Handwritten Digit Classification

Robin Chartrand
rchartra@uw.edu

February 15, 2024

Abstract

A low dimensional approximation of a dataset of images of handwritten digits from the MNIST database was used to train four different classifiers: Ridge regression classification, K-Nearest Neighbors, Linear Discriminant Analysis, and Support Vector Machines. Each classifier was tested and compared on their ability to determine the digit written in a given image sample from both the training and testing datasets. Of the four models, K-Nearest neighbors had the best combination of both cross validated accuracy and efficiency.

1 Introduction

Using a computer to distinguish between handwritten numerical digits has important applications in handwriting recognition software used by banks, businesses, and government offices to efficiently and accurately process handwritten documents. Such methods need to have very high accuracy to be implemented in practice and as a result many rigorous machine learning algorithms have been developed over the years to perform such image classification tasks. In this study we analyzed four different classification algorithms' ability to classify images of handwritten digits from the Modified National Institute of Standards and Technology database. The algorithms selected were Ridge regression classification, K-Nearest Neighbors, Linear Discriminant Analysis, and Support Vector Machines which were each trained on a portion of the dataset before being cross validated and used to evaluate the accuracy on a new test dataset.

2 Theoretical Background

High dimensional data can be difficult to visualize and work with. When implementing classification algorithms lower dimensional representations of the original data are often sought after as they greatly increase the efficiency of the classification methods. Principal Component Analysis (PCA) is the standard method for dimensionality reduction in machine learning. It is done by performing Singular Value Decomposition on the data array to decompose it into three matrices representing the rotation V^T, scaling Σ, and change of basis U performed by the array when used as an operator:

$$A = U\Sigma V^T$$

The scaling matrix Σ is a diagonal matrix whose elements are known as singular values σ_i. The change of basis matrix U is composed of modes which, when the data is properly centered, represent the directions of greatest variance of the data. The data can therefore often be approximated by a representation composed of the first k modes of the change of basis matrix, with the number of modes k chosen such that the representation captures a sufficient amount of the data's cumulative energy:

$$CE_k = \sum_{i=1}^{k} \frac{\sigma_i^2}{E_{tot}}$$

Four different classification methods were compared in this study. The first method is Ridge classification which converts the target labels into -1 and 1 and then performs ridge regression to discriminate between the two classes [3]. Ridge regression (3) is a modification to linear regression (1)(2) known as regularization which helps to relax singularities that may arise in A^TA.

$$f(x) = \beta_0 + \sum_{j=1}^{d} \beta_j \psi(x)_j \tag{1}$$

$$\beta_{MLE} = \arg\min_{\beta} \frac{1}{2\sigma^2} ||A\beta - y||^2 = (A^TA)^{-1}A^Ty \tag{2}$$

$$\beta_{reg} = \arg\min_{\beta} \frac{1}{2\sigma^2} ||A\beta - y||^2 + \frac{\lambda}{2}||\beta||_2^2 = (A^TA + \sigma^2\lambda I)^{-1}A^Ty \tag{3}$$

This modification ensures $A^TA + \sigma^2\lambda I$ is always invertible allowing us to always find the maximum likelihood coefficients β for the distribution which best divides the data.

The second method is the K-Nearest Neighbors method. This method works by determining the k closest training points to a new test point, and then assigning the new point to the most common class amongst its k neighboring points. Its accuracy and efficiency are affected by the choice of k whose optimal value is highly data-dependent [3]. The third method is Linear discriminant analysis. It doubles as both a method of performing dimensionality reduction and a classification method. It reduces the dimensions of the data and then maximizes the distance between means of classes and minimizes variance within classes, i.e. the ratio of between-class variance and the within-class variance [2]. It then identifies the linear combination of features that best divides the classes [1]. The fourth and final method is the support vector machines method. It works by finding the optimal set of hyper-planes that divide the training data points into classes [3]. It does so by minimizing the distance of the hyper-plane to the subset of data points closest to the class boundaries, known as "support vectors". This can be boiled down to a set of optimization problems that can be solved analytically [5].

3 Algorithm Implementation and Development

The data was first split into a set of training images and labels and a set of testing images and labels. PCA was then performed on the training dataset using the Scikit Learn decomposition library [3] and the cumulative energy captured by each PCA mode was calculated. It was found that 59 modes captured 85% of the data's original energy so the training and testing data sets were projected onto truncated 59 mode PCA space.

Ridge classification was then performed using the Scikit Learn `RidgeClassifierCV()` method [3]. The classifier was first used to distinguish between various pairs of digits to see which digits were easier/harder for the classifier to distinguish between. The classifier was then used to classify all ten digits in the dataset. For each classification trial the mean classification accuracy on the training and test set was calculated. Cross validation was then performed to determine a more robust sense of the accuracy by averaging the accuracy on the training set across five instances of the model and calculating the standard deviation. This was done using the Scikit Learn `cross_val_score()` method [3].

The K-Nearest Neighbors algorithm was then implemented using the Scikit Learn `KNeighborsClassifier()` method [3]. The training and testing data was first scaled using the Scikit Learn `StandardScaler()` method. To determine the ideal k neighbors to use, the model was trained, cross validated, and scored on the training data for multiple values of k. The k with the maximum accuracy was selected and used in the final model. The model was once again tested on both the training and testing data and then cross validated on the training data to determine its accuracy. The Linear Discriminant Analysis and Support Vector Machine algorithms were then also implemented in a similar fashion using the Scikit Learn methods `LinearDiscriminantAnalysis()` and `SVM()` [3]. Both models were then also tested on the training and testing data and cross validated.

4 Computational Results

PCA was first performed on the data providing 784 principal components; the first 16 of which are shown as reconstructed images in Figure 1a.

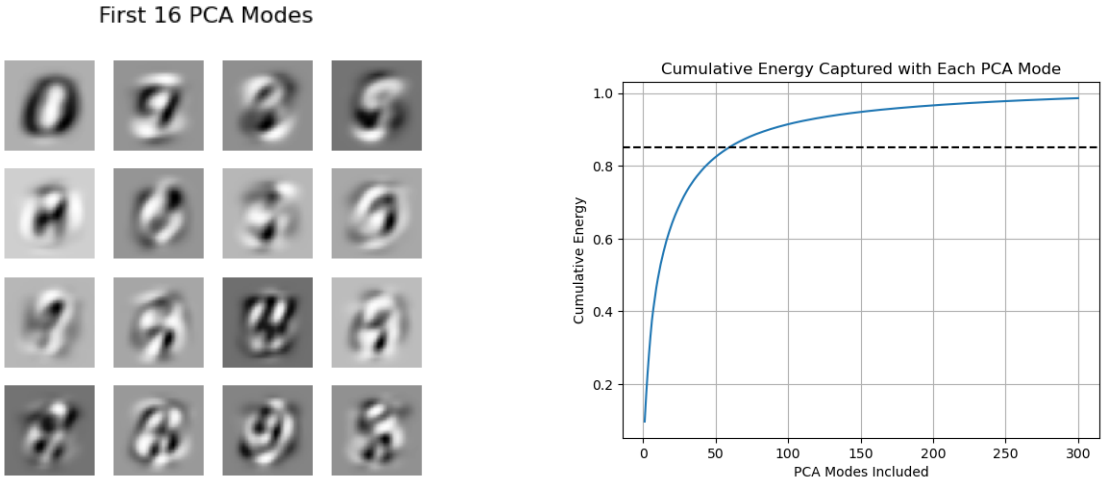

Figure 1: a.) The first 16 PCA modes (columns of the U matrix) of the dataset reconstructed as 28 x 28 images. b.) The cumulative energy added to the representation by each of the first 300 of the 784 modes of the data. The 85% threshold is marked by the dashed line.

The first mode appears to capture a circular outline similar to a zero, and the following modes capture more and more complex shapes present in the digits. The cumulative energy captured by each additional mode was calculated (Figure 1b) and it was determined that 85% of the energy within the images could be captured by just 59 of the original 784 dimensions. When the truncated approximation is used to reconstruct the original images (Figure 2) it is clear that while the approximation isn't perfect, each digit maintains its shape and can still be clearly identified. Due to its success this lower dimension approximation was used instead of the original data set in the subsequent classification algorithms.

The ridge regression classification algorithm was trained and then tested on three subsets of two numbers: one and eight, three and eight and two and seven. The accuracy of the model on each subset is shown in Figure 3. The model was significantly better at distinguishing between two and seven than the other two

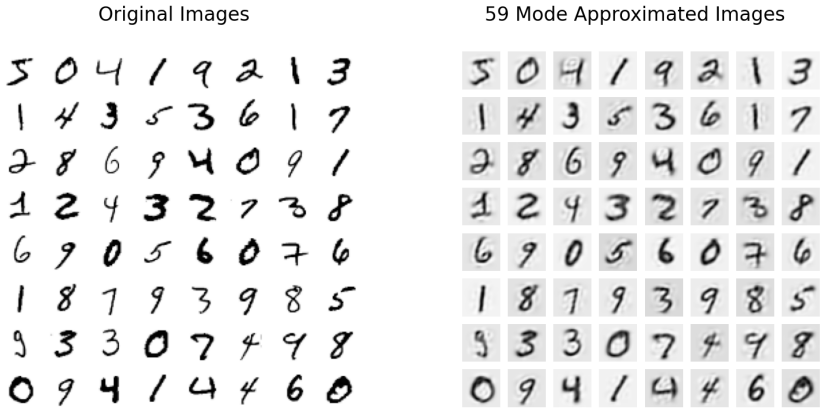

Figure 2: The first 64 images of the original dataset compared with the approximated images formed by the first 59 principal components.

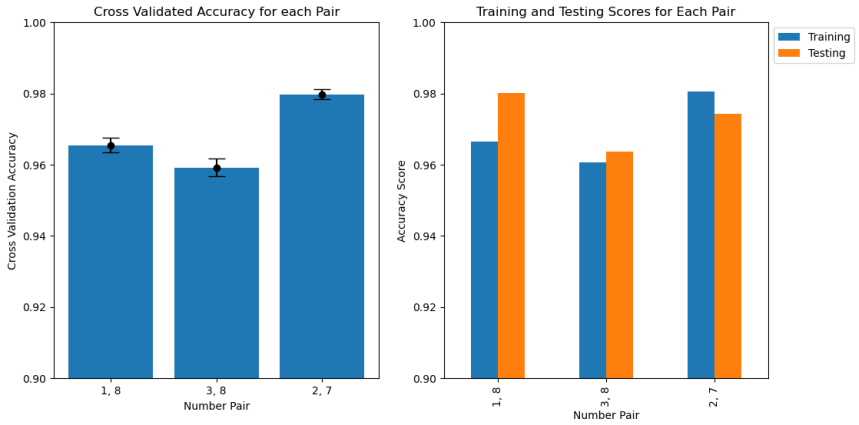

Figure 3: The mean cross validated accuracies of the Ridge classification method on each number pair is shown on the left. The error bars indicate a single standard deviation from the mean. On the right the training and testing scores of a single classifier on each number pair is also shown.

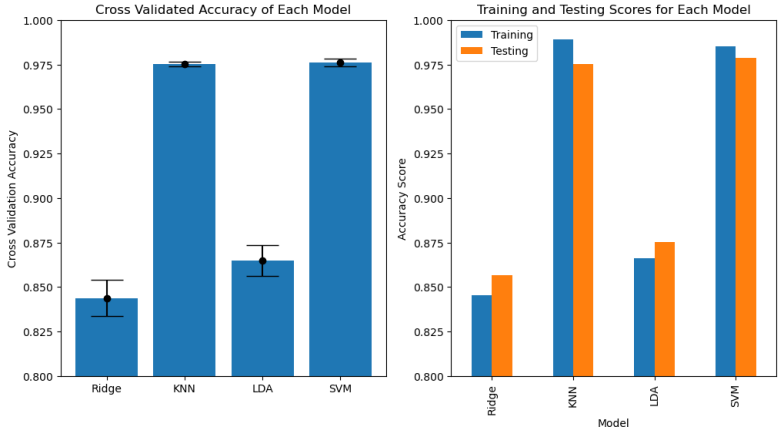

Figure 4: The mean cross validated accuracies of each multiclass classification method is shown on the left. The error bars indicate a single standard deviation from the mean. On the right the training and testing scores of a single instance of each classifier is also shown.

pairs with an accuracy of approximately 98%. This is a little shocking since to our eyes two and seven look more alike than one and eight, but this is apparently not the case for the algorithm. One and eight did however have the next best accuracy at approximately 96%, although it wasn't significantly better than three and eight which had the lowest accuracy of also approximately 96%. It makes sense three and eight were the hardest to distinguish as they are very similar digits. Ultimately ridge regression classification was very good at distinguishing between each pair of digits with high accuracies overall. It is also of note that for pairs one and eight and three and eight the testing accuracy was higher than the training accuracy. It is only so by about one to two percent however this is not the result one would typically expect.

Multiclass classification was then performed by four different methods, ridge regression, K-nearest neighbors (KNN), linear discriminant analysis (LDA), and support vector machines (SVM). The results of the classifications are shown in Figure 4. KNN and SVM both performed equally well with around 97% accuracy and leagues better than both LDA and Ridge regression. LDA had around an 87% accuracy and was significantly better than ridge regression which only had an accuracy of 84%. Accuracy is not the only factor that goes into choosing an algorithm however. When working with large datasets as in our case, efficiency is also a major factor in model choice. When comparing the training times for each model (Figure 5) the best model becomes very clear.

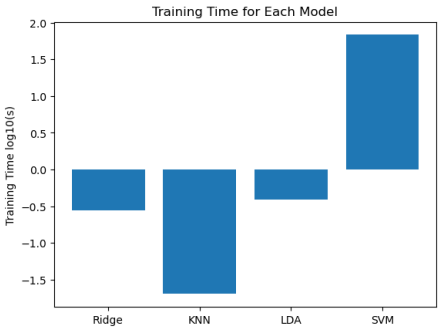

Figure 5: The log base 10 of the training time of a single instance of each model.

SVM is known for being inefficient for large datasets [4] due to the computational complexity of the underlying optimization problem it is solving, and indeed it took over three orders of magnitude longer to train an SVM model on the training set than a KNN model. Part of KNNs stand out efficiency is that for our data the optimal k neighbors to use was to use was a mere 3 neighbors (Figure 6). Using $k = 1$ would've worked just as well, however 3 was chosen to include more of the data in the classification process.

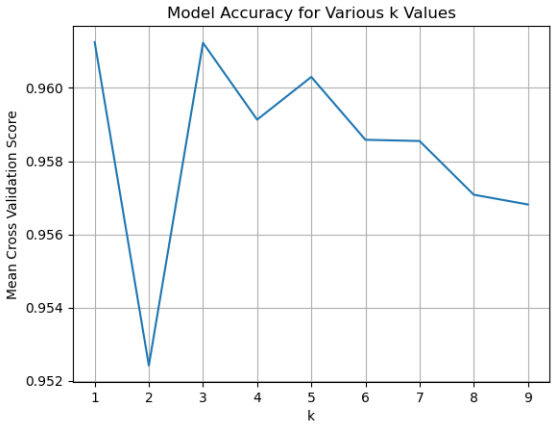

Figure 6: The accuracy of a single KNN classifier for different values of parameter k.

While identifying this optimal k value is a slow process of training multiple KNN models, once identified the KNN training process is very quick which is ideal for our large dataset. Coupled with being tied for having the greatest accuracy of all four models, KNN is the obvious best algorithm for our classification task.

5 Summary and Conclusions

Principal component analysis is an effective method of dimension reduction that is very important for the efficiency of learning algorithms which tend to struggle with high dimensional data. Ridge regression classification is very good at distinguishing between two digits, though it is better at distinguishing between certain digit pairs over others. When it comes to multiclass classification however Ridge regression does not perform well compared to methods such as K-Nearest neighbors and support vector machines which are both around 13% more accurate. While both of these methods have a similar high accuracy, K-Nearest neighbors is ultimately a better choice as it is much more efficient than the support vector machines method when working with such a large dataset.

6 Acknowledgements

I would like to thank Professor Shlizerman for his instruction on PCA and classification algorithms. I am also grateful to Grace Zhou for her discussion on reconstructing data from a truncated PCA space.

References

[1] IBM. What is linear discriminant analysis?, Nov 2023.

[2] N. Mohanty, A. L.-S. John, R. Manmatha, and T. Rath. Chapter 10 - shape-based image classification and retrieval. In C. Rao and V. Govindaraju, editors, *Handbook of Statistics*, volume 31 of *Handbook of Statistics*, pages 249–267. Elsevier, 2013.

[3] F. Pedregosa, G. Varoquaux, and et. al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[4] S. Schlag, M. Schmitt, and C. Schulz. Faster support vector machines. *Journal of Experimental Algorithmics (JEA)*, 26:1 – 21, 2018.

[5] I. Zoppis, G. Mauri, and R. Dondi. Kernel methods: Support vector machines. In S. Ranganathan, M. Gribskov, K. Nakai, and C. Schönbach, editors, *Encyclopedia of Bioinformatics and Computational Biology*, pages 503–510. Academic Press, Oxford, 2019.