CSC371 Project 2 Report: Hand-Written Digits Recognition

Niya Ma, Eric Xu

Abstract

In this paper, we demonstrate the process of training a model using various machine learning techniques to process and recognize handwritten digits from 0 to 9 in the MNIST database[1]. We use scikit-learn's vector classification method as well as the K-nearest neighbor method and are able to produce satisfactory results.

Introduction

Most organizations and individuals in the United States rely on mailing services for tax filing and delivering many other important documents. Therefore, efficiency and security are much needed in the mailing system. To improve mail sorting efficiency, we will demonstrate how we trained a model to recognize handwritten digits.

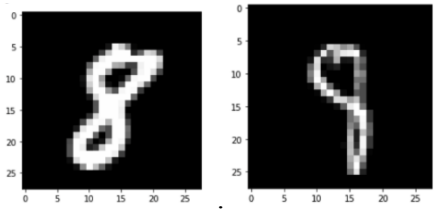
Background

The MNIST dataset is a subset of NIST. The MNIST database contains 60,000 training examples and 10,000 testing examples. The objects in MNIST are 28*28 grayscale images of handwritten digits, centered and normalized. The training set consists of 30,000 images from SD-3, digits written by Census Bureau employees, and 30,000 images from SD-1, which are collected among high school students. The mix of SD-3 and SD-1 diversifies data sourcing and enables the model to recognize handwritten digits from different demographics.

Experiments

Data Exploration

Since the goal of the model is to predict hand-written digits, we loaded the dataset using sciki-learn[2] and plotted the images to have an intuitive sense.



(two images of digit 8)

We found that the resolution of each image is quite high, therefore if we reduce the dimension of pixels of images, we can decrease the operation time significantly.

Dimension Reduction

We averaged adjacent pixels to turn each image from 28*28 to 14*14, which nearly does not affect the accuracy of the model but improves runtime significantly.

Normalization

For normalizing the dataset, we initially chose the way of normalization identical to Project 1, which is normalized_data = (data - mean)/(standard deviation). However, we eventually decided to normalize the data to 0 to 1 by using the following algorithm, x = (x - np.amin(x)) / (np.amax(x) - np.amin(x)), because in this way most of the data points will be zeros and the computer will perform faster than the previous method.

Hyperparameter Tuning

When using scikit-learn package to construct the model, we have to decide the value of gamma, the coefficient in the model. We conducted a test to find the best value. The test result is as follows:

---

[1] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine, 29(6), 141–142. http://yann.lecun.com/exdb/mnist/.

[2] Sklearn load digits https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html

- The accuracy rate is at its peak, which is 98%, between 0-0.42
- Accuracy rate decreases starting from 0.43

```
Classification report for classifier SVC(gamma=0.05):
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       980
           1       0.98      0.99      0.99      1135
           2       0.97      0.98      0.97      1032
           3       0.97      0.98      0.98      1010
           4       0.98      0.97      0.98       982
           5       0.98      0.97      0.98       892
           6       0.99      0.98      0.98       958
           7       0.97      0.96      0.97      1028
           8       0.97      0.97      0.97       974
           9       0.97      0.96      0.96      1009

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000

Classification report for classifier SVC(gamma=0.43):
              precision    recall  f1-score   support

           0       0.99      0.98      0.99       980
           1       0.99      0.99      0.99      1135
           2       0.90      0.99      0.94      1032
           3       0.99      0.97      0.98      1010
           4       0.98      0.98      0.98       982
           5       0.98      0.97      0.98       892
           6       0.99      0.97      0.98       958
           7       0.98      0.96      0.97      1028
           8       0.97      0.98      0.97       974
           9       0.98      0.95      0.96      1009

    accuracy                           0.97     10000
   macro avg       0.98      0.97      0.97     10000
weighted avg       0.98      0.97      0.97     10000

Classification report for classifier SVC(gamma=0.5):
              precision    recall  f1-score   support

           0       0.99      0.98      0.99       980
           1       0.99      0.99      0.99      1135
           2       0.84      0.99      0.91      1032
           3       0.98      0.97      0.98      1010
           4       0.98      0.97      0.97       982
           5       0.98      0.96      0.97       892
           6       1.00      0.96      0.98       958
           7       0.99      0.96      0.97      1028
           8       0.97      0.97      0.97       974
           9       0.98      0.93      0.96      1009

    accuracy                           0.97     10000
   macro avg       0.97      0.97      0.97     10000
weighted avg       0.97      0.97      0.97     10000
```

For convenience, we chose 0.1 as our final value for gamma.

Models:

KNN:

- 1 neighbor, 1000 training examples: 87.66 accuracy score
- 2 neighbor's average floored, 1000 training examples: 87.6

Vector Classification(sklearn.svm.SVC)[3]: scikit-learn built-in function for creating models.

- gamma: 0.1, accuracy 98%

Analysis:

Another model, LeNet-5[4] was trained on this dataset using CNN in 1998 with a testing accuracy of 98.74%. In 2020, a model built by Jay Gupta[5] using 5 convolutional layers reached an accuracy of 99.4%.

Results

The final accuracy for the scikit-learn constructed model is 94% on not dimension-reduced data, and 98% on dimension-reduced images, which suggests that loss of information can sometimes lead to better prediction. Another advantage of the scikit-learn model is that it has much less runtime than hand-written KNN code.

Conclusion

Compared to the dataset we analyzed in the last project, this dataset is a lot cleaner with more obvious patterns. However, there are very ambiguous handwritten digits in the dataset, leading to imperfect.

Ethics

- The model is for Arabic numerals(0 to 9) not other numerical systems. So the model might not perform well on datasets of other systems.
- The dataset being used describes digits in grey levels, so the model might not perform well on datasets other than grey-level images, such as RGB images.

---

[3] Sklearn Recognizing hand-written digits. https://scikit-learn.org/stable/auto_examples/classification/plot_digits_classification.html

[4] LeNet-5. LeCun et al., 1998. Gradient based learning applied to document recognition

[5] https://towardsdatascience.com/going-beyond-99-mnist-handwritten-digits-recognition-cfff96337392.

Contributions

Niya: Coded data normalization, pixel reduction, KNN model. Participated in SKLearn models. Wrote Abstract, Introduction, Background, and Experiments.

Eric: Participates in data normalization, pixel reduction, and Hyperparameter determination. Writes in Experiments, Conclusion, Results, and Ethics part of the paper