ML Final Report

Student B99902016 秦睿謙
B99902046 林琮堯
B99902047 張鈞為

Outline

I. Data transform, feature finding

A. Transfer data back to Image:

To find useful feature of the data and to make it easier to do some image processing tactic, we transferred the data back to image.

B. Filter:

From image, we found that there are many point in the image that is isolated and pale, most of them are not things we care (that is: have nothing to do with the word.) As a result, we make an absolute value filter. We cleaned all pixels which are smaller than 100/255, because we thought them as noises.

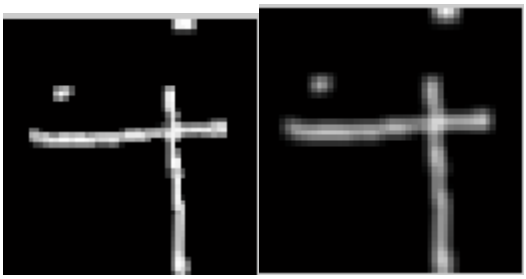
C. Normalization:

From image, we found the words are sited in different place of images, and with different size. As a result, raw data is very messy. To make them normalized, we make all words be in the same size, and site in the middle. We also transform image size from 105*122 to 200*200 for further image process.

D. Image processing:

We found some 筆劃 which should be continuous are divided into more than one piece (Due to our filter or ugly words). So we used matlab function imdilate (image dilate) to make words be more smooth and continuous.
In neural network part, we want to make words become more blur, so we also tried Gaussian mask method.

These two pictures below show the difference between the original word and the word processed by Gaussian mask.



E. Shrink image:

After the previous process, we found that the result turns to be better. However, training was very slow. To solve this, we shrank the size of the image from 200*200 to 100*100. This made training faster and remained the quality of data.

F. Output:

We output data in two ways. In first try, we used 0/1 output by applying a threshold on each pixel. If a pixel is larger than the value, output value is 1, otherwise is 0. In second try, the threshold remained, but output value is original double value.

II. Algorithm

A. SVM:

Starting from SVM, we use the package: "libsvm" to help us. We tune the parameters of svm by 5-fold cross validation. The linear version of svm was replaced by kernel of degree one, since we thought they are very similar. The result shows in following tables. The column is the value of cost of each error in soft margin svm, the row is the value of gamma in kernel, and the value is accuracy in percentage. This validation was done before the second half of data was release.

For d = 1:

g c	0.05	0.075	0.1	0.125	0.15	0.175	0.2
0.0025	58.95	61.37	62.79	63.65	64.58	65.10	65.64
0.005	62.79	64.58	65.64	66.09	66.37	66.68	66.92
0.0075	64.58	65.88	66.39	66.76	67.02	67.15	67.20
0.01	65.64	66.37	66.92	67.18	67.20	67.10	66.89
0.0125	66.09	66.78	67.18	67.17	67.00	66.76	66.40
0.015	66.39	67.02	67.20	67.04	66.74	66.37	65.83

For d = 2:

g c	0.05	0.075	0.1	0.125	0.15	0.175	0.2
0.0025	46.46	54.05	57.12	59.04	59.96	60.82	61.91
0.005	61.91	64.02	65.47	66.34	67.08	67.51	67.78

0.0075	65.99	67.41	68.09	68.47	68.55	68.52	68.55
0.01	67.78	68.53	68.58	68.78	68.81	68.83	68.88
0.0125	68.57	68.65	68.73	68.91	68.86	68.81	68.79
0.015	68.55	68.81	68.84	68.86	68.76	68.73	68.76

I used a shell script to run all this training. I also gave d=3 a try but it showed that it is much worse than d=1 or 2 because of overfitting. Value "g" can be considered as how small we want the margin is (for a gaussian kernel $g \propto \sigma$.)

Both table for d=1, 2 shows that the margin should be quite large and the cost should be small enough but not too small. This is because even though we has normalize the data and eliminated isolated point the data still has large noise (quarter of the image was cut off.) This tells that we need more tolerance on error, but also give enough penalties for each error.

B. Random Forest

For Random Forest algorithm, I use python package "scikit-learn". The package implements not only random forest but also lots of other algorithms. The random forest has two parameters to set. First is number of trees in the forest, second is the max depth of a tree to develop. Because random forest is an aggregate algorithm, and all its basic algorithms are simple, after a constant number of trees, the performance of random forest will only vary slightly. Because of the high cost of generating trees, I first try to find the constant number of trees that produce stable performance. We ran some tests and found that 2500 trees are enough. And I do validation on the parameter of how deep a tree should be. Following are statistics I record from validation (this statistics was recorded on validation on second phase of the competition):

Depth	20	35	50	75	100
oob score	0.0306	0.0267	0.0278	0.0279	0.0280

The score is not equals to the accuracy or error rate, it may relate to something similar to error rate but not equal. After this validation I use 5000 trees (although 2500 trees is stable enough, 5000 trees was a little bit better) and depth 35 as our choice. The result pushed to the judge server has about 74% accuracy.

A more random algorithm:

The random forest algorithm above does many random choose when building trees, like choosing the best random selected split data set to split a node in a tree, bootstrapping. But another implementation about random forest in this package called "ExtraTrees" does even more randomness. As in

random forests, a random subset of candidate features is used, but instead of looking for the best thresholds with purest data sets, thresholds are drawn at random for each candidate feature and choose the best of these thresholds to split a node.

This algorithm did a little better than original one with 76.04% accuracy. The reason of the better performance is that the more difference between the trees the lower the variance of the random forest (since we generate enough trees.)

C. Convolutional Neural Network

After the data is processed there are a few ways to approach Hand writing recognition. We did a few researches on neural network, and picked a few model for comparison, such as Multidimensional Recursive Neural Network, Deep Belief Net, Deep Boltzmann Machine, and Convolutional Neural Network. After a series of comparison and evaluation, we chose convolutional neural network since it has great performance on 2D images, yet others are not so good or has excessive power.

Our program is based on Theano, a deep learning toolkit for python, and is written in Python. After pre-processing, the image is cropped and minimized to 50*50, and we sent them into the first layer. The first layer is a convolutional layer with 100 maps and 3*3 filter, after that is a layer which does max pooling of 2*2. Following by 150 maps 3*3 filter and 2*2 max-pooling, 200 maps 3*3 filter, 2*2 max-pooling, 500 fully connected neurons and 1000 fully connected neurons and lastly a logistic regression neuron. That will sum up to be:

Input->100C3MP2, 150C3MP2, 200C3MP2, 500FN, 1000FN, LogReg->output

Another algorithm we implement is Deep Belief Net. We build a 1500, 1000, 500, 300, 200, 5 layered DBN 400 epochs, and pretrained each layer with 100 epochs. Still the result was not as good as expected, they were around 80% on the first phase testing. We failed fast and dumped this method.

For convolutional network, we've tested a few models with different parameters:

50C3MP2, 50C3MP2, 50C3MP2, 500FN, LogReg	87.26%
50C3MP2, 100C3MP2, 150C3MP2, 500FN, LogReg	89.44%
100C3MP2, 150C3MP2, 200C3MP2, 500FN, 1000FN, LogReg	91.26%

This model has scored 92.15% at the second phase test.

D. RBF Network

This algorithm was implemented by ourselves with help of Lloyd's algorithm

tools implement by python package “scikit-learn”. The make this algorithm work with multi-classes classification, we used one versus one technique. Although the data set was smaller than original after classified them by 12 classes, the algorithm still very slow. As a result, I used the technique of feature selection.

The feature selection is done by using the random forest in previous try which has feature selection as its function provided. The original data each has 50*50=2500 feature. After feature selection we got about 1400 feature only. This reduction of feature had huge impact on the time cost of training and predicting. Here I had a short talk with TA Ian Lin. He said that after feature selection the performance usually will be better than previous data set because the most important feature stay and less important features are removed. But the result I got was not in such case. I guessed the reason is the dispersed feature. For example “龍” may cover many point in image that other words do. Feature selection may remove some feature like this, which cause the lower performance.

Although feature selection result in lower performance, it is a good way to find a good parameters set. I then used the parameters set to train on original data. The accuracy of the RBF network is 72.26.

E. Blending

After the above work we make a blending on previous output. The first try we blended the result of SVM, random forest, and convolutional neural network with their best parameters. The blending used the uploaded result (the accuracy) as weight of each algorithm, which are SVM=79.10%, Random Forest=74.41% and Convolutional Neural Network=92.15%. The result turns to be 86.55% which is lower than Deep Neural Network.

In the next try, we used SVM, random forest (extra trees version), convolutional neural network, RBF network to blend. The accuracy is 85.54%.

Both of the blending reduced the accuracy. Blending usually help to improve a little on accuracy, but in this case the accuracy reduce. The reason I though is that I have too small number of model. There are 12 classes and I have only 4 models. And by the way, the noise is quite large. Maybe Convolutional neural network predicts correctly on a word but other three predict wrong. This may cause the reduction of accuracy. Maybe it will be a good try to make more models with different parameters set of each algorithm and then blending it again.

III. Contribution

A. B99902016: SVM, Random Forest, RBF Network, Blending
B. B99902046: Paper reading, Convolutional Neural Network (Best)
C. B99902047: Data preprocessing

IV.Reference

A. Alex Graves and J. Schmidhuber Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks. *In Advances in Neural Information Processing Systems*. 2009.
B. Dan Ciresan and J. Schmidhuber Multi-Column Deep Neural Networks for Offline Handwritten Chinese Character Classification. *IDSIA / USI-SUPSI Technical Report No. IDSIA-05-13. 2013*