Final Report For Digit Classification

WANG Huanchen, WANG Piaohong

1 Introduction

In this project, we plan to work on the handwritten digit classification, which is a traditional classification task in the *ML* area. Our project is based on the subset of the MNIST digits. The dataset has 10 classes (digits 0 through 9) with 4000 images (400 images per class), which are 2000 samples for training and 2000 samples for testing, and each image has 784 pixels (28×28). We would like to try some classical machine learning algorithms which we have learned in the lecture to evaluate their performance on the handwritten digit classification and find the approach to improve the results.

Image classification is a very traditional task in the computer vision area. It is the basis for other senior tasks such as target location, detection, and segmentation. In recent years, the deep learning method has dominated the image classification task [1]. However, some traditional machine algorithms can also perform well on the classification task, especially for some simple datasets. Therefore, in this project, we try several traditional machine learning algorithms for handwriting digit classification. To empower the performance of these algorithms, we also do the reduction and extraction of features from images for better classification performance. We believe this can greatly help the algorithms identify the key features of different digits.

2 Methodology

Generally, we try two different pipelines to classify the handwriting digit images.

Firstly, we train different algorithms directly based on the features of the gray value, which also serves as the baseline of our experiment. These algorithms are included: 1). **Support Vector Machine (SVM),** 2). **K Nearest Neighbor (kNN),** 3). **Logistic Regression,** 4). **Perceptron,** 5). **Naive Bayesian.** In addition, we focus on the hyperparameters tune for better performance of classification in each algorithm, especially on the SVM and kNN. What's more, we also reduce the amount of training data available by selecting a subset of the samples to find which models are more robust with fewer data and which tend to overfit.

Secondly, we propose a new framework to improve the performance of classification, which includes several following steps: 1). Recover the original image from the grey value features. 2). Reduction of useless features in the images and extraction of some critical features from the recovered images. 3). Re-train the model on each feature extraction method and compare their performance on time, space cost, and classification accuracy with the baseline method. These methods are included: 1). **Principal Component Analysis (PCA),** 2). **Linear Discriminant Analysis (LDA),** 3). **Scale-invariant Feature Transform (SIFT). [2]** Our pipeline can be depicted in Figure 1.

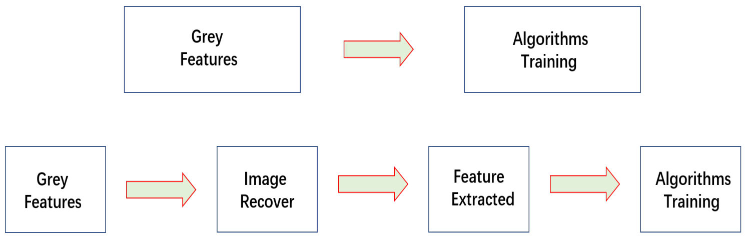


Figure 1: Pipeline of the project

3 Experiments

3.1 Experimental Setting

In this project, we will train five different kinds of models on the digit 4000 handwriting digit datasets. We adopt accuracy (acc) Equation 3.1 as our performance indicator, which is calculated as:

$$\frac{1}{N}\sum_{i=1}^{N}\mathbf{1}(\bar{y}_i = y_i) \tag{1}$$

We choose digit 4000 datasets as our training dataset, which include 4000 handwriting digits and their labels. We randomly select 50% from them as the test set. And for the parameter estimation, we do the normalization for the data feature initially.

3.2 Baseline

3.2.1 Implementation Details

Table 1: Results of each algorithm without tuning

Models	mean Accuracy	std.
SVM	**0.9426**	0.0039
3-NN	0.9143	0.0049
Logistic Regression	0.8713	0.0059
Perceptron	0.8452	0.0073
Naive Bayesian	0.8295	0.0066
1-NN	0.9160	0.0035

For the first pipeline, the baseline is the traditional models (e.g., SVM) directly trained in the digit 4000 datasets with the random 50% training dataset in default. We don't introduce any features except for the original 28X28 gray features. For our method, we introduce PIL[1] [3] and OpenCV[2] [4] library for recovering and extracting the features from datasets. For the

[1]`https://pillow.readthedocs.io/`
[2]`https://opencv.org/`

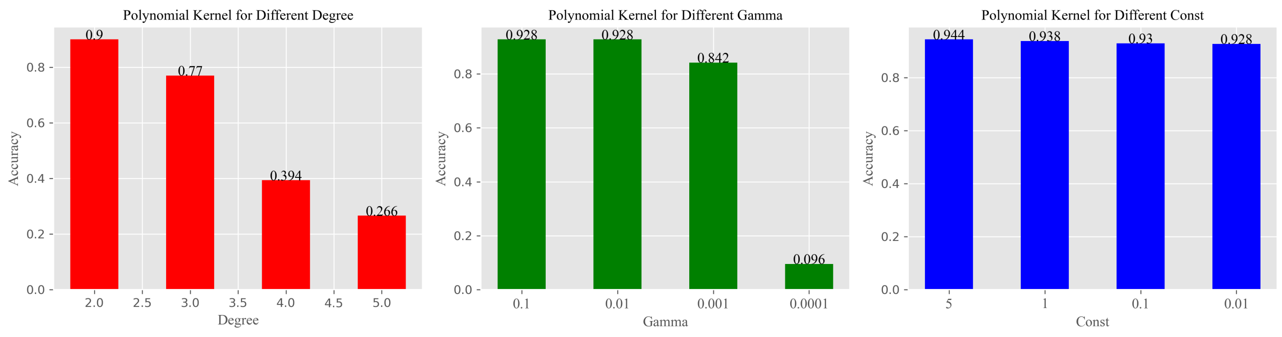
training of the algorithm, we adopt the implementation in the skicit-learn[3] [5] library for our training process.

3.2.2 Results On digit4000

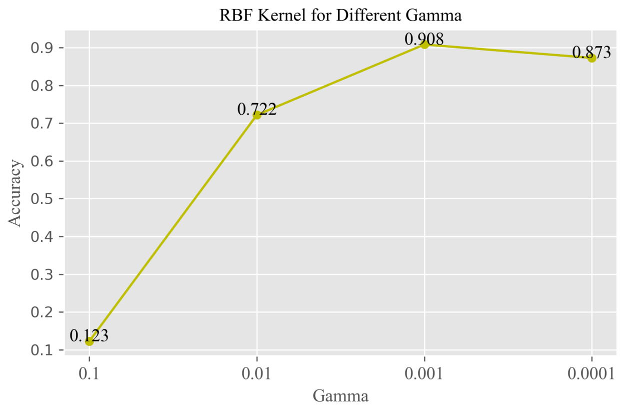
To ensure a confident result, we run multiple (10) trials with different random 50% subsets in different algorithms and take the average accuracy. Here, we report the accuracy of our methods without fine-tuning hyperparameters and baseline on digit4000 datasets in Table 1.

3.3 Estimation of Parameter and Kernel Function

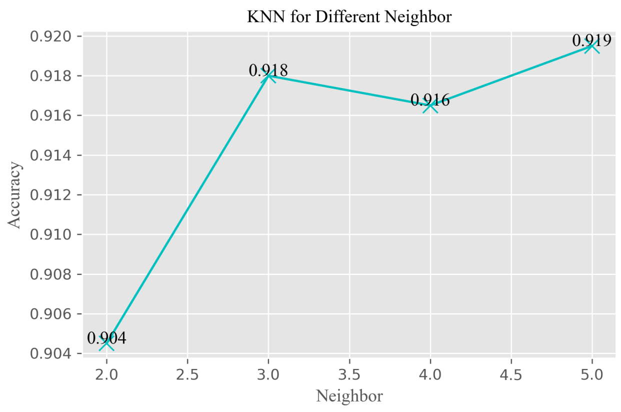
For these above algorithms, we normalize the dataset for data pre-processing and do the estimation of parameters and kernel function on some algorithms.



(a) Results of each parameters in polynomial function



(b) Results of each coefficient in rbf function



(c) Results of each number of nearest neighbor in kNN

Figure 2: Results of estimation of parameters in each algorithm

3.3.1 To the SVM,

we focus classification performance on the different kernel functions and their different parameters.

[3]`https://scikit-learn.org/`

First, we run the basic SVM with the linear kernel function as Equation 2 in 50% training set and get the mean accuracy is **0.9040** with nearly 0 standard variances.

$$\text{linear:} \qquad k(x,z) = x^T Az, \qquad \text{where A is a positive definite matrix.} \tag{2}$$

Second, we run the SVM with the polynomial kernel function as Equation 3.3.1 in 50% training set. With the polynomial kernel function, we give a set of the coefficients α, degrees of the polynomial q, and the constants c for the estimation parameters and find the best choice for the performance of classification.

$$\text{polynomial:} \qquad k(x,z) = \alpha(x^T z + c)^q, \qquad \text{where q is a positive integer.} \tag{3}$$

For the degree of polynomial q, we give the **{2, 3, 4, 5}**, and the results of accuracy are displayed in the left subfigure of Figure 2a. For the coefficients α, we give the **{0.1, 0.01, 0.001, 0.0001}**, and the results of accuracy are displayed in the center of Figure 2a. For the constant c, we give the **{5, 1, 0.1, 0.01}**, and the results of accuracy are displayed in the right subfigure of Figure 2a

Third, we run the SVM with the Gaussian(rbf) kernel function as Equation 3.3.1 in 50% training set. With the rbf kernel function, we give a set of the coefficients α for the estimation parameters and find the best choice for the performance of classification. For the coefficient α, we give the **{0.1, 0.01, 0.001, 0.0001}**, and the result of accuracy are displayed in the Figure 2b

$$\text{Guassian:} \qquad k(x,z) = exp(-\alpha||x - z||^2), \qquad \text{for } \alpha > 0. \tag{4}$$

According to the results, there are some conclusions about the SVM kernel function and parameter in the digit 4000 dataset:

3.3.2 To the kNN,

we focus classification performance on the different numbers of the nearest neighbors K. We give the **{2, 3, 4, 5}**, and the results of accuracy are displayed in Figure 2c. According to the results, we can get that all of the results perform well due to the uniformly sampled digist4000 dataset, and when the number of nearest neighbors is 5, kNN methods have the highest accuracy.

3.4 Different Training Dataset Size

We try to reduce the number of training data to find which models are more robust with less training data. For the size of the training dataset, we give the
textbf0.9, 0.75, 0.5, 0.25} and observe the performance of different algorithms under these settings, and the results are demonstrated in Figure 3

According to the results, we argue that the SVM gets the best performance on accuracy

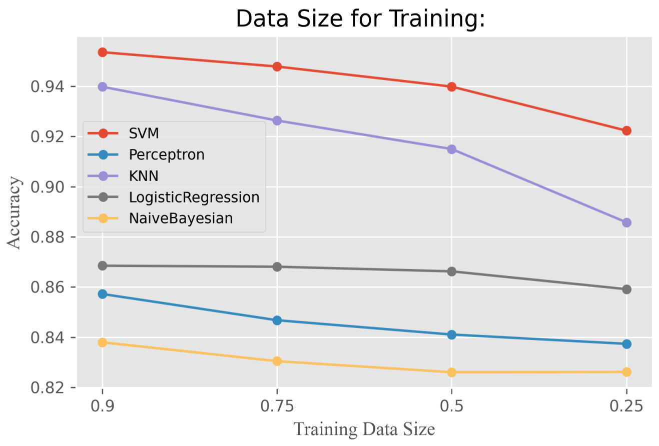


Figure 3: Results in different training sizes and algorithms

Models	90%	75%	50%	25%	std.
SVM	**0.9533**	**0.9482**	**0.9426**	**0.9225**	0.0117
Perceptron	0.8493	0.8500	0.8452	0.8386	0.0045
kNN	0.9370	0.9318	0.9143	0.8882	0.0190
Logistic Regression	0.8633	0.8698	0.8713	0.8595	0.0048
Naive Bayesian	0.8303	0.8260	0.8295	0.8229	**0.0029**

Table 2: Performance of different algorithms under various training dataset size

in each training size. However, the standard variance of Bayesian is the lowest. Therefore, we analyze that since the Naive Bayesian is based on applying Bayes' theorem and consider the conditional independence between every pair of features given the value of the class variable. Its distribution of each feature ensures it is impacted less (robust) by different training data sizes.

3.5 Feature Reduction and Extraction

As the data pre-processing on features reduction and extraction, there are below reasons for it:

- **Space.** Dataset may take up big space. Reducing features means that there are required less memory to store and process data.
- **Time**. Training a model on less data can save much time for efficiency improvement.
- **Accuracy**. Including redundant or irrelevant features means including unnecessary noise. Frequently, it happens that a model trained on less data performs better.
- **Interpretability**. A smaller model also means a more interpretable model. Explaining a model based on thousands of different factors would be unfeasible.

According to the above reasons, we select the **PCA, LDA, SIFT,** and also **t-SNE** for feature extraction. The first three focus on the training data, and t-SNE is for visualization.

For the unsupervised label method, the PCA, we keep the 95% principal components and extract the **147 features** from the 748 features in each image. And for the SIFT, each image will be extracted with a $n \times 128$ size description of key points, like the Figure 4 display. Then, we set the vocabulary package with 150 clusters, extract the features from the descriptions, and extract **150 features** in each image. For the supervised label method, the LDA, we select the

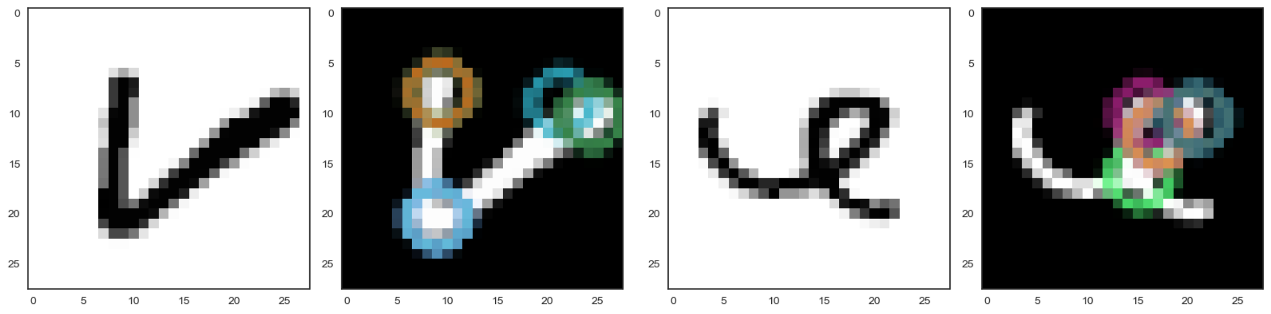


Figure 4: Key points of the image by SIFT

$\#classes - 1(9)$ **features** from the origin features in each image. After the feature extraction, there are the accuracy of SVM without any features extraction, with the PCA and SIFT, and the accuracy of LDA. What's more, each time cost of the set and the feature shape (space cost) of the set are displayed in the Figure 5

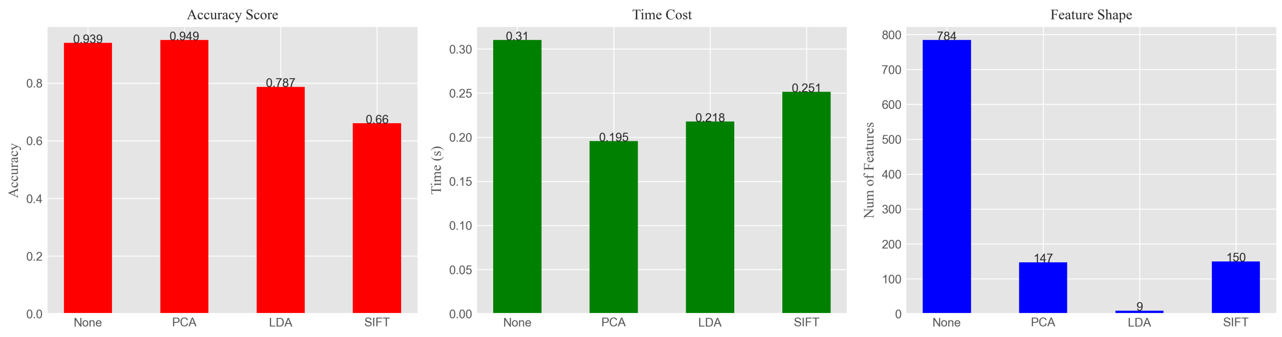


Figure 5: The comparison of each feature extraction method

According to the results, we can indicate that: 1). To accuracy, the PCA only keeps the **147 features** of the image but get the highest score among these methods without pre-labeling. 2) Since the PCA is an unsupervised method for feature reduction, it spends the lowest time **(0.195sec)** for training among these methods. 3). To space cost, since the maximum feature extraction from LDA is $\#classes - 1(9)$ based on the knowing label. However, it just gets **0.787%** accuracy.

Besides, we also do the feature extraction in constraint (only keep 2 features for visualization) among the t-SNE, PCA, and LDA. The results are shown in Figure 6. We can get that the t-SNE can well divide each class into different components. Because its loss function pays more attention to the local features instead of the global features, which lets it visualize each class better but not for the training model.

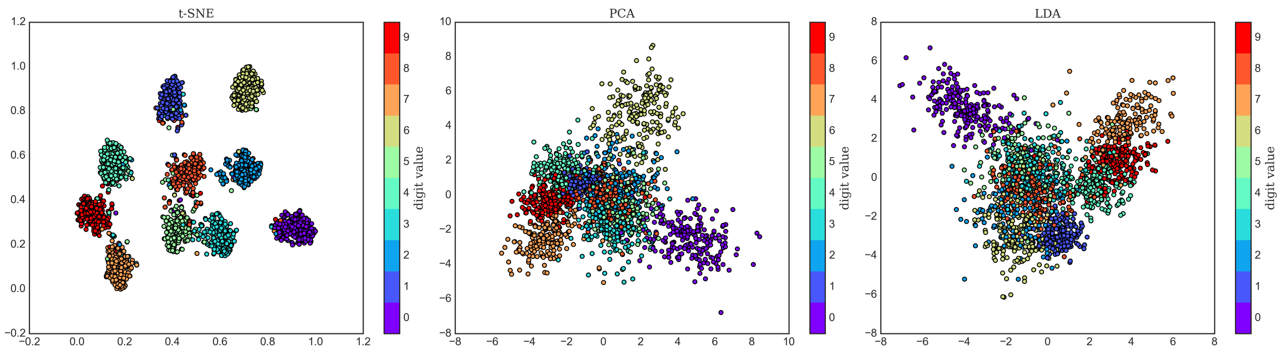


Figure 6: The visualization of 2 features extraction in each method

4 Conclusion

In this project, we initially run the baseline on some classical machine learning algorithms and evaluate and analyze their performance of results on the handwritten digit classification. Besides, we investigate some algorithms with different parameters and find the fine tune for improving their performance. Meanwhile, we do the normalization, feature reduction and extraction in data pre-processing to observe their influence on the final results.

Based on the above results in our experiments, we can conclude that the SVM classifier does better, especially in polynomial kernel function with **2 degrees, 0.1 coefficient, and 5 constant** can get the **0.9440%** accuracy on 50% training dataset. Each feature extraction method can strongly reduce the origin features for subsequent training and save space and time. However, only the PCA can help improve the accuracy of classification.

References

[1] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural computation*, vol. 29, no. 9, pp. 2352–2449, 2017.

[2] P. C. Ng and S. Henikoff, "Sift: Predicting amino acid changes that affect protein function," *Nucleic acids research*, vol. 31, no. 13, pp. 3812–3814, 2003.

[3] "Image module," 2022. `https://pillow.readthedocs.io/en/stable/reference/Image.html`.

[4] "Introduction to sift (scale-invariant feature transform)," 2022. `https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html`.

[5] "Api reference," 2022. `https://scikit-learn.org/stable/modules/classes.html`.