AMATH 482 Homework 3

Trevor Ruggeri

Due: March 7, 2025

This report presents an analysis of the MNIST dataset using various machine learning techniques. We perform Principal Component Analysis (PCA) to reduce dimensionality and use Ridge and K-Nearest Neighbors (KNN) classifiers for classification tasks. The effectiveness of these methods is evaluated and compared. Additionally, we implement an alternative classifier, the Support Vector Machine (SVM), and compare its performance with the previous classifiers. Results indicate that dimensionality reduction via PCA combined with appropriate classifiers can effectively classify handwritten digits with high accuracy.

1 Introduction and Overview

The MNIST dataset, comprising 70,000 images of handwritten digits, is a benchmark in the field of machine learning. This project explores various techniques to classify these images accurately. We employ Principal Component Analysis (PCA) for dimensionality reduction and use Ridge and K-Nearest Neighbors (KNN) classifiers for classification tasks. Additionally, we implement a Support Vector Machine (SVM) classifier and evaluate its performance.

2 Theoretical Background

2.1 Principal Component Analysis (PCA)

PCA is a statistical technique that transforms high-dimensional data into a lower-dimensional space while preserving as much variance as possible. The principal components are orthogonal vectors that capture the most variance in the data.

Steps:

1. **Standardization:** Standardize the data to have a mean of zero and a standard deviation of one:
$$\bar{X} = \frac{X - \mu}{\sigma},$$
where X is the original data matrix, μ is the mean of the data, and σ is the standard deviation.

2. **Calculate Covariance Matrix:** Compute the covariance matrix of the standardized data $\bar{X}$:
$$\text{Cov}(\bar{X}) = \frac{1}{n-1}\bar{X}^T\bar{X},$$
where n is the number of data samples.

3. **Eigenvalues and Eigenvectors:** Calculate eigenvalues and eigenvectors of the covariance matrix, i.e. the vectors v_i and scalars λ_i such that
$$\text{Cov}(\bar{X})v_i = \lambda_i v_i.$$

4. **Projection onto Principal Components:** Project the original data onto the principal components to obtain the new feature set.

$$Z = \bar{X}V,$$

where V is the matrix of eigenvectors.

2.2 Ridge Classifier

The Ridge classifier is a linear classifier that applies L2 regularization to prevent overfitting. It minimizes the sum of squared residuals while adding a penalty for large coefficients.

Mathematics:

1. **Objective Function:**

$$\text{Loss}(\omega) = \frac{1}{2n}\sum_{k=1}^{n}(y_k - \omega \cdot x_k)^2 + \alpha||\omega||^2,$$

where n is the number of samples, y_k are the target values, x_k are the feature vectors, ω is a vector of weights, and α is the regularization parameter.

2. **Optimization:** Solve the objective function and find the optimal weights.

$$\omega^* = \text{argmin}(\text{Loss}(\omega))$$

2.3 K-Nearest Neighbors (KNN) Classifier

KNN is a non-parametric classifier that classifies a data point based on the majority class among its k-nearest neighbors in the feature space.

Mathematics:

1. **Distance Calculation:** Calculate the Euclidean distance between the query point and all points in the training set:

$$d(x_q, x_i) = \sqrt{\sum_{k=1}^{d}(x_{q_k} - x_{i_k})^2},$$

where x_{q_k}, x_{i_k} are the k-th features of the query point and the i-th training point, respectively.

2. **Neighbor Selection:** Select the k-nearest neighbors based on the calculated distances.

3. **Majority Voting:** Assign the class label based on the majority class among the k-nearest neighbors.

2.4 Support Vector Machine (SVM)

SVM is a powerful classifier that finds the optimal hyperplane separating different classes by maximizing the margin between them. It can handle both linear and non-linear data using kernel functions.

Mathematics:

1. **Hyperplane Calculation:** The hyperplane is defined by the equation:

$$\omega \cdot x + b = 0,$$

where ω is the weight vector and b is the bias.

2. **Margin:** The margin is defined as the distance between the hyperplane and the nearest data points (support vectors).

$$\text{Margin} = \frac{2}{||\omega||}$$

3. **Optimization Problem:** To maximize the margin, minimize $||\omega||$ subject to the constraints that all data points are correctly classified:

$$\min \frac{1}{2}||\omega||^2$$

subject to $y_i(\omega \cdot x_i + b) \geq 1$ for all i.

3 Algorithm Implementation and Development

3.1 Data Preparation and PCA Analysis

1. **Reshape Images**: Convert the 28x28 images into 1D vectors and stack them into matrices for training and testing using `np.reshape`.
2. **Perform PCA Analysis:** Apply PCA to the training set to reduce dimensionality while retaining most of the variance, using `PCA` from `sklearn.decomposition`
3. **Examine Cumulative Energy:** Analyze the cumulative energy of the principal components to determine the number of components k needed to capture 85% of the variance, using `np.cumsum` for this calculation.
4. **Reconstruct Images Using Truncated PC Modes:** Verify the quality of image reconstruction using the first k principal components, using `inverse_transform` from `PCA` to project back into the real-world space.

3.2 Subset Selection Function

Create a function that selects samples of specific digits from the training and testing sets, using `np.isin` to identify specific digits in our training and testing datasets.

3.3 Classification and Cross-Validation

Extract subsets of the training and testing data with the digits 1 and 8 using the function above, project the data onto k-PC modes, train a Ridge classifier on the subset, and evaluate the model using cross-validation. Repeat this process with the pairs of digits 3,8 and 2,7. To implement this, `RidgeClassifier` from `sklearn.linear_model` and `cross_val_score` from `sklearn.model_selection` were used.

3.4 Multi-Class Classification with Ridge Regression and KNN

Apply Ridge and KNN classifiers to the entire k-PC mode projected dataset and evaluate their performance. To implement this, `RidgeClassifier` from `sklearn.linear_model` and `KNeighborsClassifier` from `sklearn.neighbors` were used.

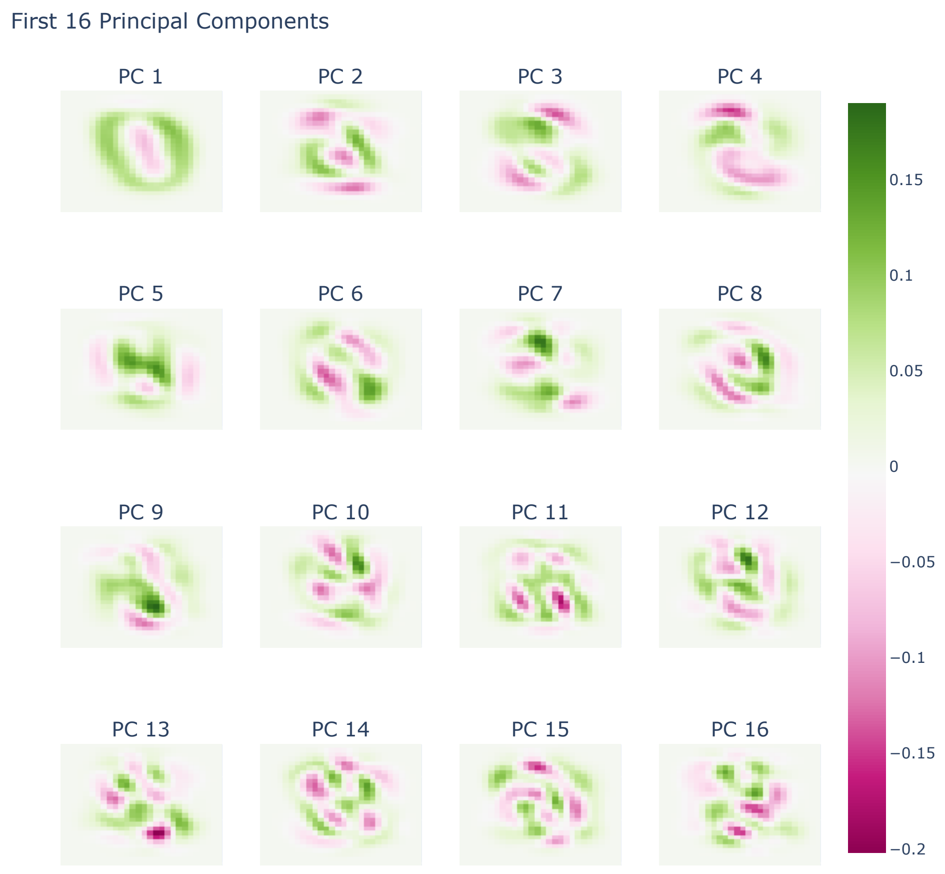
3.5 Alternative Classifier (SVM)

Use an SVM classifier as an alternative method and compare its results with Ridge and KNN classifiers. To implement, `SVC` from `sklearn.svm` was used.

4 Computational Results

Graphed below are the first 16 PC modes.

4.1 PCA Results:



Loading [MathJax]/extensions/MathMenu.js

Figure 1: First 16 Principal Components

Writing a function `find_k_for_energy` to calculate the number of modes necessary to approximate 85% of the cumulative energy, we calculated this number of modes to be $k = 59$. Images of 8 reconstructed digits from 59 truncated PC modes are shown in Figure 2 below.

4.2 Subset Selection and Classification Results

Using the function `select_digit_subset`, we were able to select subsets of the training and testing data with the digits 1 and 8 and project them into k-mode PCA space.

- Average Cross-Validation Accuracy for Digits 1,8: 0.9643
- Test Accuracy for Digits 1,8: 0.9801

We then repeated the same process for the pairs of digits 3,8 and 2,7.

- Average Cross-Validation Accuracy for Digits 3,8: 0.9587
- Test Accuracy for Digits 3,8: 0.9642
- Average Cross-Validation Accuracy for Digits 2,7: 0.9797

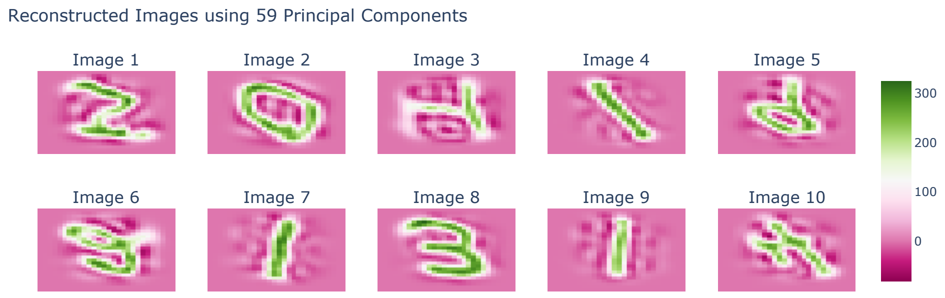


Figure 2: First 10 Reconstructed Images

- Test Accuracy for Digits 2,7: 0.9748

From the results above, we can note that the accuracy of the Ridge Classifier on the test subsets for the pairs 3,8 and 2,7 are slightly lower than for the pair 1,8. Intuitively, this makes sense, as the digits 1 and 8 seem more distinguishable to the human eye than 3,8 or 2,7.

4.3 Multi-Class Classification Results

- **Ridge Classifier Multi-Class Cross-Validation Mean Accuracy:** 0.8436
- **Ridge Classifier Multi-Class Test Accuracy:** 0.8563
- **KNN Classifier Multi-Class Cross-Validation Mean Accuracy:** 0.9747
- **KNN Classifier Multi-Class Test Accuracy:** 0.9755

4.4 Alternative Classifier (SVM) Results

- **SVM Cross-Validation Mean Accuracy: 0.9190**
- **SVM Test Accuracy: 0.9293**

Even though the SVM classifier was trained on the scaled training data, rather than projecting down onto k-mode PCA space, both the cross-validation mean accuracy and testing accuracy were less than the KNN classifier's accuracies on the same respective datasets, which points to the robustness and effectiveness of both PCA dimensionality reduction and KNN classification. The SVM classifier did perform better than the Ridge classifier in both catagories, however the computational cost is worth noting, as the SVM classifier trained on the entire scaled dataset took over 160 times longer to train and test than the Ridge and KNN classifiers.

5 Summary and Conclusions

This project demonstrated the effectiveness of PCA for dimensionality reduction and its impact on classification accuracy using Ridge and KNN classifiers. The results indicate that PCA can significantly reduce the dimensionality while retaining most of the relevant information. Both Ridge and KNN classifiers performed well, with KNN outperforming the Ridge classifier in the multi-class classification. The SVM classifier, as an alternative approach, showed competitive results and reinforced the robustness of SVM in handling classification tasks.

5.1 Acknowledgements

I would like to thank Rohin Gilman for their guidance and support throughout this project. Special thanks to my classmates for their valuable discussions and feedback.

5.2 References

1. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

2. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

3. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning. Springer.

4. Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. MIT Press.