COMP90051 Statistical Machine Learning
Project: Character Recognition from Historic Documents

Group members:
Chao Liang (cliang3@student.unimelb.edu.au)
Jie He (hej3@student.unimelb.edu.au)
Tingxuan Zhang (tingxuanz@student.unimelb.edu.au)

Introduction

Optical Character Recognition (OCR) is an important machine learning application. In the project, our task is to identify identities of characters from historical documents. Every character in the training set has its own identity (the label), we need to use the training set to train a classifier and then predict every character's identity in the test set. So this is a typical classification problem. In order to solve this problem, we try two different methods: Random Forest and Convolutional Neural Network. In this report, we will discuss these two methods and compare them.

For convenience and clarity, we will use the following notations in our report.

When discussing Random Forest, we use:
- RF: Random Forest
- m: The number of features to consider when looking for the best split

When discussing Convolutional Neural Network, we use:
- CNN: Convolutional Neural Network
- P: padding
- S: stride of filter

1. Random Forest

1.1. Description

RF is based on decision tree. We will first describe decision tree briefly.

A decision tree contains two different kinds of nodes: internal(non-leaf) nodes and leaf nodes. Each internal node is labeled with an input feature. Each leaf node is labeled with a class. When classifying an instance with decision tree, we test the instance with one feature. Then assign the instance to a sub node based on the result of test. Do this recursively until the instance reach a leaf node. The instance is labeled with the leaf node's class.

RF grows many decision trees. To classify a new instance, put the instance down each of the trees in the forest. Each tree gives a classification, and we say the tree "votes" for that class. The forest chooses the classification which has the most votes over all the trees in the forest.

Each tree is grown as follows:

1. If the number of cases in the training set is N, sample N cases at random with replacement, from the original data. This sample will be the training set for growing the tree.
2. If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
3. Each tree is grown to the largest extent possible. There is no pruning.

RF runs efficiently on large data bases and can handle hundreds of input features without feature deletion. In this project, the training set contains 50,000 instances and each instance has 13*33 = 429 features. Using RF, we don't need to worry about the time of training and don't need to process the features, which makes RF easy to develop. This is the motivation why we try RF first.

1.2. Training Execution

We use python with scikit-learn (a.k.a sklearn) to implement our RF.
During training, we set different number of trees and use different m values.

1.3. Analysis

When we increase number of trees in the RF, the accuracy first becomes better but then keeps at the same level. According to (Breiman, 2001), the generalization error for RF converges to a limit as the number of trees in RF becomes large. Our observation is agreed to this.

In order to get better performance, we need to try tune other options.
It is shown that error rate of RF depends on:

- correlation between any two trees in the forest. Increasing the correlation increases the forest error rate.
- strength of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.

Reducing m reduces correlation and strength. Increasing m increases both. So we try different m values. And we get better accuracy when m = sqrt(number of features).

RF gives a good performance. However, it is difficult to get even better performance because m is the only adjustable parameter that RF is sensitive and we have already tried to tune it. That is why we don't choose RF as our final approach.

2. Convolutional Neural Network

2.1 Description

As this project requires us to work with images, it is natural to try some methods that are suitable for classifying images. It turns out that CNN is a good option.

Most important components of CNN are convolutional layers and pooling layers.

The convolutional layer consists of a set of learnable filters. During the forward pass, each filter is convolved across the width and height of the input volume, computing dot product between the entries of the filter and the input and producing a 2D activation map that gives the responses of that filter at every spatial position. The network learns filters that activate when they see some specific type of feature at some spatial position in the input.

The pooling layer is used to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network.

Usually, we need to deal with a lot of parameters when classifying images. By using pooling layer and using parameter sharing in convolutional layer, CNN can reduce the amount of parameters and computation in the network. Besides, by stacking layers, the the network can first create good representations of small parts of the input, then assemble representations of larger areas from them. This can help to achieve better generalization on vision problems. These properties motivate us to try CNN.

2.2 Network architecture

We use the method of constructing a network in VGG (Simonyan, 2014) to build a network. For each convolutional layer, the filter is 3*3, S = 1, P = 1. For each pooling layer, the filter is 2*2, S = 2, P = 0. The architecture is shown in the following table.

input
Conv3-32
Conv3-32
Pooling
Conv3-64
Conv3-64
Pooling
FC-1024
FC-1024

FC-98
output

Table 1

2.3 Analysis

The performance of CNN can be improved by adding more convolutional layers but the time of training will also be increasing. So we do a trade-off and choose this network which has a pretty good performance and the training time is affordable.

For the provided training data, there are 50,000 instances and each has 33*13=429 features. We need to train a large amount of parameters to develop a classifier. This will easily lead to overfitting. CNN uses parameter sharing and pooling to control the number of parameters, which can avoid overfitting. This is one reason why CNN works for the data. Besides, by stacking different layers, CNN can first detect small edges and color patches, then compose these into smaller shapes and build more complex detectors. In the way, CNN can capture most features of images. This is also a reason why it works for the training set.

3. Features Analysis

According given features with csv files, this research used feature transformations and generated new features within different models training. The approaches include adjust the gray scale, fix word location, add features and adjust the data distribution. After implementation, using evaluations to analyze the results, such as accuracy, timeliness, and memory consumption. The following content about the analysis the features in dataset with decided-parameters model as above. And comparing the results between them.

- Delete the “error” data
  There have some errors occurred in csv file, such as rows contain all zero data. Then, deleting these rows in file.

- Modify the gray scale
  The original data used 0-255 gray scale to represent the images, this research used threshold (value=127) to transform the data into binary form. It indicates the image just has two value, the gray value greater than 127, set the value equal to 1, others situation set to 0. This approach can reduce the shadow part of the image, and make the picture clearer. What’s more, as CNN model, it is impossible to just use two value (0/1) to process. Thus, using the value divide by 255, getting the float results between 0 and 1 with CNN model.

Accuracy / Status	RF	CNN
Before	78.081%	74.682%
After	79.653%	77.110%

Table 2

The above table shows the accuracy about before and after adjustment of above two steps. It is obvious that the adjustment in features can get higher accuracy. In addition, before modifying the data, it executed with very slow speed in models. Thus, modifying the gray scale can improve timeliness and reduce the memory consumption in these models.

- Truncation letters of image in fixed-size
  There have a lot of blank spaces are meaningless, so cut out each image of letter as much as possible and adjust in fixed-size. It allows model can extract the features better. However, it did not get high accuracy in these models, and just remain highest accuracy (around 79%) in random forest model.

- Add new features
  This project added projections with width and length regarded as new features in dataset (Yampolskiy,2007). However, this approach cannot use in CNN model. The process in CNN model need an image file, which means it cannot exist any other features in it. But it did not get higher accuracy in random forest model.

- Adjust dataset distribution
  Considering the details in dataset, and draw the plot of dataset distribution as below.

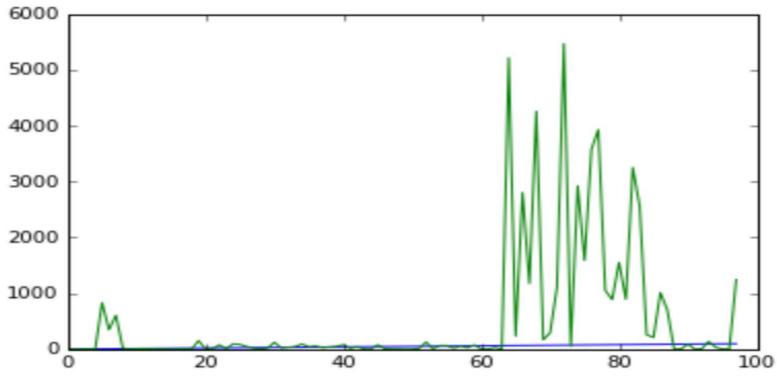

Figure 1

It is obvious that the dataset is imbalance, such as some classes have fewer instances, other hold relatively large instances. This situation can be solved by oversample method (imbalance-learn, 2016), which added some data in less part cater to the most part. Other approach used undersample, which means used most part cater to less part (both of them used imblearn package). However, theses ways would produce two "extreme" problems. As oversample method, there would produce many redundancy data that it cannot use in models' training. As undersample method would reduce more important information in data so that cannot execute models' training. On the other hand, this project used another method to adjust this situation. Deleting rows which class has less than 100 instances. After implementation, the result as below.

	RF	CNN
Accuracy	79.676%	80.393%

Table 3

4. comparison and conclusion

RF is more efficient and spends less time to train the classifier. CNN spends more time to train. CNN can achieve better accuracy compared with RF. But CNN is prone to overfitting, while RF won't worry about this problem. This can be shown by calculating the difference of training set's accuracy and testing set's accuracy.

For this project, our only goal is to achieve a better accuracy. CNN fulfills this requirement; thus we choose CNN as our final approach.

Reference:

Breiman, L. (2001) 'Random Forests', Machine Learning, 45(1), pp. 5–32. doi: 10.1023/a:1010933404324.

Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014)

Yampolskiy, R.V. (2007) *Feature extraction approaches for optical character recognition*. Rochester, NY: Briviba Scientific Press.

imbalanced-learn (2016) *Imblearn.Under_sampling.NeighbourhoodCleaningRule — imbalanced-learn 0.2.0.Dev0 documentation*. Available at: http://contrib.scikit-learn.org/imbalanced-learn/generated/imblearn.under_sampling.NeighbourhoodCleaningRule.html (Accessed: 5 October 2016).