Letter Recognition with Different Classifications Using Convolutional Neural Network

Yaxin Wang

Research School of Computer Science
ANU College of Engineering & Computer Sciences
Australian National University
u6342442@anu.edu.au

Abstract. The objective is to design a classifier to recognize letters. The data we choose is the Letter Image Recognition Data from David J, Slate [1]. We use the 16 features as the inputs and set 26 outputs to represent the 26 capital letters. To get a better accuracy, we tried Decision Tree Classification, Maximum Likelihood Classification and Neural Network Classification and compare these results with the original results to get the better one. Decision Tree Classification can get better implementation results in certain areas but the accuracy we get is lower than the original one. Then, for the Maximum Likelihood Classification, we implement it but the result is not as good as the original one. And for the Neural Network Classification, we try different kinds of activation functions and pooling to get the one with better accuracy. At last, the accuracy I gets is worse than the accuracy of a paper using the same dataset.

Keywords: We would try to design a classifier to recognize letters.

1 Introduction

Recognition of letters is very important and the data we choose is the Letter Image Recognition Data created by David J, Slate [1]. The data contains more than 2000 instances extracted from letters raster scan images. Each of the instance has a letter category which is from ‘A’ to ‘Z’ and is also represented by 16 numerical features. The aim of our project is try to get a high-level classifier which can classify different letter with better accuracy.

We train the network with Stochastic Gradient Descent as the optimizer and use cross-entropy to evaluate the performance. As the input and output layers must be numeric, we use number 0-25 to represent the category ‘A’ to ‘Z’. As 0-1 range of the input data will get a better accuracy, we normalize all the features by constraining them to (0-1) range. We can separate the dataset in to training set and test set. We randomly choose 80% instances as the training set for training the model and the rest 20% instances as the test set for testing. Except this way, we can also separate the dataset into three parts, training set, validation set and test set, similarly, we can randomly choose 60% instances as the training set, 20% instances as the test set and 20% instances to be the validation set. After that, we choose to use the 16 features as the inputs and implement the Convolutional Neural Network. At last, what we are trying to do is to correctly identify the 26 capital letters by learning from the 16 features in the training set.

2 Method

Here are some methods we use. The first one is the original method. Then the following three methods are the techniques

from the paper I choose to improve the accuracy.

2.1 The original method

For the original Neural Network mentioned is a three-layer Neural Network, which contains 16 features as inputs and 26 outputs to represent the 26 capital letters. In order to have a higher accuracy, we need to choose a suitable model and avoid overfitting. We separate the dataset into three parts, we randomly choose 60% data for training set, 20% for test set, and the rest 20% as the validation set. To prevent overfitting and get a better model, we use 10-fold cross-validation and compare the results, then we determine the features for training, learning algorithm and the layout of the neural network. After that, to avoid overtraining, we train the neural network until it has got the highest accuracy. To make the result more reliable, we implement it for ten times and get the average. For the training set, the accuracy is 80.02%, for the test set, the accuracy is 78.11%. It means we have about 80% accuracy. To get a better accuracy, we also try some other ways, including Decision Tree Classification, Maximum Likelihood Classification and Neural Network Classification.

2.2 Decision Tree Classification

The Decision Tree Classification means we can solve a classification problem by asking a series of carefully crafted questions about the attributes of the test records, each time we receive an answer, a follow-up question is asked until we reach a conclusion about the class label of the record, the series of questions and their possible answer can be organized in the form of a decision tree, which is a hierarchical structure consisting of nodes and directed edges [2].

To build the tree, we use the decision tree learning algorithm called CART. It can give us a procedure to decide which questions to ask and when. CART stands for classification and Regression Trees. To begin, we need to add a root node for the tree, all nodes receive a list of rows as input and the root will receive the entire training set. Then each node will ask a true or false question about one of the features. And in response to this question, we split the data into two subsets. These subsets then become the input to two child nodes we add to the tree. The goal of the question is to produce the purest possible distribution of the labels at each node as we proceed down [7].

The trick to build an effective tree is to understand which question to ask and when. And to do that, we need to quantify how much a question helps to unmix the labels. We can quantify the amount of uncertainty at a single node using a metric called Gini impurity and quantify how much a question reduces that uncertainty using a concept called information gain. We use these to select the best question to ask at each point, and given that question, we can recursively build the tree on each of the new nodes. We continue dividing the data until there are no further questions to ask, at which point we can add a leaf [7].

Using this way, we build the Decision Tree Classifier using the CART, and we use the same training set and test set. After training the Neuron Network, the accuracy of the training set is 76.31% and the accuracy of the test set is 74.45%.

2.3 Maximum Likelihood Classification

Maximum likelihood estimation is a method that determines values for the parameters of a model. The parameter values are found such that they maximize the likelihood that the process described by the model produced the data that were actually observed [8].

The basic idea is that, given an unknown parameter θ and a sample data D_N, the maximum likelihood estimates $\theta^$ is the value for which the empirical likelihood $L_N(\theta)$ has a maximum [9].

$$\hat{\theta}_{\text{ml}} = \arg\max_{\theta \in \Theta} L_N(\theta)$$

The estimator $\hat{\theta}_{ml}$ is called the maximum likelihood estimator (m.l.e.). In practice, it is usual to consider the log-likelihood $l_N(\theta)$ instead of $L_N(\theta)$ since, being $\log(\cdot)$ a monotone function, we have [9].

$$\hat{\theta}_{\text{ml}} = \arg\max_{\theta \in \Theta} L_N(\theta) = \arg\max_{\theta \in \Theta} \log(L_N(\theta)) = \arg\max_{\theta \in \Theta} l_N(\theta)$$

The likelihood function quantifies the relative abilities of the various parameter values to explain the observed data, the principle of m.l. is that the value of the parameter under which the obtained data would have had highest probability of arising must be intuitively our best estimator of θ, in other terms the likelihood can be considered a measure of how plausible the parameter values are in light of the data [9].

2.4 Neural Network Classification

To find a better network topology, we change the layout and activation functions for the Convolutional Neural Network to see whether the classification can get a better result. The original one we use contains different combination of activation functions and max pooling for the two pooling layers. To get a more reliable result, in every case, we implement it for ten times and then get the average.

First, we change the activation functions of the CNN all together without changing others. The results are shown in the table below.

The activation Function	Accuracy of training set	Accuracy of test set
Linear Function	78.63%	76.32%
Sigmoid Function	80.02%	78.11%
Tanh Function	79.47%	77.92
ReLu Function	77.51%	75.28%
Current Function	80.02%	78.11%

From the table we can know that when we use the original combination of the activation functions, we can get a better accuracy, when we change them as shown in the table, the accuracy will decrease.

Then, we different kinds of pooling, and everything else remains to be same. Then, we implement them, the results are shown in the table below.

Different Pooling	Accuracy of training set	Accuracy of test set
Max Pooling	80.02%	78.11%
Average Pooling	79.35%	77.69%
Sum Pooling	77.67%	76.35%

As shown in the table, when we use the max pooling which is what we use, we can get the highest accuracy, when we use other kinds of pooling, the accuracy will drop.

3 Results and Discussion

For the letters recognition data, we try to use a variety of methods to improve the accuracy of the classifier. At first, we tried the Decision Tree Classification, and the accuracy is not improved. Then, we tried the Maximum Likelihood

Classification, the result is also not improved. After that, we tried different topologies of the Neural Neatwork and changing the number of hidden nodes and hidden layers, then we choose the one with the best accuracy.

At last, the best result we get is 78.11% accuracy for the test set. While the accuracy from a paper which use the same dataset is 89.755% [6]. The classifier used in the paper is Gaussian classifier, which may have an influence on the result.

4 Conclusion and Future Work

In this research, we try to improve the accuracy of the classifier, we tried Decision Tree Classification, Maximum Likelihood Classification and Neural Network Classification. We implement these methods and compare the results to get the better one. But compared with other results using the same data, the accuracy is relatively low. The reasons for the less accuracy may be the followings.

At first, the number of inputs is 16, some of them are not necessary which can be removed. When there are some unnecessary inputs, the result may be affected. Secondly, some inputs can be added, like the ratio of the height and the weight. Thirdly, we can still try some other classifiers, including Euclidean distance classifier, Gaussian classifier, Knn classifier and so on.

5 Reference

1. Letter Recognition Data Set: http://archive.ics.uci.edu/ml/datasets/Letter+Recognitions
2. Classification: Basic Concepts, Decision Trees, and Model Evaluation, https://www-users.cs.umn.edu/~kumar001/dmbook/ch4.pdf
3. Maximum likelihood classification: http://eoedu.belspo.be/en/guide/maxvrai.asp?section=3.6.2.4
4. How Maximum Likelihood Classification works: http://desktop.arcgis.com/en/arcmap/10.3/tools/spatial-analyst-toolbox/how-maximum-likelihood-classification-works.htm
5. CLASSIFYING DRY SCLEROPHYLL FOREST FROM AUGMENTED SATELLITE DATA: COMPARING NEURAL NETWORK, DECISION TREE & MAXIMUM LIKELIHOOD : Bustos, R. A., & Gedeon, T. D. (1995). Decrypting Neural Network Data: A GIS Case Study. In Artificial Neural Nets and Genetic Algorithms (pp. 231-234). Springer, Vienna
6. LETTER IMAGE RECOGNITION DATA: https://notendur.hi.is/benedikt/Courses/PROJECT.pdf
7. Let's Write a Decision Tree Classifier from Scratch - Machine Learning Recipes #8: https://www.youtube.com/watch?v=LDRbO9a6XPU
8. Probability concepts explained: Maximum likelihood estimation: https://towardsdatascience.com/probability-concepts-explained-maximum-likelihood-estimation-c7b4342fdbb1
9. The principle of maximum likelihood: https://www.otexts.org/1437