Handwritten Digit Classification via SVD

Student: Aditya Mittal
University of California, Davis
adimittal@ucdavis.edu

Professor: Luze Xu
Course: MAT 167 - Applied Linear Algebra
Department of Mathematics
University of California, Davis
lzxu@ucdavis.edu

December 10, 2023

1 Abstract

This report investigates the application of Single Value Decomposition (SVD) in handwritten digit classification, comparing the accuracy of predicted outcomes across various low-rank approximations and between conventional machine learning methods such as Logistic Regression and Support Vector Machines. The study utilizes the usps.mat dataset and the results indicate that SVD, particularly at rank k = 17, performs extremely well along logistic regression and support vector machines in achieving a high classification accuracy of greater than apporximately 95%. The report concludes by highlighting the efficacy of SVD in handwritten digit classification and its potential advantages over conventional methods.

2 Introduction

The classification of handwritten digits is a very important task in machine learning, and serves as a foundational problem in computer vision and pattern recognition applications. Its significance is underscored by its broad range of practical uses, including character recognition in postal services, the development of automated form processing systems, and more.

To achieve high accuracy in predictions, many methods including logistic regression, K-Nearest Neighbors, Random Forests, and Neural networks have been

developed and are widely used today. In addition to these conventional machine learning models, this report explores the application of Single Value Decomposition (SVD) in handwritten digit classification. SVD is a matrix factorization method that decomposes a matrix into three sub-matrices, capturing the inherent structures and patterns with the data. SVD has found extensive applications in signal processing, image compression, and the Pagerank algorithm.

The report explores a technique called low-rank approximation, leveraging SVD to approximate the matrices with a lower-ranked counterpart. The idea behind low-rank approximation is to preserve the top-k singular values and singular vectors, where k is less than the original rank of the matrix. By employing this approach to approximate the dataset matrix, we aim to keep the most important features and reduce computational complexity.

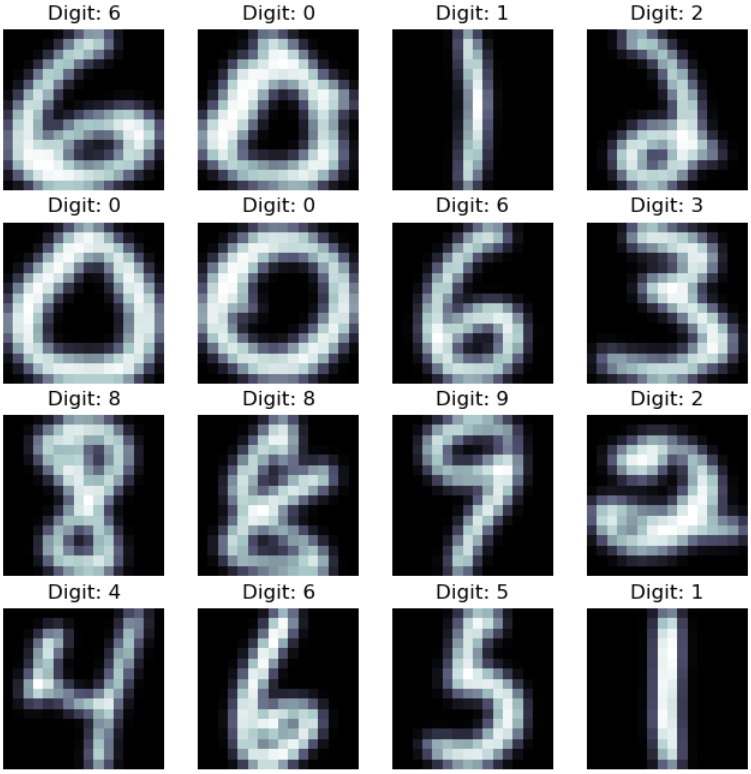
The overarching goal of this report is to classify handwritten digit predictions and compare classification accuracy rates, specifically comparing the impact of low rank approximation using SVD on classification outcomes. Results from machine learning such as logistic regression and Support Vector Machines will be compared as well. This comparative analysis aims to understand the trade-offs between accuracy and reducing size of dataset, providing valuable insights into the effectiveness of low-rank approximation in the context of handwritten digit classification.

3 Dataset

For the empirical study of conducting handwritten digit classification, this report utilizes the dataset taken from usps.mat. The dataset includes four matrices: train_patterns, test_patterns, train_labels, and test_labels. The dimensions of both 'patterns' arrays are 256 $\times$ 4649, while both 'labels' arrays of size 10 $\times$ 4649. The train patterns and test patterns represent a raster scan of 16 X 16 gray-level pixel intensity and are normalized to range between [-1,1]. Each column represents a single digit j between 0 and 9 - our training and test data each have 4649 entries of data. The train labels and test labels variables contain true information regarding the digit images.

In terms of data usage in train and test labels, if the handwritten digit image represented by train patterns[:, j] corresponds to digit i, then train labels[i, j] are assigned a value of +1. All other entries in train labels[:, j] are set to -1.

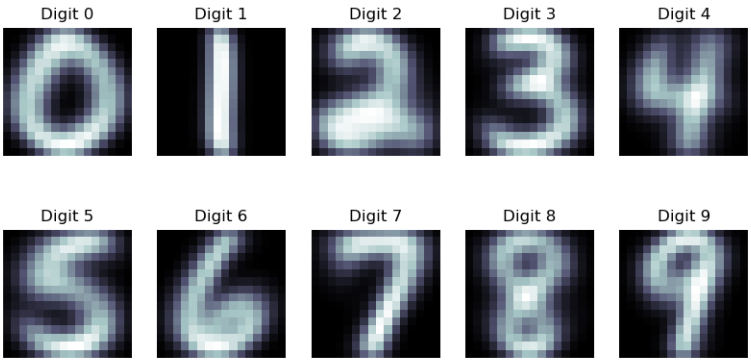
To provide a visual representation of the dataset, the first 16 images from the train_patterns dataset are displayed below:



First 16 Digits of train_patterns

Using training data, I computed the mean digits of across each digit j. The resulting mean digits are organized in a matrix named train_aves which has dimensions of 256×10. The outputs are displayed below.

The plot displays the mean image pooled across each digit j using the training data. It highlights what the average gray-level pixel intensity looks like across each individual digit between 0 - 9 in the training set.



Mean Digits of train_patterns

4 Methodologies

In this study, the objective is to predict the accuracy of handwritten digits using the `usps.mat` dataset. The training data, which is denoted as `train_patterns` $\in \mathbb{R}^{256\times4649}$, will be used to compute the average gray-level pixel intensity across each individual digit. The digits are then pooled into another matrix and then I computed the mean across each row to produce `train_aves` $\in \mathbb{R}^{256\times10}$. Here, each column represents the mean raster values for digit j.The visualization for the first 16 digits in the training data along with their corresponding mean digits is displayed above to offer a foundational understanding of the dataset.

The focus of this report is directed in three distinct experiments.

4.1 Simple Classification Experiment

In the first portion of the experiment, I do not use SVD and rather focus on establishing a baseline accuracy using a simple classification experiment. The primary objective is to display classification accuracy and its corresponding confusion matrix – shown in the results section below.

To begin, I created a new matrix denoted as `test_classif` $\in \mathbb{R}^{10\times4649}$. The matrix is populated by the Euclidean distance (or its square) between each image in `test_patterns` and each individual digit $0 \leq j \leq 9$ in `train_aves`. Each column refers to the single digit in `test_patterns`, and each i-th row represents the distance of the test digit to the i-th mean digit from our training data.

Then, the classification results, i.e., the predicted labels from our `test_patterns`, are obtained by indexing the smallest value in each column of `test_classif`.

These predicted labels are then placed into a vector named `test_classif_res` $\in \mathbb{R}^{1\times 4649}$. This is our predicted $\hat{Y}$.

Note: The code documentation includes two distinct methods, both of which show identical results in regards to the classification rates and confusion matrices.

4.2 Singular Value Decomposition via Low Rank Approximation

In the SVD-based classification experiment, the goal is to represent each test digit image in terms of the singular vectors obtained from the SVD of the corresponding digit set in the training data. In particular, I will use low rank approximations using SVD to compare results across different values of k.

Useful Definitions:

Single Value Decomposition (SVD): SVD refers to the the factorization of a matrix A in m x n into the following form:

$$A = U\Sigma V^T \tag{1}$$

- Σ comprises the singular values of A, representing the lengths of the principal semiaxes of an ellipsoid.
- U consists of the left singular vectors of A, which are unit vectors along the principal semiaxes.
- V^T comprises the right singular vectors of A and serves as the preimage of u_i.

Low-Rank K Approximation:

For a matrix $A \in \mathbb{R}^{m\times n}$, there exist two sets of k orthonormal columns $v_1, \ldots, v_k$ and $u_1, \ldots, u_k$ such that:

$$A = \sum_{i=1}^{k} \sigma_i u_i v_i^T \tag{2}$$

σ_i is the i-th singular value, u_i is a i-th left singular vector, and v_i is the i-th right singular vector. The goal of low-rank approximation is to represent A using a low-rank matrix A_k by keeping the k most significant singular values and their corresponding vectors. As such, I computed SVD (using the function svds) across various values of k to approximate our matrix and then compare the respective classification results.

For the methods, I first pooled the images corresponding to the j-th digit in `train_patterns` and computed the rank k SVD of that set of images. I put the left singular vectors of our SVD into a tensor `train_u` of dimensions $256 \times k \times 10$.

Here the $256 \times k$ matrix corresponds to the left k singular vectors of U for any single handwritten digit (hence, the third dimension of size 10 for the 10 digits). They are stacked 10 times for each digit $0-9$. Then, I computed the expansion coefficients of each test digit image with respect to the k singular vectors of each train digit image set. The tensor `test_svd` contains the expansion coefficients for each test digit image with respect to the k singular vectors of each training digit image set. Each set of coefficients corresponds to a specific digit label, capturing how each test digit image can be expressed in the subspace defined by the singular vectors of its corresponding training digit set. The resulting 3D array `test_svd` has dimensions $k \times 4649 \times 10$. The first dimension (k) is the number of singular vectors used in the expansion. The second dimension (4649) represents each single entry of the test digit. The third dimension (10) represents the digit label (0 to 9).

Next, I computed the error between each original test digit image and its rank k approximation using the j-th digit images in the training dataset. The idea of this classification is that a test digit image should belong to class j if the corresponding rank k approximation is the best approximation (i.e., the smallest error) among 10 such approximations.

In particular, the error can be computed using the Frobenious norm as the following:

$$\sum_{i=1}^{4649} \|test\_patterns[:, i] - U_{j,k} \cdot test\_svd[:, i, j]\|_F^2 \tag{3}$$

In this context, the smallest error corresponds to the predicted label. Each value within this error matrix shows a discrepancy between the original and approximated pixel values for a given test digit image and its predicted digit class. Here, test_patterns[:,i] represents the i-th digit in test data and $U\_j, k \cdot test\_svd[:, i, j]$ represents the rank-k approximation of the pooled images in the training data using the corresponding left singular vectors U_j, k for the digit j. Thus, for each digit j (where j ranges from 0 to 9), this equation computes the total squared error between the original test digit image and its rank-k approximation using the left singular vectors U_j, k.

Ten error values are computed for each image i, then the errors are stored them in another matrix `test_svdres` of size 10×4649. The index containing the smallest error is then chosen to be the predicted label for test_patterns via the low rank approximation. A function `usps_svd_classification()` has been created that allows us to compute prediction labels and classification rates for any k rank SVD. This function can be found in the documentation of the code. For $k = 1 : 50$, I ran the function to compare the accuracy rate and results by plotting a figure of the overall classification rates versus the rank k. I found that $k = 17$ SVD has the best classification rate.

4.3 Comparison of Methods

In addition to comparing the error rates from the initial experiment and various rank-k approximations utilizing Singular Value Decomposition (SVD), I showed the results produced by machine learning algorithms Logistic Regression and Support Vector Machines. Accuracy measures from both methods are included in the results section as well. It was found that both provided similar accuracy rates compared to the SVD.

To run the model with these conventional ML algorithms, I first transposed the training data such that each row represented a digit entry while the columns represented the raster scans. The true labels were indexed based on the location of the +1 in train_patterns so that both X_train and y_train can be used to fit the model. After the models were fitted, I ran them using the test_patterns data to get our accuracy rates.

Note: Both ML algorithms were created using sklearn package in python, documentation is provided in the attached zip file. sklearn would need to be installed prior to testing the code directly.

5 Results

5.1 Simple Classification Experiment Results

Based on the results of our simple classification experiment, I noticed that the classification rate was 0.846 - I was accurate approximately 84.6% of the time on test_patterns. The confusion matrix below displays our predicted values (columns) vs actual values (rows).

```
[[656   1   3   4  10  19  73   2  17   1]
 [  0 644   0   1   0   0   1   0   1   0]
 [ 14   4 362  13  25   5   4   9  18   0]
 [  1   3   4 368   1  17   0   3  14   7]
 [  3  16   6   0 363   1   8   1   5  40]
 [ 13   3   3  20  14 271   9   0  16   6]
 [ 24  11  13   0   9   3 353   0   1   0]
 [  0   5   1   0   7   1   0 351   3  34]
 [  9  19   5  12   6   6   0   1 253  20]
 [  1  15   0   1  39   2   0  24   3 314]]
```

Confusion Matrix (Actual, Predicted) of Simple Experiment

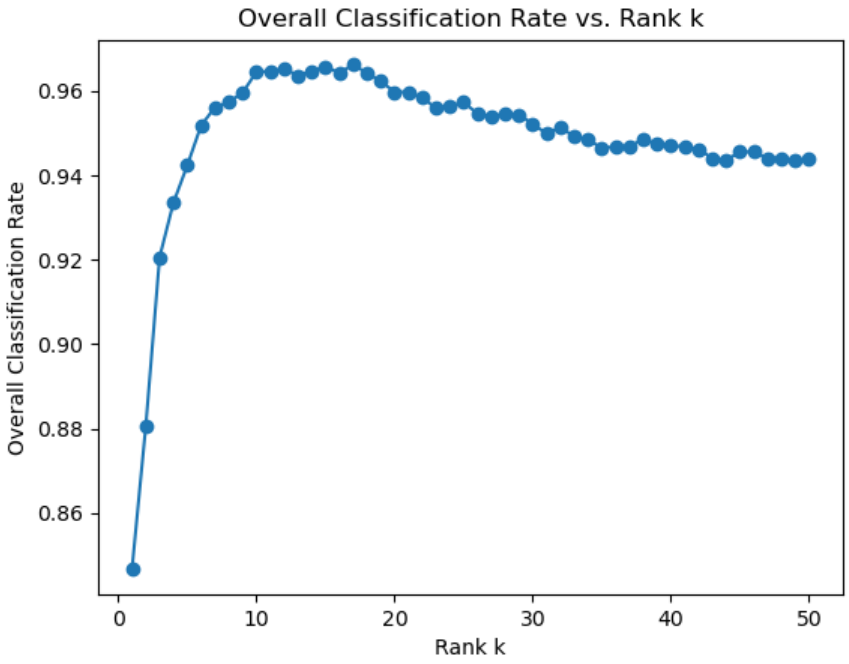
The diagonal elements of the confusion matrix represent the number of correctly classified instances for each class. I notice that this experiment produced fairly

accurate results across each of the digits. The values off the diagonal are the misclassifications. For example, the element in the first row and second column (1) suggests that an instance of class 1 was misclassified as class 2. The value in the fifth row and tenth column (40) indicates a relatively higher number of instances of class 4 misclassified as class 9. Similarly, this can also be seen in the last row and the fifth column containing 39 misclassifications, suggesting that a high number of true 9's were incorrectly labeled as 4. With this, our results indicate that the experiments produced quite a few misclassifications between digits 4 and 9.

As we can see, the results from the confusion matrix show that the errors were quite ranging across each digit. This initial experiment sets a baseline accuracy using the training data.

5.2 Singular Value Decomposition Results

For the second part of this experiment, I computed low rank approximations using k = 1:50. The plot below displays the overall classification rates across different values of k.



Overall Classification Rate vs. Rank k

The results of this plot are quite informative; as expected, the classification accuracy rose significantly as k increase, particularly in range between 1 to 5.

Notably, we can see that the accuracy rates go from below 0.85 to approximately 0.95. Surprisingly, this surpassed the accuracy achieved in the initial part of our experiment.

As k continued to increase, specifically beyond k = 20, the error rate dropped and then seemed to plateau. This may be attributed to overfitting as there may be an excessive fit to the training data when k increases, making the predictions less adaptable to the variations from test data. Thus, the accuracy rates seemed to slightly go down as the model struggled to predict as effectively with higher k. This explain why the results with lower rank k approximation produced higher classification rates than our initial experiment.

Based on our plot, the classification rate is the highest at rank k = 17. More specifically, the classification rate for test_patterns at k = 17 is 96.4%, an extremely high accuracy. I have displayed the associated confusion matrix of our predictions as well.

```
[[772   2   1   3   1   1   2   1   3   0]
 [  0 646   0   0   0   0   0   0   0   1]
 [  3   6 431   6   0   3   1   2   2   0]
 [  1   1   4 401   0   7   0   0   4   0]
 [  2   8   1   0 424   1   1   5   0   1]
 [  2   0   0   5   2 335   7   1   1   2]
 [  6   4   0   0   2   3 399   0   0   0]
 [  0   2   0   0   2   0   0 387   0  11]
 [  2   9   1   5   1   1   0   0 309   3]
 [  0   5   0   1   0   0   0   4   1 388]]
```

Confusion Matrix of Low Rank Approx. via SVD (k = 17)

Compared to our results in part a, it is clear that the confusion matrix produces significantly lesser error rates as the values on non-diagonal entries are values closer to 0 (indicating lesser misclassifications). Compared to the initial experiment, we can see that our misclassifications between digits 9 and 4 are now much more accurate as well. With this, it is clear than rank k = 17 approximation performs quite well on this dataset.

5.3 Comparison of Methods

The table at the bottom displays accuracy rates produced by Logistic Regression, Support Vector Machines, the classification experiment for baseline results (part I), and the low rank k = 17 approximation via SVD (part II):

The classification rates appear to be approximately same for each method (mi-

Table 1: Classification Rates for Different Methods

Method	Classification Rate
Logistic Regression	0.942
Support Vector Machines	0.951
Simple Classification Experiment	0.846
SVD Low-Rank Approximation (rank 17)	0.964

nus the baseline). The accuracy rates for all three algorithms are extremely high ¿90%, which suggests that they are all very effective in predicting handwritten digit classifications. In particular, the rank 17 SVD has a slightly higher accuracy than both logistic regression and SVM. It is essential to note that the effectiveness of these techniques may vary across datasets, so testing these methods with diverse set of datasets will provide a more comprehensive understanding.

6 Conclusion

In conclusion, this report highlights the effectiveness of low-rank approximation via Singular Value Decomposition in handwritten digit classifications. The results, particularly at rank k = 17, showcase a very high accuracy rate of 96.4%. This accuracy surpassed our baseline experiment and even outperformed conventional methods such as Logistic Regression and Support Vector Machines. Again, note that this difference can be attributed to the individual dataset itself so additional experiments with other dataset may instill a more detailed understanding.

The advantages of SVD in this context lie in the ability to capture only the most essential features of the dataset while significantly reducing the computation complexity via low rank approximation. This method improved the accuracy of our results by mitigating potential overfitting issues observed with higher values of k. The visualizations of mean digits further emphasize the interpretability of SVD in understanding and representing the underlying patterns in handwritten digit data.

In terms of drawbacks, while SVD exhibits impressive predictive accuracy in this application, the performance may vary in other uses and datasets. Additionally, the latter ML methods demonstrate high accuracy as well, and may be preferred due to their simplicity and ease of implementation.

Overall, the choice between various different methods depends on the specific project and inherent characteristics of the dataset itself. In this case, SVD with low-rank approximation proves to be a powerful tool for handwritten digit classification, offering a balance between accuracy and computational efficiency.