard writing habits**. But for 4 and 9, NN faces an obstacle.

2.1.2 A new try and result and comparison

Why not a Support Vector Machines? Perhaps after reduce dimension, any types of classifiers can perform well. Following this thought, we can do some comparison to see such a problem more clearly.

This problem should be nonlinear and involved. But after dimension reduction, we are easier to catch the linear relationship between the input and output. We can see from the SVM result.

With the help of LIBSVM[7], we resort to C-SVM which aims at optimizing

$$\min_{\epsilon_i, w, b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \epsilon_i$$

$$\text{s.t. } y^{(i)}(\boldsymbol{w^T x^{(i)}} + b) \geq 1 - \epsilon_i,\ i = 1, 2, ..., m,\ \epsilon_i \geq 0$$

Utilizing different kernel function with different degrees, we can control the capability of handling nonlinear dataset.

We use the same models as a Final Report of George[8], the difference is that we use them on the data projected on PCs, while George use them on the original data. The results are in Table.2

The gap of error rate between two algorithms ($\triangle\epsilon = \mathbf{7\%}, \mathbf{0.4\%}$) show that the nonlinear pattern changes into some kind of linear pattern with principal components' transformation.

Algorithm	Error Original	Error PCs
SVM Linear	11.1%	12.1%
SVM 4 poly	4.1%	11.7%

Table 2: Comparison between Results on Two Datasets

2.1.3 SUMMARY

That SVM and NN based on dimension-reduced data by PCA has a similar performance: **87.9% and 92.2%**, and also the similar performance of SVM-poly(6) and SVM-linear, both means that a linear method performs as well as a nonlinear method because of the shrinkage of nonlinear pattern in dataset after finding the principal components(PCs). And from that, we suppose that combining many time-saving methods with PCA may get a pleasing result under a restraint of time or other cost.

2.2 1NN Based On Local PCA and Euclidean Distance

To increase the accuracy, we want to catch digits own features respectively, which derives Local PCA[9].

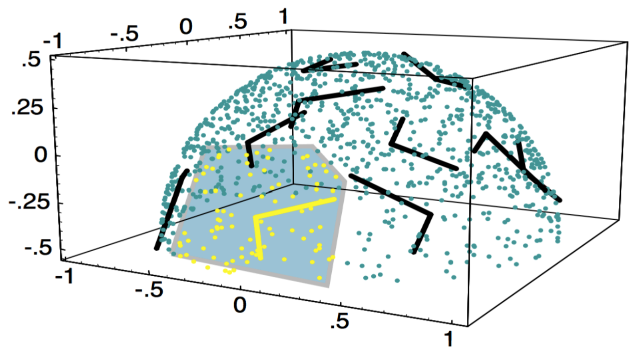

Figure 5: Local PCA in A Space[9]

Local PCA is a sort of PCA under conditioning neighbor or kernel density. **To take the different shapes of written digits into consideration, we conveniently define the neighbor of digit, such as 3, is all the observations with label '3' in the training set**. Of course, we can also define a smaller neighbor by a constant Euclidean distance between the same digits, but we just attempt to do a simple try about such a method and for some extreme cases, such a neighbor may not include them.

And then when we get 10 subspaces $\mathbb{R}^m_{(K)}, (K = 0, 1, ...9, m < n)$ consist of different PCs. we can classify a new observation $\boldsymbol{x} \in \mathbb{R}^n$ by calculating its distance from each subspace and assign it into the group from which the distance is the smallest.

Algorithm

Suppose $\boldsymbol{x} \in R_n$, then we will assign $\boldsymbol{x}$ into group K

$$K = \underset{K}{argmin}\, \mathcal{D}(\boldsymbol{x}, \mathbb{R}^m_{(K)}), K = 0, 1, ..., 9$$

where $\mathcal{D}(\cdot)$ is Euclidean distance.

How to define the distance between a vector and a space? Let $B_k = [b_1, b_2, ..., b_k], k < n$ be the basis vectors of a space and $\boldsymbol{x} \in \mathbb{R}^n$ be a new observed vector. We should find a vector $\alpha \in \mathbb{R}^k$ on the space B_K from which the distance to x is the smallest:

$$\min_\alpha \|B_k\alpha - \boldsymbol{x}\|_2$$

By intuition, α should be projection of x on the space.

$$\begin{aligned}\min_\alpha \|B_k\alpha - \boldsymbol{x}\| &= \min_\alpha (B_k\alpha - \boldsymbol{x})^T(B_k\alpha - \boldsymbol{x}) \\ &= \min_\alpha (\alpha^T\alpha - \alpha^T B_k^T\boldsymbol{x} - \boldsymbol{x}^T B_k\alpha + \boldsymbol{x}^T\boldsymbol{x})\end{aligned}$$

Then take the derivative:

$$2\alpha^T - 2\boldsymbol{x}^T B_k = 0 \Rightarrow \alpha = B_k^T\boldsymbol{x}$$

which is the projection of $\boldsymbol{x}$ on the B_k space, the same as the intuition.

2.2.1 Experimental result

So we first get PCs, then get distances used to classify.

According to the theorems in 2.1, 2.2, 2.3, we can find that principal components can help us to minimize the reconstruction error, which means that they can be used for lower dimension approximation. And SVD decomposition can help us quickly get the top principal components of any real matrix, while PCA can only be carried on when the matrix is a square covariance matrix. Thus, generally, this is a good time-saving method instead of calculating the covariance matrix and doing spectral decomposition. Somehow, SVD and PCA will get same results in many cases. We get each digit's PCs, and we choose the first 20. The **low-dimension approximation** of digits is in Figure 6. The prediction result of testing dataset is in Table 3.

2.2.2 SUMMARY

We compare this result with former result, although the accuracy rate of two methods are both $\boldsymbol{\theta} = \mathbf{92.2}\%$, we find that result of 1NN nearly promotes all the accuracy of classification expect that of digit '8'. And the mistaken classification of '9' and '4' has reduced a little. But from the approximated images, we can tell that digit '4' and '9' really have patterns in common. It seems to be a common obstacles before both approaches.

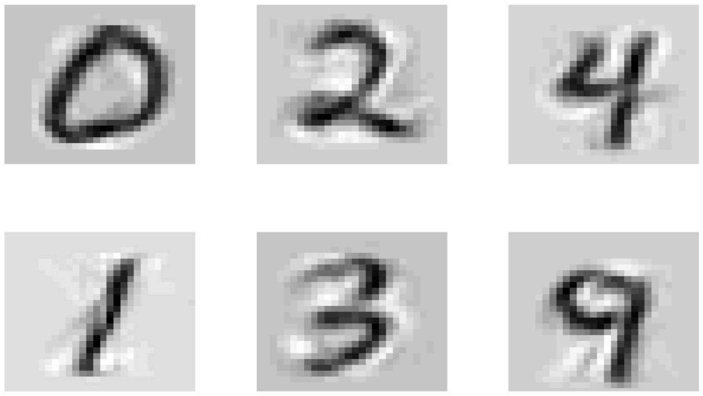

Figure 6: 20 PCs approximated digits 0-4,9

ori v.s pre	0	1	2	3	4	5	6	7	8	9
0	105	0	0	0	0	0	1	0	0	1
1	0	101	1	0	0	0	0	0	1	1
2	3	1	109	0	0	0	0	2	1	0
3	2	1	2	89	0	3	0	1	1	2
4	1	1	0	0	70	0	1	0	0	5
5	2	0	0	2	0	82	2	0	4	0
6	2	0	0	0	0	0	92	0	1	0
7	0	1	0	0	1	0	0	98	0	3
8	0	3	1	3	0	1	2	0	74	3
9	1	1	0	1	5	1	0	4	2	102

Table 3: Comparison between Original Labels and Predicted Labels Local PCA

2.3 Similarity in Tangent Distance[10]

In many papers, authors propose an invariant distance–Tangent Distance. The accuracy of the classifier depends a lot on the size of training set. Thus, we not only need the image of digit '3', but images of left-shifted or right-shifted '3', and so on. Tangent distance is a kind of distance can regard the transformed '3' and standard '3' as nearly a same point. Let $s(x, \alpha)$ be the transformation of $x \in \mathbb{R}^n$ with the parameter α, so the set of all transformations

$$s_p = \{y | \exists \alpha, y = s(x, \alpha)\}$$

And it's obvious that $s(x, 0) = \boldsymbol{x}$, which means that no transformation on x happens. If α is a k-dimension vector, under some regular conditions, s_p may be a smooth plane in whole space. Actually, when we have a new observation a, we want to find if a and x are included in common group, the distance between a the $s(x, \alpha)$ plane generated by all the transformation of x. Because $s(x, \alpha)$ is not a linear plane, which means the distance are not easy to calculate. Instead, some researches use the tangent plane at point x to remedy. By the help of Taylor expansion[?] of $s(x, \alpha)$, around $\alpha = 0$, which means the region near x

$$s(x, \alpha) = s(x, 0) + \alpha \frac{\partial s(x, \alpha)}{\partial \alpha} + O(\alpha^2) \approx P + L_P \alpha$$

While the linear form is what we want. We can do the same thing to new observation a after generating some transformation on it. Then we can get a Euclidean distance between to approximated linear plane.

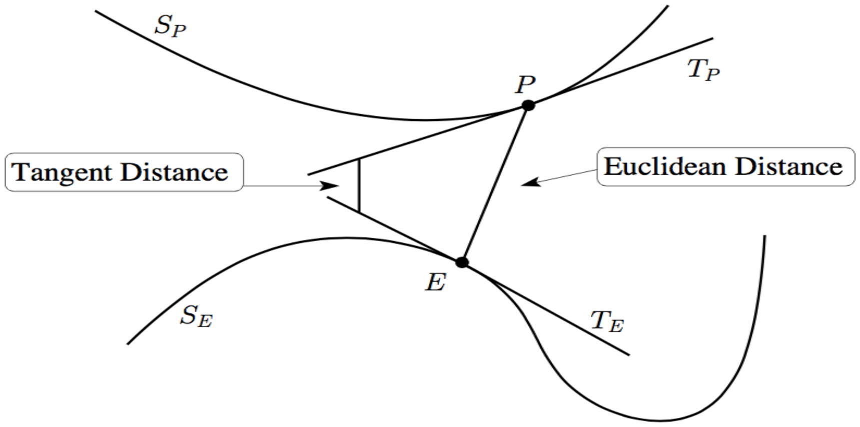

Figure 7: Tangent distance and Euclidean distance

2.3.1 SUMMARY

Surprising: The thought is kind of same as what we follow in Local PCA,, because that we use all the same digit, such as all the digit '3', to get the k-dimension PCs equals to that we get a k-dimension tangent plane of standard digit '3', if $\alpha = k$. Of course there are also some papers using SVD method to find tangent plane. Thus the result of this method will not differs a lot from the result of 1NN.

2.4 Discriminant of Analysis and "4 or 9" Problem

How to increase the accuracy of classifying 4 and 9? In this article, we propose a method based on PCA of differences between 4 and 9. The differences can show 4 and 9's own features. And we use all the 4 and 9 digits in the training set to sample a set of 4s and a set of 9s both with size of 100. After calculating $100 \times 100 = 10000$ differences, we are able to estimate $\boldsymbol{E}(x_9 - x_4)$. The point larger than zero shows that digit 9's not 4's own features, vise versa. And we can also have a more direct figure to show the obvious region on the paper where they are different in Figure 8.

Based on PCA of these different features, we will get a subspace including them. We can also project vector x_4 and x_9 on the subspace. As we have found that after reducing the dimensions by principal components, many nonlinear patterns fade away, which allows us to try LDA(Linear Discriminant Analysis) or QDA(Quadratic Discriminant Analysis).

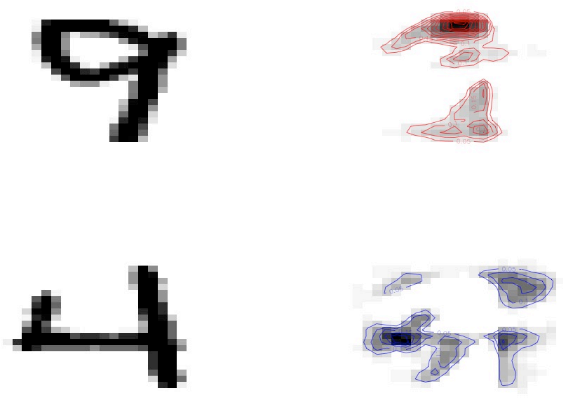

Figure 8: Differences between Digit 4 and 9

The difference between LDA and QDA is whether the covariance matrices are homogeneous, if yes, then LDA; if not, then QDA.

QDA Decision Rule[11]

Classify a new observation $\boldsymbol{x}$ *into the population that has the largest quadratic score function:*

$$s_i^Q(\boldsymbol{x}) = -\frac{1}{2}log|\boldsymbol{S_i}| - \frac{1}{2}(\boldsymbol{x}-\bar{\boldsymbol{x}})'\boldsymbol{S_i^{-1}}(\boldsymbol{x}-\bar{\boldsymbol{x}}) + \log(p_i), i = 0, 1, ..., 9$$

where p_i *is the proportion of group i,* $\boldsymbol{S}_i$ *is the sample covariance matrix of group i.*
The Figure 9 shows the different boundaries under LDA (dashed) and QDA.

2.4.1 Experimental result

We can find the PCs of the differences we calculate before, the same as what we have done in former section. But this time we need to choose 80 PCs to satisfy the 95% explained-variance-criteria.

Then we project our data vectors on the space based on these PCs to be ready to do LDA and QDA. We have 4s and 9s (610 in total) in the same training set as before, and 4s and 9s (195 in total) in the same test testing set. The results are in Table 4. We use both model to predict 4s and 9s in both training and testing set for comparison.

2.4.2 SUMMARY

So we find that according to methods used in former sections, the accuracy rate of classifying 4 and 9 is **91.6%**, **94.5%**, while this result show that after getting the difference we can increase the accuracy rate to **95.3%**, even to **99.6%**. Although, the promotion is

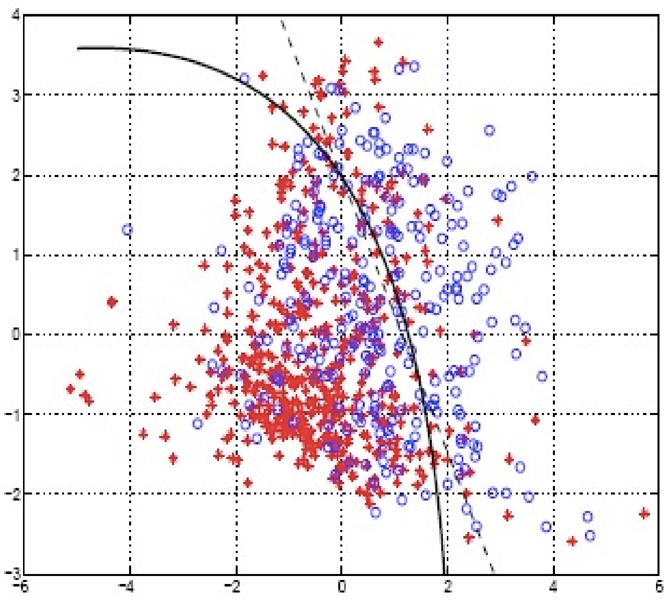

Figure 9: Difference between LDA and QDA

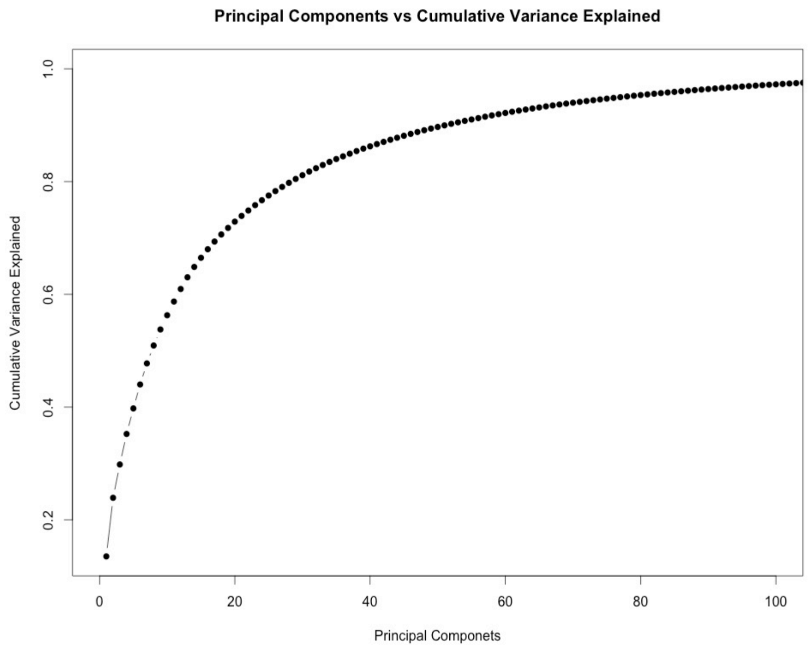

Figure 10: Screeplot of Differences

not very significant, but such a procedure really show us some intrinsic and unique patterns of digit 4 and 9. So we propose that if we can catch all the unique patterns of all the digits, then the accuracy may be improved.

	Datasets	Training		Testing	
Models	pre vs org	4	9	4	9
LDA	4	297	8	75	7
	9	13	292	3	110
QDA	4	308	0	76	7
	9	2	300	2	110

Table 4: Comparison between LDA and QDA

3 Conclusion

After trying different combination of methods, we can find that PCA really is some kind of important roles in written digits recognition tasks. And we can do some comparison from accuracy and time.

(1). Neutral Network based on Global PCA: most of time are used to train the NN model. It costs about 2 minutes and gets the 92.2% accuracy rate.

(2). SVM based on Global PCA: It really saves time but its accuracy rate is not high—approximated 88%.

(3). 1-Nearest-Neighbor method based on Local PCA: Such a procedure really promote accuracy of classifying almost all the digits a little, but not so good on recognize 3 and 8, 4 and 9. While its accuracy rate is also 92.2%, it is a really time-consuming method. Calculating the distances between testing observations with 10 subspaces costs about 10 minutes. This fact tells us a pure **memory-based** methods is much more time-consuming than **learning-based** methods.

(4). LDA and QDA: In this section, we discuss about the unique patterns existing in written digits. We try to solve "4 or 9" problem with the help of, after trying to find their unique patterns, LDA and QDA models. And the result show that we can improve the accuracy a little, and we suppose if all the unique patterns of each digit can be found, the classification accuracy will be promoted a lot. By the way, most of time are spent by generating the differences between vectors x_4 and x_9 to find the unique patterns.

Actually, combination of different method do really get some better results or save more time when applying them in real world.

Referrances

[1] Gaurav Jain, Jason Ko, *Handwritten Digits Recognition.* Master's thesis, 11/21/2008.

[2] José Israel Pacheco, *A Comparative Study for the Handwritten Recognition Problem.* Phd's thesis, 2011

[3] Alvin C.Rencher, William F.Christensen, *Methods of Multivariate Analysis Third Edition.* 2012.

[4] H. Zou, T. Hastie, R. Tibshirani, *Sparse Principal Component Analysis.*

[5] L.Elden. *Matrix Methods in Data Mining and Pattern Recognition.* 2007.

[6] Andrew Ng, Jiquan Ngiam, *Neutral Network.* Internet: http://deeplearning.stanford.edu /wiki/index.php/UFLDL_Tutorial, 7/4/2013.

[7] Chih-Chung Chang and Chih-Jen Lin, *LIBSVM : a library for support vector machines.* Internet: http://www.csie.ntu.edu.tw/ cjlin/libsvm, 2011.
[8] George Margulis, *Optical Character Recognition: Classification of Handwritten Digits and Computer Fonts.* CS229 Final Report.
[9] N. Kambhatla, T. K.Leen, *Dimension Reduction by Local Principal Component Analysis.*
[10] Patrice Y.Simard, Yann A. Le Cun. *Transformation Invariance in Pattern Recognition: Tangent distance and Propagation.* Microsoft Research.
[11] PennState Eberly College of Science, *STAT505-10.6: Quadratic Discriminant Analysis* Internet: https://onlinecourses.science.psu.edu/stat505/node/97, Online Course.