Machine Learning
Final Project

Team: hahaha
R01942054 林家蓉 R01942068 賴威昇

February 22, 2014

1 Introduction

In this project, we are asked to solve a classification problem of Chinese characters. The training set contains 6144 examples. Two testing sets contain 3072 examples respectively. Since the original feature has 12810 dimensions, which is composed of image pixel intensity, directly using machine learning techniques is inefficient and may not produce good results. For example, training by LIBSVM[1] with default parameters gets 0.152018 accuracy on test1 dataset, and training by LIBLINEAR[2] gets 0.324544 accuracy on test1 dataset. As a result, we decide to perform pre-processing to the raw input data and extract statistical features before applying machine learning tools. Our method reaches 0.940755 accuracy on test2 dataset in the phase-2 competition. We will discuss the pre-processing steps, the extracted features, and the machine learning techniques we use in this report.

2 Data Pre-processing

In order to apply image processing techniques, we reconstruct character images from the raw data first, and store images in `bmp` format to avoid compression artifacts.

2.1 Thresholding

To turn grey image into binary image, we threshold the input image at pixel value 20.

2.2 Normalization

We adopt bi-moment normalization[3] method to reduce the shape variation between images of the same class. We denote the input image and the normalized image by $f(x, y)$ and $g(x', y')$, respectively. Normalization is implemented by coordinate mapping.

$$\begin{cases} x' = x'(x, y), \\ y' = y'(x, y). \end{cases} \tag{1}$$

We compute the centroid of the input image (x_c, y_c). Then, we can compute one-sided second-order moments:

$$\begin{cases} \mu_x^+ = \sum_{x>x_c} (x - x_c)^2 f_x(x), \\ \mu_x^- = \sum_{x<x_c} (x - x_c)^2 f_x(x), \\ \mu_y^+ = \sum_{y>y_c} (y - y_c)^2 f_y(y), \\ \mu_y^- = \sum_{y<y_c} (y - y_c)^2 f_y(y). \end{cases} \tag{2}$$

The bounds of input image are then reset to $[x_c - \delta_x^-, x_c + \delta_x^+]$ and $[y_c - \delta_y^-, y_c + \delta_y^+]$, where$\delta_x^+ = \beta\sqrt{\mu_x^+}$, $\delta_x^- = \beta\sqrt{\mu_x^-}$, $\delta_y^+ = \beta\sqrt{\mu_y^+}$, and $\delta_y^- = \beta\sqrt{\mu_y^-}$.

Then, we can map the original image to new image using parameters above accordingly:

$$\begin{cases} x' = (x - x_c)\frac{W_2}{\delta_x} + x'_c, \\ y' = (y - y_c)\frac{H_2}{\delta_y} + y'_c. \end{cases} \tag{3}$$

where W_2 and H_2 indicate the width and height of the normalized image respectively.
In our experiment, we set x'_c to $\frac{W_2}{2}$ and y'_c to $\frac{H_2}{2}$. In addition, we set $W_2 = H_2 = 60$. The effect of normalization is demonstrated in Fig. 3 and Fig. 4. Observing the figures, we can see that blank(black) area and noise(isolated white dot) are cleared out.

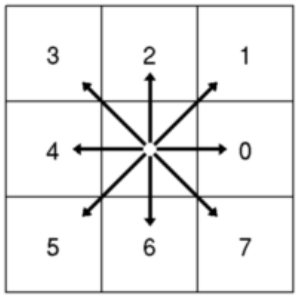

Figure 1: Eight directions of chaincode.

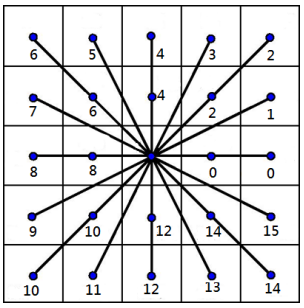

Figure 2: Sixteen directions of chaincode.

Figure 3: No.1917 image in test1 dataset

Figure 4: No.1917 image in test1 dataset after normalized to 60×60

3 Feature Extraction

We find the features best suited for this task are direction histogram features, because they are relatively immune to the missing patch of the character. Therefore, we choose Normalization Cooperated Chaincode Feature (NCCF) as our main feature.

3.1 Normalization Cooperated Chaincode Feature (NCCF)

The feature extraction is accomlished in two steps: directional decomposition, and feature sampling.

3.1.1 Directional Decomposition

Directional decomposition results in a number of direction planes (the smae size as the normalized image), $f_i(x, y), i = 1, ..., N_d$. We first decompose the skeleton of the character into eight directions, then extend to 12 and 16 directions.

At a white pixel (x, y) denoting the values of 8-connected neighbors counterclockwise as p_k, $k = 0, 1, ..., 7$, with the east neighbor being P_0. For $k = 0, 2, 4, 6$, if $p_k = 0$, check p_{k+1}: if $p_{k+1} = 1$ (chaincode $k + 1$), $f_{k+1}(x, y)$ increases by 1; otherwise, if $p_{(k+2)\%8} = 1$ (chaincode $(k + 2)\%8$), $f_{(k+2)\%8}(x, y)$ increases by 1. Fig. 5 shows the 8 direction planes of normalized No.1917 image in test1 dataset.

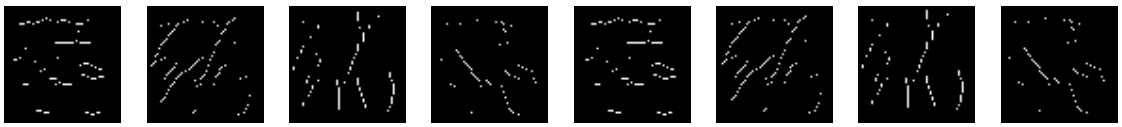

Figure 5: Direction planes of No.1917 in test1 dataset (8 directions)

3.1.2 Extension to More Directions

In addition to 8 directions, we also experiment with 12-derection and 16-direction features. The extension of skeleton decomposition to more directions is straight forward. We set N_d standard directions with angle interval $360/N_d$. Using a 5×5 kernel like Fig. 2, we can decompose the skeleton of the character into 16 direction planes using algorithm similar to the one we use to extract 8-direction feature. A 12-dimension feature can be extracted likewise.

3.1.3 Image Sampling

Conventionaly, the direction plane is partitioned into a number of block zones (as shown in Fig. 6) and the averaged value of each zone is viewed as a feature value. Previous work[4] suggests that using overlapped blocks (as shown in Fig. 7) instead of basic blocks partition can further improves accuracy. The reason is that overlapped blocks alleviate the effect of stroke-position variation at the boundary. In our experiments, we find

that using overlapped block zones achieves higher accuracy than basic block zones. We compare results of 4×4 basic block zones with 3×3, 4×4, 5×5 overlapped block zones.

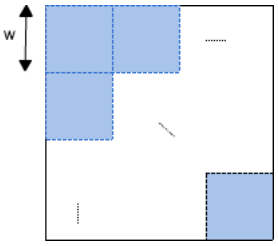

Figure 6: basic blocks

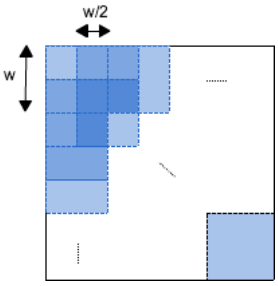

Figure 7: overlapped blocks

4 Learning Algorithm

We use scikit-learn[5], a python package that supports lots of machine learning and data mining algorithm, as our main tool. We choose 4 algorithms to compare: SVM with RBF kernel, SVM with linear kernel, random forest, and logistic regression. For each algorithm, we choose parameters by grid search and cross-validation. During phase-1 competition, we use 5-fold cross-validation on the training dataset to select parameters, and then we train the whole training dateset with the best parameter to predict test1 dataset. During phase-2 competition and experiments in this report, we merge the training dataset and test1 dataset together for training.

4.1 Models

4.1.1 RBF SVM

The implementation of `SVC` in scikit-learn is based on LIBSVM[1]. We choose the RBF kernel, and perform parameters grid search with $C \in \{2^0, 2^1, \cdots, 2^6\}$, $\gamma \in \{2^{-4}, 2^{-2}, \cdots, 2^4\}$.

4.1.2 Linear SVM

`LinearSVC` in scikit-learn is implemented in terms of LIBLINEAR[2], which has more flexibility to select the penalty and loss function and is more efficient than `SVC` with linear kernel. We perform grid search with $C \in \{2^0, 2^1, \cdots, 2^6\}$, penalty function $\in \{l1, l2\}$, and loss function $\in \{l1, l2\}$. However, the case that penalty function $= l1$ and loss function $= l1$ is not supported in scikit-learn.

4.1.3 Random Forest

Random Forest[6] is a meta algorithm that fits a number of decision tree on various sub-samples of the dataset. With averaging, it can improve the predictive accuracy and also control over-fitting. We search the number of estimators(trees) from $\{50, 100. \cdots, 300\}$. Although more estimators may lead to better results, we find that the performance does not improve much when using more than 300 esitmators.

4.1.4 Logistic Regression

We search for penalty $\in \{l1, l2\}$ and $C \in \{2^{-2}, 2^{-1}, \cdots, 2^4\}$, where C is the inverse of regularization strength.

4.2 Feature Normalization

In most situations, feature normaliation is an important step that improves the numerical stability of the training algorithm solvers. However, since NCCF is a kind of normalized histogram, the range of feature value is already in $[0, 1]$. As a result, we do not have to normalize feature values in our experiment.

4.3 Feature Selection

Feature selection can help us find the most important feature and remove some less useful features to further enhance the performance. We choose F-score[7] as our selection criterion, which can be calculated simply and efficiently. We use the python code provided by LIBSVM[1]. After calculating the F-score for each dimension,

[1] `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/#feature_selection_tool`

we sort features in the training file and testing file. Then, we can select first-N features and decide how many features we should use with cross-validation.

5 Experiment and Discussion

5.1 Model and Feature

We first extract NCCF feature with 3 different set of direction planes (8, 12, 16) and 3 different overlapped block zones(overlap-3 means 3×3, overlap-4 means 4×4, and overlap-5 means 5×5 block zones). The following tables show the cross-validation accuracy and test2 dataset accuracy with 4 different learning models.

Feature	Dimension	CV-accuracy	test2-accuracy	parameter
r60-bimoment-NCCF-8	128	0.910590	0.915365	C = 4, gamma = 16.0000
r60-bimoment-NCCF-12	192	0.904947	0.914388	C = 4, gamma = 16.0000
r60-bimoment-NCCF-16	256	0.910698	0.920573	C = 4, gamma = 16.0000
r60-bimoment-NCCF-8-overlap-3	200	0.917100	0.924154	C = 4, gamma = 4.0000
r60-bimoment-NCCF-8-overlap-4	392	0.923937	0.930339	C = 4, gamma = 1.0000
r60-bimoment-NCCF-8-overlap-5	648	**0.928927**	**0.935221**	C = 2, gamma = 0.2500
r60-bimoment-NCCF-12-overlap-3	300	0.914604	0.913737	C = 4, gamma = 1.0000
r60-bimoment-NCCF-12-overlap-4	588	0.920247	0.928711	C = 4, gamma = 0.2500
r60-bimoment-NCCF-12-overlap-5	972	0.916992	0.925781	C = 2, gamma = 0.2500
r60-bimoment-NCCF-16-overlap-3	400	0.917209	0.915365	C = 8, gamma = 1.0000
r60-bimoment-NCCF-16-overlap-4	784	0.925673	0.929036	C = 8, gamma = 0.2500
r60-bimoment-NCCF-16-overlap-5	1296	0.918186	0.926107	C = 4, gamma = 0.0625

Table 1: Experiment results of RBF SVM

Feature	Dimension	CV-accuracy	test2-accuracy	parameter
r60-bimoment-NCCF-8	128	0.848633	0.855469	C = 32, penalty = l1, loss = l2
r60-bimoment-NCCF-12	192	0.847873	0.847005	C = 32, penalty = l2, loss = l2
r60-bimoment-NCCF-16	256	0.857639	0.863281	C = 16, penalty = l2, loss = l2
r60-bimoment-NCCF-8-overlap-3	200	0.862847	0.863281	C = 8, penalty = l2, loss = l2
r60-bimoment-NCCF-8-overlap-4	392	0.869900	0.866211	C = 1, penalty = l2, loss = l1
r60-bimoment-NCCF-8-overlap-5	648	0.874457	0.872070	C = 1, penalty = l2, loss = l2
r60-bimoment-NCCF-12-overlap-3	300	0.866428	0.868490	C = 4, penalty = l2, loss = l2
r60-bimoment-NCCF-12-overlap-4	588	0.873155	0.880534	C = 2, penalty = l2, loss = l1
r60-bimoment-NCCF-12-overlap-5	972	0.870334	0.870768	C = 1, penalty = l2, loss = l1
r60-bimoment-NCCF-16-overlap-3	400	0.874457	0.877604	C = 4, penalty = l2, loss = l2
r60-bimoment-NCCF-16-overlap-4	784	**0.881185**	**0.885417**	C = 1, penalty = l2, loss = l1
r60-bimoment-NCCF-16-overlap-5	1296	0.874566	0.874349	C = 1, penalty = l1, loss = l2

Table 2: Experiment results of Linear SVM

Feature	Dimension	CV-accuracy	test2-accuracy	parameter
r60-bimoment-NCCF-8	128	0.890951	0.895508	n-estimators = 250
r60-bimoment-NCCF-12	192	0.883247	0.894206	n-estimators = 300
r60-bimoment-NCCF-16	256	0.879558	0.893229	n-estimators = 300
r60-bimoment-NCCF-8-overlap-3	200	0.891709	0.892904	n-estimators = 250
r60-bimoment-NCCF-8-overlap-4	392	0.902235	0.904948	n-estimators = 300
r60-bimoment-NCCF-8-overlap-5	648	**0.904731**	**0.914714**	n-estimators = 300
r60-bimoment-NCCF-12-overlap-3	300	0.871745	0.880534	n-estimators = 300
r60-bimoment-NCCF-12-overlap-4	588	0.895291	0.904297	n-estimators = 250
r60-bimoment-NCCF-12-overlap-5	972	0.891601	0.902669	n-estimators = 300
r60-bimoment-NCCF-16-overlap-3	400	0.874674	0.875977	n-estimators = 250
r60-bimoment-NCCF-16-overlap-4	784	0.895941	0.908203	n-estimators = 250
r60-bimoment-NCCF-16-overlap-5	1296	0.892687	0.899740	n-estimators = 300

Table 3: Experiment results of Random Forest

Feature	Dimension	CV-accuracy	test2-accuracy	parameter
r60-bimoment-NCCF-8	128	0.846029	0.859701	C = 16, penalty = l1
r60-bimoment-NCCF-12	192	0.848523	0.850911	C = 16, penalty = l1
r60-bimoment-NCCF-16	256	0.857747	0.867513	C = 16, penalty = l1
r60-bimoment-NCCF-8-overlap-3	200	0.862847	0.863607	C = 8, penalty = l1
r60-bimoment-NCCF-8-overlap-4	392	0.868707	0.871745	C = 8, penalty = l2
r60-bimoment-NCCF-8-overlap-5	648	0.875759	0.874349	C = 4, penalty = l2
r60-bimoment-NCCF-12-overlap-3	300	0.865560	0.864583	C = 16, penalty = l2
r60-bimoment-NCCF-12-overlap-4	588	0.875759	0.880534	C = 4, penalty = l2
r60-bimoment-NCCF-12-overlap-5	972	0.877171	0.876302	C = 1, penalty = l2
r60-bimoment-NCCF-16-overlap-3	400	0.874566	0.872396	C = 8, penalty = l1
r60-bimoment-NCCF-16-overlap-4	784	0.882053	**0.887370**	C = 8, penalty = l2
r60-bimoment-NCCF-16-overlap-5	1296	**0.885417**	0.882487	C = 2, penalty = l2

Table 4: Experiment results of Logistic Regression

From Table 1 to Table 4, we can see that SVM with RBF kernel not only achieves the best accuracy on test2 dataset (0.935221) but also has over 0.90 average accuracy. 8-direction NCCF with 5×5 overlap block zones and 16-direction NCCF with 4×4 overlap block zones have better performance than other features.

5.2 Feature Selection

Next, we merge some features into a high dimension training dataset and then apply feature selection technique on it. This process picks out dimensions that are more useful. We merge r60-bimoment-NCCF-8-overlap-3, r60-bimoment-NCCF-8-overlap-4, and r60-bimoment-NCCF-8-overlap-5 together into a 1240-dimension data, which is named r60-bimoment-NCCF-8-overlap-345. We also merge r60-bimoment-NCCF-8-overlap-3, r60-bimoment-NCCF-12-overlap-3, and r60-bimoment-NCCF-16-overlap-3 into a 900-dimension data, which is named r60-bimoment-NCCF-all-overlap-3. Noted that we already sort the dimension of these two new training data by F-score.

Dimension	CV-accuracy	test2-accuracy
100	0.840603	0.853841
500	0.916124	0.927409
1000	0.930338	0.939128
1240	0.931966	**0.940430**

Table 5: Accuracy of each selected dimension for r60-bimoment-NCCF-8-overlap-345

Dimension	CV-accuracy	test2-accuracy
100	0.838107	0.843099
300	0.905490	0.912109
500	0.916123	0.921875
700	0.922092	0.928385
751	0.921983	0.927734

Table 6: Accuracy of each selected dimension for r60-bimoment-NCCF-all-overlap-3

Experiment results show that the performance grows with the number of selected dimension. The first 700 dimensions in r60-bimoment-NCCF-all-overlap-3 reach only 0.000651 higher accaracy than using all dimensions, which means the former only classifies two more characters correctly than the latter. Therefore, we can conclude that these features have similar importance to the classification result.

5.3 Probabilistic Feature

Some models can predict the probability of each class for input data. We wonder whether this probability can be considered a useful feature. We choose r60-bimoment-NCCF-8-overlap-5 as basic feature and predict its probability using RBF SVM and Random Forest(RF).

Model to predict probability	Model to train	CV-accuracy	test2-accuracy
RBF SVM	RBF SVM	0.968964	0.925130
RBF SVM	linear SVM	0.968964	0.925130
RBF SVM	Random Forest	0.968422	0.922526
RBF SVM	Logistic Regression	0.969832	0.924154
Random Forest	RBF SVM	0.995551	0.909831
Random Forest	linear SVM	0.995551	0.916992
Random Forest	Random Forest	0.995334	0.805013
Random Forest	Logistic Regression	0.995551	0.915039

Table 7: Experiment results of the probabilistic feature

We can see that these 12-dimension probabilistic features all produce over 0.96 cross-validation accuracy. However, they do not improve test2 dataset accuracy. It is because the features overfit the training dataset.

6 Conclusion

From our experiment, we conclude that using r60-bimoment-NCCF-8-overlap-345 feature and training by RBF-SVM is the best method to predict chinese characters in this problem. We summarize our method in following steps:

1. Normalize input images to 60×60 using bi-moment normalization.
2. Extract NCCF feature with 8 direction planes, 3×3, 4×4, and 5×5 overlapped block zones respectively.
3. Merge features
4. Train a classifier with RBF SVM

We recommend SVM not only because it is the most popular machine learning tool in recent years, but also because the training process can be conducted efficiently using LIBSVM. Although RBF SVM works well in our case (using feature with around 1000 dimensions), we should keep in mind that it has some disadvantages. First, it becomes less efficient when working on large-scale problems. For example, problems with over 1 million data or features over 10000 dimensions (such as the raw input data in this project). In addition, the result of RBF SVM is not interpretable like linear model or random forest model. However, RBF SVM does work well in most problems.

Work Allocation

林家蓉 : Paper survey, image pre-processing, feature extraction, experiment, report writing
賴威昇 : Learning model selection, feature selection, probabilistic feature, experiment, report writing

References

[1] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[2] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[3] C.-L. Liu, H. Sako, and H. Fujisawa, "Handwritten chinese character recognition: alternatives to nonlinear normalization," in *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, 2003, pp. 524–528 vol.1.

[4] J. Guo, N. Sun, Y. Nemoto, M. Kimura, H. Echigo, and R. Sato, "Recognition of handwritten characters using pattern transformation method with cosine function," *Trans. IEICE Japan*, vol. 76, no. 4, pp. 835–842, 1993.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[6] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[7] Y.-W. Chen and C.-J. Lin, "Combining svms with various feature selection strategies," in *Feature Extraction*. Springer, 2006, pp. 315–324.