Digital recognization by five methods

Lanhe Gao 2018533212
SIST
ShanghaiTech University
Shanghai, China
gaolh@shanghaitech.edu.cn

Yunfei Zhang 2018533098
SIST
ShanghaiTech University
Shanghai, China
zhanyf2@shanghaitech.edu.cn

Abstract

Recognizing handwritten numbers is a very classical machine learning problem, which has a variety of solutions. In this report, we try to implement five of these methods. By comparing their performances, we hope to choose the best one.

1 Introduction

Number recognization is a widely used application, with great sophistication. There are many types of machine learning methods that can be used in this scenario. In this report, we have adopted five methods, such as Support Vector Machine (SVM), K-Nearsest Neighbors (KNN), Naive Bayes (NB), Neural Networks (NN) and Logistic Regression (LR). We will analyze the accuracy, runtime performance, confusion matrix and other standards to compare their performance, and conclude the best method.

2 Data Setting and Methods

2.1 Data setting

Our training data and testing data are collected from *mnist* dataset. They're in the same form, with 28*28 pixels, therefore, we use a 784 dimensional vector to represent each number picture. The pictures are gray images, scaling from 0 to 255.

2.2 KNN

The first method we use is KNN. We convert the images to a 784 dimensional vector for easier calculation. Since the dimension is quite big, which makes the complexity of calculating 2-norm quite big, so we use 1-norm instead. KNN calculates K nearest images that has the smalles 1-norm distance to a given test data, and labels it as the class that contains the most number of neighbors.

The code implementation is at

`https://github.com/chuansao-258/machine-learning--digital-recognition/blob/main/knn.ipynb`

2.3 SVM

We use the svm from *sklearn* package. We directly put the data and label into the model *svm.fit()* for training, then use *svm.predict()* to run prediction.

The code implementation is at

Preprint. Under review.

https://github.com/chuansao-258/machine-learning--digital-recognition/blob/main/SVM.ipynb

2.4 NB

Before coding, we tried the naive bayes model in sklearn. However, it does not work well, with only 58% accuracy, so we try to implement one by ourselves. For training data, we calculate two thing: the probabilities of an image being labeled as each class; and the probabilities of the gray value of a pixel being 0 in each class, which means we choose every pixel as a feature.

Then, the prediction works like this: given a test image, we calculate the probability multiplication of all pixel points, and choose the label with the highest probability.

The code implementation is at

https://github.com/chuansao-258/machine-learning--digital-recognition/blob/main/bayes.ipynb

2.5 LR

As a matter of fact, Logistic Regression is a binary classification method. However, we want to classify these handwritings into 10 classes, so we need to extend LR method. For each class, we devide the dataset into the ones belong to the class, and the ones do not. This is a binary classification problem, and we need to do it 10 times. Finally, we choose the class that the input has the highest probability on.

2.6 NN

3 Results

3.1 Accuracy

As displayed in the jupyter notebook, the accuracy of these methods are

Table 1: Accuracy table

KNN	47%
NB	68.41%
SVM	72.92%