AMATH 482/582: HOME WORK 2

SHENGBO JIN

CFRM Program, University of Washington, Seattle, WA
Shengboj@uw.edu

Abstract. This report aims at train a classifier to distinguish images of handwritten digits. The main method is PCA and Ridge regression. Firstly, use PCA to reduce the dimensions of original data set. Then, using reduced data, fit the Ridge regression to predict the digit in three scenarios. At last, evaluate classifier performances and explore their correlation with PCA results.

1. Introduction and Overview

Our goal is to train a classifier to distinguish images of handwritten digits from the famous MNIST data set. The data set is split into training and test sets. The training set contains 2000 instances of handwritten digits, the "features" are 16×16 black and white images while the "labels" are the corresponding digit. In addition, before training classifier, apply PCA to the original data set and extract out the new concentrated "features".

In this report, under Python environment, PCA and Ridge regression are realized by NumPy[1] and Sklearn[3] package. Matplotlib[2] is used for visualization.

2. Theoretical Background

Principal component analysis (PCA) is the process of computing the principal components and using them to perform a change of basis on the data, sometimes using only the first few principal components and ignoring the rest.

PCA is defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some scalar projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.

Consider an $n \times pn\times$ p data matrix, X, with column-wise zero empirical mean (the sample mean of each column has been shifted to zero), where each of the n rows represents a different repetition of the experiment, and each of the p columns gives a particular kind of feature (say, the results from a particular sensor).

Mathematically, the transformation is defined by a set of size l of p-dimensional vectors of weights or coefficients $\mathbf{w}_{(k)} = (w_1, \ldots, w_p)_{(k)}$ that map each row vector $\mathbf{x}_{(i)}$ of X to a new vector of principal component scores $\mathbf{t}_{(i)} = (t_1, \ldots, t_l)_{(i)}$, given by

$$t_{k(i)} = \mathbf{x}_{(i)} \cdot \mathbf{w}_{(k)} \qquad \text{for} \qquad i = 1, \ldots, n \qquad k = 1, \ldots, l$$

in such a way that the individual variables $t_1, \ldots, t_l$ of t considered over the data set successively inherit the maximum possible variance from X, with each coefficient vector w constrained to be a unit vector (where l is usually selected to be strictly less than p to reduce dimensionality).

Ridge regression is a method of estimating the coefficients of multiple-regression models in scenarios where linearly independent variables are highly correlated. It has been used in many fields including econometrics, chemistry, and engineering.

Date: February 11, 2022.

In standard linear regression, an $n \times 1$ column vector y is to be projected onto the column space of the $n \times p$ design matrix X (typically $p \ll n$) whose columns are highly correlated. The ordinary least squares estimator of the coefficients $\beta \in \mathbb{R}^{p \times 1}$ by which the columns are multiplied to get the orthogonal projection $X\beta$ is

$\widehat{\beta} = (X^T X)^{-1} X^T y$ (where X^T is the transpose of X).

By contrast, the ridge regression estimator is

$\widehat{\beta}_{\text{ridge}} = (X^T X + kI_p)^{-1} X^T y$ where I_p is the $p \times p$ identity matrix and $k > 0$ is small. The name 'ridge' refers to the shape along the diagonal of I.

3. Algorithm Implementation and Development

Step 1

1. i) Initialize *PCA()* as pca
   ii) Use *pca.fit()* to find all 256 components (same to the original data dimensions)
   iii) Plot the singular values with *plt.plot()* and *pca.signular_values_*
   iv) Plot the explained variance ratio by *pca.explained_variance_ratio_* and *plt.plot()*
2. i) Define *plot_digits()* function to help visualize data
   ii) Plot the first 16 PCA modes as 16×16 images

Step 2

1. i) Use *np.cumsum()*, *np.sum()* and *pca.singular_values* to calculate the Frobenius norm
   ii) With *np.where()*, find the number of PCA modes to approximate X_{train} up to 60%, 80% and 90% in the Frobenius norm
2. i) Define *reconstruction()* function to get the approximated data using *pca.fit()*, *pca.transform* and *pca.inverse_transform()*
   ii) Reconstructed data by different numbers of modes by *reconstruction()*
   iii) Visualize them using *plot_digits()*

Step 3

1. i) Initialize *PCA(16)* to keep 16 componets as pca
   ii) Extract the features and labels of the digits 1 and 8 from train set and test set with *np.where()*
   iii) Use *pca.transform()* to get Atrain matrix
   iv) Use *np.where()* to get Btrain vector
2. i) Initialize *RidgeCV()* with cross validation as classifier to choose best alpha
   ii) Fit the Ridge regression with Atrain and Btrain by *classifier.fit()*
   iii) Use *classifier.predict()* and *np.where()* to get predicted values and classified values

Step 4

1. Use *mean_squared_error()* to calculate mean squared error
2. Use *confusion_matrix ()* to calculate confusion matrix
3. Visualize the weights of beta using *plt.plot()* and *classifier.coef_* and find the most important mode by *np.argmax()*
4. Visualize where the 1's and 8's live in the PCA space using *pca.transform()* and *scatter3D()*

4. Computational Results

Task 1

Fit PCA with the original data and plot singular value spectrum with all 256 dimensions.

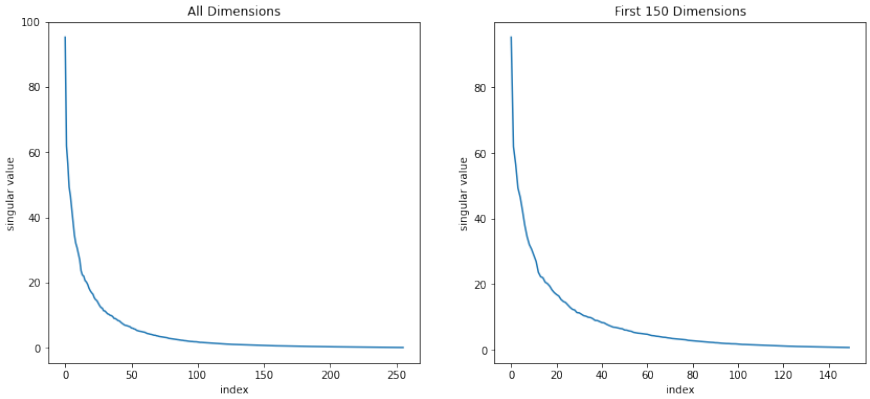

FIGURE 1. Singular value spectrum

To a large extent, the effective dimension of the data appears to be 100.

From the other side, before using 100 components, the explained variance ratio has been larger than 0.95.

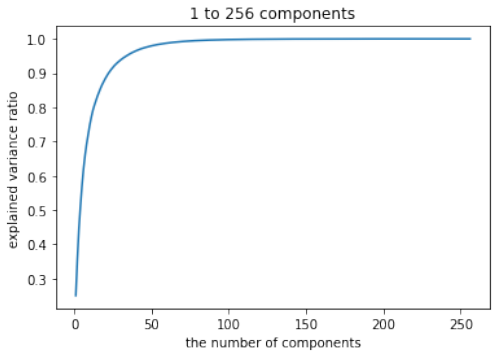

FIGURE 2. Explained variance ratio

Plot the first 16 PCA modes as 16×16 images. We can see the first 16 concentrated features from the entire data set.

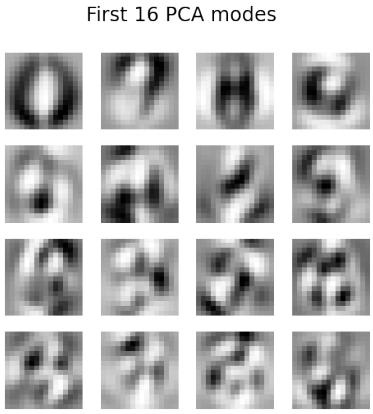

FIGURE 3. First 16 PCA modes

Task 2

By calculating the Frobenius norm, we can find the number of PCA modes to approximate Xtrain up to 60%, 80% and 90%

approximation up to	60%	80%	90%
number of modes	3	7	14

TABLE 1. Frobenius norm

Do the different low rank reconstructions by 3, 7, 16, 64 and 128 components.

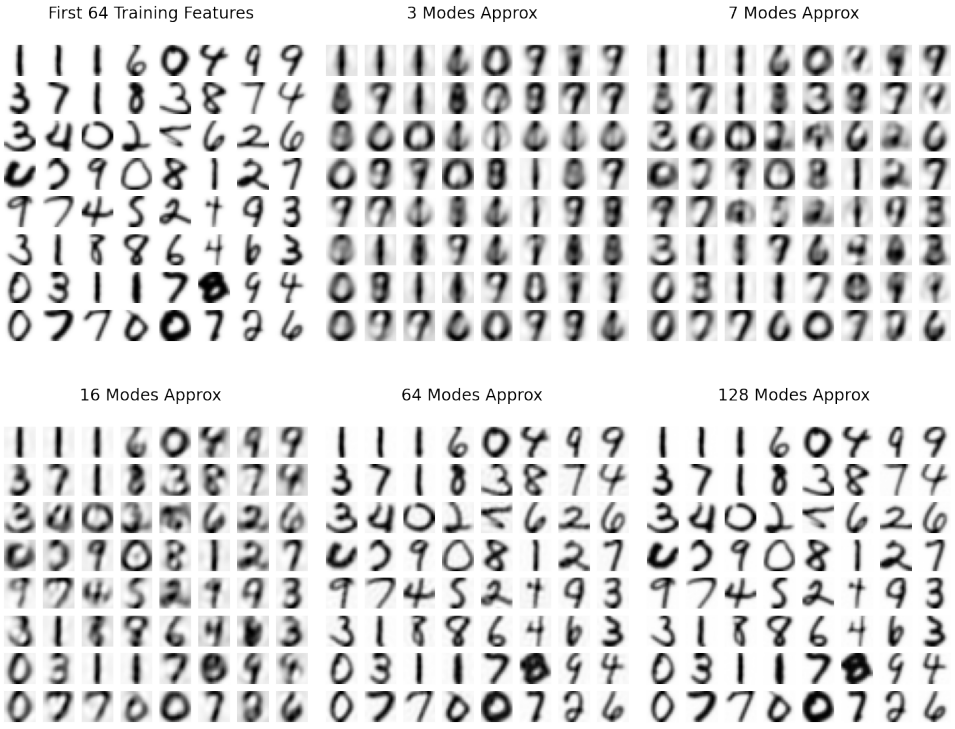

FIGURE 4. low rank reconstructions

Compared to the raw data, we can see the low rank reconstructions become recognizable starting from 16.

Task 3 & 4

For creating binary classifiers towards the pairs of digits (1, 8), (3, 8) and (2, 7), use transformed 16 dimensions data which is from PCA, to fit the Ridge regression and use cross validation to choose the best alpha (Regularization parameter of the Ridge regression). Model output:

	(1,8)	(3,8)	(2,7)
MSE_train	0.0754	0.1812	0.0928
MSE_test	0.0826	0.2591	0.1314

TABLE 2. MSE

	(1,8)	(3,8)	(2,7)
Best Alpha	10	10	10

TABLE 3. Best alpha

FIGURE 5. The weights of beta

$$\begin{bmatrix} 277 & 1 \\ 3 & 174 \end{bmatrix} \quad \begin{bmatrix} 71 & 1 \\ 1 & 23 \end{bmatrix}$$

$$\begin{bmatrix} 171 & 3 \\ 3 & 174 \end{bmatrix} \quad \begin{bmatrix} 42 & 6 \\ 0 & 24 \end{bmatrix}$$

$$\begin{bmatrix} 180 & 0 \\ 2 & 169 \end{bmatrix} \quad \begin{bmatrix} 52 & 1 \\ 1 & 44 \end{bmatrix}$$

This is Confusion Matrix above.

From the above results, comparing MSEs and Confusion Matrices in different models, we can find the performance of (3, 8) and (2, 7) classifiers apparently worse than (1, 8) classifier. In terms of human intuition, the similarity of (3, 8) and (2, 7) is much stronger than (1, 8), we can speculate, this is some kind of reason for different performances.

For further study, visualize where the 1's vs 8's, 3's vs 8's and 2's vs 7's live in the PCA space using the coefficients for modes 1, 2 and 3.

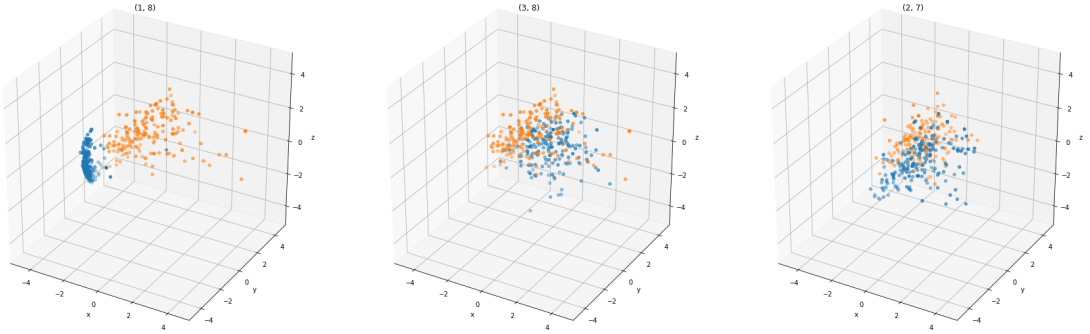

FIGURE 6. PCA space

Apparently, we can see far apart (1, 8) clouds, close together (3, 8) clouds and close together (2, 7) clouds. To a large extent, this result confirms my conjecture!

5. Summary and Conclusions

Firstly, fit PCA with the original data and plot singular value spectrum and explained variance ratio. We find the effective dimension of the data appear to be less 100. Secondly, by calculating the Frobenius norm and different low rank reconstructions, we can find appropriate number of PCA modes to approximate data. Finally, create binary classifiers with Ridge regression and evaluate their performance. Surprisingly, the machine model result is consistent with human intuition. And the PCA results confirm my conjecture.

Acknowledgements

The author is thankful to Prof. Hosseini for lectures about signal processing. I am also thankful to Dr. Owens and Dr. Levin for careful guidance about the report. Furthermore, My peers Xuqing Hu and Xiangyu Gao were helpful in the use of Latex.

References

[1] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020.

[2] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.