AMATH 482/582: HOME WORK 2

YIXUAN LIU

University of Washington, Seattle, WA
yixuanl@uw.edu

Abstract. In this project, we will train a classifier to distinguish images of handwritten digits from the MNIST data set. We will be given a training set and a test set. The classifier will be trained with the training set to distinguish several specific pairs of handwritten digits. Then we will use the classifier to distinguish those pairs of handwritten digits in the test set. After the distinguishing process, we will investigate the performance of the classier on different pairs of handwritten digits.

1. Introduction and Overview

We will train a classifier to distinguish images of handwritten digits from the MNIST data set. The data set is split into training and test sets. The training set contains 2000 instances of handwritten digits, the "features" of the training set are 16×16 black and white images, and the "labels" of the training set are the corresponding digit. The test set contains 500 instances, with the same attributes as the training set. Here is the visualization of the first 64 training features:

Figure 1. Visualization of the first 64 training features

Date: February 9, 2022.

We will train the classifier with the training set to distinguish the handwritten digits between 1 and 8, 3 and 8, and 2 and 7. Apply the classifier to the test set, we can see how well does the classifier proform in distinguishing those digits.

2. Theoretical Background

The singular Value Decomposition (SVD) has the following theorem: let $A \in \mathbb{R}^{n\times m}$ then there exist unitary matrices $U \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{m\times m}$ along with a diagonal matrix $\Sigma \in \mathbb{R}^{n\times m}$, with positive entries, so that

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

The $\{u_j\}$ the columns of U are called the left singular vectors of A and form a basis for $\mathbb{R}^n$. The $\{v_j\}$ the columns of V are called the right singular vectors of A and form a basis for $\mathbb{R}^m$. The $\{\sigma_j\}$, are non-negative (typically in descending order) $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_m \geq 0$ and are called the singular values of A [4].

The Principal Component Analysis (PCA) can reduce dimensionality, which allows us to represent high dimensional data using a few coefficients. With the given data set

$$X = x_0, ..., x_{N-1}, x_j \in \mathbb{R}^d$$

the left singular vector of X are the pricipal components of the data x_{ij} and the optimal basis in which the x_j can be represented, so from the liner algebra view point PCA is just an application of SVD [3].

Frobenins (Hilbert-Schmidt norm) is defined as

$$||A||_F = (\sum_{i=1}^{n}\sum_{j=1}^{m}|a_{ij}|^2)^{1/2} = (\sum_{j=1}^{m}||a_j||_2^2)^{1/2}$$

where $||a_j||_2$ is the Eucliden norm (2-norm):

$$||a_j||_2 = (a_j^T a_j)$$

SVD providea another way to get Frobenins norm:

$$||A||_F = (\sum_{j=1}^{\min\{m,n\}}\sigma_j^2)^{1/2}$$

where σ_j is the singular values of A [4].

Ridge Regression is a special case of regression [2]. Define

$$A = \begin{pmatrix} \psi_0(\underline{x}_0) & \psi_1(\underline{x}_0) & ... & \psi_{J-1}(\underline{x}_0) \\ \psi_0(\underline{x}_1) & ... & ... & \psi_{J-1}(\underline{x}_1) \\ ... & ... & ... & ... \\ \psi_0(\underline{x}_{N-1}) & ... & ... & \psi_{J-1}(\underline{x}_{N-1}) \end{pmatrix} \in \mathbb{R}^{N\times J}$$

If A^TA is not invertible, solve

$$\underline{\hat{\beta}} = \arg\min_{\beta\in\mathbb{R}^J} \frac{1}{2\sigma^2}||A\underline{\beta} - \underline{y}||^2 + \underline{\lambda}\,||\underline{\beta}||^2$$

The training and test mean squared error (MSE) [2] on the trained regression model $\hat{f}(\underline{x}) = \sum_{j=1}^{J} \hat{\beta}_j \psi_j(\underline{x})$ is defined as

$$\text{MSE}_{\text{train}}(\hat{f}, \underline{y}) = \frac{1}{N} \sum_{n=0}^{N-1} |\hat{f}(\underline{x}_n) - y_n|^2 \text{ for } \underline{x}_n \in X, y_n \in Y$$

$$\text{MSE}_{\text{test}}(\hat{f}, \underline{y}) = \frac{1}{N'} \sum_{n=0}^{N'-1} |\hat{f}(\underline{x}_n) - y_n|^2 \text{ for } \underline{x}_n \in X', y_n \in Y'$$

3. Algorithm Implementation and Development

We will use

- `sklearn` [6] to call
  - `PCA`
    - `n_components`
    - `.fit`
    - `.components_`
    - `.singular_values_`
    - `.transform`
  - `RidgeCV`
    - `.fit`
    - `.predict`
  - `mean_squared_error` to calculate the training and test MSE
- `numpy` [1] for mathematical operations
- `matplotlib` [5] to plot figures

We will use rank(A) by SVD to determine the dimentionality of our train set

4. Computational Results

We are given a data set that contains training and test set to distinguish images of handwritten digits. We will use $X_{\text{train}} \in \mathbb{R}^{2000 \times 256}$ to denote the matrix of training features and $Y_{\text{train}} \in \mathbb{R}^{2000}$ to denote the vector of training labels. We will use $X_{\text{test}} \in \mathbb{R}^{500 \times 256}$ to denote the matrix of test features and $Y_{\text{test}} \in \mathbb{R}^{500}$ to denote the vector of test labels. We fit PCA on X_{train} to get 256 PCA modes. Here is the visualization of the first 16 PCA modes as 16×16 images:

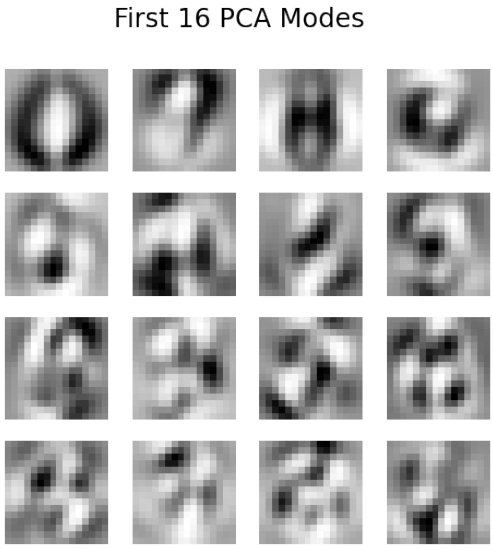

FIGURE 2. Visualization of the first 16 PCA modes

We plot the singular value of the fitted X_{train}:

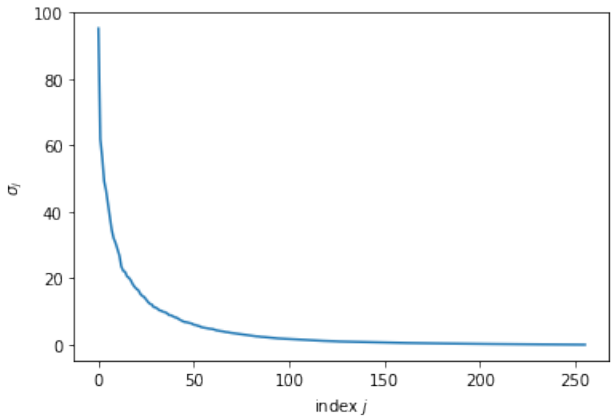

FIGURE 3. Singular value of the fitted X_{train}

We can see from the plot that the first 50-100 singular values are dominant, which indicates that X_{train} is close to a rank(50-100) matrix, so the effective dimension of the fitted X_{train} is in between 50-100.

If we want to approximate X_{train} up to 60%, 80%, and 90% in the Frobenius norm, which is approximately `190.27893628308198`, we get the minimum of PCA modes needed is `3, 7, 14`, which indicates that we do not need to use the entire PCA modes.

To train the classifier, we first extract the features and labels of the 2 digits m and n of the distinguishing digits pair from the training set. The features are representing as $X_{(m,n)}$ and the labels are representing as $Y_{(m,n)}$. By projecting $X_{(m,n)}$ on the first 16 PCA modes of X_{train}, we get a matrix A_{train} which contains 455 m's and n's with the features of the first 16 PCA modes. Assign label -1 to the images of the digit m and label +1 to the images of the digit n to get a vector $b_{\text{train}} \in \{-1, +1\}^{455}$. Then we fit the RidgeCV regression on

A_{train} and b_{train} to get a predictor to predict on A_{train} to get the predicted value of how the PCA modes distinguish the two different digits. Use the predition with b_{train} to calculate $\text{MSE}_{\text{train}}(m, n)$. Do the same procedure on the test data set to get A_{test} and b_{test}. Use the same predictor to predict on A_{test} to calculate $\text{MSE}_{\text{test}}(m, n)$.

Plug in $(1, 8)$, $(3, 8)$, and $(2, 7)$ sequentially as (m, n). We get the following output of MSE:

```
MSE_train( 1 , 8 ):  0.07545046193479041
MSE_test ( 1 , 8 ):  0.0825830416996403
MSE_train( 3 , 8 ):  0.18121824942014883
MSE_test ( 3 , 8 ):  0.2591148603960329
MSE_train( 2 , 7 ):  0.09284760866898845
MSE_test ( 2 , 7 ):  0.13143223499625878
```

We can see that the digit pair (1, 8) has the smallest $\text{MSE}_{\text{train}}(m, n)$ and $\text{MSE}_{\text{test}}(m, n)$, and the digit pair (3, 8) has the greatest $\text{MSE}_{\text{train}}(m, n)$ and $\text{MSE}_{\text{test}}(m, n)$. We can explain this just by the character of digit 1, 3, and 8. Comparing to the pair (1, 8), digit 3 and 8 have more similarities in the structure of the figure.

5. Summary and Conclusions

We use PCA modes and RidgeCV regression to train our classifier to distinguish digits and get the mean squared error of the training set and test set for 3 pairs of digits. When comparing $\text{MSE}_{\text{train}}(m, n)$ and $\text{MSE}_{\text{test}}(m, n)$, the $\text{MSE}_{\text{test}}(m, n)$ is slightly higher, this might due to the smaller number of instances in the test set. Moreover, there are similarities in the structure of different digits, especially in handwritten form, it could be harder to distinguish the digits with more similarities. However, with a relatively small mean squared error, we can conclude that our classifier is successful.

Acknowledgements

The author is thankful to every thought in the AMATH 582/482 Discord server, especially the help of classifying the methods and parameters in `sklearn`.

References

[1] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020.

[2] B. Hosseini. Evaluating sl models. University of Washington-Seattle (LOW 216), Jan 2022. AMATH 482/582.

[3] B. Hosseini. Principal component analysis. University of Washington (LOW 216), Jan 2022.

[4] B. Hosseini. Review of linear algebra. University of Washington (LOW 216), Jan 2022.

[5] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[6] 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.