PCA-BASED FEATURE EXTRACTION FOR HANDWRITTEN DIGIT RECOGNITION: EVALUATING RIDGE AND KNN CLASSIFIERS ON MNIST

CHENAB

Applied and Computational Mathematics, University of Washington, Seattle, WA
chenab@uw.edu

Abstract. The following paper discusses various classifiers were trained on the MNIST data set to solve the problem of distinguishing images handwritten digits using the concept of classification. The classifiers were trained by processing the data by extracting the principal component modes till an 85% energy of the original data was reached. This speed up computation considerably.

1. Introduction and Overview

The MNIST data set is a large database of handwritten digits, it contains 60,000 training images and 10,000 testing images. Each image is 28x28 pixels, which makes our X_{train} shape 784×60000. This paper will discuss the development of classifiers that distinguish between the images of the digits. The primary classifiers used were Ridge, K-Nearest Neighbors and SVM. The classifiers were trained after projecting the image data into k-PCA space, where k is the number of modes capturing 85% of the energy in terms of Frobenius norm.

For each classifier, its best parameter was found using cross validation on the test data. Also, for each classifier its testing accuracy, cross validation score, a classification report containing information about precision, recall and f1 score was created. Finally, confusion matrices were created to visualize the prediction trends for each digit. The following sections will go over the details of the key concepts and implementation.

2. Theoretical Background

The **Singular Value Decomposition** essentially decomposes any matrix $A \in R^{n\times m}$ into the following:

$$A = U\Sigma V^T$$

U is a unitary matrix with $U \in R^{n\times n}$

V is a unitary matrix with $V \in R^{m\times m}$

Σ is a diagonal matrix $\Sigma \in R^{n\times m}$ with all elements assumed to be non-negative and ordered from largest to smallest magnitude.

Principal Component Analysis is a dimensionality reduction technique that is used for **removing redundancy** and noise. It involves reducing the number of variables in a data set while capturing **as much information** (represented by variance) about the system as possible. We also get information about the most important directions along which the data varies the most. PCA involves diagonalizing the covariance matrix using the **SVD**, thus giving a relation between the **singular values** and the **eigenvalues**.

We will be using PCA to **reduce redundancy** in our data set, while capturing as much information as possible and to **ncreasing performance and efficiency** by dimensionality reduction.

An important concept we will use to gauge the amount of information being captured is the **Frobenius Norm**. Frobenius norm essentially captures the total variance (or energy of the system). It is given by

$$||A||_F = \sqrt{\sigma_1^2 + \sigma_2^2 + ... + \sigma_r^2}$$

where σ represents the singular values ordered by magnitude.

Date: March 25, 2025.

Ridge Classifier: Ridge regression involves introducing L2 regularization to the Ordinary Least Squares Regression, such that:

$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^J} \|A\beta - Y\|^2 + \frac{\lambda}{2}\|\beta\|_p^p$$

This is done to reduce the large variances in the coefficient estimates, and to improve numerical stability. Solving for β gives us $\beta = (A^T A + \lambda I)^{-1} A^T Y$.

Now, we use this regression framework for **binary classification** by encoding the class labels by distinguishing between the signs of the predictions, ie. class $\in$ {-1,1} if $\{pred < 0,\ pred > 0\}$. For multiclass classification, we use the strategy of **one-hot encoding** to extend y to multi-output, where for k classes, k different models are trained and the $argmax(ypred_k)$ is chosen as the predicted class.

K-Nearest Neighbors: In this we choose **k** number of neighbors. Then for each new sample, we compute the euclidean distance of that point in PCA space with every other point in the training data. Then we choose the k points closest to that point. And classification is done based on whichever class is most common by majority among the k-nearest neighbors. KNN applies well to multiclass classifications as well with no major modifications

K-fold Cross Validation: It is a procedure for estimating an error metric of a model. It involves chosing different subsets of data and using cross-validation on them. The data set is divided into k folds, and trained on the data sets that is labelled "train". This process is repeated k times for different subsets and the average of the model fit metric is used as the cross validation score.

3. Algorithm Implementation and Development

The following algorithm was implemented to perform PCA Analysis and classification of the images of the handwritten digits.

1.) Initial Setup and Data Processing

We first load from the MNIST database our training data, train labels, test data and the test labels using the helper code provided. We store the transpose of the training data into a matrix X_{train} upon which further PCA Analysis and classification training is performed.

2.) Obtaining Principal Components

Next, we use the **PCA fit** function from the **sklearn** library to fit the X_{train} dataset. First of all, it centers the X_{train} to be mean zero. This is very important because without centering, the mean value can skew the covariances leading to components that do not reflect the mean rather than the intrinsic variation in the data. After centering, it uses the **SVD** on the centered data matrix to decompose into U (which contains the left singular vectors, that are the eigenvectors of the covariance matrix of X_{train}), Σ, diagonal matrix containing the singular values sorted in descending order, and V^T containing the right singular vectors transposed. The PCA modes thus are spatial modes. The first 16 PC modes are then plotted as 28×28 images.

3.) Calculating k modes to reach 85% energy using the Frobenius norm and image reconstruction The Frobenius norm captures the amount of variance in the data. Therefore, we use to to get the minimum number of modes we need to capture certain percentages of the original information/variance.

This is essentially done by finding the ratio: $\frac{\sigma_1^2+\sigma_2^2+...+\sigma_k^2}{\sigma_1^2+\sigma_2^2+...+\sigma_r^2}$ Where k is the minimum modes needed to get 85%, and the denominator represents the total energy by including all the singular values. We use the **PCA explained variance ratio** function from the **numpy** library to get the ratio and the k modes needed for 85 %.

Then we reconstruct the original mode using our k-PCA modes to get an approximation. This is done using **PCA inverse transform** function, and image is plotted using the plotting function provided in the helper code.

4.) Selecting subsets of particular digits (all samples) from the We then write a function that selects all the subsets of particular digits ,passed as a list, from X_{train}, Y_{train}, X_{test} and Y_{test} and transforms each one of these into the subsets. The **isin** function of the numpy library is used to extract these subsets.

5.) Applying the Ridge classifier to distinguish between digits [1,8], [3,8] and [2,7]

We select the corresponding subsets using our function and then apply the Ridge classifier to the projected subsets in k-PCA modes using the **RidgeClassifierCv** from the **sklearn** library. We compute the cross

validation score for each pair using k-fold cross validation with default cv = 5. We also analyze the accuracy the different values of the ridge parameter alpha using **accuracy score** on the **test** data and our predictions, and report the best value of alpha that maximizes the accuracy using **numpy argmax**.

5.) Multiclass Classification on all digits using Ridge and K-NN classifiers

Next, we use the Ridge classifier for all digits. This is done using **one-hot encoding** that is taken care of us by python's **RidgeClassifier**. We use **GridSearchCV** to get the best alpha using cross validation. Next, we use KNN to classify all the digits using **KNeighborsClassifier**. For this, we check the accuracy of different k values in predicting the test data and choose the best k using **argmax** on the accuracies array.

Then for both the classifiers, We print a classification report that contains scores for precision, recall and f1-score.

4. Computational Results

Upon performing the PCA analysis, we plot the first 16 PC modes as 28×28 images:
And we get the following graph:

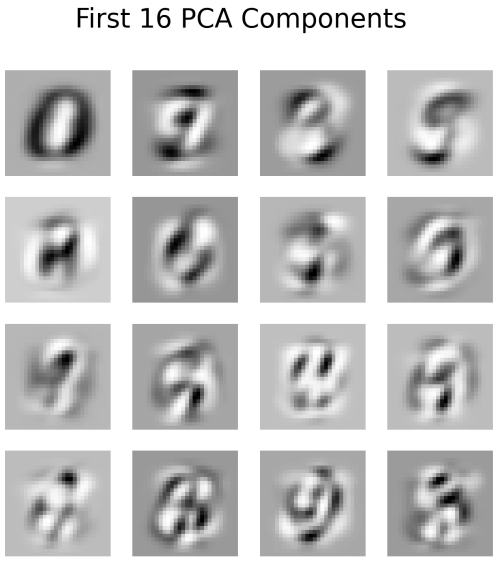

Figure 1. Plot of the first 16 PC Modes as 28x28 images

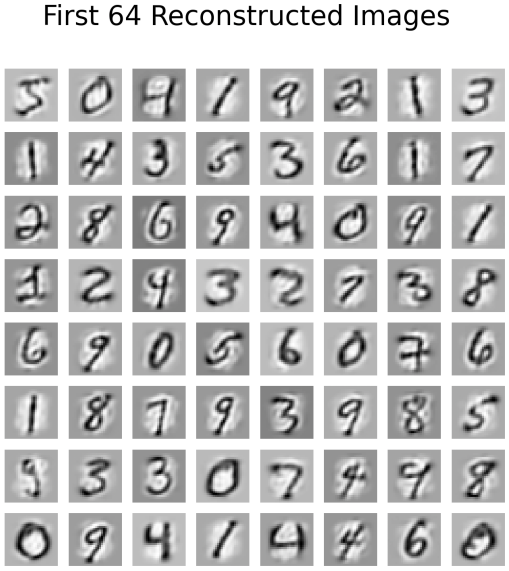

Figure 2. Reconstructed images using 59 PC modes

We also determine the number of modes needed to capture 85% energy, which turns out to be k = 59 PC modes. Next, we move on to ridge classification of pairs of digits [1, 8], [3, 8] and [2, 7] by applying the ridge classifier on training data by projecting onto 59 PC modes. We apply cross validation and get the following graphs of accuracy vs alphas, that include the best alpha and corresponding test set accuracy. The alphas are chosen on a log scale from 10^{-10} to 10^{10}

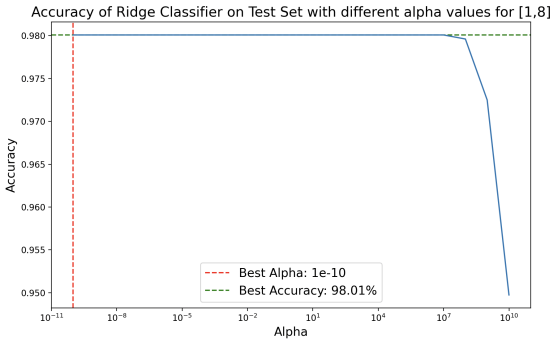

FIGURE 3. Accuracies for different alpha values for [1,8]

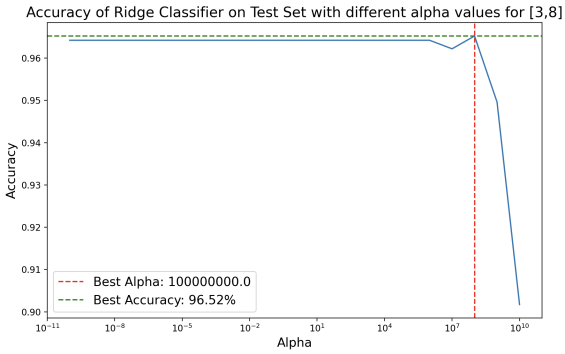

FIGURE 4. Accuracies for different alpha values for [3,8]

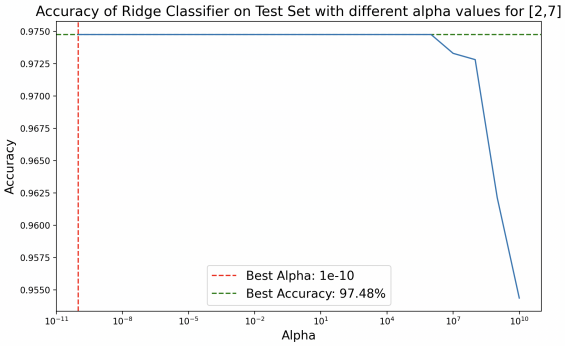

FIGURE 5. Accuracies for different alpha values for [2,7]

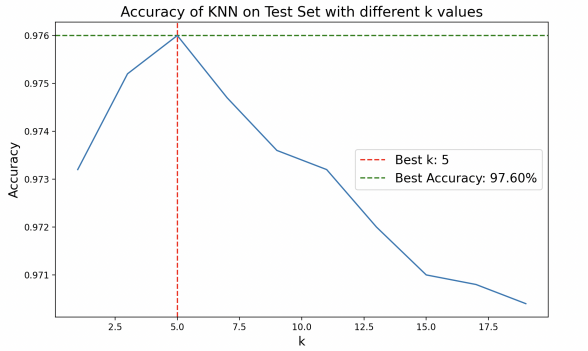

FIGURE 6. Plot of Test Accuracy vs K values for the KNN classifier

The following are the K-fold cross validation scores with cv = 5
0.96 accuracy for [1, 8] with a standard deviation of 0.00
0.96 accuracy for [3, 8] with a standard deviation of 0.01
0.98 accuracy for [2, 7] with a standard deviation of 0.00

We notice that [3,8] has less accuracy than [2,7] by a small margin. This could be explained either by structural similarities in the symbols for 3 and 8, which to some extent, look more similar with curved edges, than say more distinct pairs like [1, 8] and [2, 7] that are farther in the feature space, which makes classification more accurate. We also notice that the best alpha that maximizes accuracy for the ridge classifier on test data, is way larger (approximately 10^8 for [3, 8], and is the smallest (10^{-10}) for [1,8] and [2,7]. This suggests that a significant amount of regularization is needed for classifying [3, 8] accurately, showing a large number of irrelevant or noisy features are present. In comparison, [1,8] and [2, 7] have almost no regularization.

Next, for the multiclass classification with all the digits, we get the following accuracy for the **Ridge classifier** on the test data: **85.61%**. This accuracy was for the best alpha = 1e-10, which was found using **GridSearchCV**. The average k-fold cross validation score is **84.44%**.

Moving on to KNN, we use the **KNeighborsClassifier**. We trained the KNN classifier for various K-values from 1, 3, 5, ... 19. The prediction accuracy on the test data for each k value was computed (cross-validation) to get the best k which is found to be 5, as seen in the figure 6 above.

The following were the classification reports:

TABLE 1. Classification Results: Ridge Classifier vs. KNN

	Ridge Classifier				KNN			
Class	**Prec**	**Rec**	**F1**	**Supp**	**Prec**	**Rec**	**F1**	**Supp**
0	0.89	0.96	0.92	980	0.98	0.99	0.98	980
1	0.82	0.97	0.89	1135	0.98	1.00	0.99	1135
2	0.91	0.79	0.84	1032	0.98	0.97	0.98	1032
3	0.84	0.86	0.85	1010	0.97	0.97	0.97	1010
4	0.83	0.88	0.85	982	0.98	0.98	0.98	982
5	0.90	0.67	0.77	892	0.97	0.97	0.97	892
6	0.87	0.93	0.90	958	0.98	0.99	0.98	958
7	0.85	0.87	0.86	1028	0.97	0.97	0.97	1028
8	0.83	0.80	0.81	974	0.98	0.96	0.97	974
9	0.85	0.82	0.83	1009	0.97	0.96	0.97	1009
Accuracy		0.86		10000		0.98		10000
Macro avg	0.86	0.85	0.85	10000	0.98	0.98	0.98	10000
Weighted avg	0.86	0.86	0.85	10000	0.98	0.98	0.98	10000

Moreove, the following confusion matrices were created to better visualize how are classifiers are performing for each digits.

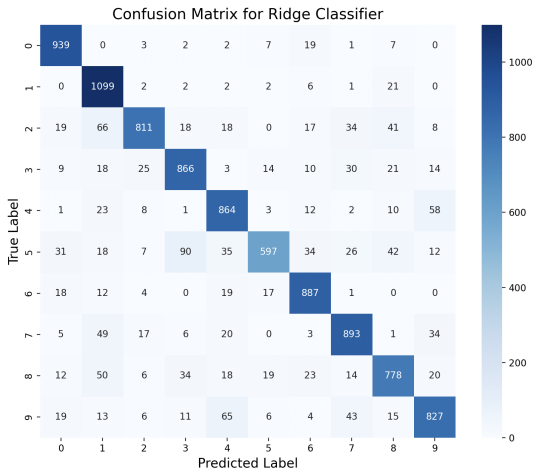

FIGURE 7. Confusion Matrix for the Ridge classifier

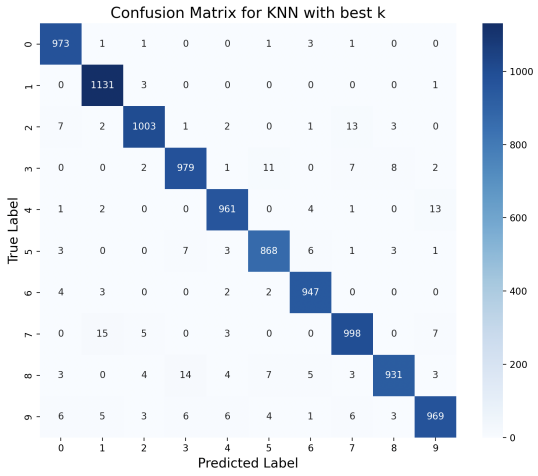

FIGURE 8. Confusion Matrix for the KNN classifier

Clearly, the KNN classifier is superior than the Ridge at multiclass classification with an Accuracy of around 98% compared to 86% for the Ridge.

5. SUMMARY AND CONCLUSIONS

In conclusion, we effectively used Principal Component Analysis and Classification based on the Ridge Classifier and the KNN classifier to succesfully distinguish between images of digits from the MNIST data set, achieving a peak accuracy of 97.6% for multiclass classification using KNN. In the process, weh also made several visualizations to support our findings.

REFERENCES

1.) Kutz, J. Nathan. Data-Driven Modeling Scientific Computation: Methods for Integrating Dynamics of Complex Systems and Big Data. Oxford: Oxford University Press, 2013. Chapter 15, "Matrix Decompositions," 375-390. Sections include "The Singular Value Decomposition (SVD)," 375-378; "The SVD in

Broader Context," 379-383; "Introduction to Principal Component Analysis (PCA)," 384-386; "Principal Components, Diagonalization and SVD," 387-389; "Principal Components and Proper Orthogonal Modes," 390. Chapter 17, "Unsupervised Machine Learning": "Data Mining, Feature Spaces and Clustering" (p. 443) "Unsupervised Clustering Algorithms" (p. 447)

2.) Instructor Notes and Helper Code

6. Additional Possible Approaches

We used the Quadratic Discriminant Analysis (QDA) classifier for an alternative classifier. It is a classifier with a **quadratic decision boundary**. It does this by fitting Gaussian distributions, having different means and variances, to each class. I performed cross validation on the reg parameter using **GridSearchCV**. The best parameter was found to be **0.0**, meaning no regularization is needed. QDA accuracy with the best parameter was found to be **96.11%** and the k-fold cross validation score (cv=5) was **95.87%**.

Table 2. QDA Classification Report

Class	Precision	Recall	F1	Support
0	0.98	0.99	0.98	980
1	1.00	0.96	0.98	1135
2	0.93	0.97	0.95	1032
3	0.95	0.96	0.96	1010
4	0.97	0.98	0.98	982
5	0.96	0.96	0.96	892
6	0.99	0.96	0.98	958
7	0.99	0.93	0.96	1028
8	0.89	0.96	0.92	974
9	0.96	0.94	0.95	1009
Accuracy		0.96		10000
Macro avg	0.96	0.96	0.96	10000
Weighted avg	0.96	0.96	0.96	10000

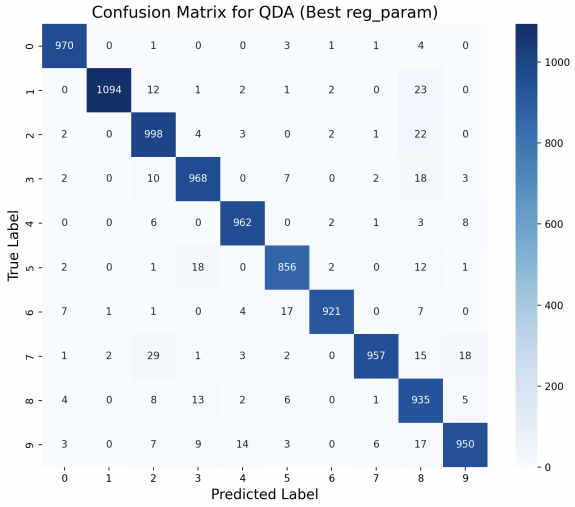

Figure 9. Confusion Matrix for QDA

This classifier performs much better than the Ridge Classifier, however marginally lower than the KNN Classifier.