AMATH 582: HOMEWORK 3

KATIE WOOD

Applied Math Department, University of Washington, Seattle, WA
kvo24@uw.edu

Abstract. We compare the performance of three different machine learning classifiers on the MNIST data set. This data set consists of thousands of images of handwritten digits. We employ the Ridge, K-Nearest Neighbors, and Linear Discriminant Analysis classifiers from Python's Sci-kit Learn package. We also use Principal Component Analysis to reduce the dimensionality of the data set prior to classification.

1. Introduction and Overview

Correctly classifying images of handwritten digits is essential for banks and the postal service. Recently, a computational data analysis class at the University of Washington was tasked with training three different classifiers to recognize handwritten digits. The scientists on the project received tens of thousands of gray-scale images of handwritten digits, presorted into training and testing sets. They then used Principal Component Analysis to reduce the dimensionality of the data set. Finally, they trained and tested the Ridge, K-Nearest Neighbors, and Linear Discriminant Analysis classifiers on pairs of digits (1, 8), (3, 8), and (2, 7), as well as a multi-class classification of all digits 0-9.

2. Theoretical Background

The Ridge classifier uses linear regression to create a decision line (or curve) with which to differentiate different digits. The K-Nearest Neighbors classifier determines to which class a given data point belongs by taking a vote of the classes of its k-nearest neighbors. In this assignment, we are performing supervised learning because we are given labels y_{train} and y_{test} that indicate the true digit associated with each image.

Below, we recap the theory behind using the SVD for dimension reduction. The Singular Value Decomposition, or SVD exists for all matrices. The decomposition is usually written:

$$A = U\Sigma V^\dagger \tag{1}$$

where $A \in \mathbb{C}^{m\times n}$, $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ are unitary, and $\Sigma \in \mathbb{C}^{m\times n}$ is diagonal [1]. The *Principal Components* of the data contained in A are given by the column vectors of U.[1]

Since U is unitary, we have

$$U^T A = \Sigma V^T \tag{2}$$

To project a dataset x onto k Principal Component modes, we thus multiply $U_k^T x = x_k$. Here, the subscript k indicates that U_k^T is truncated to the first k columns of the original matrix U (which are the first k rows of U^T). This truncation is achieved either by setting columns $k+1$ onward to be zeros, or simply deleting those columns.

[1]Consistent with the data we are working with, here we only consider real A.

Lastly, we mention the Frobenius norm, which is used to measure the *energy* of a matrix [1]:

$$||A||_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2} \tag{3}$$

By summing the energy cumulatively across the first k Principal Component modes of A, we aim to capture a large percentage of the behavior of A. This is made possible, mathematically, by the fact that the singular values contained in Σ (lying on its main diagonal), appear in decreasing order:

$$\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \tag{4}$$

where the σ_j are the singular values of A. Because U and V are unitary, their column (or row) vectors have length 1 in the 2-norm. Thus we can measure the energy of A by focusing only on the magnitudes of A's singular values. If A's singular values decrease quickly, the dominant contribution to A's energy comes from its first few singular values. Thus our aim is to use these first k singular values, and their corresponding k Principal Component modes, to understand the majority of the behavior of A. To reduce the dimension of the data significantly, we want $k \ll m$.

3. Algorithm Implementation and Development

Our computational approach relied primarily on Python's machine learning package Sci-Kit Learn. We used classifiers, accuracy measures, cross validation, and principal component analysis methods from Sci-Kit Learn. We also made extensive use of Python's scientific computing package NumPy. Visualization was provided by Python's Matplotlib package.

4. Computational Results

We began by compiling all the training samples into the matrix X_{train} and all the testing samples into X_{test}. We then plotted the first 16 Principal Component modes as 28×28 images:

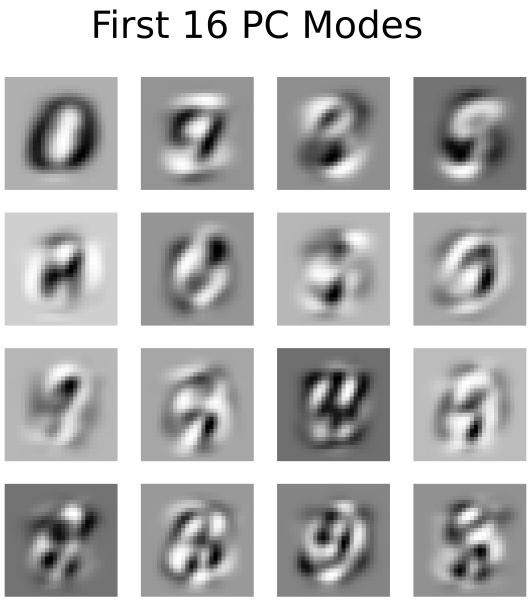

Figure 1. The first 16 PC modes

We then inspected the cumulative energy of X_{train}'s singular values to determine $k =$ the number of PC modes needed to approximate 85% of the energy. We found that $k = 59$. We then plotted the first 64 training images reconstructed from 59 truncated PC modes.

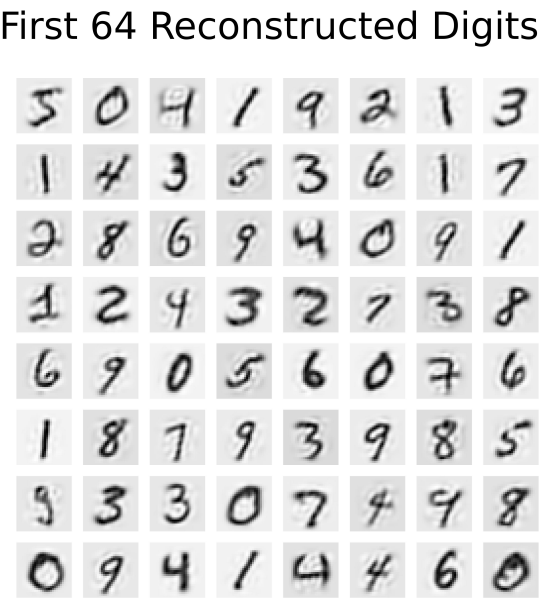

FIGURE 2. The first 64 training images reconstructed from 59 PC modes

Inspecting the above images, we notice a slight degradation from the original images provided in the assignment. However, the digits are still distinguishable. Thus, we validate that 59 PC modes provides a reasonable approximation to the 784-dimensional data set.

Next, we projected X_{train} onto 59 PC modes using the method described in 2 . We then selected all samples of the digits 1 and 8 from X_{train}, X_{test}, Y_{train}, and Y_{test}, and applied the Ridge classifier (linear) to distinguish between these two digits. We obtained a training score of 0.965 and a testing score 0.981. We performed cross-validation and found an average training accuracy of 0.963 with a standard deviation of 0.00310.

We repeated the above process for pairs of digits 3, 8 and 2, 7. For 3, 8 we obtained a training score of 0.961 and a testing score of 0.963. Cross-validation yielded an average training accuracy of 0.959 with a standard deviation of 0.00571. For 2, 7 we found a training score of 0.981 and a testing score of 0.973. Cross-validation gave 0.981 average accuracy with a standard deviation of 0.00235.

Intuitively, we might expect that the Ridge classifier would perform the best on the digit pair 1, 8 because visually these two digits look more different than 2, 7 or 3, 8. However, we actually found the highest scores for 2, 7. Across the board, the Ridge classifier performs well (over 95% accuracy) each of the digit pairs.

We then performed multi-class classification with three different classifiers: Ridge, K-Nearest Neighbors (KNN), and Linear Discriminant Analysis (LDA). For Ridge, we found a training score of 0.846, a testing score of 0.857, and a cross-validation score of 0.844 with a standard deviation of 0.0101. For KNN, we found a training score of 0.989, a testing score of 0.976, and a cross-validation score of 0.975 with a standard deviation of 0.00130. Lastly, for LDA, we found a training score of 0.866, a testing score of 0.876, and a cross-validation score of 0.865 with a standard deviation of 0.00869. Below, we plot the confusion matrix for each classifier:

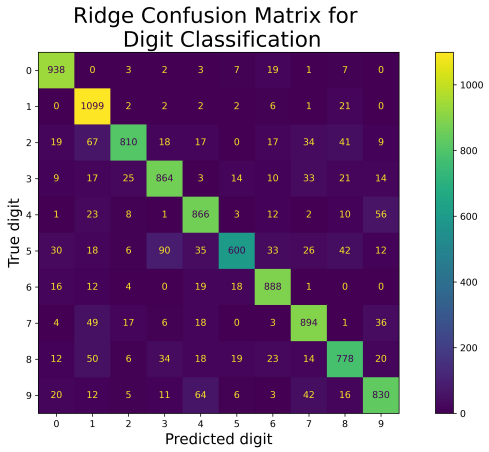

FIGURE 3. Multi-class classification: Confusion matrix for the Ridge classifier

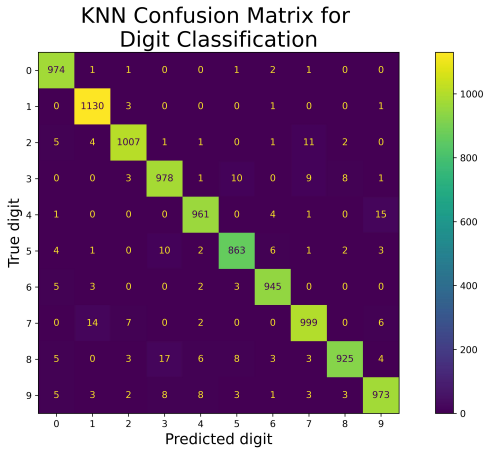

FIGURE 4. Multi-class classification: Confusion matrix for the KNN classifier

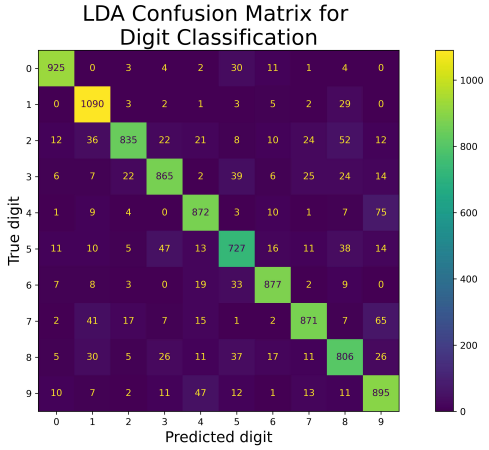

FIGURE 5. Multi-class classification: Confusion matrix for the LDA classifier

For additional interest, we also plotted the data samples along their first two and three LDA components, to visualize how much separation Linear Discrimant Analysis achieves between the 10 digits:

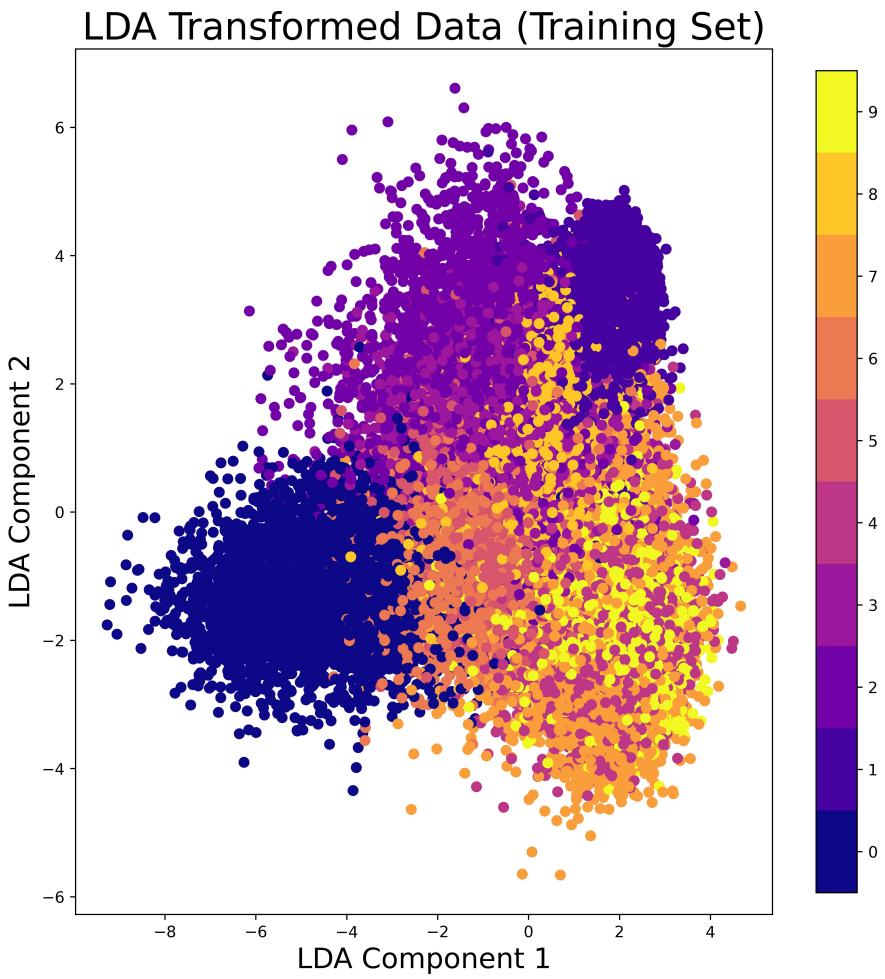

FIGURE 6. Visual separation with the first two LDA components

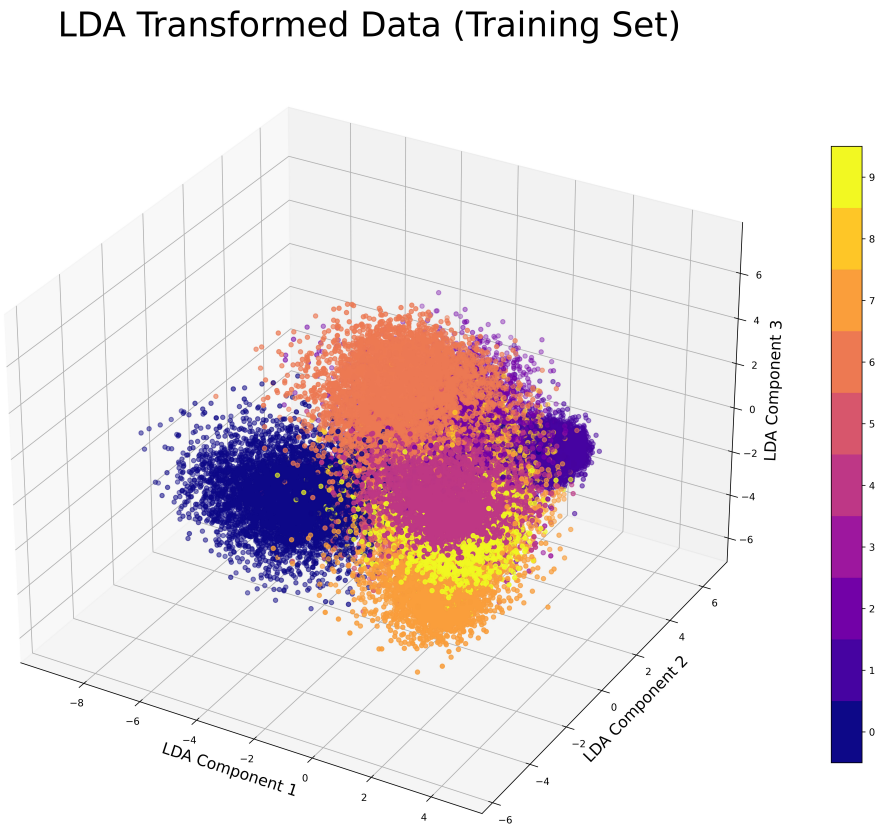

FIGURE 7. Visual separation using the first three LDA components

5. Summary and Conclusions

We found that $k = 59$ Principal Component modes captured 85% of the energy of the training data. There were slight differences in performance of the Ridge classifier on different pairs of digits. There was a greater difference between the classifiers in their ability to classify all of the digits. The K-Nearest Neighbor classifier performed the best for the MNIST data set 4.

In a future project, we could implement another classifier such as SVM and compare its performance with the classifiers already used.

Acknowledgements

The author is thankful for the Homework 3 recitation.

References

[1] L. Trefethen and D. Bau. *Numerical Linear Algebra*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, 1997.