AMATH 482/582: HOME WORK 3

REBECCA WANG

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

Abstract. This project explores handwritten digit classification using the MNIST dataset. Principal Component Analysis (PCA) is applied to reduce dimensionality while retaining essential features. Various classifiers, including Ridge Regression, K-Nearest Neighbors, and Support Vector Machines, are evaluated through cross-validation and test accuracy. Binary and multi-class classification tasks compare performance across different digit pairs. An alternative classifier is also introduced to assess its effectiveness. The study highlights the impact of dimensionality reduction and model selection on classification accuracy.

1. Introduction and Overview

Machine learning plays a crucial role in handwritten digit recognition, with the MNIST dataset being a widely used benchmark for evaluating classification algorithms. This project focuses on applying Principal Component Analysis (PCA) to transform high-dimensional image data into a more compact and meaningful representation. By leveraging PCA, we extract essential features while reducing computational complexity, ensuring efficient classification.

To assess classification performance, we implement Ridge Regression, K-Nearest Neighbors (KNN), and Support Vector Machines (SVM). The study includes both binary and multi-class classification tasks, comparing accuracy across different digit pairs and evaluating the impact of dimensionality reduction on model performance. Additionally, an alternative classifier is introduced to expand the comparison. Through cross-validation and test evaluations, this project highlights the effectiveness of different classifiers and the role of PCA in enhancing digit recognition.

2. Theoretical Background

2.1. Principal Component Analysis (PCA). Principal Component Analysis (PCA) is a **dimensionality reduction technique** that transforms high-dimensional data into a lower-dimensional space while preserving the most significant variance. It is commonly used for feature extraction and noise reduction. PCA finds new basis vectors (principal components) by computing the **eigenvectors** of the **covariance matrix** of the data. PCA can be computed using SVD: $X = U\Sigma V^T$, where the principal components are the first k columns of V. The explained variance ratio determines the number of principal components k needed to retain a significant portion (e.g., 85%) of the dataset's variance.

Application in this project: PCA is applied to reduce the original **784-dimensional** images into a smaller feature space. The first **16 principal components** are visualized as eigenimages. The **cumulative energy** of singular values determines the optimal number of components.

Date: March 3, 2025.

2.2. Ridge Regression (Ridge Classifier). Ridge Regression is a **linear classification model** that extends ordinary least squares regression by adding an **L2 regularization term**: $\lambda \sum_i \beta_i^2$ to prevent overfitting by shrinking the model coefficients towards zero. The L2 penalty encourages smaller coefficients, reducing the impact of less relevant features and improving generalization.

Objective Function:

$$\beta^* = \arg\min_{\beta} ||X\beta - y||_2^2 + \lambda||\beta||_2^2$$

where λ (alpha) is a hyperparameter controlling the penalty on large coefficients. The L2 regularization term, defined as:

Regularization Impact:

- **Small** $\lambda \rightarrow$ Low regularization, model is flexible but may overfit.
- **Large** $\lambda \rightarrow$ High regularization, reduces variance but may underfit.

Application in this project: Ridge regression is used for **binary classification** (digits 1 vs. 8, 3 vs. 8, 2 vs. 7). Cross-validation is used to determine the best **regularization parameter** λ.

2.3. K-Nearest Neighbors (KNN) Classifier. The **K-Nearest Neighbors (KNN)** algorithm is a **non-parametric classification method** that assigns a class label based on the majority class of the k nearest neighbors in the feature space. The most common distance measure is **Euclidean distance**:

$$d(x_i, x_j) = \sqrt{\sum_{m=1}^{d} (x_{im} - x_{jm})^2}$$

Effect of k:

- **Small** $k \rightarrow$ More sensitive to noise, high variance.
- **Large** $k \rightarrow$ Smoother decision boundaries, may underfit.

Application in this project: KNN is used for **multi-class classification** on the entire dataset. Performance is compared against Ridge Regression and Support Vector Machines.

2.4. Support Vector Machine (SVM). SVMs are **supervised learning models** that find the optimal decision boundary (hyperplane) maximizing the **margin** between different classes. **Linear SVM** is a hyperplane of the form:

$$f(x) = w^T x + b$$

where w is the weight vector and b is the bias. When data is not linearly separable, SVMs use **kernel functions** to project it into a higher-dimensional space. **Regularization Parameter (C):** Controls the trade-off between maximizing margin and minimizing classification error.

Application in this project: Used as an **alternative classifier** to compare against Ridge and KNN. Evaluated through **cross-validation and test accuracy**.

2.5. Cross-Validation. Cross-validation is used to **evaluate model performance** and **tune hyperparameters**.

K-Fold Cross-Validation means the dataset is split into k equal parts, and the model is trained on $k-1$ folds while tested on the remaining fold. The process is repeated k times, and the final accuracy is averaged.

Application in this project: 5-Fold Cross-Validation is used for Ridge, KNN, and SVM classifiers. Helps in **selecting the best hyperparameters** (e.g., best λ for Ridge Regression).

2.6. Accuracy Metrics. **Accuracy** is the primary evaluation metric in this project, defined as:

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}}$$

Application in this project: Used to compare **cross-validation accuracy** and **test accuracy**. Helps in assessing the effectiveness of different classification models.

2.7. Confusion Matrix. A **confusion matrix** provides a detailed breakdown of correct and incorrect classifications. It is structured as an $n \times n$ table where:

$$C_{ij} = \text{Number of instances with actual class } i \text{ but predicted as } j$$

The diagonal elements C_{ii} indicate correctly classified instances, representing the number of times each digit was correctly predicted. Higher values along the diagonal suggest better classification accuracy. Off-diagonal elements represent misclassifications, showing instances where the model confused one digit with another. The confusion matrix is plotted as a heatmap to highlight classification patterns, helping identify common misclassification errors and areas for model improvement. **Application in this project:** Confusion matrices are computed for Ridge Regression, KNN, and SVM to analyze classification performance. Helps analyze classification performance and error distribution. Heatmaps are used to visualize correct and incorrect predictions.

3. Algorithm Implementation and Development

The implementation of this project leverages the following Python packages for efficient computation and visualization:

- **NumPy** – Numerical operations and matrix manipulations.
- **Struct** – Handling binary file formats (MNIST dataset).
- **Matplotlib** – Visualization of digit images, PCA components, and model performance.
- **Pandas** – Data handling and cross-validation result management.
- **Scikit-learn** – Machine learning tools, including:
  - **PCA** – Dimensionality reduction.
  - **Ridge Classifier, KNN, SVM** – Classification models.
  - **GridSearchCV** – Hyperparameter tuning.
  - **StandardScaler** – Feature standardization.
  - **Accuracy Score & Confusion Matrix** – Model evaluation.
- **Seaborn** – Enhanced visualization of confusion matrices.

3.1. Data Preprocessing. The MNIST dataset is loaded and reshaped, with each 28×28 image flattened into a 784-dimensional vector. Standardization using `StandardScaler` ensures consistent feature scaling, improving the performance of distance-based classifiers like KNN and regularized models like Ridge Regression.

3.2. Principal Component Analysis (PCA). PCA is applied to reduce computational complexity by projecting data onto the first k principal components, capturing 85% of the total variance. This transformation retains key features while filtering noise. The first 16 principal components are visualized as 28×28 images to illustrate the dominant patterns in the dataset.

3.3. Digit Subset Selection. A function extracts specific digits from the dataset by selecting samples matching target labels from X_{train} and X_{test}. This enables focused binary classification tasks, such as distinguishing between digits 1 and 8.

3.4. Binary Classification with Ridge Regression. For binary classification, data is projected onto the reduced PCA space and classified using Ridge Regression. Cross-validation determines the optimal regularization parameter α, balancing bias and variance. The final model is evaluated on the test set to assess accuracy.

3.5. Comparison Across Digit Pairs. The classification procedure is repeated for different digit pairs (3,8) and (2,7) to compare classification difficulty. This analysis reveals how digit similarity affects model performance in a reduced-dimensional space.

3.6. Multi-Class Classification. Multi-class classification is performed using Ridge Regression and KNN. Grid search optimizes hyperparameters, and model performance is compared based on test accuracy to determine the most effective classifier.

3.7. Confusion Matrix and Error Analysis. Confusion matrices analyze misclassification patterns, providing insight into model weaknesses. Heatmaps generated with `seaborn` visualize classification errors, highlighting commonly confused digits.

3.8. Alternative Classifier SVM. An SVM classifier is implemented and evaluated against Ridge Regression and KNN using the same PCA-transformed dataset. By comparing test accuracy, the most effective classification approach is identified, demonstrating SVM's strengths in handling complex decision boundaries.

4. Computational Results

4.1. Principal Component Analysis (PCA). PCA was applied to the MNIST dataset to reduce dimensionality while preserving essential digit structures. Each image was reshaped into a 784-dimensional vector and transformed into a lower-dimensional space. Figure 1 presents the first 16 principal components, which reveal fundamental stroke patterns, and the cumulative explained variance plot, which shows the proportion of variance captured by each component. These visualizations illustrate how PCA efficiently compresses data while maintaining key discriminative features.

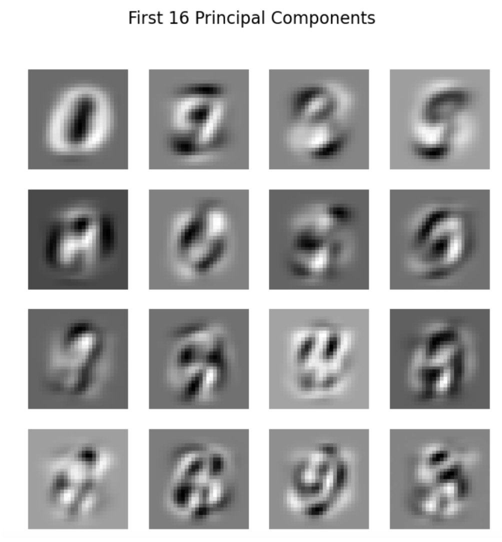


FIGURE 1. First 16 Principal Components

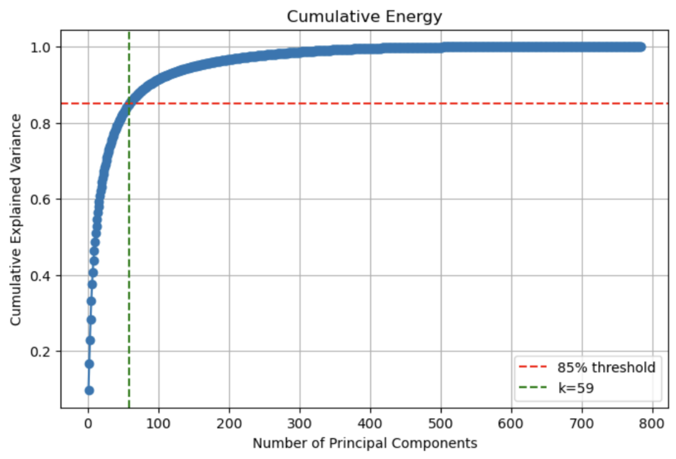


FIGURE 2. Cumulative Energy

4.2. Cumulative Energy and Image Reconstruction. To determine an optimal dimensionality reduction, the cumulative explained variance was analyzed. Figure 2 shows that $k = 59$ principal components retain 85% of the total variance, striking a balance between data compression and information retention. The reconstructed images in Figure 3 illustrate that even with reduced dimensions, digit structures are well-preserved, confirming PCA's ability to maintain key features while filtering noise.

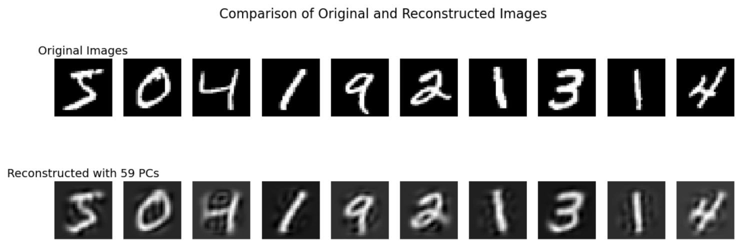

Comparison of Original and Reconstructed Images
Original Images
Reconstructed with 59 PCs

FIGURE 3. Comparison of Original and Reconstructed Images

4.3. Subset Selection of Digits. A subset of the dataset was extracted to focus classification on specific digits. The function filters the training and test datasets to retain only the selected digits. Given the selection of digits {1, 8}, all corresponding samples were extracted from X_{train}, y_{train}, X_{test}, and y_{test}, forming new subsets X_{subtrain}, y_{subtrain}, X_{subtest}, and y_{subtest} as:

$$X_{\text{subtrain}} : (12593, 784), \quad y_{\text{subtrain}} : (12593), \quad X_{\text{subtest}} : (2109, 784), \quad y_{\text{subtest}} : (2109)$$

4.4. Classification of Digits 1 and 8. A Ridge classifier was used to distinguish between digits 1 and 8, with PCA reducing the dataset's dimensionality. The training and test data were projected onto the first k principal components, and hyperparameter tuning was conducted using cross-validation over different values of the regularization parameter α. The best cross-validation accuracy achieved was 0.9643, and the final test accuracy was 0.9801, indicating strong generalization. Performance remains stable across different values of α, demonstrating the robustness of Ridge regression in this classification task.

4.5. Classification of Digit Pairs. The Ridge classifier was applied to digit pairs (3,8) and (2,7) to compare classification performance. As shown in Figures 4 and 5, accuracy remains stable across different regularization values, confirming the model's robustness. Table 1 shows that digits (2,7) achieved higher accuracy compared to (3,8), suggesting that (2,7) are more distinct in the reduced PCA space, while (3,8) may have overlapping features, making classification slightly harder.

Digit Pair	**Best CV Accuracy**	**Test Accuracy**
(3,8)	0.9588	0.9637
(2,7)	0.9800	0.9748

TABLE 1. Classification performance for digit pairs (3,8) and (2,7)

4.6. Multi-Class Classification with Ridge and KNN. Multi-class classification was performed using Ridge and KNN classifiers to evaluate their effectiveness on the full MNIST dataset. The Ridge classifier maintained stable performance across different regularization values, as shown in Figure 6, while the KNN classifier achieved the highest accuracy at $k = 3$ before declining with larger neighbor values. The test accuracy for Ridge was 0.8553, with a cross-validation accuracy of 0.8436, whereas KNN outperformed it with a test accuracy of 0.9755 and a cross-validation accuracy of 0.9752, demonstrating that non-parametric methods can be more effective when sufficient training data is available.

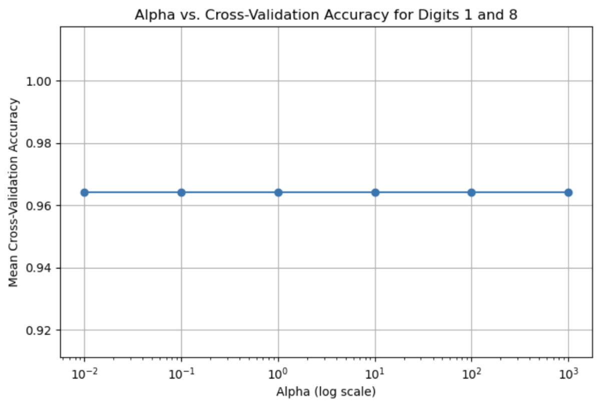


Figure 4. Cross-validation accuracy for digits 1 and 8

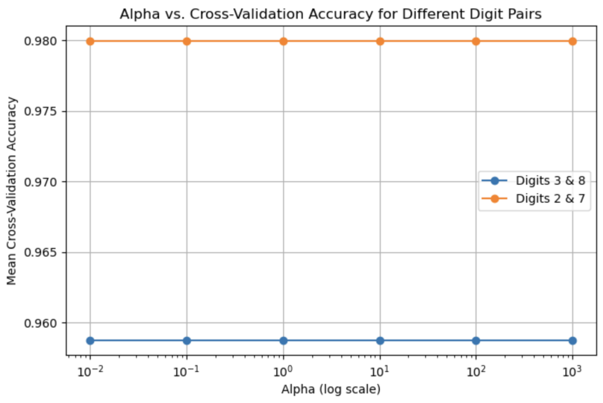
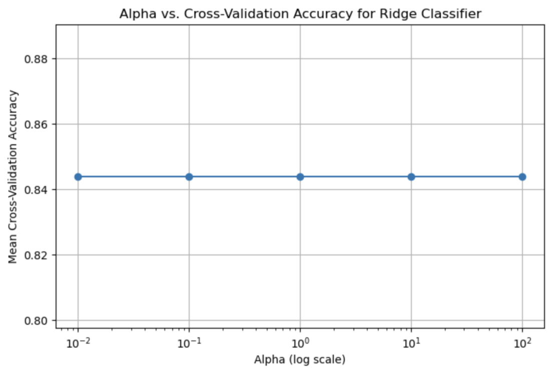
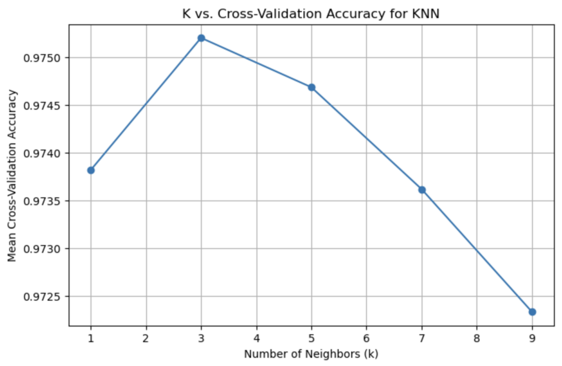


Figure 5. Cross-validation accuracy for digit pairs (3,8) and (2,7)



Ridge Classifier



KNN Classifier

Figure 6. Comparison of Ridge and KNN classification performance

4.7. Comparison with SVM. To further improve performance, an SVM classifier was implemented and compared with Ridge and KNN. SVM outperformed both, achieving a test accuracy of 0.9840. Figure 7 presents confusion matrices for each classifier. Ridge misclassified more instances, particularly in overlapping digit classes, while KNN improved accuracy but still exhibited occasional confusion. SVM provided the highest accuracy with fewer misclassifications, confirming its superior capability in handling complex decision boundaries.

5. Summary and Conclusions

This project explored handwritten digit classification using PCA for dimensionality reduction and compared the performance of Ridge Regression, KNN, and SVM classifiers. PCA effectively compressed the data while preserving key features, enabling efficient classification. Ridge Regression demonstrated stable performance but was outperformed by KNN, which achieved higher accuracy when the optimal number of neighbors was selected. SVM further improved classification results, achieving the highest test accuracy and demonstrating superior handling of complex decision boundaries. The findings highlight the importance of dimensionality reduction, model selection, and hyperparameter tuning in optimizing classification accuracy for handwritten digits.

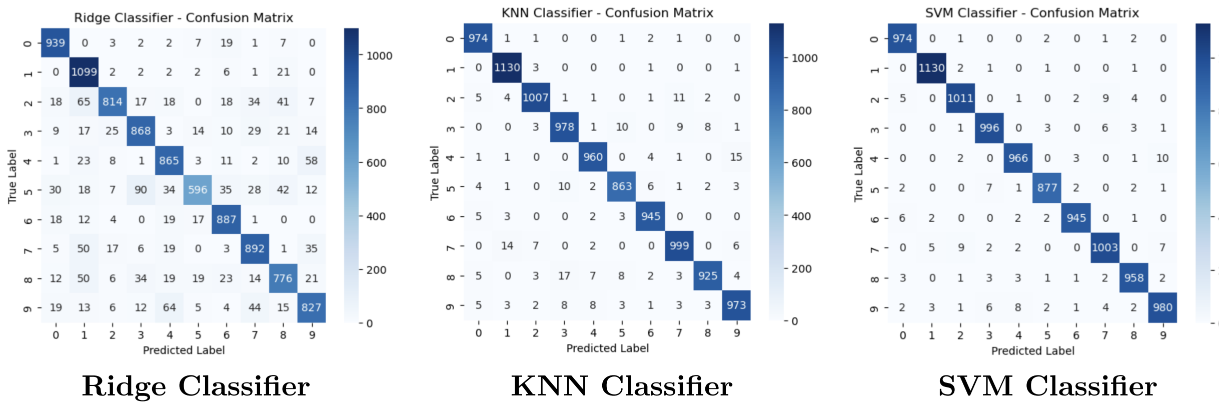


Figure 7. Comparison of confusion matrices for Ridge, KNN, and SVM classifiers

Acknowledgements

This author thanks Professor Frank for providing useful information to tackle the problem through the hints in the assignment, as well as the HW3-Helper.ipynb code. The author is also grateful to TA Rohin Gilman for providing theoretical background to assist in constructing my code. Finally, my peers Liam Lin and Zhaohan Yin were also helpful when I ran into issues related to coding.

References

[1] J.N. Kutz, *Methods for Integrating Dynamics of Complex Systems and Big Data*, Oxford, 2013.
[2] N. Frank, *Lecture 11 Notes*, Feb 2 2025.
[3] N. Frank, *Homework 1-TA Slides*, January 2025.