CLASSIFYING DIGITS

ABHISHEK SHARMA

Mechanical Engineering Department, University of Washington, Seattle, WA
as711@uw.edu

Abstract. In this work, we train classifiers to distinguish between pairs of digits from the MNIST dataset. We use Principal Component Analysis (PCA) to reduce the dimensionality of the dataset and then use ridge regression in the lower dimensional space to predict the labels given the input images.

1. Introduction and Overview

The MNIST dataset [1] is a standard machine learning dataset of handwritten digits. In this work, we take 2000 instances of training images and 500 instances of test images to train as classifier to discriminate between the following pairs of digits- $(1, 8), (3, 8), (2, 7)$. We use PCA to identify the important features in the data and reduce the dimensionality of the problem. Then, we use ridge regression to learn a function f that maps the lower dimensional projection of the training data to the desired labels. We then test this function for how closely does it predict the labels for the test set. We report the mean squared error (MSE) as the error metric for train and test sets.

2. Theoretical Background

Natural data often exists on a lower dimensional manifold than the number of degrees of freedom available to capture the data. In other words, the data can often be reduced to a lower dimension than it is available in. There are several benefits to dimentionality reduction like data compression, storage efficiency and interpretability. Another potential benefit is that it can alleviate the problems of learning a classfier or regressor in high dimensions i.e. curse of dimensionality [2]. There are several methods to reduce the dimensionality of the data- SVD [3], PCA [4], autoencoders [5], which preserve the relevant information in the data while also reducing the dimensionality. In this work, we use PCA which involves doing the eigendecomposition of the covariance matrix corresponding to the centered *Training* data matrix (mean subtracted along each dimension) $X \in R^{m \times n}$ (eq. 1), where m is the number of sample and n is the number of dimensions (usually $m >> n$).

$$Cov[X] = \frac{1}{n-1} X X^T = \frac{1}{n-1} U \Sigma^2 U^T \tag{1}$$

$$X = U \Sigma V^T \tag{2}$$

The matrix Σ^2 is a diagonal matrix consisting of the eigenvalues of $X^T X$ arranged in descending order. Σ are also the singular values of the matrix X (eq. 2). The column vectors in V are also called the principal components of the data. The principal component associated to the higher eigenvalue (Σ^2) is more important and accounts for a higher variance (given by $\frac{1}{n-1}\Sigma^2$) in the data. To reduce the dimensionality of the data we can keep only a desired number (say $k < n$) of these

Date: February 12, 2022.

principal components. Then, we can project the original data on to these k principal components to reduce the dimensionality of the problem to k. A principled way to decide the number of dimensions is to set a desired level of variance to be explained in the data and select the number of PCs that meet that criteria. Another way is to measure the number of PCs that can approximate the data X to a desired level in Frobenius norm eq. (3).

$$||X||_F^2 = \sum_{j=1}^{n} \sigma_j(X)^2 \tag{3}$$

where σ_j are the diagonal entries of Σ in a descending order. Dimensionality reduction is often done to simplify the problem. Usually the end goal in machine learning it to build models the predict outcomes, dependent variables or classify data, This involves learning a mapping from the features in the data to the desired output, which is accomplished by training a classifier or a regressor on a part of data called *Training set*. Then, the performance of the learned model is estimated on a *Test set* which is an approximation to how the model with perform in the real world setting.

One such regression method is *Ridge regression* [6] which fits a linear model to the data i.e. the outputs are the weighted sum of the input features. The weights are the coefficients ($\hat{\beta} \in R^k$) of the linear model learned by minimizing the error between the predicitons and the desired label, as shown in eq 4.

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \, ||A\beta - y||^2 + \lambda||\beta||^2 \tag{4}$$

where $A \in R^{m \times k}$ is the input matrix with m training examples and n features, $y \in R^m$ are the associated ground truth, and λ is the regularization term. λ is hyperparameter tuned using cross validation [] to find an optimal model with low bias and low variance. (See Bias-variance tradeoff [7]) The optimal model achieves the best generalization performance. The commonly used performance metric for regression problem is mean squared error (MSE). The NSE for the ridge regression problem with m examples can be computed as shown in eq. 5

$$MSE = \frac{1}{m} \times ||A\hat{\beta} - y||_2^2 \tag{5}$$

where $A \in R^{m \times k}$, $\hat{\beta} \in R^k$, $y \in R^m$.

3. Algorithm Implementation and Development

All the algorithms were implemented in Python [8]. Numpy package [9] was used for data array manipulations, sklearn [10] was used for PCA and Ridge regression. All the visualizations and figures were generated using plotly [11] matplotlib [12]. We used *RidgeCV* from sklearn to perform ridge regression a long with the default *Leave-one-out* cross validation. Regularization parameter (λ) were selected from a uniform grid of 10^5 points between in the range $[10^{-4}, 10]$. The outputs of the regressor are then converted to -1 for negative outputs and $+1$ for positive outputs and the corresponding class is assigned to the input.

4. Computational Results

First we explored the dimensionality of the dataset. The singular values of centered data matrix are shown in Fig. 1. We see that the singular values (σ_j) drop-off very rapidly with $\sigma_{37} < 0.1\sigma_1$ and $\sigma_{130} < 0.01\sigma_1$. Thus, the data lies on a low dimensional manifold and can be compressed using a few dominant principal components. Fig 1 shows the first 16 principal components. The training data has a lot of 0s (328) and digits with round features- 6s (185), 2s (180), 8s (177), 3s (174), 9s (165) and 5s (145) as compared to those with sharp features- 1s (278), 4s (197) and 7s (171). This might explain the shape of the first four principal components being largely round. The imbalance in the number of digits might be detrimental to classification performance on the test set if the distribution of digits in the test set is not the same.

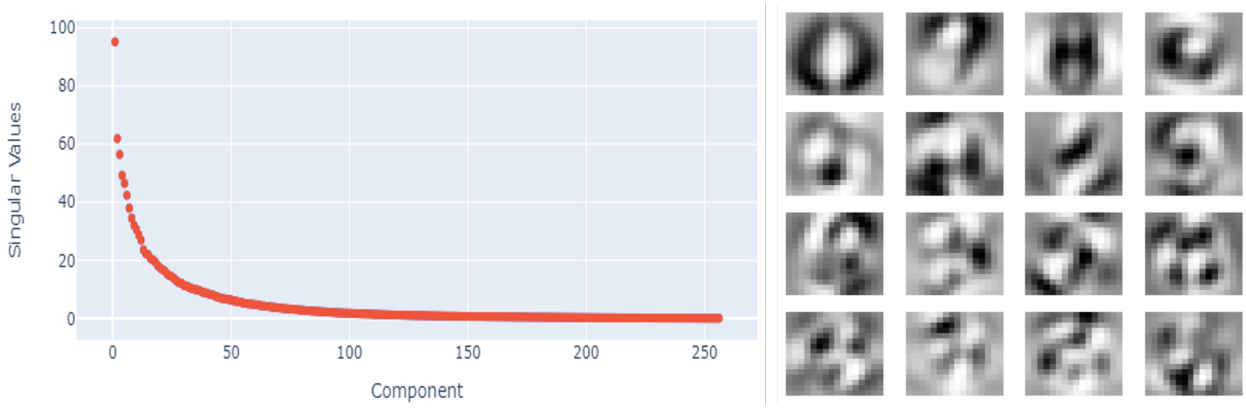

Figure 1. **Left:** The singular value (σ_j) spectrum of the training data. The values drop off quickly ($\sigma_{37} < 0.1\sigma_1$), indicating that the data lies on a lower dimensional manifold and existence of dominant modes in data. **Right:** The first 16 principal components plotted as picture. First row contains first 4 PCs (left to right), the second row contains PCs 5-8 and so on

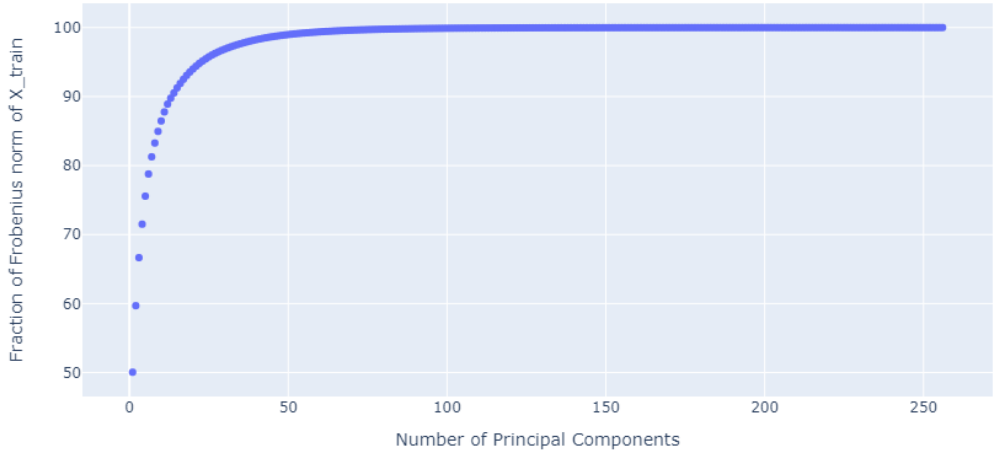

Figure 2. The fraction of Frobenius norm of X_{train} accounted for as a function of the number of principal component used.

Fig. 2 shows the Fraction of Frobenius norm accounted for as function of number of principal components considered. We find that *at least* 60%, 80%, and 90% of the Forbenius norm is accounted for by sum of first 3, 7, and 14 principal components respectively.

Fig 3 show the comparison between the full rank digits (centered data) on the left and their low rank approximation using First 14 principal components on the right. This demonstrates that images can be compressed into coefficients corresponding to the first few principal components.

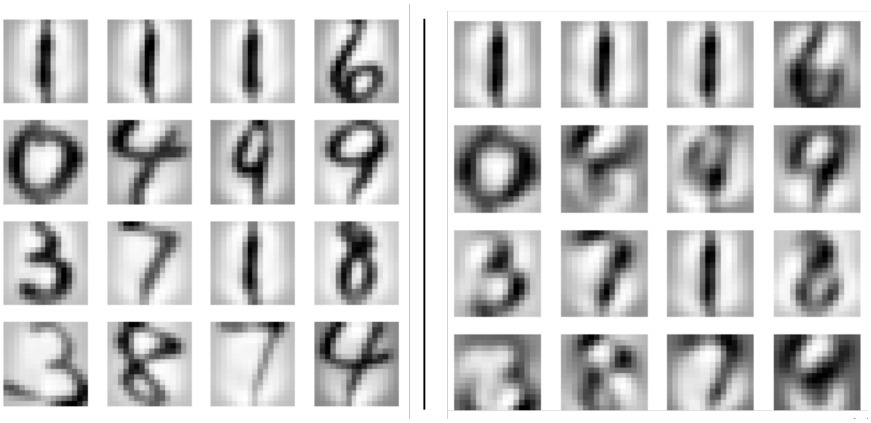

FIGURE 3. **Left:** Original centred data Right: Low rank approximation using first 14 principal components

Nest we show the performance of linear model obtained using ridge regression in Table 1. The optimal λ values for (1,8), (3,8) and (2,7) were 9.803, 10.0, and 7.595 respectively. We find that the model performance is quite good for the pair (1,8). The MSE is 0.0826 for the test set (RMSE $\approx$ 0.287) on the output range of [-1,1]. The performance drop between Train and Test set is also relatively small. The performance drops significantly for the (3,8) case. The MSE for test set is 0.2591 (RMSE $\approx$ 0.509). The MSE for the test set is almost twice that of the training set, showing worse generalization performance. The performance for (2,7) is not as good as (1,8) but still better than (3,8). This could be understood from Fig 4

	$\text{MSE}_{(1,8)}$	$\text{MSE}_{(3,8)}$	$\text{MSE}_{(2,7)}$
Train	0.0754	0.1812	0.0924
Test	0.0826	0.2591	0.1324

TABLE 1. Predictor performance (MSE) on training and test sets for classification of the following pair of digits: (1,8), (3,8) and (2,7).

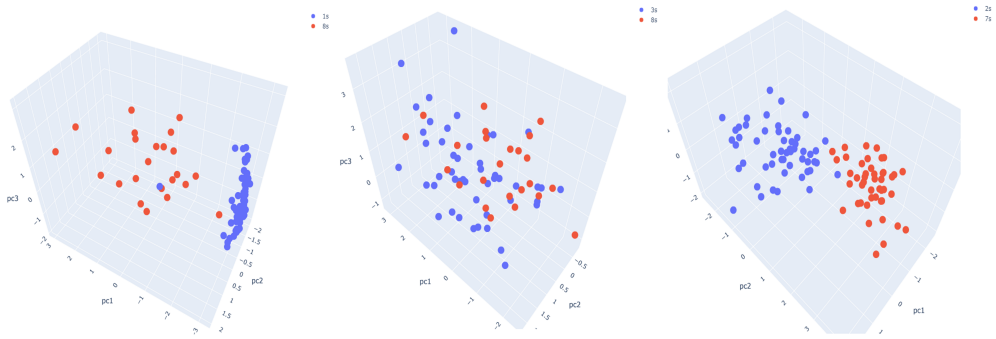

FIGURE 4. The projection of Test data in space spanned by first 3 prinicipal components. Left:(1,8), Center:(3,8), and Right:(2,7)

In Fig. 4, we see the possible cause of aforementioned variations in performance. Even with first 3 principal components 1 and 8 are quite separable thus it is easier for simple linear model to learn a decision boundary. This explains the high classification accuracy for (1,8) (See the confusion matrix in Fig 5). For the (3,8) case it is impossible to distinguish between 3 and 8 in the 3-dimensional space. We can imagine something similar might be happening in the 16 dimensional space in which regressor was learned. We might need more PCs which carry finer details, in oreder to have a better accuracy for (3,8). For the (2,7) case, we see that data point are separable (Explaining high classification accuracy in Fig 5) but the distance between the clouds isn't as big (there are quite a few point close to the imaginary separation boundary) which might be the reason for decrease in MSE for the test set. On a higher level of analysis, 2 and 7 are distinct 2s are "round" and 7s are "sharp". Which might account for why classifier work good enough for separating them using just the course information from first 16 pcs. This isn't the case with 3 and 8 which are both "round".

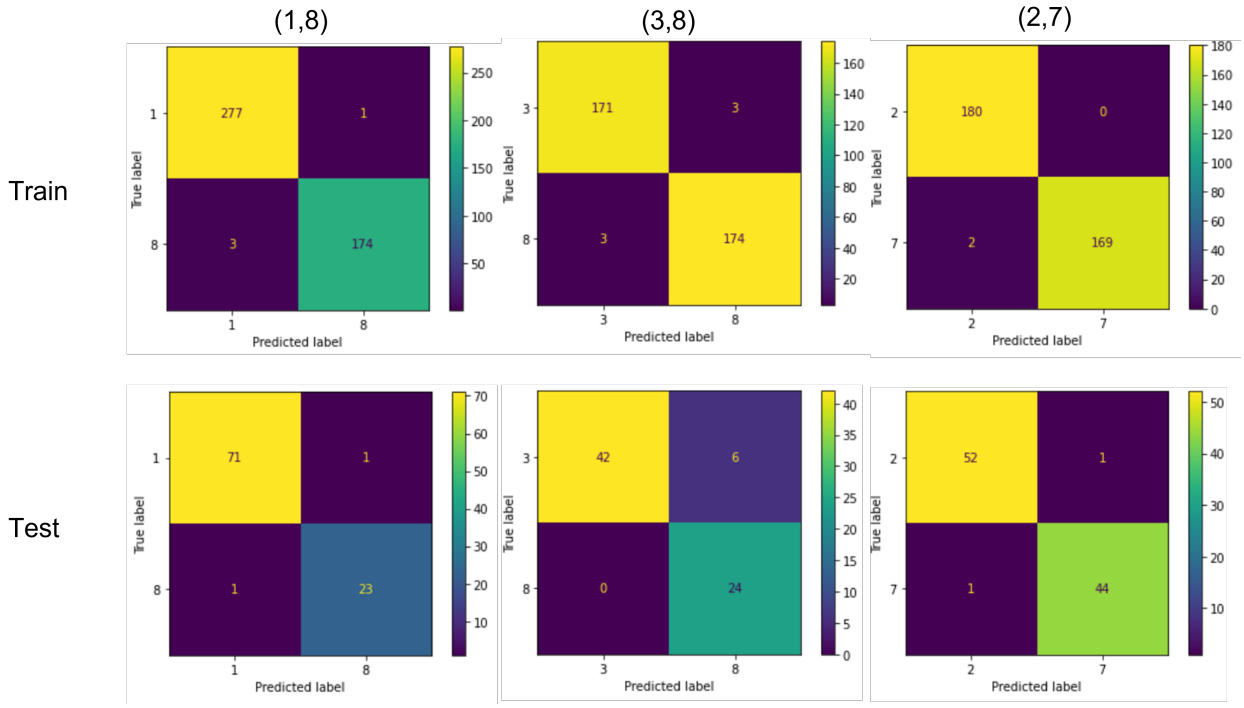

FIGURE 5. Top row show confusion matrix for Training set for classification of (1-blue,8-red): Left, (3-blue,8-red):centre, and (2-blue,7-red): Right. The bottom row shows the confusion matrix for the test set.

5. Summary and Conclusions

In this report, we analyzed the performance of binary digit classification on the pairs of digit (1,8), (3,8) and (2,7) from the MNIST dataset, following dimensionality reduction to 16-dimensional space spanned by first 16 principal components learned from a training data of 2000 images. We found that it was easy to do classification for (1,8) and (2,7). For (3,8) the loss of information contained in removed PCs might have caused degradation in performance. Adding more principal components might resolve this.

Acknowledgements

The author is thankful to Prof. Bamdad Hosseini for useful discussions about the Principal Component Analysis, Regression and demonstrations associated with these methods. We are also

thankful to Katherine Owens for promptly responding to queries regarding the sklearn package and Nicole Nesbihal for clarifications regarding dimensionality reduction. Furthermore, the discussion thread on canvas was extremely helpful.

References

[1] "The mnist database of handwritten digits." `"http://yann.lecun.com/exdb/mnist/`, 1998. Accessed on 2/12/22.
[2] "Curse of dimensionality." `"https://en.wikipedia.org/wiki/Curse_of_dimensionality`, 2022. Accessed on 2/12/22.
[3] S. Tanwar, T. Ramani, and S. Tyagi, "Dimensionality reduction using pca and svd in big data: A comparative case study," in *International conference on future internet technologies and trends*, pp. 116–125, Springer, 2017.
[4] M. Ringnér, "What is principal component analysis?," *Nature biotechnology*, vol. 26, no. 3, pp. 303–304, 2008.
[5] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *arXiv preprint arXiv:2003.05991*, 2020.
[6] G. C. McDonald, "Ridge regression," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 1, no. 1, pp. 93–100, 2009.
[7] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias–variance trade-off," *Proceedings of the National Academy of Sciences*, vol. 116, no. 32, pp. 15849–15854, 2019.
[8] G. Van Rossum *et al.*, "Python programming language.," in *USENIX annual technical conference*, vol. 41, p. 36, 2007.
[9] T. E. Oliphant, *A guide to NumPy*, vol. 1. Trelgol Publishing USA, 2006.
[10] 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*, vol. 12, pp. 2825–2830, 2011.
[11] P. T. Inc., "Collaborative data science," 2015.
[12] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in science & engineering*, vol. 9, no. 03, pp. 90–95, 2007.