Handwritten digit recognition through PCA and Lasso method

Fang Linjiajie	Xiao Ziliang	Xue Zexiao	Zhang Wenyong
20382284	20378386	20403674	20377289

1 Problem

How to recognize a handwritten digit through PCA and Lasso method?

2 Models and Inference

We firstly get data from the webpage and split it into two parts. We randomly picked seventy percentage to be training data and the rest of it to be test data. And use PCA on training data to find the main principal components of ten different digits. In order to classify the test data into one digit group, we use the first several principal components as independent variables and simulate the test data by linear combination of these principal components. We make use of the Lasso method or Linear regression to realize the classification. Finally, we use the test data to find the accuracy of the classification.

There are several questions during the process of solving the problem.

1. How many principal components should we use?
2. Whether using Lasso method or Linear regression?
3. If we choose to use Lasso method, which lambda should we choose?
4. What is the accuracy of this classification?

2.1 Data Split

We aim to pick seventy percentage data to be training data. It is not reasonable to pick first seventy percentage as training data. So we randomly pick data into training dataset until the percentage reaches seventy percentage and leave the rest data as test data.

2.2 PCA

PCA is among the most popular statistical tools applied in data analysis and many other disciplines. We acquire about one thousand training data from each digit. Using PCA method can reduce the dimension of existing samples and at same time, it can also maintain

as much information in the data as possible. We want to find the principal components of each digit which can explain more than eighty percentage variance provided. The following are the number of principal components required to explain up to about eighty percentage contribution of total variance.

digit	0	1	2	3	4	5	6	7	8	9
# of PC	17	8	30	29	26	27	20	18	28	19
contribution	0.808	0.808	0.801	0.804	0.806	0.806	0.807	0.804	0.801	0.805

Table 1 : number of principal components which can explain 80% of variance

As the Table 1 shows, in order to find the similar contribution of the principal components, the number of principal components varies from eight to thirty. On account of the complexity of different digits, the number of principal components contributing eighty percentage of the variance exists big difference. For instance, there just need eight components for digit 1, since 1 does not change much written by different people. When we do it in programming, we use forty principal components for each digits for convenience. The following shows the first twenty five principal components of the digit 9, the contribution of the PC, and the centralized figure.

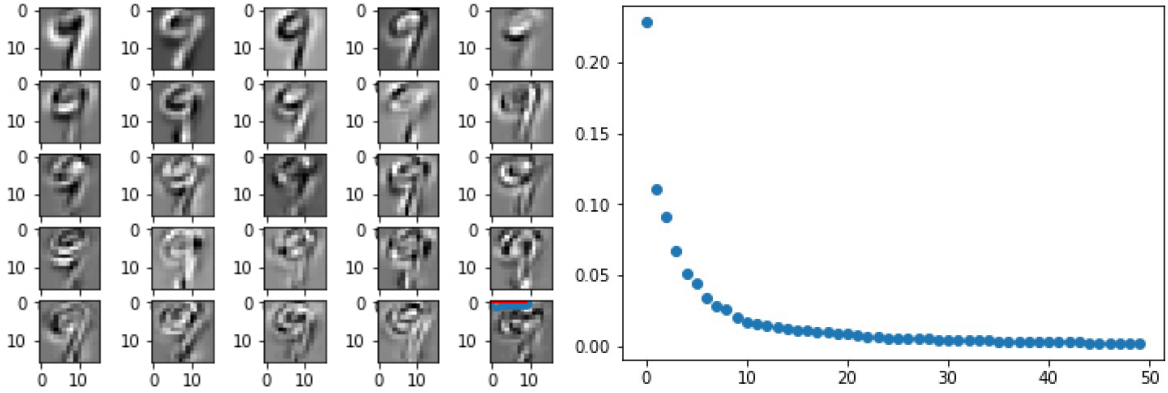

Figure 1: First 25 principal components of digit 9

The PCA method can be effective to find common factors in each training image. For instance, when we look at Figure 2, the first 25 principle components explain most variance except mean image 9.

2.3 Lasso

Principal components store over 80% information of certain digital images and operate dimension reduction from 256-dimension to about 20-dimension without losing so much

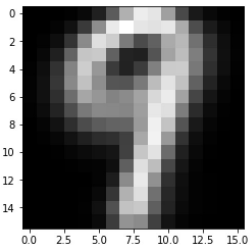

Figure 2: Centralized figure

information. An idea about classification with respect to principal components would be very reasonable because typical shape of numbers are mostly explained by the reduced affine plane.

After principal component analysis, given an undetermined image X, say, an image of hand-written 7, can be approximately explained by linear combination of principal components from the corresponding number-7, i.e.

$$X = \mu + \sum_{l=1}^{k} a_l PC_l + \epsilon.$$

where $||\mu||$ is the mean of number-7 images and turns out to be a constant vector. An image being similar to 7 will be very close to the affine plane spanned by principal components and $||\mu||$, which leads to relatively small $||\epsilon||$.
A technique to build regression problem is to consider each pixel of our given image to be explained by pixels from principal components. The sample size for creating regression is 256, i.e. (here, μ_0 is a scalar rather than a matrix)

$$X_{ij} - \mu_{ij} := Y_{ij} = \sum_{l=1}^{k} a_l PC_{l,ij} + \epsilon \quad i, j = 1, ..., 16$$

in vector form, it is simple linear regression without considering intercepter:

$$Y_n = \sum_{l=1}^{k} a_l PC_{l,n} + \epsilon \quad n = 1, ..., 256$$

A desirable result for correct classification leads to higher R^2 since more information of pixels are explained by linear model. Higher R^2 can also be interpreted by smaller noise by $||\epsilon||$. After operating regression with each number's principal components, we treat the highest R^2 as the signal for right classification:

$$G(X) = \arg\max_k (R_k^2),$$

here R_k^2 relates to linear regression from principal components of number-k.

A simple linear regression considers all the variables and may contain nuisance variables that do not reflect important features that characterize the image. Thus L_1-penalty from Lasso would be preferred for variable reduction. Empirical attempts also give results that Lasso works better than simple linear regression (showing later). The Lasso regression takes the form:

$$\arg\min_a ||Y_n - \sum_{l=1}^{k} a_l PC_{l,n}||_2 + \lambda||a||_1 \quad n = 1, ..., 256$$

In order to find a fitted λ, we tried λ from 0 to 0.3. However, after considering the magnitude of the centralized data, around 10^{-2}. We finally decide to choose $\lambda = 0.002$ to penalize the parameters. The results verify that $\lambda = 0.002$ is the best selection which make a significant difference between lasso and linear regression.

After Lasso regression, we choose the largest R^2 to classify the given image to a certain group. Since R^2 represents the percentage of variance explained by regression model, a higher R^2 is prefered. In which,

$$R^2 = 1 - \frac{||y - \hat{y}||^2}{||y - \bar{y}||^2}$$

3 Results

As the figures show, the vertical axis is the R^2 and the horizontal axis is the digit. So the figures show the R^2 of different input when using Lasso method or Linear regression for classification.

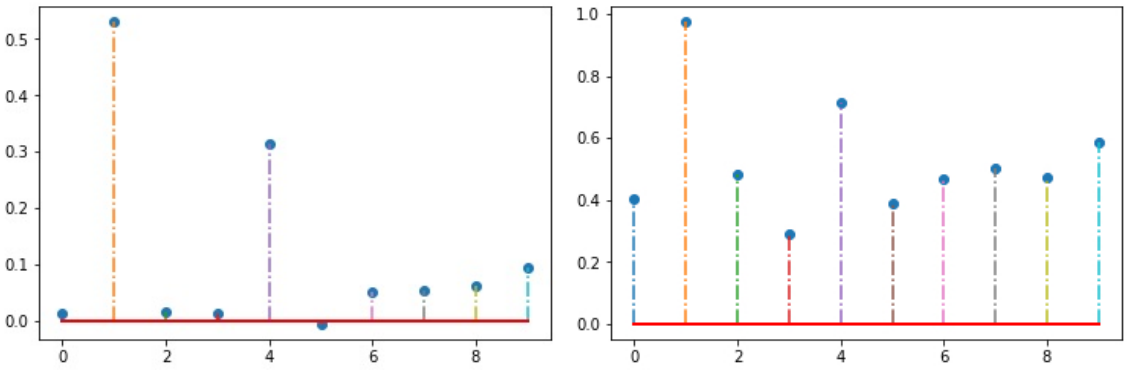

Figure 3: Lasso method($\lambda = 0.002$) and Linear regression for input 1

digit	0	1	2	3	4
Lasso R^2	0.0145	0.5296	0.0162	0.0145	0.3139
Linear R^2	0.4024	0.9740	0.4802	0.2909	0.7117
digit	5	6	7	8	9
Lasso R^2	-0.005	0.0518	0.0543	0.0629	0.0957
Linear R^2	0.3891	0.4679	0.5039	0.4724	0.5882

Table 2 : R^2 of digit 1

We first choose digit 1 for easy checking, because 1 is the easiest one to recognize. The input hand written digit 1 is the 10_{th} row of test data. In figure 3, the left figure is the result for Lasso method with $\lambda = 0.002$. The R^2 of 1 is above 0.5, much more higher than other digits. The second larger R^2 comes from digit 4. We find that the R^2 of the digit 4 is slightly bigger than half of R^2 of digit 1, as we have known that both these two have same vertical chirography. So we can get this information quantitatively from lasso method.

Compared with the right figure of figure 3, using Lasso method has strong advantage. Although Linear regression gives the right classification, the output R^2 are similar. The R^2 of digit 1 is 0.97 and the R^2 of digit 4 is 0.71, and one can easily see that it has relatively tiny difference. Besides, the R^2 of other digit is also relatively large, so we cannot make strong decision of classification. Because Lasso method can ignore some nuisance PCs to reduce their contribution.

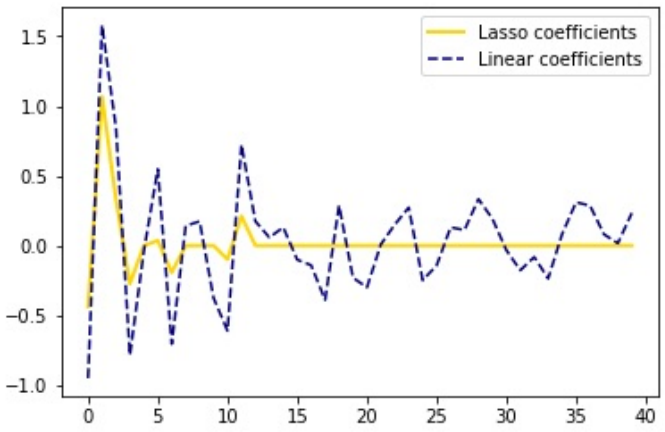

Figure 4: The coefficient of the regression for input 1 and PC1

In figure 4, the yellow line shows the coefficients of top forty principal components when using lasso regression, and the blue dash line shows the coefficients when using linear regression. It is obvious that the yellow line become zero after 12_{th} principal components,

because lasso could be used for variable selection.Compared with the yellow line, the dash line oscillates a lot and all variables are involved.

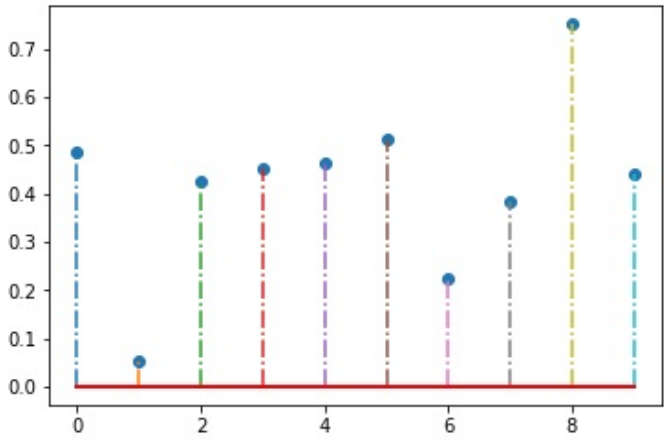

Figure 5: Lasso method($\lambda = 0.002$) for input 8

And then we try a relatively tough one: digit 8. Compared with other digit, digit 8 seems like digit 0, digit 3, digit 6 and digit 9. It is probably the most difficult one to classify. The test data is random picked from the training data. The result shows in the figure 5 using the Lasso method with $\lambda = 0.002$. The R^2 of digit 8 is over 0.7, which is also high as last time we try digit 1. But this number has less preponderance when contrast with the R^2 of other digits. We can see that the R^2 of digit 1 is near 0, and the R^2 of digit 0, 2, 3, 4, 5, 9 are at the same level as 0.4. So there exists more difficulties in classifying digit 8 than digit 1.

4 Analysis of Model Accuracy

After building our classification model, we applied our model to check each image of each digit. The results give us the accuracy for correct classification for each digital. The accuracy of each number is defined by the rate of correct classification of certain number to corresponding group among all attempts:

$$Accuracy\ Rate_k = \frac{\sum_{x_i \in N_k^{test}} I(\arg\max_l(R_l^2(x_i)) = k)}{N_k^{test}}$$

Our classification method works really well since all the empirical outcomes give us considerable high rate of accuracy. All the accuracy is above 95 percentage except digits 3 and 4. It is reasonable since handwritten images of 4 vary a lot due to relatively complex handwriting, and handwritten images of 3 is always confusing for similarity to 8 and 5. The accuracy of 1 reaches extremely high 100 percentage, which concludes that the handwriting of 1 does not differ much since the alone vertical line of 1 gives strong feature to distinct it from other digits.

digit	0	1	2	3	4
Accuracy Rate	0.9888	1.0000	0.9909	0.9394	0.9388
digit	5	6	7	8	9
Accuracy Rate	0.9760	0.9750	0.9639	0.9755	0.9742

Table 3 : Accuracy rate for different digits

5 Conclusion

After solving the question we raised on, we make following conclusions:

The PCA method can be effective to find common factors in each training data. Upon to about 20_{th} principal components, the variance can be explained above eighty percentage.

Lasso method has more advantages than linear regression. From regression between the test data and principle components from each group (For instance, we use one matrix which is on behalf of digit 1 as the test data to find its relationship with principle components of different digits), we can easily find that lasso method can distinguish more accurately from other digits than linear regression because the significance of R^2 of the digits due to Lasso is much larger than simple linear regression.

The lasso method can not only make the relatively accurate classification, but also tell us more details about similar characterization among different digit pictures. Linear regression, however, have much less ability to mine such detail information of such characteristics.

Our classification model built by PCA and related pixel-wise linear regression works really well on testing data. The accuracy rates are all above ninety percentage. However, in order to satisfy practical application, we need to make calibrations to add significance statement to pick out confusing image as ?unknown? for further artificial recognition. The tradeoff of sacrificing efficiency may lead to ideal accuracy. A simple rule for judgement significance statement could be the difference between the largest two R^2, a relatively small difference of R^2 is preferred as uncertainty.

In conclusion, one can use PCA to obtain common factors of training data and then apply the lasso method to recognize the test digit data with more accuracy.

6 Data Source

Handwritten digits

Sources: http://statweb.stanford.edu/ tibs/ElemStatLearn/datasets

Data description: normalized handwritten digits, automatically scanned from envelopes by the U.S. Postal Service. The original scanned digits are binary and of different sizes and orientations; the images here have been descanted and size normalized, resulting in 16 x 16 grayscale images (Le Cun et al., 1990).