Rijal Aayush; International Journal of Advance Research and Development

INTERNATIONAL JOURNAL OF ADVANCE RESEARCH AND DEVELOPMENT

IJARnD

(Volume 4, Issue 7)

Available online at: www.ijarnd.com

Machine learning implementation to recognize manuscripts

Aayush Rijal

Student, Kalinga Institute of Industrial Technology, Bhubaneswar, Odisha

ABSTRACT

A field of studies in artificial intelligence, computer vision, and pattern recognition is handwritten character recognition. In paper files, images, touch-screen devices, and other sources, a computer performing handwriting recognition is said to be able to obtain and identify characters and transform them into machine-encoded form. Its implementation can be discovered in optical character recognition, handwritten document transcription into digital papers, and more sophisticated smart character recognition technologies.

Keywords— *Ml, Ai, Support Vector Machines (SVMs), Nearest Neighbor (NN), Principle Component Analysis (PCA)*

1. INTRODUCTION

A field of studies in artificial intelligence, computer vision, and pattern recognition is handwritten character recognition. In paper files, images, touch-screen devices, and other sources, a computer performing handwriting recognition is said to be able to obtain and identify characters and transform them into machine-encoded form. Its implementation can be discovered in optical character recognition, handwritten document transcription into digital papers, and more sophisticated smart character recognition technologies.

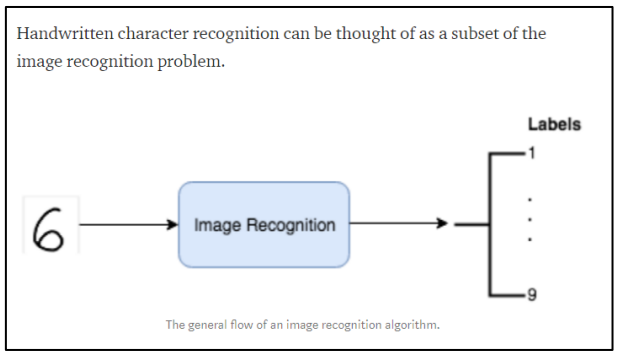

Fig. 1: General flow of an image recognition algorithm

The algorithm basically requires a picture (picture of a handwritten digit) as an input and produces the probability that the picture belongs to distinct classes (machine-encoded numbers, 1–9). In this blog post, with a mixture of machine learning methods, I will elaborate on my strategy for solving this issue.

Problem Statement Classification of handwritten numbers is the issue with this project. The objective is to take a handwritten digit picture and determine what the digit is. The numbers range from one (1) to nine (9).

We will look into the methods for solving the issue using Support Vector Machines (SVMs) and Nearest Neighbor (NN). The activities concerned are as follows:

- Download the MNIST dataset
- Preprocess the MNIST dataset
- Train a classifier that can categorize the handwritten digits
- Apply the model on the test set and report its accuracy

The dataset for this problem will be downloaded from kaggle, which was taken from the famous MNIST (Modified National Institute of Standards and Technology) dataset. Metrics the precision score will be used to quantify our model's output. The precision will inform us of the correct classification of the proportion of our test information. Precision is a useful metric option as it will be simple to compare the output of our model with the performance of the benchmark as it utilizes the same metric. Our dataset is also balanced (equal number of training examples for each label), making the precision suitable for this issue.

Data Exploration Some original information exploration shows a total of 42,000 samples and 784 characteristics in our training set. Each sample in the dataset represents a picture with a height of 28 pixels and a width of 28 pixels, thus a total of 784 pixels. For a total of 10 distinct labels, each picture is labeled with its respective category which is the real digit from 0 to 9.

Fig. 2: Some examples of the dataset

Exploratory Visualization We counted in the instruction set the number of events of each item. The figure below shows how these labels are distributed. From the figure, it is evident that the allocation means our dataset is balanced.

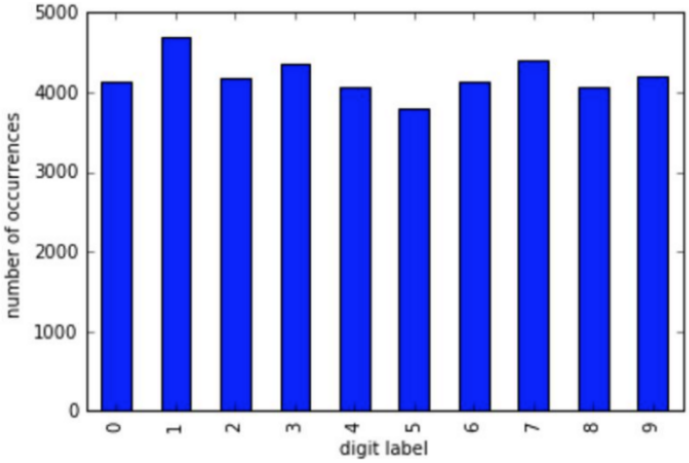

Fig. 3: number of occurrences of each label in the dataset

We would also like to understand more about average intensity, which for the distinct numbers is the average value of a pixel in a picture. Intuition informs me that the number "1" will have a lower intensity on average than the number "8."

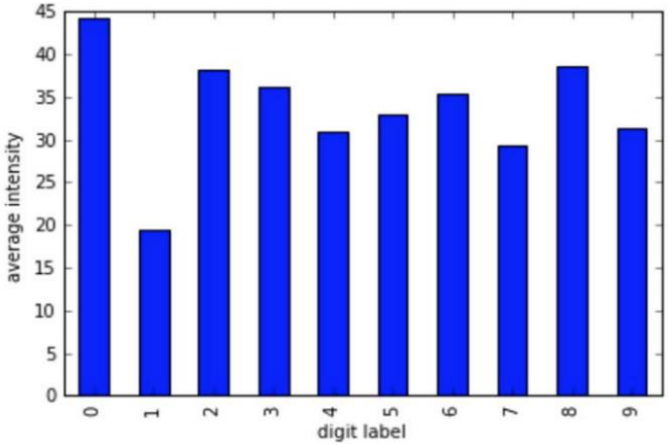

Fig. 4: Average intensity of each label in the dataset

There are variations in intensities, as we can see, and our intuition was right. "8" has an intensity higher than "1." Also, "0" has the greatest intensity, which is surprisingly even greater than "8." This could be ascribed to distinct individuals writing their numbers differently. Calculating the normal intensity deviation provides a value of 11.08 showing that there is some variation in how the numbers are written.

Algorithms and techniques, it has been shown that supporting vector machines (SVMs) can be applied to the recognition of images and handwritten characters [4]. Due to the high dimensionality of our input space, i.e. 784 characteristics, SVMs are efficient in high dimensional spaces, so it makes sense to use SVMs for this research. However, in big datasets, SVMs do not perform well as the training time becomes cubic in the dataset size. This could be a problem as there are 42,000 samples in our dataset which is quite big. To address this problem, we will implement a method suggested by research undertaken at Berkeley University of California to train a support vector machine to collect closest neighbors in a solution called "SVM-KNN"[2]. Training an SVM on the entire information collection is slow and it is not as natural to extend SVM to various classes as Nearest Neighbor (NN). Nevertheless, SVMs often perform better than other classification techniques in the neighborhood of a tiny amount of instances and a tiny amount of courses.

As an original pruning phase, we use NN and conduct SVM on the lower but more meaningful set of examples requiring cautious discrimination. This strategy represents how humans conduct coarse categorization: when presented with a picture, human observers can respond in as little as 150ms to coarse queries such as the presence or lack of an animal and, of course, can say which animal is given sufficient time[6]. The inspiration behind the "SVM-KNN" method was this method of rapid categorization, followed by consecutive finer but slower discrimination.

Benchmark To benchmark our model, we will use the benchmark model used by kaggle to create projections on the test set using a straightforward random forest model. The precision score of the benchmark model is 0.93, in other words, 93 percent of the test information can be properly labeled. We will assess how our final solution is compared to this model. We will strive to have greater precision than the benchmark model in our final solution.

Data Preprocessing Since the original dimension is quite large (784 input characteristics), it becomes necessary to reduce the dimensionality. First, we extract from the initial information on the main parts. We do this by fitting the instruction set with a Principle Component Analysis (PCA), then transforming the information with the PCA fit. To convert the dataset, we used the scikit-learn Python library's PCA module with n components set to 60. As shown in the figure below, roughly 97 percent of total information (in terms of total variance retained) can be interpreted by the first 60 main components, which is sufficient to represent the information in the original dataset. We, therefore, select the first 60 main parts as the characteristics extracted.

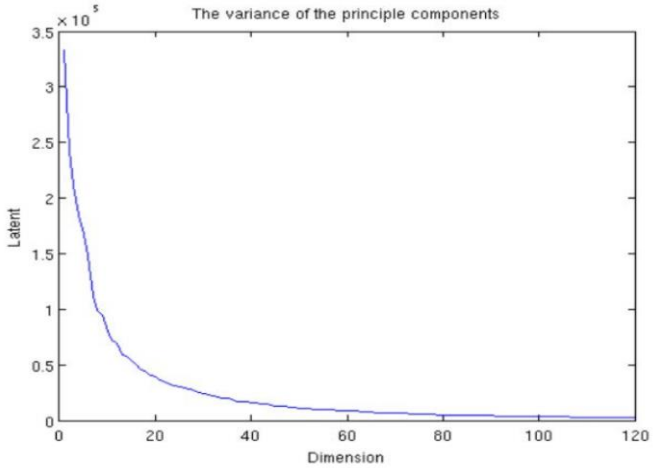

Fig. 5: Variance of principle componets

We also used cross-validation to divide the dataset into training and testing sets, maintaining 40% of the testing information. We use the scikit-learn Python library's StratifiedShuffleSplit module passing through the label values as one of the parameters. StratifiedShuffleSplit returns to train and test indices to divide the information into train and test sets while maintaining the proportion of each class sample. Implementation Our easy implementation of SVM-KNN goes as follows: we calculate the Euclidean query distances to all training examples for a query and select the closest K neighbors. If the neighbors of K have the same labels, the query will be labeled and exit. Otherwise, we are calculating the pair distances between the K neighbors, converting the distance matrix to a kernel matrix and applying SVM multiclass. The resulting classifier is lastly used to label the request.

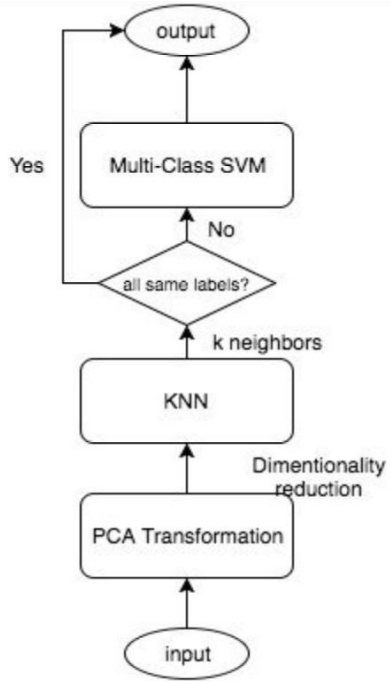

Fig. 6: Flow chart

All the algorithms used in our application were from the Python library, version 0.17.1 of the scikit-learn. We used PCA to reduce dimensions, StratifiedShuffleSplit to cross-validate, KNeighborsClassifier to discover the closest neighbors and SVC to train our multi-class SVM.

Model Evaluation and Validation We extract 60 main parts in our original application and use k=2 for KNN and C=1.0 for SVM parameters. A validation set was used to assess the model during growth. I divide the dataset into sets of practice and testing. The final hyperparameters were selected because among the tested pairs they performed the best. The highest outcomes were achieved with a final value of k=3 and C=0.5. For our model, a low k value makes sense because we are attempting to discover the few samples where NN has difficulty setting a decision boundary and apply SVM to conduct a more coarse-grained classification.
I use a cross-validation technique (StratifiedShuffleSplit) on the dataset to verify the robustness of the final model to ensure that the model is well generalized by using the entire dataset for both training and testing. The model classified the handwritten characters consistently with a precision of 97 percent.

Conclusion The 0.9714 classification precision is greater than the benchmark classification precision (0.93514). Therefore we can conclude that our model is sufficient to solve the issue of classifying handwritten characters in the MNIST dataset as it is capable of categorizing well with a precision that is very near to people. However, in a restricted domain, our model is helpful. To address a larger issue of acknowledging various numbers in a picture or acknowledging arbitrary multi-digit text in unconstrained natural pictures, some modifications would have to be made.

2. REFERENCES

[1] http://yann.lecun.com/exdb/publis/pdf/matan-90.pdf
[2] http://www.vision.caltech.edu/Image_Datasets/Caltech101/nhz_cvpr06.pdf
[3] http://www.johnwinn.org/Publications/papers/WinnCriminisi_cvpr2006_video.pdf
[4] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.441.6897&rep=rep1&type=pdf
[5] https://en.wikipedia.org/wiki/Support_vector_machine#Applications
[6] https://www.ncbi.nlm.nih.gov/pubmed/8632824
[7] https://github.com/chefarov/ocr_mnist/blob/master/papers/knn_MNIST.pdf

