USE MULTIPLE CLASSIFIERS TO DISTINGUISH HANDWRITTEN DIGITS FROM MNIST DATASET

EDRIA LI

Applied Mathematics Department, University of Washington, Seattle, WA
yijueli@outlook.com

Abstract. We successfully trained the classifiers to distinguish images of handwritten digits from the MNIST dataset. We also visualized the digit features in lower PCA modes and in confusion matrix. Comparison of different pairs of digits were performed. Ridge Classifier, LDA, KNN, and SVM were performed to test accuracy of training; cross validation of the above-mentioned methods were performed as well to see the robustness of each classifier.

1. Introduction and Overview

We are provided with Yann Lecun‘s MNIST data of images of handwritten digits that has been split into training and test sets. The training set is of size 784×60000 where each digit is represented in pixels of size 28×28.

Our task is to train different classifiers to distinguish the images of digits and provide analysis for the results. Given the sheer volume of data, we first used `sklearn` PCA method to visualize the first 16 PCA modes for the effect of dimensional reduction. We then calculated and compared the preserved energy of different PCA modes for our supervised learning. We observed that when $k = 59$, the cumulative energy ratio reaches 85.02%. We then reconstructed the projected data and visualized the digits. We first examined digit pairs for binary linear training. Pairs of 1, 8, 2, 7, 3, 8 were investigated when `RidgeClassifer` were used for linear regression. We finally explored multi-class classifiers to distinguish all the digits; `RidgeClassifierCV`, `KNeighborsClassifier` (KNN), and `LinearDiscriminantAnalysis` (LDA) were tried and distinguished accuracy rates for each method was documented. Moreover, Support Vector Machines (SVM) method was used where both `SVC` and `LinearSVC` were investigated. Based on all the training methods explored, we found that KNN achieved highest accuracy around 96% with standardized data among Ridge, KNN, and LDA. Further, if standardized data were used for training, LinearSVC may converge faster than using merely projected data. Non-linear SVM achieved the highest accuracy rate of 98.33% among all methods tried.

2. Theoretical Background

2.1. **Principle Component Analysis (PCA).** PCA is to dimensionally reduce a large data sets by projecting the large dataset into lower dimension along its principle axis. PCA projection can be very useful to represent data matrix $\mathbf{X}$ in a new basis of U (i.e. the left singular vectors as new basis) where the full projection is: $U^\top \cdot \mathbf{X} = \Sigma V^\top$. To lower the dimensions of $\mathbf{X}$, we further truncate the new basis by keeping the first k columns of U and setting all remaining columns to zero, i.e. U_k, where k denotes the modes of PCA: $U_k^\top \cdot \mathbf{X} = \widetilde{\mathbf{X}}$.

2.2. **Energy and Frobenius Norms.** The following cumulative energy approximation is used to estimate how much energy do we still have in our representation of projected matrix using Frobenius norm: $\sum E_j = \sum_{k=1}^{j} \frac{\sigma_j^2}{E}$, $E = \|A\|_F$, where $\|A\|_F = \|\Sigma\|_F$.

2.3. **Model Background.**
RidgeClassifier. RidgeClassifier is a classifier using Ridge regression, which addresses the least squares problem by imposing a penalty on the size of the coefficients [1]. The ridge coefficients minimize a penalized residual sum of squares: $\min_\beta \|A\beta - y\|_2^2 + \lambda\|\beta\|_2^2$. The solution is $\beta = (A^\top A + \lambda I)^{-1}A^\top y$.
LDA. LDA is a classifier with a linear decision boundary based on logistic regression with Sigmoid logistic function $f(x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}}$, which is also to solve min $\|f(x) - y\|_2^2$.
KNN. KNN is K nearest neighbors to compute distance of test points to the trained data and found the nearest neighbor. Euclidean distance is usually used.
SVM. A Support Vector Machine (SVM) constructs a hyperplane in a high or infinite dimensional space, which can be used for classification [1]. Given training vectors $x_i \in \mathbb{R}^p$, $i = 1, \cdots, n$ in two classes, and a vector $y \in \{1, -1\}^n$, the goal is to find $w \in \mathbb{R}^p$ and $b \in \mathbb{R}$ such that the prediction given by $\text{sign}(w^\top \phi(x) + b)$ is correct for most samples. SVC solves the primal problem: $\min_{w,b,z} \frac{1}{2} w^\top w + C\sum_{i=1}^{n} z_i$ subject to $y_i(w^\top \phi(x_i) + b) \geq 1 - z_i, z_i \geq 0, i = 1, \cdots, n$.

3. Algorithm Implementation and Development

we used `Python` in which we import `numpy`, `struct`, `PCA`, `RidgeClassifierCV`, `KNeighborsClassifier`, `LinearDiscriminantAnalysis`, `LinearSVC` and `SVC`, `cross_val_score`, and `StandardSCaler` for computing; we also import `matplotlib` and `ConfusionMatrixDisplay` for plotting.

Here are the detailed algorithms as shown in Algorithm 1 to 7:

Algorithm 1 First 16 PCA Modes

```
1: Load MNIST training set $X_{\text{train}}$ of size $784 \times 60000$.
2: Compute PCA of $X_{\text{train}}^\top$ with 16 components using PCA from sklearn.decomposition
3: Plot the first 16 Modes.
```

Algorithm 2 Cumulative Energy Ratio

```
1: Apply PCA over 784 modes to obtain singular value matrix $\Sigma$
2: for $j = 50, \ldots, 59$ do
3:     Truncate $\Sigma$ by setting $(j+1)$-th and on element to zero as $\widetilde{\Sigma}_j$
4:     Compute the power ratio by $E_{\text{ratio}} = \frac{\|\widetilde{\Sigma}_j\|_F^2}{\|\Sigma\|_F^2}$.
5:     Plot the ratio.
6: end for
7: Reached 85% ratio when $k = 59$
```

Algorithm 3 Reconstruct Training Images Using 59 PCA Modes

```
1: Use PCA mode when $k = 59$, compute PCA of $X_{\text{train}}^\top$ with 59 components as $X_{\text{proj}}$
2: Reconstruct the $X_{\text{proj}}$ to original size and stored as $X_{\text{train\_recon}}$
3: Restore the result by transpose $X_{\text{train\_recon}}$
4: Plot first 64 reconstructed training images
```

Algorithm 4 Use RidgeClassifier for Training

```
1: Use RidgeClassifierCV model on the desired training set
2: Obtain accuracy rate on test set
3: Use cross_val_score and set parameter $cv = 5$
4: cross-validate on desired training data
5: Use scores.mean() and scores.std() to obtain cross-validation result of mean accuracy and
   standard deviation respectively.
6: Plot Confusion matrix
```

Algorithm 5 Use LDA for Training

```
1: Import from sklearn.discriminant_analysis
2: Use LinearDiscriminantAnalysis model on the projected training set
3: Obtain accuracy rate on test set
4: Cross validate the model same as in Algorithm 4
```

Algorithm 6 Use KNN for Training

```
1: Standardized the training data and test data by StandardSCaler
2: for $k_{\text{value}} = 1, \dots, 10$ do
3:     Compute KNeighborsClassifier model by setting parameter n_neighbors $= k_{\text{value}}$
4:     Cross validate the model same as in Algorithm 4
5:     append the average accuracy to the scores list
6: end for
7: Plot the accuracy score for each $k_{\text{value}}$
8: Set the $k_{\text{value}}$ to $k_{\text{opt}}$ with argmax accuracy and Use the $k_{\text{opt}}$ as the parameter to test model
```

Algorithm 7 Use SVM for Training

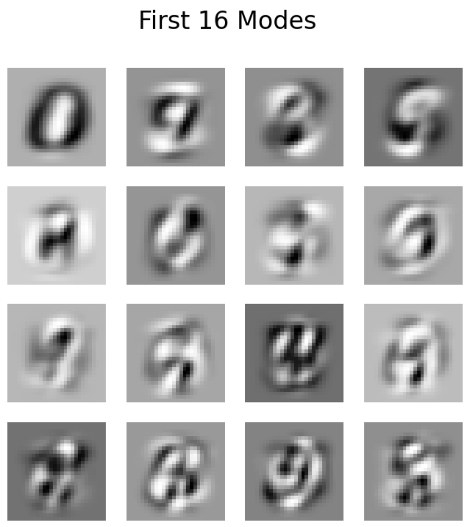
```
1: Standardized the training data and test data by StandardSCaler
2: Use LinearSVC (linear) and SVC (non-linear) model on the standardized projected training set
3: Obtain accuracy score on the standardized test set
4: Cross validate the model same as in Algorithm 4
```

4. Computational Results

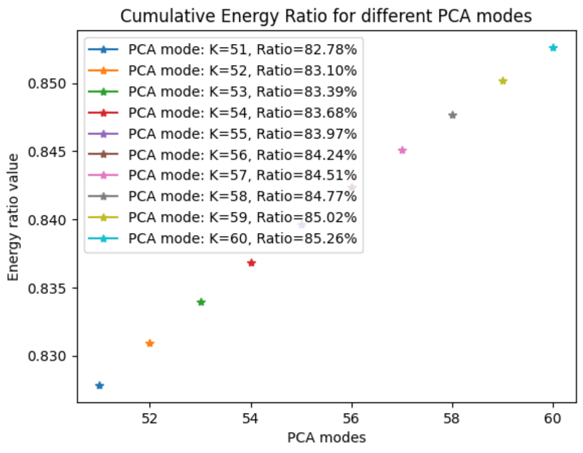
4.1. **Plot First 16 PCA modes.** We first loaded the training data files into one matrix X_{train} and applied `PCA` with `n_components` $= 16$. We can see the plot after reshaping the 16 features into 28×28 subimages of 16 as shown in Figure 1a.

4.2. **Cumulative Energy Ratio.** We tried out different k values of PCA modes and found out that when $k = 59$ the projected matrix can preserve cumulative Frobenius norm above the threshold of 85% as shown in Figure 1b with a ratio of 85.02%. We will use this k value for the following analysis.

4.3. **Reconstruction of Training Images.** We then project the training set where PCA modes $k = 59$ and inversely transform the training images to observe image resolution. The resulting reconstructed image is shown in Figure 2a as compared to our original training images as shown in Figure 2b. We can clearly see that the reduced dimension preserved the image quality pretty well with all digits represent their critical features.

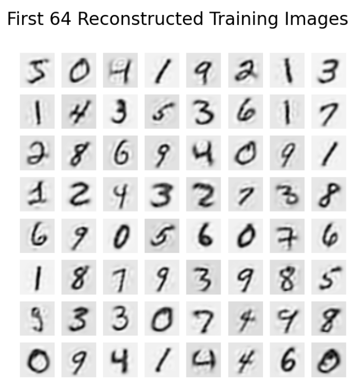


(A) First 16 Modes of PCA.



(B) Energy Ratio for Different PCA Modes

FIGURE 1. 1a First 16 Modes of PCA. 1b Energy Ratio for Different PCA Modes.



(A) First 64 Reconstructed Images

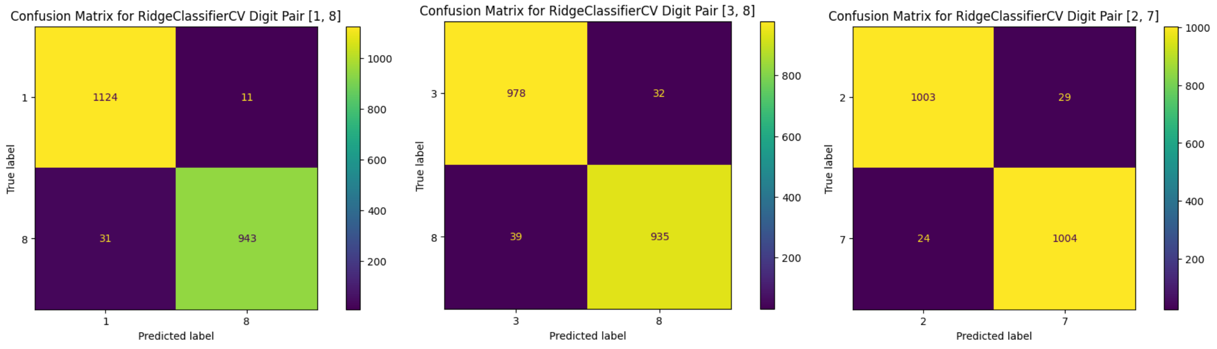
(B) First 64 Original Images

(C) First 64 of Digit Pair {1,8}

FIGURE 2. 2a First 64 Reconstructed Training Images with 59 modes; 2b First 64 Original Training Images with Full Features; 2c First 64 Images of Digit Pair {1,8} from Original Images.

4.4. **Binary Classification of Digit Pair {1,8},{2,7} and {3,8}.** We split the training and test dataset and first observed binary classification result of digit pairs $\{1,8\}, \{2,7\}$, and $\{3,8\}$. For pairs $\{1,8\}$, we tested and plot our first 64 images from the subdata set with assurance as shown in Figure 2c. We used Algorithm 4 for training and obtained test accuracy of 98% with cross-validation mean accuracy of 96.43% and a standard deviation $\sigma = 0.0028$. The Confusion matrix as in Figure 3a shows that 1124 cases of 1 and 943 cases of 8 are rightly distinguished with merely 11 cases of 1 and 31 cases of 8 wrongly classified.

We further tested out our binary classification for pairs {2,7} and {3,8} and observed that test accuracy lowered to 97.43% for {2,7} (cross-validation mean 97.97% with $\sigma = 0.00156$) and further lowered to 96.42% for {3,8} (cross-validation mean 95.82% with $\sigma = 0.0061$) where confusion matrix for each pair as shown in Figure 3b and 3c . We expected such a drop in accuracy due to the fact that features of {3,8} and {2,7} are far closer and more ambiguous in differences than distinguished features of {1,8}.



(A) Confusion Matrix for {1,8} (B) Confusion Matrix for {3,8} (C) Confusion Matrix for {2,7}

FIGURE 3. The figure above shows the Confusion Matrix for different digit pairs with 3a for {1,8}, 3b for {3,8}, and 3c for {2,7}, respectively.

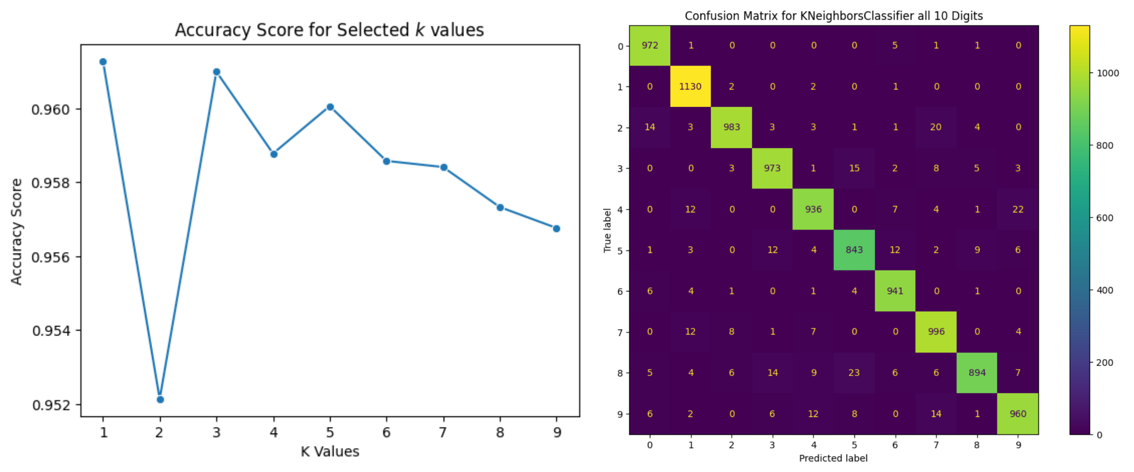
4.5. **Multi-Class Classification Using Ridge, LDA, KNN.** We further explored the multi-class classification methods to distinguish all 10 digits in RidgeClassifier (Algorithm 4), LDA (Algorithm 5), and KNN (Algorithm 6). For uniform comparisons, we standardized all projected training data for results interpretation. We listed our results in the following table: We examined

Method Name	**Test Accuracy**	**Cross-Validation Accuracy**
RidgeClassifierCV	85.51%	$\mu = 84.38\%$, $\sigma = 0.0094$
LDA	87.54%	$\mu = 86.53\%$, $\sigma = 0.0086$
KNN ($k_{\text{opt}} = 1$)	96.28%	$\mu = 96.13\%$, $\sigma = 0.0021$
Linear SVM	90.63%	$\mu = 89.79\%$, $\sigma = 0.0061$
Non-linear SVM	98.33%	$\mu = 98.03\%$, $\sigma = 0.0021$

TABLE 1. Test Result for 5 Multi-class Method. For KNN model, we observed optimal accuracy when parameter of neighbors $k = 1$. The highest accuracy was achieved through non-linear SVM model with test accuracy 98.33%.

our best-result-method KNN (within the three methods of Ridge, LDA and KNN) in more depth below. First, with trial and error, we examined different values of k, i.e. number of neighbors. A little bit to our surprise, the result shows that when $k = 1$ the accuracy score reached maximum (Figure 4a). We expected k values to be around 3, but its accuracy score showed a shy of 0.0283% (96.1%) compared with our optimal k (96.1283%). We suspect that given we tested on projected test data instead of original test data, we will need a smaller diameter for good neighbors. Confusion matrix for 10-class classification is shown in Figure 4b as we can see from our intuition, digits with some similarities in features will have more errors in classification, for example, for digit pairs {3,5}, {2,7}(as we already observed before),{4,9}, and {5,8}. Moreover, we can see some trend of symmetry in miss-classified pairs from the confusion matrix which echoed our intuition as well.

4.6. **Multi-Class Classification Using SVM.** We finally trained our model using SVM and we examined both linear and non-linear models using Algorithm 7. We first tried the linear model *without* preprocessing the data, we observed a test accuracy of 81.17% with cross-validation mean accuracy of 78.31% and $\sigma = 0.0187$, lowest accuracy and highest cross-validation standard deviation among all methods used. Moreover, the runtime using `Python` on personal laptop needed about 14 minutes to go through projected data. We then preprocessed the data using StandardSCaler and saw a huge improvement in runtime, the result converge way faster with merely 17 seconds. Moreover, the test accuracy grows to 90.63 % (other details see Table 1). Such huge benefit of



(A) KNN Accuracy for Selected Parameter Value (B) Confusion Matrix for KNN for All Digits

FIGURE 4. The two figures above showed our best-result model for multi-class classification. We reached 96.28% accuracy result using KNeighborsClassifier (KNN) with optimal k value of 1. 4a is the accuracy score for selected k values from 1 to 9, and 4b shows the confusion matrix for 10 digits comparison.

standardizing data before training the model explains the scaling effect which may lead to disastrous outcome if omitted. The best accuracy score finally come from the non-linear SVM where overall test accuracy reached 98.33% with cross-validation mean accuracy $\mu = 98.03\%$ and $\sigma = 0.0021$.

5. Summary and Conclusions

To distinguish the digit images from MNIST dataset, we first dimensionally reduced the features to preserve at least 85% energy of the original data. We examined binary classification for three digit pairs. We then examined 5 different training models (Ridge, LDA, KNN, SVM (both linear and non-linear)) in detail for multi-class classification. We successfully distinguished 10 digits and visualized through confusion matrix. We observed highest accuracy of KNN (96.28%) for the first three models, and non-linear SVM (98.33%) for all models examined.

Acknowledgements

The author is thankful to Prof. Eli for useful discussions and code samples about different classification methods such as LDA, RidgeClassifierCV, and KNN.

References

[1] 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.