AMATH 482: HOMEWORK 3

TIANRUI ZHOU

Amath Department, University of Washington, Seattle, WA
tianrz5@uw.edu

Abstract. This report is about my program of training classifiers to distinguish images of handwritten digits from the famous MNIST data set [1]. The content will be divided into five sections: Introduction and Overview, Theoretical Background, Algorithm Implementation and Development, Computational Results, and Summary and Conclusions. They will elaborate on how I used Principal Component Analysis, Machine Learning concept, and Ridge classifier to process the raw data given. I used Python code as my data processing tool and generated visual results with it. I will provide detailed explanations of my understanding of the mathematical foundations of the methods and algorithms. I will present my computational results and my conclusion for this project.

1. Introduction and Overview

I am working on the famous MNIST handwritten digits dataset in order to train classifiers to distinguish and classify those digits. In the dataset, it has 60,000 training images and 10,000 testing images. I will train my classifier using the training set, and then, use testing images for evaluation of my classifier [1].

The goal of this project is to employ Ridge, KNN, and LDA classifiers for computing accuracy, facilitating comparison of their accuracy in distinguishing between different digits. For the Ridge classifier, a specific focus will be placed on evaluating its performance in distinguishing between pairs of digits. Finally, we are going to discern the situations in which each classifier, among Ridge, KNN and LDA, performs well, and identify which one achieves the highest accuracy in the digit classification task.

2. Theoretical Background

My goal of the whole algorithm is using math foundation of Principal Component Analysis and Machine Learning concept to first project data to truncated PCA space, and then apply classifiers of Ridge, KNN, and LDA to distinguish between digits for both training data and testing data.

First, I need to plot the first 16 PC modes as 28×28 images. I reshape each image in training data and testing data into vector and combine the vectors into matrices for `Xtraindata` and `Xtestdata` respectively [1]. The `Xtraindata` has the shape of $784 \times 60,000$, and `Xtestdata` has the shape of $784 \times 10,000$. Then, I use PCA function in `sklearn` to compute the PC modes of the training data. The dimension of the truncated 16 PC modes is 784×16.

The mathematical foundation for the truncated PCA modes is that, the basis of PCA modes can be extracted through the columns of the left singular vector U matrix from SVD: $A = U\Sigma V^T$.

Date: February 22, 2024.

Then, the result of the singular value decomposition can be used to calculate the cumulative energy:

$$E_i = \frac{\sum_{j=1}^{i} dS_j^2}{\sum_{j=1}^{n} dS_j^2}$$

It calculates the cumulative energy by dividing a certain proportion of the squared singular values by the total sum of squared singular values. dS is the singular value, j means the j th singular value, and E_i stands energy for mode i.

Then we need to project the data to truncated k-PCA space, where contains 85% of the total energy. The projection of a given sample z onto k-PC modes is achieved by $z_k = U_k^T z$, where U_k^T is the transposed U matrix from SVD with first k column vectors and column vectors from $k+1$ and onward are set to be zero vectors.[2] In our case, I got the result that $k = 59$ is the number of PC modes that contains 85% of the total energy.

Then, we use the projected data to evaluate the Ridge classifier's performance in distinguishing between the two digits in each of the three pairs of digits 1&8, 3&8, and 2&7. In order to do this, I extract X(the image data) and y(the actual digit name) data for each of the pairs, and construct into subsets. Then, we are now able to run the Ridge classifier on each of the pairs, and compute the accuracy. Then, we use Ridge, KNN, and LDA classifier to distinguish all the digits using the projected data.

Each time when implementing the classifier, I do cross-validation to examine the performance of the classifier to the training data. The cross-validation shuffles the data, divides the data into k folds, then computes accuracy and shows standard deviation on them.[10]

3. Algorithm Implementation and Development

In the python code, I used some software packages to help me do the calculation and analysis.

The `NumPy`[3] library is used for numerical operations and array manipulations. Some built-in functions in this package are really crucial to my program. `np.fromfile`[7] is used to construct an array from data in a text or binary file; `np.dtype` defines the data type of the elements; `np.transpose`[8] returns an array with axes transposed of the input array; `np.cumsum`[6] returns the cumulative sum of the elements along a given axis; `np.argmax`[5] returns the indices of the maximum values along an axis.

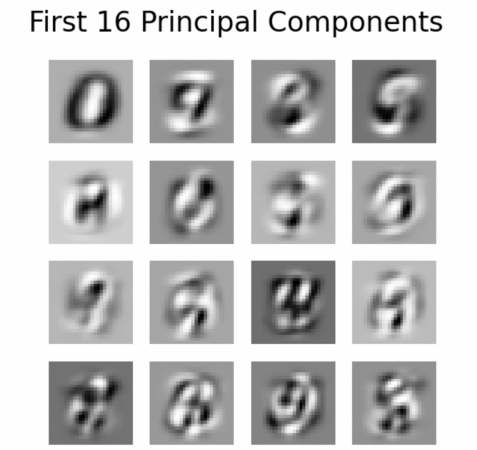
The `sklearn.decomposition`[9] module is used to perform matrix factorization and dimensional reduction, including Principal Component Analysis (PCA). The `sklearn.linear_model`[9] module introduces the Ridge classifier function. The `sklearn.model_selection` module is intended to assist with cross-validation. The `sklearn.neighbors` module is used for the KNN classifier. The `sklearn.discriminant_analysis` module is employed to implement the LDA classifier.

The package `Matplotlib`[4] is used to make visualization of my data.

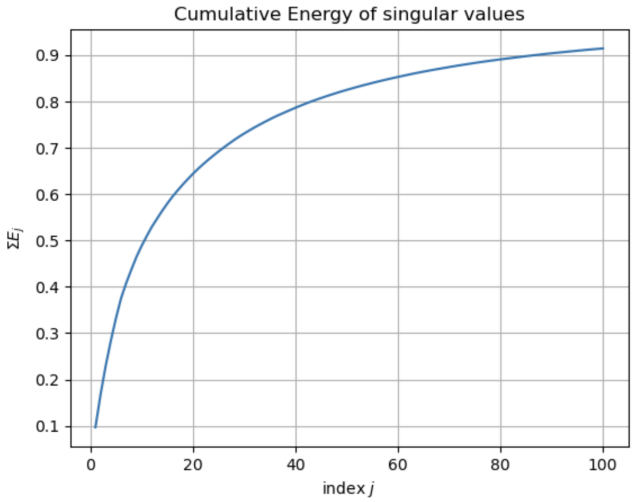
4. Computational Results

Through using mathematical foundation and programming packages, I have computed and plotted out my computational results. They include using PCA analysis to plot the first 16 PC modes as 28×28 images; The value k of PC modes needed to approximate 85% of the energy of singular values; For Ridge classifier, I evaluate its accuracy performance in distinguishing between the two

digits in each of the three pairs of digits 1&8, 3&8, and 2&7; Lastly, I use Ridge, KNN, and LDA classifier to distinguish all the digits using the projected data, and compute their accuracy for training and testing data.



(A) The first 16 PC modes as 28 × 28 images



(B) The cumulative energy of the singular values

First, I reshape each image into a vector and combine the vectors into matrices for `Xtraindata` and `Xtestdata` respectively [1]. Then, I compute the truncated 16 PC modes of the training data, with the dimension of 784×16. Figure 1a shows the plot of the first 16 PC modes as 28×28 images.

Figure 1b plots the cumulative energy for different numbers of PCA modes. The x-axis stands for how many PCA modes are kept, the y-axis represents the corresponding cumulative energy (1.0 refers to 100%). As the number of PCA modes kept increases, we observed that the cumulative energy also increases. Through my computation, we need $\mathbf{k = 59}$ numbers of PC modes to approximate **85%** of the energy.

Digit Pair	**Training Acc (%)**	**Mean CV (%)**	**CV StdDev (%)**	**Testing Acc (%)**
1 vs. 8	96.649	96.427	0.267	98.009
3 vs. 8	96.044	95.886	0.605	96.371
2 vs. 7	98.028	97.938	0.195	97.427

TABLE 1. Ridge classifier accuracy of training and testing for different digit pairs, along with mean cross-validation scores and the according standard deviations.

After creating a function which extract $X_{subtrain}$ and $y_{subtrain}$, I select the subset of digit pairs of 1 & 8, 3 & 8, and 2 & 7, and apply the linear Ridge classifier to distinguish between these two digits in each pair respectively. Table 1 shows my computational results in training accuracy, testing accuracy, and the mean cross-validation scores and the according standard deviations for using Ridge classifier on those digit pairs. Each accuracy result indicates how well the Ridge classifier do to distinguish between the two digits in each pair.

From Table 1, we can clearly see that digit pair 3 vs 8 exhibit lower accuracy in training (96.044%), testing (96.371%), and mean cross-validation score (95.886%) compared to the other two pairs. Also, the standard deviation 0.605% in cross-validation is higher, which means higher variability in performance across different folds, than that of the other two pairs. One possible

reason for this notable difference in the accuracy is that 3 and 8 look really similar, especially in handwriting circumstances. Therefore, it may be challenging for the classifier to distinguish these two similar dataset.

Digit pairs 1 vs 8 and 2 vs 7 both have nice performances in accuracy. Their cross-validation standard deviations stay in a low level, which suggests consistent performance across different folds.

Classifier	**Training Acc (%)**	**Mean CV (%)**	**CV StdDev (%)**	**Testing Acc (%)**
Ridge	84.5	84.4	0.957	85.6
KNN	98.292	96.098	0.148	96.1
LDA	86.662	86.512	0.863	87.52

TABLE 2. Multi-Class classifier accuracy results of training and testing for Ridge, KNN, and LDA classifiers, along with mean cross-validation scores and the according standard deviations.

Then, I use all the digits and perform multi-class classification with Ridge, KNN and LDA classifiers. Table 2 shows my computational results in training accuracy, testing accuracy, and the mean cross-validation scores and the according standard deviations for using those three different classifiers. Each accuracy result indicates how good each classifier's performance in distinguishing between all the digits.

It is evident that KNN appears as the top-performing classifier in this multi-class classification work. It has notably about 10% higher accuracy in training, testing and the mean cross-validation score than the Ridge and LDA classifier. However, Ridge and LDA demonstrate better adaptation to the testing data, since their testing accuracy is higher than the training accuracy. For KNN, the testing accuracy 96.1% is slightly lower than the training accuracy 98.292%

5. Summary and Conclusions

This project is about training different classifiers, including Ridge, KNN, and LDA, to distinguish images of handwritten digits from the famous MNIST data set. I first use training data to train my classifiers, and then, use testing images to evaluate how well my classifier do when dealing with new data.

Through my computation results of Ridge classifier accuracy of training and testing for three different digit pairs, we may infer that the visual similarity of digits plays a role in influencing classifier performance. And for all the digits in MNIST data, KNN classifier performs the best with testing accuracy of 96.1% among the three classifiers. Despite this, it is crucial to acknowledge that the most suitable classifier for a specific application dataset may vary.

Acknowledgements

The author is thankful to Prof. Eli Shlizerman for useful information posted on discussion board about downloading the data, and the useful information about this project during lectures. We are also thankful to peer Ziyi Zhao and all the classmates on Discord for meaningful discussion of logic of the coding part, and helping me debug my code successfully.

References

[1] Directions, reminders and policies. PDF, 2024. File: AMATH482582_Homework3.pdf.

[2] Directions, reminders and policies. PDF, 2024. File: AMATH482582_Homework2.pdf.

[3] 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.

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

[5] NumPy Community. numpy.argmax. *NumPy Documentation*, 2022. https://numpy.org/doc/stable/reference /generated/numpy.argmax.html.

[6] NumPy Community. numpy.cumsum. *NumPy Documentation*, 2022. https://numpy.org/doc/stable/reference/ generated/numpy.cumsum.html.

[7] NumPy Community. numpy.fromfile. *NumPy Documentation*, 2022. `https://numpy.org/doc/stable/reference/generated/numpy.fromfile.html`.

[8] NumPy Community. numpy.transpose. *NumPy Documentation*, 2022. https://numpy.org/doc/stable/reference/ generated/numpy.transpose.html.

[9] scikit-learn developers. scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] E. Shlizerman. Amath 482 lecture 16 notes, 2024. University of Washington.