Coding Project 4: Teaching a Computer to Recognize Written Numbers

Hao Li
Department of Applied Mathematics
University of Washington
haoli25@uw.edu

Abstract

In this project, we explore the task of teaching a computer to recognize handwritten digits. To achieve this, we use wavelet decomposition to analyze the images and extract relevant features. Next, we apply singular value decomposition (SVD) to the wavelet coefficients to reduce the dimensionality of the feature space and extract the most informative features. We then use linear discrimination analysis (LDA) to classify digit 0 or 1, achieving a high accuracy of 99.72%. Building on this initial success, we extend our algorithm to classify all 10 digits with an overall accuracy 94%. Moreover, we did a comparison with other machine learning algorithms, namely support vector machines (SVMs), decision trees, and neural networks and found out that SVM provides the best overall accuracy at 98%.

1 Introduction

The human visual system is one of the wonders of the world. Consider the following sequence of handwritten digits:

most of us can easily recognize those digits are: 504192 [1]. To teach a machine to recognize these digits, we need to feed them large datasets to learn patterns and relationships within the data, and hopefully, they can generalize the patterns to recognize unseen new data. We call such algorithm: Machine Learning (ML). Depending different tasks, ML can be classified as supervised, unsupervised, and reinforcement learning. In this project, we are performing a supervised learning algorithm by feeding the machine 30,000 images[1] with correspond labels.
Now we are going to introduce some theoretical background behind our algorithms and provide the numerical steps in section 3. Plots and results analysis are listed in section 4.

2 Theoretical Background

Assume readers already had backgrounds for SVD. We are going to introduce some backgrounds for wavelet transform and LDA in this section.

[1]30k dataset includes all 10 classes but during digit 0 or 1 classification, we have only 6370 images for these two digits.

2.1 Wavelet tranform

Wavelet-transformation is similar to Fourier-transformation but with additional special properties of the wavelets. In short words, it decomposes an image into wavelets representing different frequencies and orientations, enabling analysis of signals at different scales. In image classification, wavelet-based approaches extract relevant features such as texture and edges, improving classification accuracy. In our case, we are using Haar wavelet as our base function. The Haar wavelet's mother wavelet function can be defined as:

$$\psi(t) = \begin{cases} 1 & \text{if } 0 \leq t < \frac{1}{2} \\ -1 & \text{if } \frac{1}{2} \leq t < 1 \\ 0 & \text{otherwise.} \end{cases}$$

After transformed to the wavelet basis, we can keep the most informative features and remove the redundant features by applying SVD.

2.2 Linear Discriminant Analysis

In image classification, the goal is to assign an input image to one of several predefined classes based on its features. LDA is a supervised learning algorithm that works by finding the optimal linear projection of the input data onto a lower-dimensional space that maximizes the separability of the different classes with means μ_1 and μ_2. Thus, the key idea for LDA is to find a suitable projection that maximizes the distance between the inter-class data while minimizing the intra-class data [2]. Mathematically, we want to construct a project $\boldsymbol{w}$ such that

$$\boldsymbol{w} = \text{arg} \max_{\boldsymbol{w}} \frac{\boldsymbol{w}^T \boldsymbol{S}_B \, \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_W \, \boldsymbol{w}} \tag{1}$$

where $\boldsymbol{S}_B, \boldsymbol{S}_W$ are between-class-variance and within-class-variance which are given by

$$\boldsymbol{S}_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T \tag{2}$$

$$\boldsymbol{S}_W = \sum_{j=1}^{2} \sum_{\boldsymbol{x}} (\boldsymbol{x} - \mu_j)(\boldsymbol{x} - \mu_j)^T \tag{3}$$

By learning such projection $\boldsymbol{w}$, machine should be able to discriminate two different digits by projecting the data on it.

3 Numerical Methods

The basic algorithm can be implemented as follows:

i Reshape the training and test data to a `(n_features, n_samples)` shape and use `dwt2(data, 'haar')` to conduct the wavelet transform.

ii Apply `numpy.linalg.svd(,full_matrices=False)` of the transformed data to extract important features and project it onto the principal components.

iii For binary classification, we only pick 0's and 1's from the training dataset. Then we calculate the within class, between class variances, and projection line $\boldsymbol{w}$ according to formula (1), (2), and (3).

iv After orient all the 0's on the lower end and all the 1's on the higher end, we can calculate the threshold value.

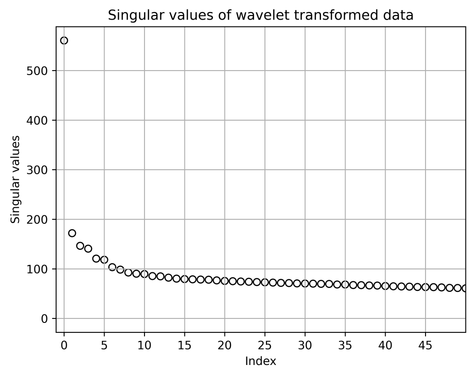
v Load our test dataset. After wavelet transform and principal-components projection, we project the data onto our best line $\boldsymbol{w}$ which gives us the best separation of these two classes.

vi Compare with `Y_test` and compute the accuracy.

vii Repeat steps i$\sim$vi for each rest of the digits and note that we need to loop through the digits and conduct LDA on each digit vs. all the other digits.

4 Results

4.1 Plots



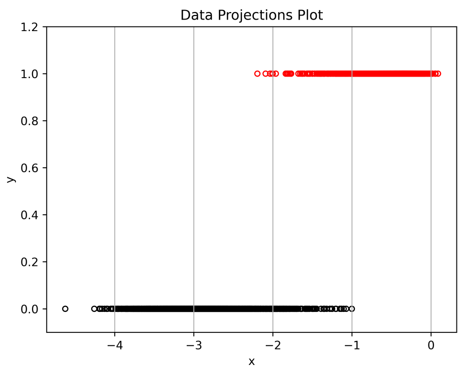


Figure 1: First 50 singular values (left); Data project onto the best line (right)

From the singular value plot (left), we see the first 10 singular values dropped drastically and then decay very slowly. Hence, the first 15 features will be a good approximation.
From the data projection plot (right), we see our best projection line $\boldsymbol{w}$ does a pretty good job by separating classes 0 (black) and 1 (red). Most of the data (dense parts) are clearly separated but there are still some overlaps which might cause some mistakes in our test data. After projecting our test data onto the line, the accuracy of successfully classified these two digits is 99.72%. Considering we use smaller (1062 test data) and very balanced (500 vs. 562) datasets, it should have very hight accuracy. However, later if we loop through the digits and conduct LDA on each digit vs. all the other digits, we will generate very unbalanced data, which will for sure decrease the accuracy as we see in the next table.

4.2 Table of accuracy for all 10 classification

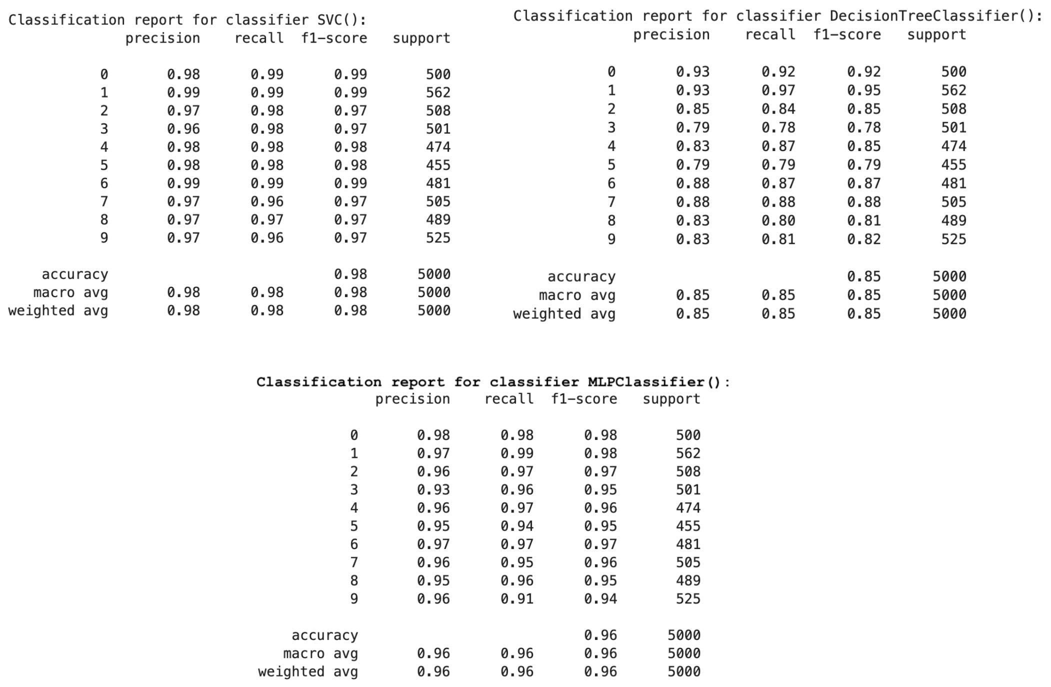
Digits	0	1	2	3	4	5	6	7	8	9
Accuracy %	96	98	93	92	91	91	96	96	93	93

As we expected, since we are doing binary classification (one digit vs. all the other digits), the other digits will have much more images. Intuitively, machine will see other digits more frequently than the one that it wants to classify, which will cause some drop in accuracy. However, the overall accuracy (the average of all 10 classification) is 94% which is still a very good result in image classification. In terms of each digit, we see 1 has the highest accuracy (98%). This is probably because 1 has the simplest shape among all these digits which made machine easier to generalize its pattern. In contrast, 4 and 5 have relatively more complex shape (like they don't a clear geometric pattern. People can write them in very different ways). Hence, they have the lowest classification accuracy.

4.3 Results of other ML algorithms under the same datasets

From `sklearn` library, I import `svm, DecisionTreeClassifier`, and `MLPClassifier` to classify 10 digits under the same datasets.
For `svm` and `DecisionTreeClassifier`, I used the default parameter settings which can be found in their documentation: `sklearn.svm.SVC`, `sklearn.tree.DecisionTreeClassifier`.
For `MLPClassifier`, here is my parameters setting:

```
mlp = MLPClassifier(hidden_layer_sizes=(286,), activation='logistic',
                                alpha=1e-5,
                    solver='sgd', tol=1e-8, random_state=1,
                    learning_rate_init=.01, verbose=True)
```



```
Classification report for classifier SVC():
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       500
           1       0.99      0.99      0.99       562
           2       0.97      0.98      0.97       508
           3       0.96      0.98      0.97       501
           4       0.98      0.98      0.98       474
           5       0.98      0.98      0.98       455
           6       0.99      0.99      0.99       481
           7       0.97      0.96      0.97       505
           8       0.97      0.97      0.97       489
           9       0.97      0.96      0.97       525

    accuracy                           0.98      5000
   macro avg       0.98      0.98      0.98      5000
weighted avg       0.98      0.98      0.98      5000
```

```
Classification report for classifier DecisionTreeClassifier():
              precision    recall  f1-score   support

           0       0.93      0.92      0.92       500
           1       0.93      0.97      0.95       562
           2       0.85      0.84      0.85       508
           3       0.79      0.78      0.78       501
           4       0.83      0.87      0.85       474
           5       0.79      0.79      0.79       455
           6       0.88      0.87      0.87       481
           7       0.88      0.88      0.88       505
           8       0.83      0.80      0.81       489
           9       0.83      0.81      0.82       525

    accuracy                           0.85      5000
   macro avg       0.85      0.85      0.85      5000
weighted avg       0.85      0.85      0.85      5000
```

```
Classification report for classifier MLPClassifier():
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       500
           1       0.97      0.99      0.98       562
           2       0.96      0.97      0.97       508
           3       0.93      0.96      0.95       501
           4       0.96      0.97      0.96       474
           5       0.95      0.94      0.95       455
           6       0.97      0.97      0.97       481
           7       0.96      0.95      0.96       505
           8       0.95      0.96      0.95       489
           9       0.96      0.91      0.94       525

    accuracy                           0.96      5000
   macro avg       0.96      0.96      0.96      5000
weighted avg       0.96      0.96      0.96      5000
```

Figure 2: `metrics.classification_report` for `svm` (left), `DecisionTreeClassifier` (right), and `MLPClassifier` (bottom)

From the report, we can see SVM has outperform all other algorithms with f1-score range from 97% - 99% and overall accuracy at 98%. The second best performance is neural networks with an overall accuracy 96%. Our method LDA is in the third place whereas DecisionTree gives us the worst accuracy.
From my understanding, this is because DecisionTree is very bad at dealing with high-dimensional data. Intuitively, DecisionTree draws a 2D line to separate all the data points, once the data are in high-dimension, DecisionTree will have hard time to classify them. Other the hand, SVM is very good at generalize the high-dimensional hyperplane which performs well in image classification as it should be. Neural network should give the similar results as SVM but I didn't have time to tune the best parameters so, I would expected a very close performance as SVM after tuning the parameters.

5 Conclusion

In this project, we have presented a successful approach to recognizing handwritten digits using linear discrimination analysis (LDA). Our results demonstrate that this method is effective and achieves high accuracy for recognizing the digits 0 and 1, as well as for all 10 digits. We also compared our approach with other machine learning algorithms, including support vector machine (SVM), decision tree, and neural networks, and found that SVM provided the best overall accuracy at 98%. Further work could focus on extend our LDA algorithm to multi-classes with more balanced data distribution.

6 Acknowledgment

Thanks for the help from Katie's and Professor Amin's office hour and users from the `# project-help` channel in `Amath 4/582` discord server.

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[2] J.N. Kutz. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*. Data-driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data. OUP Oxford, 2013.