Mini-Project Report

Data Compression and Digits Classification

Group Member:
FAN Min 20401482
PAN Hui 20360107
PAN Mengxin 20388379

March, 12, 2017

Abstract

Our group select hand-written digit dataset on the provided website, which is part of the mnist database. In this project, we implemented principal component analysis (PCA) to do dimensional reduction, in order to compress the original data and achieve high-quality recovery as well.

Inspired by the geometric meaning of PCA, we intuitively built two models for digits classification: angle method and distance method based on PCA, which are both highly-performing and time-saving. From another perspective, PCA can be regarded as feature transformation. Transformed features in low-dimensional space enabled us to train some models and avoid the curse of dimensionality, which refers to the phenomenon that as dimension goes up, the complexity of algorithms increases drastically. We can see how the time in training phase exploded, as the dimension of feature increase in logistic regression algorithm.

We also adopted support vector machine (SVM) with transformed features. From theoretical view, logistic regression and SVM have similar performance and our group proved this in the experiments. The advantage of SVM over logistic regression is substantially time-consuming and achieving competitive prediction accuracy meanwhile.

Logistic regression can be treated as a neural network without hidden layers. Logistic regression with linear kernel could only approximate the convex boundaries, while the neural network with hidden layers (even only one hidden layer) possesses the ability to approximate non-convex boundaries. The result of neural network algorithm we ran also demonstrated the power of neural network.

Workload: Group members discussed and shared each idea with the others, then decided the final project plan together. FAN Min is responsible for Section 1, 2 in I and Section 3, 4 in Part II. PAN Mengxin is responsible for Section 5 in Part II. PAN Hui is responsible for Section 6 of Part II.

Contents

I Compression and Recovery **2**

1 Principal Component Analysis **2**

2 Compression and Recovery **3**

2.1 Compression of Hand-written Digits 3

2.2 Recovery from the Compressed Data 3

II Classification **5**

3 Geometric Methods based on PCA **5**

3.1 Angle Method 6

3.2 Distance Method 7

3.3 Discussion 8

4 Logistic Regression **10**

4.1 Brief Introduction 10

4.2 Experiments 10

5 Support Vector Machine (SVM) **11**

6 Neural Networks (NNs) **13**

6.1 Introduction of Neural Networks 13

6.2 Experiment Setup 14

Part I

Compression and Recovery

In this part, we will briefly introduce some notations and recall theory of Principal Component Analysis (PCA). Then PCA is implemented on the hand-written digit dataset and the recovery performance with different number of principal components is demonstrated in section 2.

1 Principal Component Analysis

The idea of principal component analysis can data back to the beginning of the 20th century, found in these two papers [Pea01] and [Hot33]. The purpose of PCA is, roughly speaking, to find a good low-dimensional affine space to represent high-dimensional data.

Denote data as $\mathrm{D} = [d_1, d_2, \ldots, d_n] \in \mathbb{R}^{p\times n}$, where n is the number of observations in data and p is the dimension of observations. We are going to find a k-dimensional affine space in $\mathbb{R}^p$ to approximate these n observations. Suppose affine space can be parameterized by $\mu + \mathrm{U}\beta$, where $\mu \in \mathbb{R}^p$ is the origin of affine space and $\mathrm{U} = [u_1, u_2, \ldots, u_k] \in \mathbb{R}^{p\times k}$ consists of k-columns of an orthonormal basis of the affine space and β is the coordinate under this affine space. The coordinates of all observations D under the affine space is $\mathrm{B} = [\beta_1, \ldots, \beta_n] \in \mathbb{R}^{k\times n}$.

The criteria of selecting affine space is to minimize the loss of "energy", which is indicated by Euclidean distance:

$$\min_{\beta,\mu,\mathrm{U}} \sum_{i=1}^{n} \|d_i - (\mu + \mathrm{U}\beta_i)\|^2, \tag{1.1}$$

After deduction, we know that μ is the average of all observations and U is formed by the first k left singular vectors, which is the first k columns of the left singular matrix ($\hat{U}$ in equation 1.3) of $\hat{\mathrm{D}} = [\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_n]$, where

$$\mu = \frac{1}{n}\sum_{i=1}^{n} d_i \tag{1.2}$$

$$\hat{\mathrm{D}} = \hat{U}\hat{\Sigma}\hat{V}^{\mathrm{T}}, \quad \hat{d}_i = d_i - \mu. \tag{1.3}$$

Equation 1.3 is the singular value decomposition of $\hat{\mathrm{D}}$.

Suppose n is larger than p, which means the number of data is more than the dimension of data. $\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq \cdots \geq \hat{\lambda}_p$ are diagonal elements of $\hat{\Sigma}$, called the singular values of $\hat{\mathrm{D}}$. The quantities that could indicate the loss of "energy" is showed below:

- Total Variance:

$$\mathrm{trace}(\hat{\Sigma}_n) = \sum_{i=1}^{p} \hat{\lambda}_i, \tag{1.4}$$

- Percentage of Variance explained by top-k principal components:

$$\mathrm{p}_{\mathrm{var}}(k) := \frac{\sum_{i=1}^{k} \hat{\lambda}_i}{\mathrm{trace}(\hat{\Sigma}_n)} \tag{1.5}$$

The columns of $\hat{\mathrm{D}}$ can be decomposed as $\hat{d}_i = \hat{d}_{i\|} + \hat{d}_{i\perp}$, where $\hat{d}_{i\|} \in \mathrm{span}(\mathrm{U}) = \mathrm{span}(u_1, \dots, u_k)$ and $\hat{d}_{i\perp} \in \mathrm{span}^{\perp}(\mathrm{U}) = \mathrm{span}(u_{k+1}, \dots, u_p)$. From the properties of the singular value decomposition, we could have relationships below:

$$\sum_{i=1}^{p} \hat{\lambda}_i^2 = \mathrm{trace}(\hat{\Sigma}_n \hat{\Sigma}_n^{\mathrm{T}}) = \mathrm{trace}(\hat{\mathrm{D}}\mathrm{D}^{\mathrm{T}}) = \sum_{i=1}^{n} \hat{d}_i^2, \tag{1.6}$$

$$\sum_{i=k+1}^{p} \hat{\lambda}_i^2 = \min_{\beta,\mu,\mathrm{U}} \sum_{i=1}^{n} \|d_i - (\mu + \mathrm{U}\beta_i)\|^2 = \sum_{i=1}^{n} \hat{d}_{i\perp}^2, \tag{1.7}$$

$$\sum_{i=1}^{k} \hat{\lambda}_i^2 = \sum_{i=1}^{n} \hat{d}_{i\|}^2. \tag{1.8}$$

From these relationships, we can clearly see that $\mathrm{p_{var}}(k)$ in equation 1.5 is a good indicator of how best the affine space approximates original data. In practice, different fields have different rules to choose k. For example, computer vision field chooses k such that $\mathrm{p_{var}}(k) \geq 90\%$.

2 Compression and Recovery

We select hand-written digit dataset, where images are in gray scale. And each image has 16*16 pixels, that is, each digit has 256 features. There are 7291 digits in training set and 2007 digits in testing set. After principal component analysis, the number of features could change from 256 to k, which could be substantially less than 256. For instance, if $k = 25$, we compress original data such that it only needs, approximately, 1/10 previous storage to store these data and could achieve high-quality recovery meanwhile.

2.1 Compression of Hand-written Digits

Denote hand-written digit training data as $\mathrm{D} = [d_1, d_2, \dots, d_n] \in \mathbb{R}^{256\times 7291}$, where 256 is the number of pixels in each image and 7291 is the number of observations in training set. After adopting PCA algorithms repeatedly, we could have the relationships of k and $\mathrm{p_{var}}(k)$ in figure 2.1 and in table 2.1.

k	7	11	17	29	55	88
$\mathrm{p_{var}}(k)$	51.01%	61.27%	70.04%	80.41%	90.14%	95.20%

Table 2.1: The relationship between k and $\mathrm{p_{var}}(k)$

The experiment indicates that it only needs 29 or 55 features to achieve over 80% or 90% variance. Compared with original 256 features of each digits, the transformed data $\mathrm{B} = [\beta_1, \dots, \beta_n] \in \mathbb{R}^{k\times n}$ are highly compressed and storage-saving.

2.2 Recovery from the Compressed Data

From section 1,we could know that $\tilde{d}_i$ is the approximation of each digit d_i:

$$\tilde{d}_i = \mathrm{U}\beta_i + \mu, \tag{2.1}$$

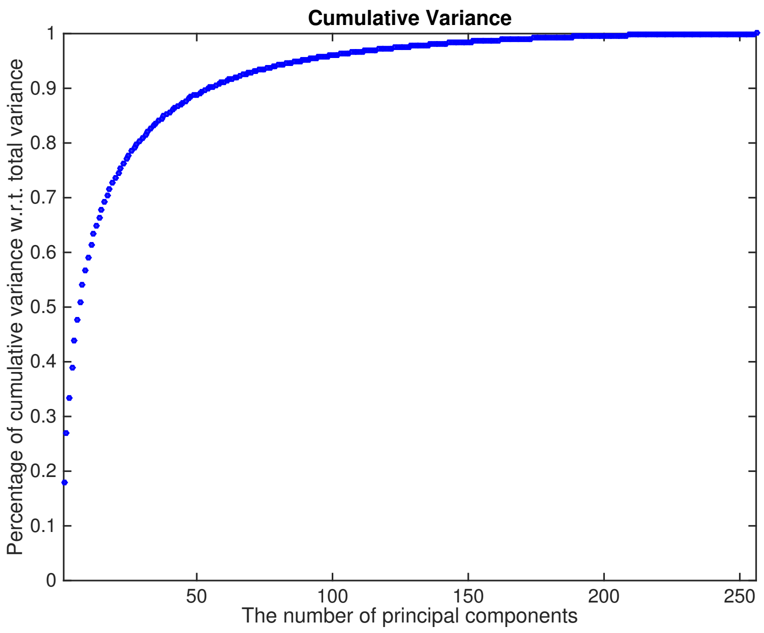

Figure 2.1: Cumulative variance: x-axis is the number of principal components, y-axis is the value of $\mathrm{p}_{\mathrm{var}}(k)$ corrosponds to different k.

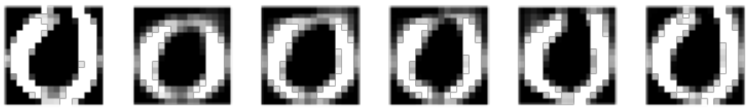

Figure 2.2: Recovery: one of digit 0

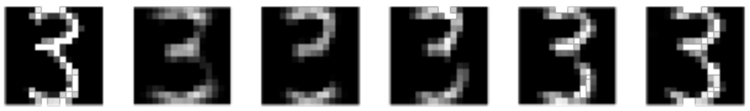

Figure 2.3: Recovery: one of digit 3

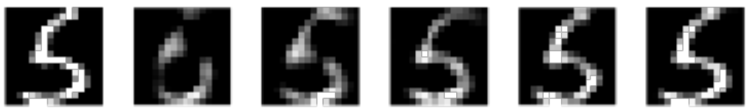

Figure 2.4: Recovery: one of digit 5

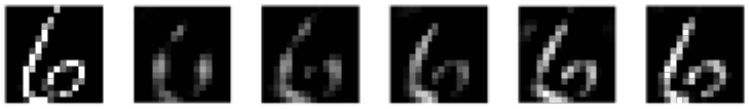
Figure 2.5: Recovery: one of digit 6

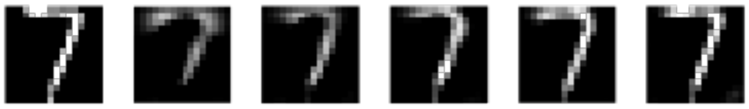
Figure 2.6: Recovery: one of digit 7

where β_i is the coordinate of d_i under the affine space:

$$\beta_i = \mathrm{U}^{\mathrm{T}}(d_i - \mu). \tag{2.2}$$

Five figures were randomly picked from the training set. From left to right, images in the figure are original ones and recoveries of them with $k = 7, 11, 17, 29, 55$.

From this experiments, when $k = 29$, the recovery of digits are very similar with the original one. It means that choosing $k = 29$ could save about 9/10 storage and the images are almost the same with the original data ($\mathrm{p}_{\mathrm{var}}(k) \geq 90\%$).

Part II

Classification

Given hand-written digit dataset, a natural question is how to recognize or predict which digits the image is. Dataset contains training set and its label, testing set and its label. In order to do validation, we divide training set into two part, the first 80% is new training set and the left 20% is validation set. Training set, validation set, testing set and their labels are denoted as $\mathrm{D} = [d_1, \ldots, d_{5832}] \in \mathbb{R}^{256\times 5832}, \mathrm{V} = [v_1, \ldots, v_{1459}] \in \mathbb{R}^{256\times 1459}, \mathrm{T} = [t_1, \ldots, t_{2007}] \in \mathbb{R}^{256\times 2007}$ and $\mathrm{Y}_{\text{train}} \in [0, 1, \cdots, 9]^{5832}, \mathrm{Y}_{\text{valid}} \in [0, 1, \cdots, 9]^{1459}, \mathrm{Y}_{\text{test}} \in [0, 1, \cdots, 9]^{2007}$.

3 Geometric Methods based on PCA

Principal Component Analysis has clear geometric meaning, which is to find a low-dimensional affine space to minimize the mean square error/distance of the original digits and their approximations. Inspired by this, we proposed two geometric methods for digits classification based on principal component analysis. And these two methods both have high performance in the testing set.

3.1 Angle Method

3.1.1 Geometric Explanation

Each point could be regarded as a vector. Intuitively, the angle between vector and subspace can be a good indicator to demonstrate the similarity of them.

In order to adopt the idea above, the first step is to centralize data with the mean of training set $\mu = \frac{1}{5832}\sum_{i=1}^{5832} d_i$ such that all vectors and subspaces share one common origin. Denote $\hat{\mathrm{D}}, \hat{\mathrm{V}}, \hat{\mathrm{T}}$ as centralized data, where $\hat{d_i} = d_i - \mu, \hat{v_i} = v_i - \mu, \hat{t_i} = t_i - \mu$. Then, $\hat{\mathrm{D}}$ is separated into 10 sets corresponding to 10 digits, as $\hat{\mathrm{D}}^0, \hat{\mathrm{D}}^1, \cdots, \hat{\mathrm{D}}^9$.

Given an appropriate k, we can foresee that the best approximate subspace of $\hat{\mathrm{D}}^0, \hat{\mathrm{D}}^1, \cdots, \hat{\mathrm{D}}^9$ lie in different directions. These k orthogonal directions of each subspace have explicit form as below:

$$\hat{\mathrm{D}}^i = \hat{\mathrm{U}}^i \hat{\Sigma}^i \hat{V}^{i\mathrm{T}}, \quad i = 0, \cdots, 9, \tag{3.1}$$

where equation 3.1 is singular value decomposition, the columns of $\hat{\mathrm{U}}^i \in \mathbb{R}^{256\times256}$ are an orthonormal basis of $\mathbb{R}^{256}$, these k orthogonal directions of each subspace is the first k columns of $\hat{\mathrm{U}}^i$.

Denote $\{\alpha_i\}_{i=1}^{256}$ as the angle between one vector u and an orthonormal basis $[u_1, \ldots, u_{256}]$. α is the angle between u and the k-dimensional subspace spanned by $[u_1, \ldots, u_k]$. They have relationships as below:

$$\sum_{i=1}^{256} \cos^2 \alpha_i = 1, \tag{3.2}$$

$$\cos \alpha_i = u_i^{\mathrm{T}} \frac{u}{\|u\|}, \tag{3.3}$$

$$\alpha = \arcsin \sqrt{1 - \sum\nolimits_{i=1}^{k} \cos^2 \alpha_i}. \tag{3.4}$$

Therefore, the smaller α is, the greater $\sum_{i=1}^{k} \cos^2 \alpha_i$ is.

Denote $\left\{\alpha_i^j\right\}_{i=1}^{256}$ as the angle between one vector u and an orthonormal basis $\hat{\mathrm{U}}^j = [u_1^j, \ldots, u_{256}^j], j = 0, 1, \cdots, 9$. The smallest angle $\tilde{\alpha}$ between u and $\left\{\mathrm{span}(u_1^j, \ldots, u_k^j)\right\}_{j=0}^{9}$ is

$$\alpha = \arcsin \sqrt{1 - \sum\nolimits_{i=1}^{k} \cos^2 \alpha_i^{\tilde{j}}}, \tag{3.5}$$

$$\tilde{j} = \arg\max_j \sum_{i=1}^{k} \cos^2 \alpha_i^j \tag{3.6}$$

Since u has the smallest angle with the subspace $\mathrm{span}(u_1^{\tilde{j}}, \ldots, u_k^{\tilde{j}})$, vector u has the highest similarity with this subspace rather than the others. Reasonably, we can assign $\tilde{j}$ as the label of u.

3.1.2 Experiments

We implements angle method on $(\hat{\mathrm{D}}, \mathrm{Y}_{\mathrm{train}}), (\hat{\mathrm{V}}, \mathrm{Y}_{\mathrm{valid}}), (\hat{\mathrm{T}}, \mathrm{Y}_{\mathrm{test}})$. And the prediction accuracy of training, validation, testing set is showed in the figure 3.1. $k = 16$ has the highest prediction accuracy 97.62% on validation set. When $k = 16$, the testing prediction accuracy is pretty good 93.87%. The best k of testing set is 29, which has 94.42% prediction accuracy close to 93.87%, and the distance between 16 and 29 is pretty small compared with 256, the total dimension of data .

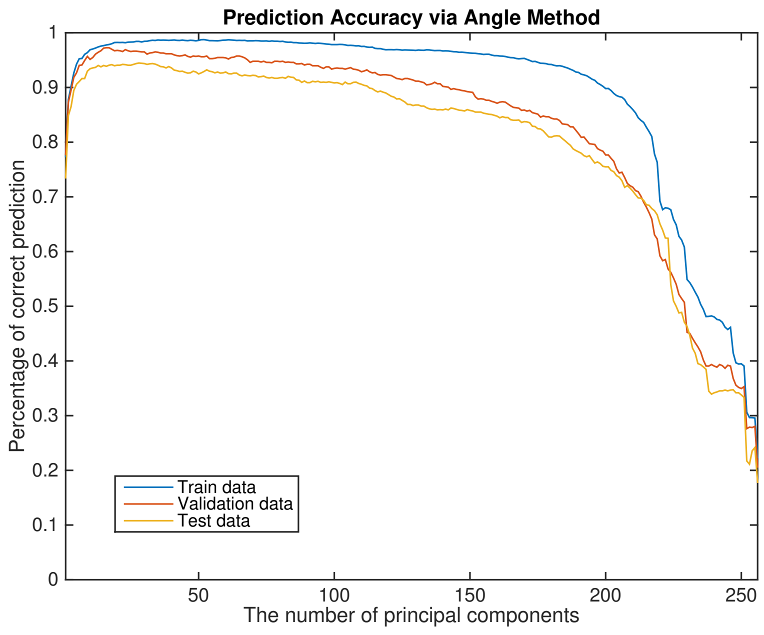

Figure 3.1: Prediction Accuracy via Angle Method

3.2 Distance Method

3.2.1 Geometric Explanation

Analogically, D is separated into 10 sets corresponding to 10 digits, as $\mathrm{D}^0, \mathrm{D}^1, \cdots, \mathrm{D}^9$. Given an appropriate k, by the virtue of principal component analysis, each set D^j has its own best approximate affine space, denoted as $\mathbf{S}^j = \{\mu^j + \mathrm{U}^j\beta | \beta \in \mathbb{R}^k\}$, where μ^j is the average of $\mathrm{D}^j = [d_1, \ldots, d_{n^j}]$,

$$\mu^i = \frac{1}{n^i} \sum_{i=1}^{n^j} d_i, \tag{3.7}$$

and U^j is the first k columns of $\hat{\mathrm{U}}^j$. $\hat{\mathrm{U}}^j$ is the left singular matrix of $\hat{\mathrm{D}}^j$,where

$$\hat{\mathrm{D}}^j = \hat{\mathrm{U}}^j \hat{\Sigma}^j \hat{\mathrm{V}}^{j\mathrm{T}}, \tag{3.8}$$

$$\hat{d}_i^j = d_i^j - \mu^j, \quad \hat{\mathrm{D}}^j = [\hat{d}_1^j, \ldots, \hat{d}_{n^j}^j]. \tag{3.9}$$

Naturally, the distance l between a vector u and an affine space $\mathbf{S} = \left\{\mu + \mathrm{U}\beta | \beta \in \mathbb{R}^k\right\}$ can indicate the similarity between them. The smaller l is, the more possible u belongs to the class corresponding to the affine space.

Given a vector u, the distance between a vector u and the affine space, $\mathbf{S}^j = \left\{\mu^j + \mathrm{U}^j\beta | \beta \in \mathbb{R}^k\right\}$, is

$$l = \sqrt{\sum\nolimits_{i=k+1}^{256} u^{\mathrm{T}} \hat{u}^j}, \tag{3.10}$$

where $\mathbf{S}^j = \mathrm{span}^{\perp}(\hat{u}^{k+1}, \ldots, \hat{u}^{256})$. The label prediction of u is

$$\tilde{j} = \arg\min_j l_j = \arg\min_j \sqrt{\sum\nolimits_{i=k+1}^{256} u^{\mathrm{T}} \hat{u}^j}. \tag{3.11}$$

3.2.2 Experiments

We implements angle method on $(\mathrm{D}, \mathrm{Y}_{\text{train}}), (\mathrm{V}, \mathrm{Y}_{\text{valid}}), (\mathrm{T}, \mathrm{Y}_{\text{test}})$. And the prediction accuracy of training, validation, testing set is showed in the figure 3.2. $k = 13$ has the highest prediction accuracy 97.46% on validation set. When $k = 16$, the testing prediction accuracy is pretty good 94.17%. The best k of testing set is 30, which has 94.77% prediction accuracy close to 94.17%, and the distance between 13 and 30 is pretty small compared with 256, the total dimension of data .

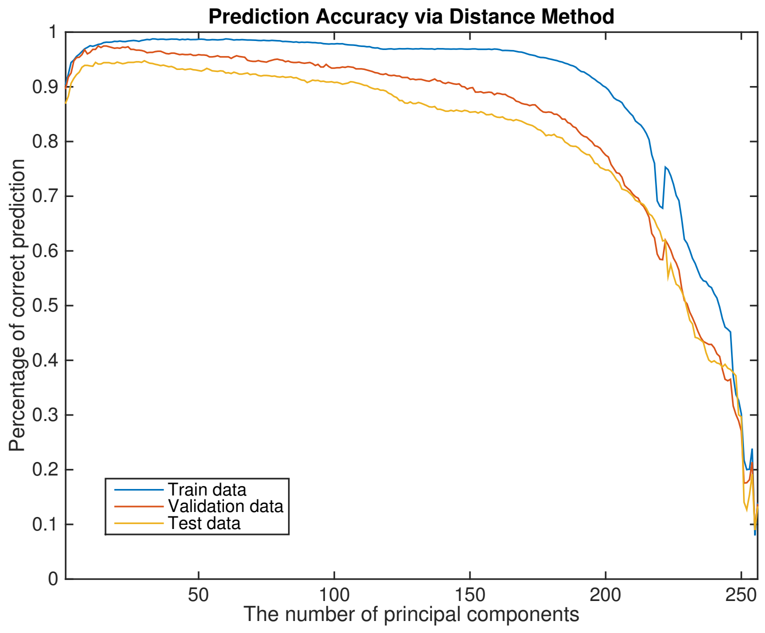

Figure 3.2: Prediction Accuracy via Distance Method

3.3 Discussion

This part, we will talk about the time complexity of these two algorithms and how to choose one of them with different k and then deal with one phenomenon in angle method.

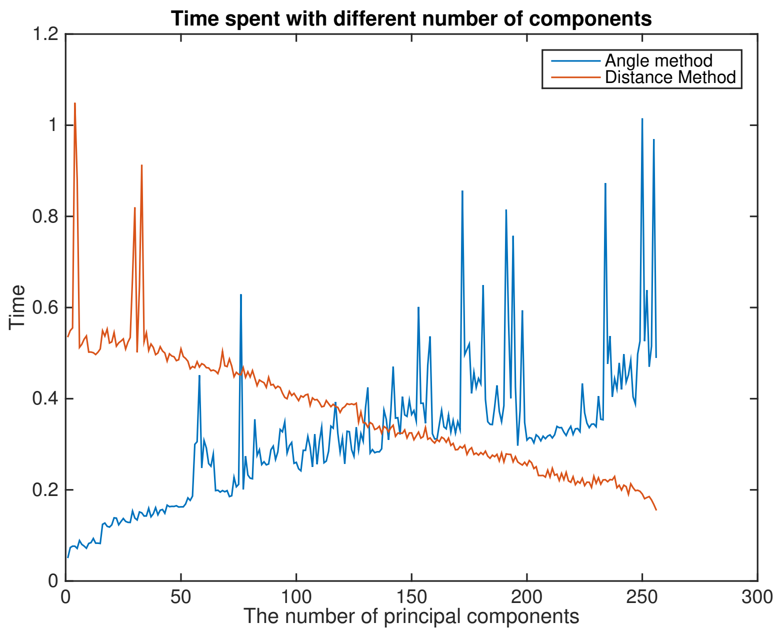

Figure 3.3: Time spent via Angle Method and Distance Method

These two geometric methods are based on principal component analysis. The key part of PCA is singular value decomposition. In our case, the time spent in singular value decomposition is less then 0.5 second. Roughly speaking, the process of principal component analysis could be regarded as training phase in machine learning field. The time spent in prediction of all training set, validation set and testing set are showed in figure 3.3.

No matter what k is, the time spent in prediction phase is always less than 1.5 seconds. Compared with classification algorithms in the following section, geometric methods in this section are time-saving and these two simple methods have even better performance than most of algorithms. Perhaps this is where mathematical beauty glows.

As k increases, time spent in angle method goes up and time spent in distance method goes down, which could be easily understood from equation 3.4 and equation 3.10. Hence, it is better to choose angle method when k is small and select distance method when k is large.

During experiments, there was an interesting but confusing phenomenon. In the angle method, the first step is centralization. In report, we used the mean of training data μ, while it seems that no matter which point we select to centralize the data, performance was very stable. For example, the prediction accuracy of the cases without centralization and with $\mu = 10 \cdot \mathbf{1}^{\mathrm{T}}$, which is comparably much larger than data whose features range from $[-1, 1]$, was even higher than centralization with the mean of training data by around 0.5%.

4 Logistic Regression

4.1 Brief Introduction

This section has two main purpose: one is to present the necessity of principal component analysis as feature transformation, the other is to compare with the geometric methods and show how good prediction performance angle method and distance method achieve with ignorable time in training and prediction phases.

Briefly speaking, logistic regression is to maximize the likelihood of observations with sigmoid function in equation 4.1, turning linear combination $w^{\mathrm{T}}x$ to probability, where x is features and w is the parameters.

$$h(w^{\mathrm{T}}x) = \frac{e^{w^{\mathrm{T}}x}}{1+e^{w^{\mathrm{T}}x}}, \tag{4.1}$$

4.2 Experiments

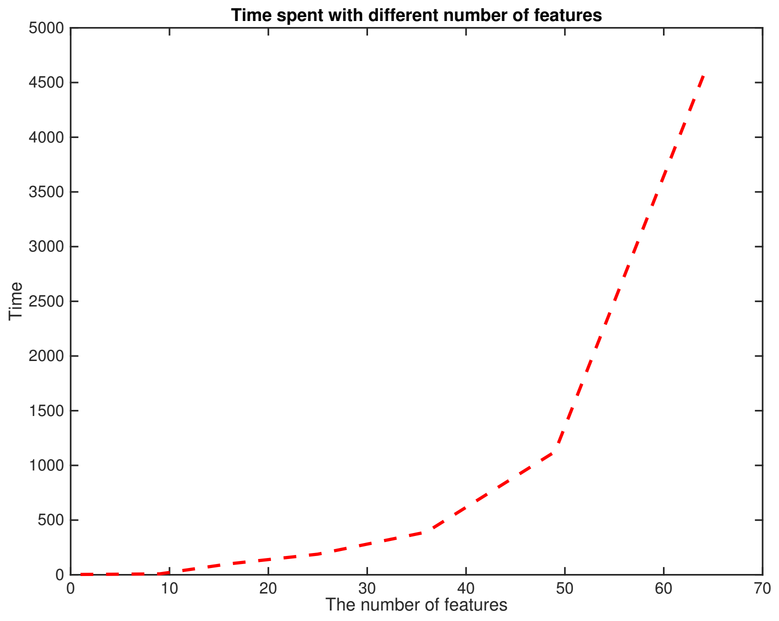

Figure 4.1: Time spent via Logistic Regression

Logistic regression doesn't have explicit and closed solution. Different optimization algorithms are adopted to tune parameters w, such as gradient descent (GD) and stochastic gradient descent (SGD), which will consume more time as the dimension of features increases.

Since principal component analysis conserves the "energy"/variance or data as much as possible, the coordinate of original data under k-dimensional affine space could be regarded as a good feature transformation from 256 dimension space to k dimension space. We picked some k to run logistic regression algorithm on Matlab. And the time spent (second) in training

k	1	4	9	16	25	36	49	64
train	41.22%	71.13%	86.30%	92.42%	95.80%	95.64%	96.16%	96.95%
valid	38.04%	72.24%	85.26%	91.57%	94.31%	94.11%	94.17%	94.31%
test	40.81%	69.31%	81.96%	88.39%	90.48%	90.43%	89.89%	89.89%

Table 4.1: Prediction accuracy of logistic regression, k is the number of features

phase with different number of features are showed in figure 4.1. It is clear that time spent exploded as the number of feature increases. It took more than 1 hour even only 64 features were selected.

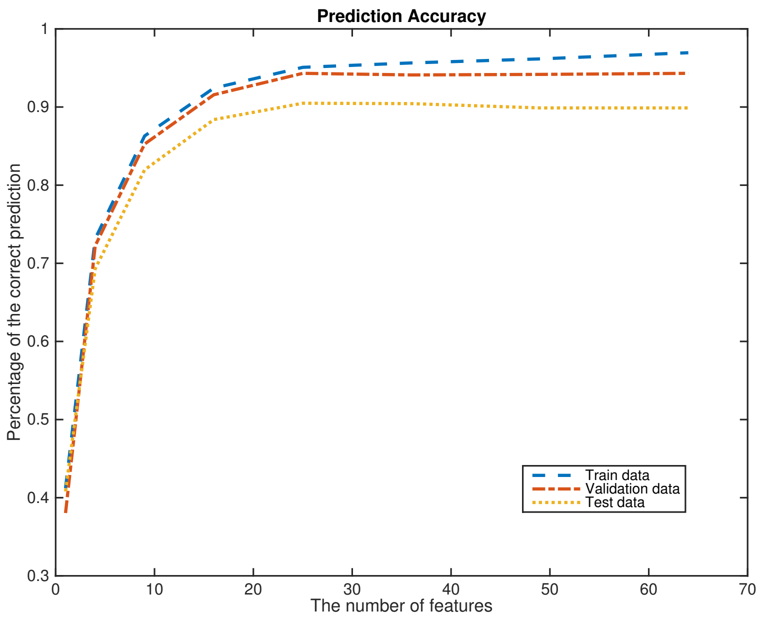

Figure 4.2: Prediction Accuracy via Logistic Regression

Even thought logistic regression required such long time in training phase, the performance was not every desirable, which was recorded in figure 4.2 and table 4.1.

5 Support Vector Machine (SVM)

In this section, the LIBSVM was utilized to train the model. LIBSVM is an advanced SVM library developed by Prof. Chih-Jen Lin from National Taiwan University[CL11].

Firstly, the raw data without PCA were used to train the LIBSVM model. Since there are 256 features, linear kernel is enough to build a reliable model with high accuracy. After experiments, KBF kernel was proved to lead to overfitting easily in this case. After using the grid search method to find the best c parameter, where c is the parameter controlling the rate

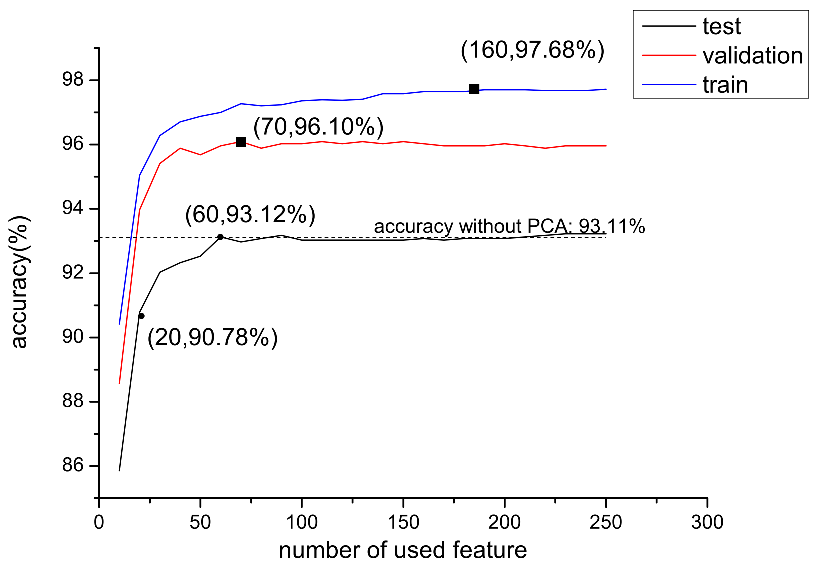

Figure 5.1: Prediction accuracy of SVM

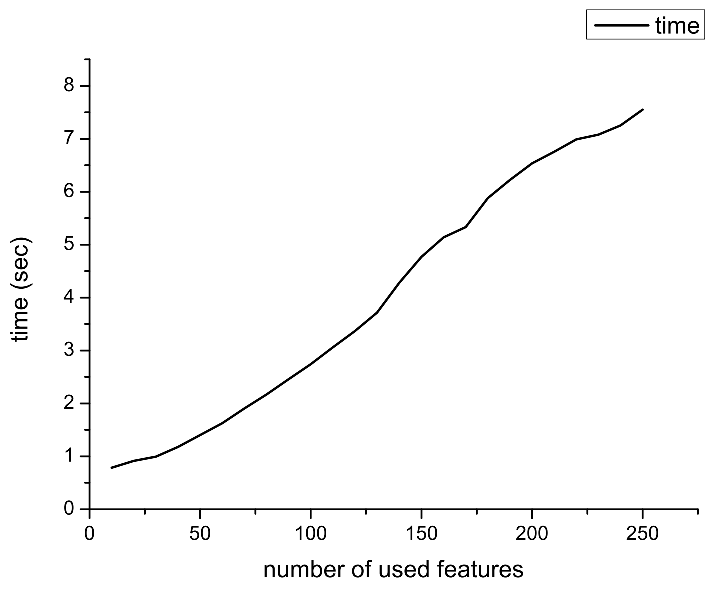

Figure 5.2: Time spent of SVM

of regularization, the highest test accuracy is 93.11%, and the training accuracy and validation accuracy are 98.87% and 96.05% respectively.

For data conducted the principal component analysis (PCA), time consumption was analyzed. After PCA, the features are arranged by the importance. 25 models were built to study the accuracy and the time consumption using different dataset. The first model only contains the first 10 features, and the second one contains the first 20 features and so on and so forth.

The result shows in figure 5.1. The test accuracy of the model built by the 20 most important features gets 90.78%, and the test accuracy of the model built by the first 60 features is 93.1%, which is almost equal to the accuracy of model built by all the 256 features. As for the time consumption, the model utilizing 60 features takes only 1.62 seconds, while the model using all the features takes 7.53 seconds. Compared to the model train by raw data, only 20% time was used by the model trained by PCA data, and they have the same accuracy.

6 Neural Networks (NNs)

Neural Networks are engineered models originally inspired by biological brain but have since evolved and achieved tremendous engineering success in applications of visual recognition since 2012[KSH12]. It is proved that Neural networks with at least one hidden layer are universal approximators [Cyb89] for any continuous functions. They are a good candidate for learning from data because of their strong learning power.

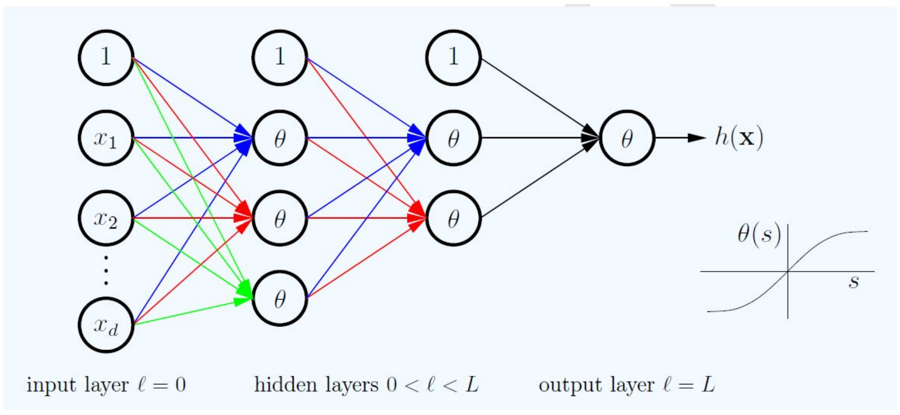

Figure 6.1: A 3-layer Neural Network [AMMIL12]

6.1 Introduction of Neural Networks

A graph representation of a feed forward neural network is showed in Figure 6.1 [AMMIL12]. This is a 3-layer Neural Network with two hidden layers and one output. The neural network hypothesis $h(x)$ is computed through forward propagating. We first initialize the input layer ($l = 0$) with training data. To get the input vector into other layers l ($l > 0$), we compute the weighted sum of the outputs from the previous layer, with weights specified in $W^{(l)}$:$s_j^{(l)} = \sum_{i=1}^{d^{l-1}} w_{ij}^{l} x_i^{l-1}$, which can be vectorized calculated by matrix multiplication:

$$s^{(l)} = (W^{(l)})^{\mathrm{T}} x^{(l-1)}. \tag{6.1}$$

After forward propagation, the output vectors $x^(l)$ at every layer $l = 0, \cdots, L$ are computed.

Then we should computer the loss function $\mathrm{E}_{in}(w)$ by comparing the difference between $h(x_n)$ and the ground truth y_n. For the sum of squares,

$$\mathrm{E}_{in}(w) = \frac{1}{N}\sum_{n=1}^{N}(h(x_n; w) - y_n)^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n^{(L)} - y_n)^2. \tag{6.2}$$

After getting the loss function, we then discuss how to minimize it to obtain the learned weights.

We here consider the sigmoidal multi-layer neural network with $\theta(x) = \tanh(x)$. To minimize the loss function, we apply the gradient descendant method to the resulting loss function. The gradient with respect to $E_{in}(W)$ is calculated by:

$$\frac{\partial \mathrm{E}_{in}(W)}{\partial W^l} = \frac{1}{N}\sum_{n=1}^{N}\frac{\mathrm{d}}{\mathrm{d}h}e(h(x_n), y_n)\frac{h(x_n)}{W^{(l)}}. \tag{6.3}$$

6.2 Experiment Setup

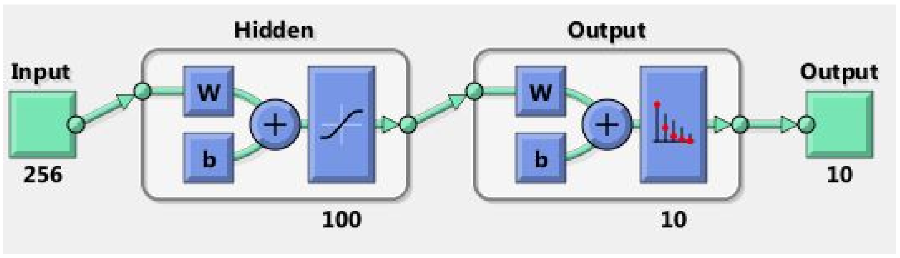

Figure 6.2: A typical Neural Network of our experiment

We explored the application of 3-layer neural network on the handwriting digits number dataset. Theoretically a large enough 3-layer fully connected network can approximate any continuous function arbitrarily well. Here we explore the influence of the size of hidden layers on the final prediction accuracy. To test the effect of PCA preprocessing before feeding the data into the network, we compare the accuracy with different input data. Figure 6.2 depicts a typical Neural Network used in our experiment. Figure 6.3 and figure 6.4 show the experiment results. Comparing these two figures, prediction accuracy increases a little after processed by PCA.

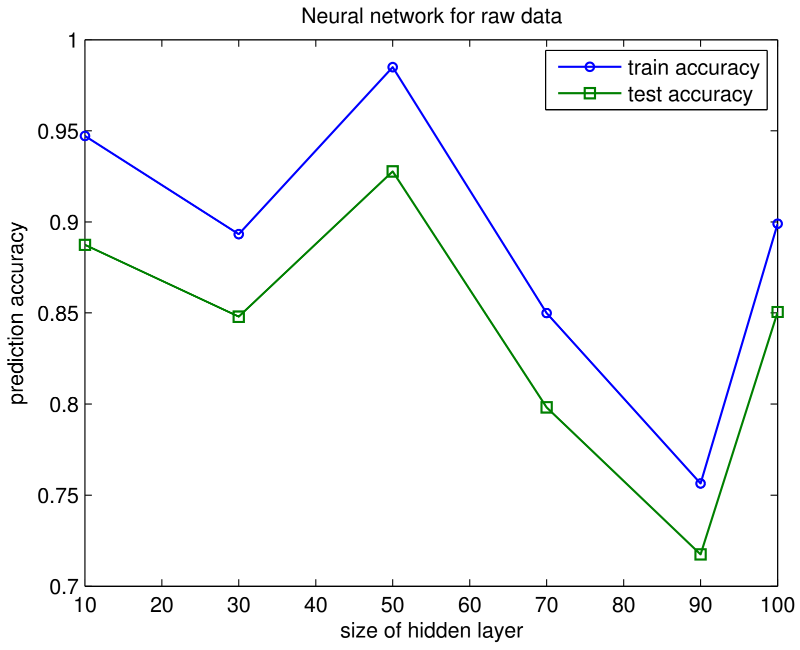

Figure 6.3: Prediction accuracy of Neural network on raw data

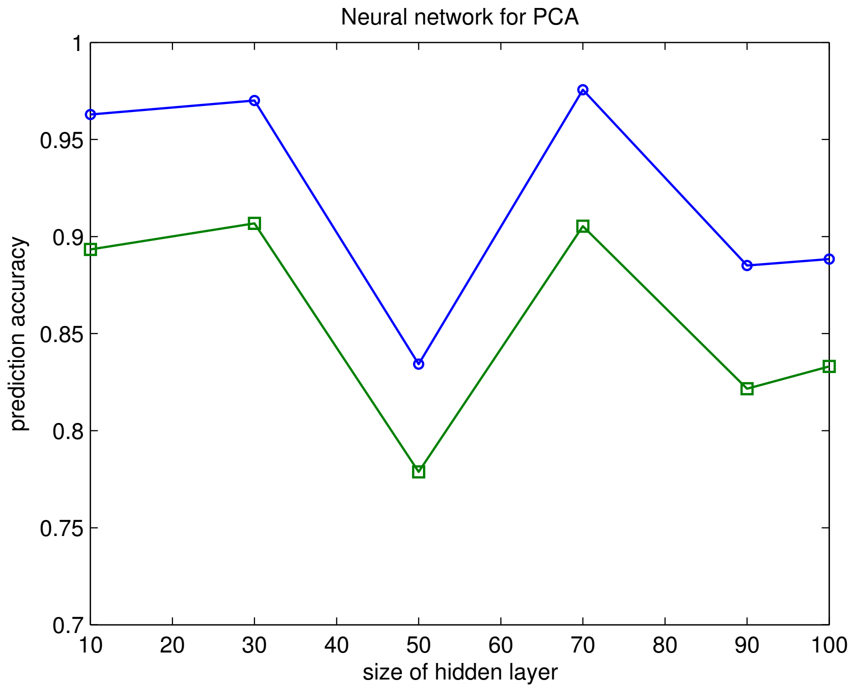

Figure 6.4: Prediction accuracy of Neural network on PCA-processed data

References

[AMMIL12] Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning from data*, volume 4. AMLBook New York, NY, USA:, 2012.

[CL11] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[Cyb89] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.

[Hot33] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[Pea01] Karl Pearson. Principal components analysis. *The London, Edinburgh and Dublin Philosophical Magazine and Journal*, 6(2):566, 1901.