AMATH 482/582: HOME WORK 3

SARAYU GUNDLAPALLI

Applied Math, University of Washington, Seattle, WA
saru07g@uw.edu

Abstract. This project focuses on training classifiers to distinguish images of handwritten digits from the famous MNIST data set. It aims to develop a system capable of correctly identifying written numbers by applying PCA and various classification techniques, which have various applications in fields like image recognition. We can approximate 85% of the cumulative energy in the Frobenius norm using the first 59 PCA modes. Subsets containing two digits each (1,8), (3,8), and (2,7) were extracted and classified using a Ridge regression classifier after projecting onto the top 59 PC modes. An extension is to test our classifier performance using different methods, like Ridge, KNN, and LDA classifiers. The report provides detailed results and analysis of the various classification approaches.

1. Introduction and Overview

Given images of handwritten digits from the Modified National Institute of Standards and Technology (MNIST) dataset, we aim to train a classifier to distinguish the images[5]. The dataset is split into 60,000 training images and 10,000 test images. X_{train} are the 16x16 black and white training "images" while "labels," which we denote as Y_{train}, are the digits. The first 64 training images are visualized in Figure 1 (from hw file).

Figure 1. First 64 digits in MNIST data.

Using Principal Component Analysis (PCA), we investigate the dimensionality of X_{train}. We determine the number of PCA modes to keep in order to approximate X_{train} 85% in the Frobenius

Date: February 23, 2024.

norm. To approximate up to 85%, we keep the first 59 PCA modes. The classifier is trained to distinguish two digits by extracting the features and labeling those two digits from the training data set, then projecting those features on the first 59 PCA modes of X_{train}. We then train and evaluate binary classifiers to distinguish between pairs of digits (1 vs 8, 3 vs 8, and 2 vs 7) after PCA projection. We assess our classifier by comparing the training and testing mean squared errors (MSE) of the classifier and cross-validating the scores. We are indeed able to train a classifier that can classify the two digits with reasonable accuracy. Three standard classifiers are examined: Ridge regression, k-nearest neighbors (KNN), and linear discriminant analysis (LDA). The results and performance of the different models are analyzed and compared.

The results shed light on the advantages and limitations of these standard machine-learning approaches for pattern classification problems.

2. Theoretical Background

2.1. **Principal Component Analysis (PCA).** [4]

PCA is an application of singular value decomposition (SVD), which produces characteristic features determined by C_X, the covariance matrix of data X. The eigenvectors of C_X are the principal components. Defining C_X if X = $\mathbb{U}\Sigma\mathbb{V}^T$, then we have:

$$C_X = \frac{1}{n-1}XX^T = \frac{1}{n-1}\mathbb{U}\Sigma V^T\mathbb{V}\Sigma^T\mathbb{U}^T = \frac{1}{n-1}\mathbb{U}\Sigma^2\mathbb{U}^T$$

If we project the data onto the left singular values and call the result Y (so $Y = U^*X$) then the covariance matrix of Y is given by:

$$C_Y = \frac{1}{n-1}\Sigma^2$$

where $\Sigma^2 = \Lambda$ where Λ is a diagonal matrix containing the eigenvalues of XX^T. In this case, the principal components corresponded to the eigen-images of greatest variance (where the variance is proportional to Σ^2). By only using the top few principle components, the number of dimensions (28x28=784) can be drastically reduced to enable faster/better classification.

2.2. **Ridge Regression Model.**

Considering a simple linear regression model for supervised learning and solving, the regression solution is found as:

$$\hat{\beta} = argmin\frac{1}{2\sigma^2}\|A\beta - Y\|^2 + \frac{\lambda}{2}\|\beta\|^{p(=2)}$$

where $\hat{\beta}$ are the predicting coefficient vectors and λ is the regularization parameter.

2.2.1. ***Mean Squared Error (MSE).***

By calculating the testing and training MSEs, we can assess the performance of the classifier. Training MSE can be defined as:

$$MSE_{train}(\hat{f}, Y) = \frac{1}{N}\sum_{n=0}^{N-1}\left|\hat{f}(x_n) - y_n\right|^2$$

2.2.2. ***Cross Validation.***

Both training and testing errors are dependent on the choice of λ, which is a model parameter. To address this, we could use K-folder Cross Validation (CV). K-folder CV splits the training data (both features and labels) into K-subsets (some as training and others as testing), iterating over k = 0, ..., K-1, removing the K-th subset, and fitting the model to the training data.

2.3. **Linear Discriminant Analysis (LDA).** Linear Discriminant Analysis (LDA) is a commonly used method in machine learning and statistics for feature extraction and dimensionality reduction. In LDA, the goal is to find a projection of the data onto a lower-dimensional space while still preserving the separability of the classes. This is done by maximizing the between-class scatter and minimizing the within-class scatter. The between-class scatter measures the distance between the means of each class, while the within-class scatter measures the variance within each class.

2.4. **K-Nearest Neighbors.** [1] KNN is a non-parametric, instance-based supervised learning technique. It does not explicitly learn a model but memorizes the entire training dataset. To classify a new data point, it searches the training set to find the k closest data points or "neighbors" to the new point based on a distance metric (e.g. Euclidean). It assigns the new point the class label that appears most frequently within the set of k nearest neighbors.

3. Algorithm Implementation and Development

The provided MNIST data is split into training and testing data. To classify digits, we investigate the dimensionality of X_{train} using scikit learn's PCA function and plot the first 16 PCA modes as 28×28 images.

1. Investigate the dimensionality of X_{train}, apply PCA fit to data
2. Calculate the cumulative energy of the singular values obtained from PCA.
3. Determine the # of PCA modes to approximate X_{train} to 85% in the Frobenius norm.
4. Write a function that selects a subset of particular digits from X_{train}, y_{train}, X_{test} and y_{test} from returns the subset as new matrices $X_{subtrain}$, $y_{subtrain}$, $X_{subtest}$ and $y_{subtest}$
5. Train a Ridge classifier to distinguish two digits (example: 1 and 8)
   i. Extract subset of training and test data for selected digits.
   ii. Initialize PCA with k=59 components, fit and transform training data, and transform test data using the same PCA instance fitted on training data.
   iii. Fit Ridge Classifier on transformed training data and corresponding labels.
6. Report the training and testing MSE of the classifier and perform cross-validation
7. Repeat the same classification procedure for pairs of digits 3,8 and 2,7.
8. Perform multi-class classification with Ridge, KNN (for various n-neighbors), and LDA classifiers for all digits and report scores, and cross-validate. We use **StandardScaler** to scale the data for the KNN model.

Packages used: NumPy[2] and scikit-learn [6] for mathematical calculations and Matplotlib[3] for visualizations.

4. Computational Results

PCA is performed on the X_{train} data and the first 16 mode are plotted as 28×28 images2

We observe that 85% of the cumulative explained variance can be approximated with only 59 PCA modes. The first 64 images are plotted in the truncated PCA space (at k = 59) and it appears that the image reconstruction is reasonable.3

We train a Ridge classifier to distinguish two digits, report the training and testing MSE of the classifier, and perform cross-validation. The training accuracy indicates the model fits the training data very well. The testing accuracy is only about 2.3% lower than training accuracy, implying that there is some overfitting occurring, but it is relatively minor1. The low standard deviation in cross-validation shows performance is reliable across different data splits, no individual splits suffer from high variance.

Further (task 5), we see that the training, testing, and cross-validation accuracies are all higher for the (2,7) pair compared to the other two pairs. A possible reason is geometric similarity, similarities

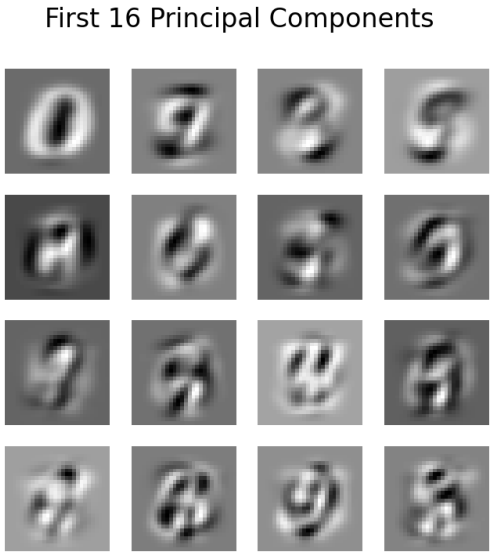
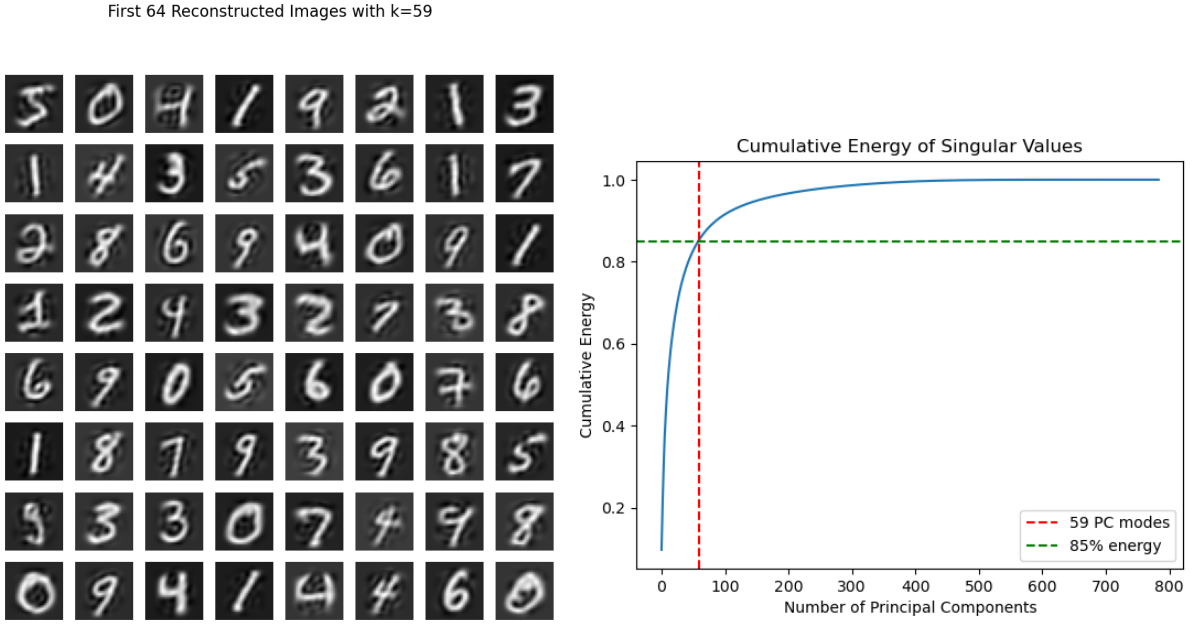


FIGURE 2. First 16 modes



(A) First 64 reconstructed images using k = 59 (B) Cumulative energy vs Number of Components (k)

FIGURE 3. Task2

between 2s and 7s are likely less than between 1s, 3s, and 8s. This allows an easier separation by the model. Possibly, 3 and 8 look more similar than 1 and 8, thereby making classifying these two digits more difficult.

Using all the digits and performing multi-class classification with Ridge, KNN, and LDA classifiers, we compare the results. LDA performs better than Ridge. LDA finds an efficient linear separation, so does better than Ridge but is still limited by linear boundaries2. KNN significantly outperforms the other two models, with testing accuracy $> 96\%$ and cross-validation accuracy $> 97\%$ for all values of k tested2.

Subset	Training score	Test score	cross-val accuracy, std. dev
1, 8	0.96521	0.94167	0.96323 , 0.00273
3, 8	0.96152	0.96370	0.95994 , 0.00551
2, 7	0.98093	0.97330	0.98061 , 0.00222

TABLE 1. Comparison of classifier results for different subsets

This demonstrates the effectiveness of a nonlinear model in separating the digits into 10 classes. A possible reason is due to the high model capacity i.e. KNN stores all the training data points and classifies new points based on nearest neighbors. This provides an extremely high model capacity to memorize fine details in the data.

Confusion matrices for the three classifier methods are shown in Figure 4

Classifier	Training score	Test score	cross-val accuracy
Ridge	0.84506	0.8562	0.84395
LDA	0.86651	0.8757	0.86471
KNN (k = 3)	0.9827	0.9612	0.97541
KNN (k = 4)	0.97833	0.9592	0.97445
KNN (k = 5)	0.97723	0.961	0.97468

TABLE 2. Comparison of results for different classifiers/methods

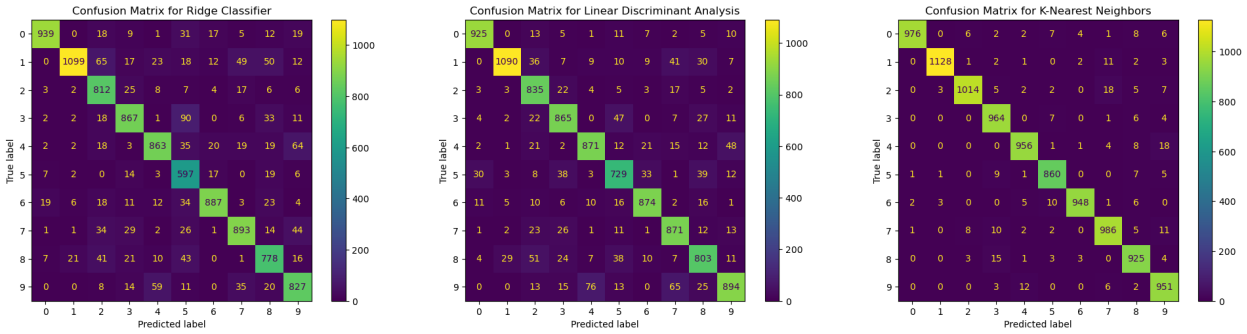


FIGURE 4. Confusion matrix for all 3 classifiers

5. SUMMARY AND CONCLUSIONS

We built a digit classifier to distinguish images of handwritten digits from the MNIST dataset. We used PCA to investigate the dimensionality of the features of the training data, which allowed us to conduct dimension reduction. We determined that to approximate X_{train} 85% in the Frobenius norm, we need the first 59 PCA modes. After extracting the features of our target digits, we use the Ridge regression model with cross-validation to train a predictor. There is some variation in MSE for the three sets of digit combinations we tested, which we hypothesize to be due to geometric differences in the digit pairs. We use different model fitting methods (Ridge, LDA, KNN) over all the digits to compare the results of each. We see that the k-nearest neighbors method (KNN) performs best, with the highest accuracy out of the methods.

Acknowledgements

The author is thankful for the suggestions/advice about the algorithm implementation from peers in AMATH 482/582. The author is thankful to the Teaching Assistants, Saba Heravi and Juan Felipe Osorio Ramírez, for their invaluable time and fruitful discussions about setting up the classifier. Further, the author is thankful to Prof. Eli Shlizerman for useful discussions about the course content.

References

[1] K-nearest neighbor(knn) algorithm. Accessed: 2024-01-24.
[2] 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.
[3] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.
[4] J. Kutz. *Methods for Integrating Dynamics of Complex Systems and Big Data*. Oxford, 2013.
[5] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010.
[6] 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.