2016 49th Hawaii International Conference on System Sciences

Online Signature Verification Based on Biometric Features

Nan Li*, Jiafen Liu*, Qing Li*, Xubin Luo* and Jiang Duan*
*School of Economic Information Engineering
Southwestern University of Finance and Economics, Chengdu, P. R. China
auroralinan2@gmail.com, jfliu@swufe.edu.cn,
liq_t@swufe.edu.cn, xubinluo@gmail.com, duanj_t@swufe.edu.cn

***Abstract*—Since current signatures are generally not verified carefully, frauds by forging others signature always happen. This paper tried to authenticate user automatically with electronic signatures on mobile device. We collected coordinates, pressure, contact area and other biometric data when users sign their name on touch screen smart phone. Then we used four different classification algorithms, Support Vector Machine, Logistic Regression, AdaBoost and Random Forest to build a specific signature verification model for each user, and compared the verification accuracy of these algorithms. The experimental result on 42 persons' dataset shows that these four algorithms have satisfactory performance on Chinese signature verification, and Adaboost has the best performance with error rate of 2.375%.**

***Keywords*-Electronic Signature; Signature Verification; Adaboost; Random Forest;**

I. Introduction

With the continuous advance of the paperless office, electronic signatures are gradually replacing handwritten signatures in various fields. The most beneficial reason for electronic signatures taking place of handwritten signatures is resource saving. To get electronic signatures, users are required to sign his/her name on the touch screen of a digital panel, a tablet or a smart phone rather than on paper. For those companies that have a large requirement for receipts or contracts, electronic signatures can help them reduce considerable expenditure. Nowadays, many communications, retailing, hotels and many other service-oriented industries are beginning to use electronic signature to authenticate user or produce non-repudiation evidence. Especially in financial industries such as banks and insurance which are in greater demand for signatures, electronic signatures are widely adopted in China. So if the reliability of electronic signatures can be guaranteed, more companies and industries are willing to use electronic signatures instead of handwritten ones.

In terms of security, electronic signatures have better reliability than handwritten signatures. Generally handwritten signatures are highly vulnerable to imitate. People have to authenticate handwritten signatures by hand, and it is neither reliable nor efficient because of judges background knowledge and limited experience. Even if we can turn the handwritten signature to an image and authenticate it automatically, it is unreliable. Because offline-handwritten signatures contain no information of writing process, we can only authenticate it by computing the similarity of shape and structure. As for electronic signatures, we could collect more information with touchscreen besides shape and structure of characters, such as writing speed, pressure, size of contact area and other biometric features. These biometric features can be used to reproduce the writing process and often reflect users writing habits. Since each person has unique writing habit and it is hard to imitate, signature verification based on biometric features is feasible.

This paper explored the feasibility of automatic signature identification on a smart device to tell whether the signer is the real user. The rest of this paper is organized as follows. We first briefly describe the related work in Section 2. The design details for our signature verification experiment are presented in Section 3. We then compared performance of 4 different classification methods in signatures verification using our experimental data in Section 4. This paper is concluded with speculation on how the current prototype can be further improved in Section 5.

II. Background and related work

In general, signature verification methods are mainly divided into 2 categories: offline verification and online verification. Offline signature verification mainly uses static graphic information to authenticate signer. While online signature verification can make use of more dynamic information, such as velocity, acceleration and pressure of writing a signature, which is more difficult to imitate. Hence online signature verification generally has higher accuracy rate than offline verification[1].

In online verification, the first thing is to extraction features from signature data, which consist of global features and local ones[2]. Global features describe characteristics of the entire signature, such as total time of writing, number of strokes, and size of signature and so on. Local features show the signature trait at a time point or during a short period, such as the local velocity and angle of a stroke. Besides direct feature extraction, hidden Markov model[3-5] and wavelet analysis[6-8] are also widely used.

Corresponding author: Jiafen Liu.

1530-1605/16 $31.00 © 2016 IEEE
DOI 10.1109/HICSS.2016.683

After feature extraction, researchers tried several different methods to verify electronic signatures. One is matching algorithm[9,10]. It compared the similarity between a test signature and the template. Classification is also widely used to verify signatures. For example, SVM (Support Vector Machine)[4,8,11,12], which is considered as one of the best classification algorithm, performs well in small samples and get a low error rate of about 3%. Artificial neural networks[6,13,14], decision tree[15] and other methods are also be adopted in signature verification.

Because there is no common signature database for test, different researchers collected their experimental data individually to test their verification schemes, and it is hard to compare their results. The First International Signature Verification Competition (SVC2004) [16] was held in 2004, and 13 teams participated in. In the competition, the champion came up with a scheme with error rate close to 2.8%. Kumiko Yasuda [17] recorded video through the webcams and tracked the pen tip moves to extract signature data, and that provided a new way for online signature verification. S. Rashidi[18] adopted DCT (Discrete Cosine Transform) as feature extraction method, and reduced EER of two tasks to 3.61% and 2.04% respectively on the signature dataset of SVC2004. Yasmine Guerbai [19] used one-class classifier for offline signature verification, and used soft threshold to improve the classification accuracy, but the final result is not really satisfactory because of the limitation of one-class classifier.

III. Experimental design and settings

We invited 42 participants to our experiment to collect signature characteristic data. Each participant is required to sign his or her own name 50 times firstly. To imitate imposters, we got a negative sample set of 20 for each participant. Since there might be more than one imposter in reality, we assigned forgery task of one person to 4 different participants and asked them to imitate the signature 5 times each. Thats to say, all 50 positive samples were generated by the names real owner, and 20 negative samples were generated by other 4 participants. We developed an App to collect data such as signatures coordinate, pressure, contact area, and etc. After extracting features from data, we built a classifier based on SVM, Logistic Regression, Adaboost and Random Forest. Then we compared the results and got conclusion.

The experiment consists of three parts. The first part is collecting the signature data, the second part is extracting features, and the third part is building a verification model. The framework of our experiment is sketched in Figure 1.

A. Data collection

Experimental data was collected using a smartphone LG-G2 based on Android 4.4, we collected a total of 42 participants data. Firstly the experimenter needed to enter his or her name, and then used a finger to sign his name 50 times on the touch screen. After that, the experimenter was asked to imitate 20 signatures as negative samples. Our APP will assign 4 other participants names to him/her randomly, and each name should be written for 5 times.

When an experimenter wrote his/her signature, our App will record the user's name, current number of writing times, current time, coordinates x, y of current position, pressure, contact area of the screen and the status of current position. When the app detected finger movement, it will create a point, thus each signature can be regarded as a series of points and it is usually between 80 and 300 points depending on the importer. Each point will be a set consists of the above 8 data items. A sample signature is shown in Figure 2.

Signature data sample is shown in Figure 3.

Features listed in Figure 3 are explained in detail as follows:

Users name (name) is entered by the participant before he or she writes name;

Current writing number (number of signature) is marked to distinguish different signatures, and its value is between 1-50 in our experiment;

Current time (time) was collected by Android systems touchEvent API MotionEvent.getEventTime();

Current Location coordinate x, y (coordinate X, coordinate Y) were gotten by Android systems API MotionEvent.getX() and MotionEvent.getY(). These API regards the left bottom of the screen as zero point; Pressure (pressure) was attained by API MotionEvent.getPressure();

The contact area of the screen (size) was collected by API MotionEvent.getSize();

The current state (status) was gotten by calling API MotionEvent.getAction().This API returns DOWN, UP and MOVE three states to denote the start, midway and the end of a stroke.

B. Feature extraction

Because each participant differs greatly in length of name and writing style, number of points of signatures varies. But classification algorithms we adopted requires equal feature numbers, so we have to extract features from raw data to get feature vector.

We use Python as programming language and extract the following 57 features listed in Figure 4 to compose feature vector.

In these features, the velocity was extracted according to the distance change per unit time between two consecutive points, and the acceleration was extracted according to the velocity change per unit time between two consecutive points. Velocity and acceleration both contain three directions, x-axis, y-axis, and in the plane.

As for features in Average Type, we extracted average velocity of the signature (x-axis, y-axis and overall), average acceleration, average pressure and average contact area.

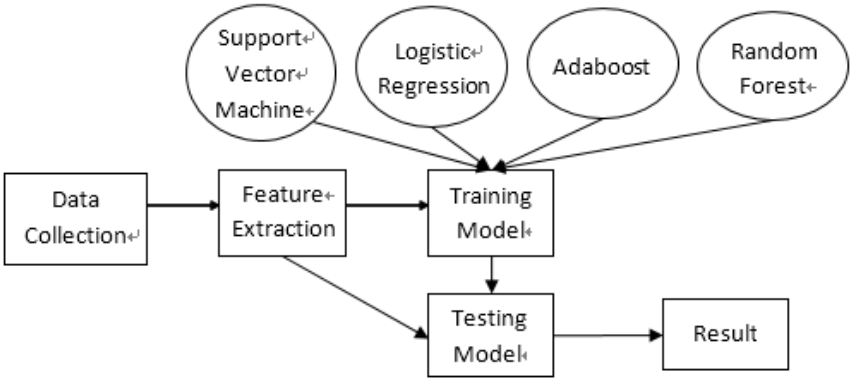

Figure 1. Flow Diagram of Experiment

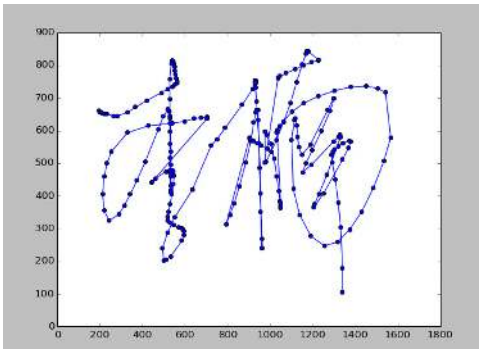

Figure 2. One of the signatures

For features in Maximum Type, we extracted maximum speed, maximum acceleration, maximum pressure and maximum contact area of the signature. Besides, we extracted the moment when these maximum values appear in the signature.

Features in Minimum Type are extracted with the same way as Maximum Type.

In Other Type features, Total Time represents the total time of writing a signature, Break Times means the amount of strokes in one signature, Total Points represents the total number of points of a signature, Maximum Coordinate, Minimum Coordinate and Average Coordinate represent the largest, minimum and the center of x, y coordinates respectively. Maximum Coordinate Position, and Minimum Coordinate Position represent the moment of the maximum and minimum coordinate appears in the signature. Size of signature is on behalf of the total area of the signature using size of the minimum rectangle to contain the signature, Total Distance of Move is on behalf of the total moving distance on the screen.

After feature extraction, the raw data was converted to a 57-dimensional feature vector. So far, we have 50 positive samples feature vectors and 20 negative samples feature vectors for each of 42 participants.

C. Verification model

We mainly used four classification algorithm to build the verification model, including SVM, Logistic Regression, AdaBoost and Random Forest.

SVM (Support Vector Machine) is a binary-class classification model. Its basic model is a maximum interval linear classifier in the feature space. For linear separable data set, SVM learns to be a linear classifier through interval maximization. For non-linear separable data set, SVM makes the data set linearly separable in the high-dimensional space by using kernel trick. Assuming that a given training data set in the feature space is:

$$T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$$

There is a hyperplane $\mathrm{f}(\mathrm{x}) = \omega^{\mathrm{T}} \cdot \mathrm{x} + \mathrm{b}$, the objective function of maximum interval classifier can be defined as

$$max \frac{1}{\|\omega\|} \quad s.t. \quad y_i \left(\omega^T \cdot x_i + b\right) \geq 1, \quad i = 1, \cdots, n$$

Logistic Regression is a log-linear model. Binomial logistic regression model is a classification model, which is represents by conditional probability distribution $\mathrm{P}(\mathrm{Y}|\mathrm{X})$, in the form of parameterized logistic distribution. The conditional probability distribution is shown below:

$$P(Y = 1|x) = \frac{exp(\omega \cdot x + b)}{1 + exp(\omega \cdot x + b)}$$

$$P(Y = 1|x) = \frac{1}{1 + exp(\omega \cdot x + b)}$$

AdaBoost is one of the most representative boosting method. For a classification problem, building a weak classifier with rough rule is easier than building a strong classifier. Boosting method obtains a strong classifier by combining several weak classifiers after learning repeatedly. AdaBoost improves the weight of which is misclassified in the previous

_id	name	numOfSign	time	coordinateX	coordinateY	pressure	size	status
1	linan	1	87641978	659.389465332031	194.745422363281	0.096000000834465	0.200000017881393	DOWN
2	linan	1	87642012	656.273010253906	200.540771484375	0.184000015258789	0.200000017881393	MOVE
3	linan	1	87642029	653.020324707031	210.050201416016	0.196000009775162	0.233333349227905	MOVE
4	linan	1	87642046	650.581970214844	220.625610351563	0.204000011086464	0.233333349227905	MOVE
5	linan	1	87642062	650.397766113281	233.072265625	0.208000004291534	0.233333349227905	MOVE
6	linan	1	87642096	644.124328613281	265.10498046875	0.208000004291534	0.233333349227905	MOVE
7	linan	1	87642113	644.0048828125	281.7060546875	0.204000011086464	0.266666680574417	MOVE
8	linan	1	87642147	656.338684082031	331.513305664063	0.204000011086464	0.266666680574417	MOVE
9	linan	1	87642164	671.1123046875	368.085754394531	0.204000011086464	0.266666680574417	MOVE
10	linan	1	87642188	691.859375	421.12744140625	0.204000011086464	0.233333349227905	MOVE
11	linan	1	87642215	715.760559082031	488.8017578125	0.200000002980232	0.266666680574417	MOVE

Figure 3. Data Sample

	Feature Name	**Expression**
Average Type	Average speed x, y, total	$\bar{v}_x \quad \bar{v}_y \quad \bar{v}$
	Average acceleration x, y, total	$\bar{a}_x \quad \bar{a}_y \quad \bar{a}$
	Average pressure	$\bar{p}$
	Average size	$\bar{s}$
Maximum Type	Maximum speed x, y, total	$V_{x.max} \quad V_{y.max} \quad V_{max}$
	Maximum acceleration x, y, total	$A_{x.max} \quad A_{y.max} \quad A_{max}$
	Maximum pressure	P_{max}
	Maximum size	S_{max}
	Maximum speed x, y, total position	$T_{v.x.max} \quad T_{v.y.max} \quad T_{v.max}$
	Maximum acceleration x, y, total position	$T_{a.x.max} \quad T_{a.y.max} \quad T_{a.max}$
	Maximum pressure position	$T_{p,max}$
	Maximum size position	$T_{s.max}$
Minimum Type	Minimum speed x, y, total	$V_{x.min} \quad V_{y.min} \quad V_{min}$
	Minimum acceleration x, y, total	$A_{x.min} \quad A_{y.min} \quad A_{min}$
	Minimum pressure	P_{min}
	Minimum size	S_{min}
	Minimum speed x, y, total position	$T_{v.x.min} \quad T_{v.y.min} \quad T_{v.min}$
	Minimum acceleration x, y, total position	$T_{a.x.min} \quad T_{a.y.min} \quad T_{a.min}$
	Minimum pressure position	$T_{p,min}$
	Minimum size position	$T_{s.min}$
Other Type	Total time	T_{whole}
	Break times	N_{break}
	Total points	N_{point}
	Maximum coordinate	$X_{max} \quad Y_{max}$
	Minimum coordinate	$X_{min} \quad Y_{min}$
	Average coordinate	$\bar{x} = \sum_{i=1}^{n} x_i \ /n \quad \bar{y} = \sum_{i=1}^{n} y_i \ /n$
	Maximum coordinate position	$t_{x.max} \quad t_{y.max}$
	Minimum coordinate position	$t_{x.min} \quad t_{y.min}$
	Size of signature	$S_{xy} = (X_{max} - X_{min}) * (Y_{max} - Y_{min})$
	Total distance of move	$D_x \quad D_y \quad$ D

Figure 4. Feature List

round and reduces the weight of which is correctly classified. The final classifier of AdaBoost is:

$$G(x) = sign(f(x)) = sign\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$$

Where $\mathrm{G(x)}$ is the final classifier, m is the number of weak classifiers, $\mathrm{G_m\,(x)}$ is the weak classifier for each step, and α_m is the weight of each classifier. Random Forest is a kind of classifier applying several trees in the sample training and predicting, and constituted by several decision tree. CART (Classification and Regression Tree) is the most frequently used. For each tree, the training set is sampled with replacement from total training samples, which means that some of the samples will appear multiple times in the training set. The features are selected without replacement from all features. Assuming that the total number of features is M, generally we select sqrt(M), 1/2sqrt(M) or 2sqrt(M) features. The advantages of Random Forest are that it can handle high-dimensional data and has a fast training speed without feature selection. Besides, Random Forest is able to detect mutual influence between features during training, and after training it can give a list about which features are more important.

We used a famous python machine learning package named scikit-learn when operating experiment with the application of the module SVC, Logistic Classifier and Random Forest Classifier. The methods to assess parameters are FAR and FRR. FRR (False Rejection Rate) represents the rate of regarding the negative samples as positive samples. FAR (False Acceptance Rate) represents the rate of treating the positive samples as the negative.

For each experimenter, we randomly selected a part of data from his positive samples and negative samples as the training dataset, and the remaining portion as a test dataset.

In model SVM and Logistic Regression, we adjusted parameters C (error term) to train, and use test set to find the best parameters C which has the lowest error rate. In model Adaboost, we adjusted the number of weak classifiers to find the best parameters n_estimators which has the lowest error rate. In model Random Forest, by immobilizing the number of trees in a forest and changing the number of features for each tree, find the best parameters max_features which has the lowest error rate. Then, calculating the rate of misclassification of each model.

Finally, we analyzed and compared these four classification algorithms and estimated the reliability of our models. In addition, we compared the pros and cons of four classification algorithms.

IV. Result and analysis

From 50 positive samples of each participant, we randomly selected 30 as training data, and the remaining 20 as test data. From 20 negative samples, 10 were randomly selected as training data, while the other 10 as test sample. Each experiment is repeated 50 times, and we chose the average value as the final result. Finally, we used FAR and FRR values to evaluate the results of the algorithms.

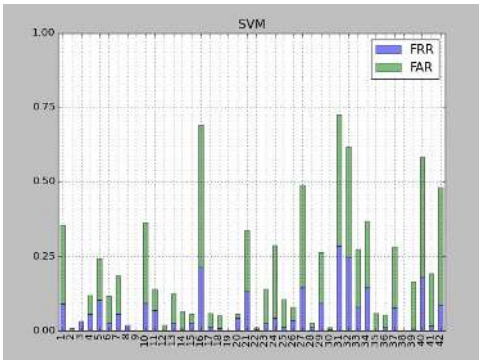

Figure 5. Result of SVM in polynomial kernel

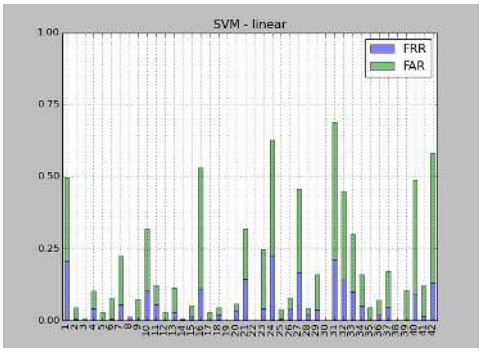

Figure 6. Result of SVM in linear kernel

A. SVM

Model SVM used ”poly” and linear kernel, expression as follows:

Parameter C (error term) selected from the following values, c_svm = [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0, 5.0, 10.0, 50.0, 100.0], the remaining parameters use the default settings. Training the feature vectors of 42 experimenters. The results are shown in Figure 5 and Figure 6.

Finally, the average FRR of 42 experimenters using poly kernel is 5.05%, the average FAR is 11.5% and the total error rate is 16.55%.

The average FRR of 42 experimenters using linear kernel is 4.43% the average FAR is 10.68% and the total error rate is 15.01%.

B. Logistic Regression

For Model Logistic Regression, parameter C (error term) selected from the following values, c_log = [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0, 5.0, 10.0, 50.0, 100.0], and the remaining parameters use the default settings. Training the feature vectors of 42 experimenters using Logistic Regression Classifier. The results are shown in Figure 7.

Finally, the average FRR of 42 experimenters is 1.625%, the average FAR is 6.17% and the total error rate is 7.795%.

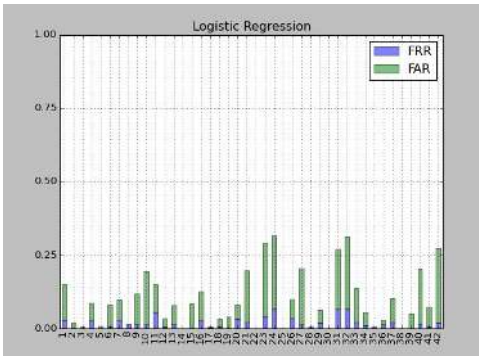

Figure 7. Result of Logistic Regression

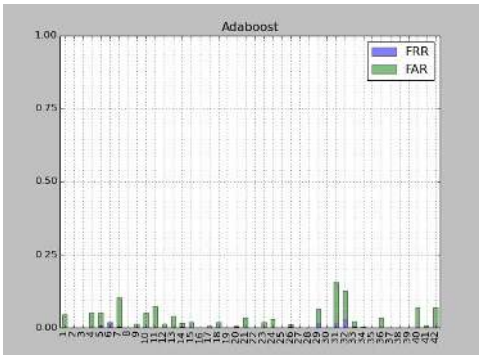

Figure 8. Result of Adaboost

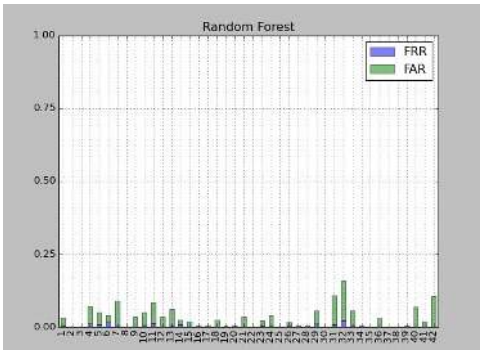

Figure 9. Result of Adaboost

Table I
TOP 10 MOST IMPORTANT FEATURES

1	Whole_Distance_y
2	Average_Size
3	Total_Time
4	Whole_Distance
5	Whole_Distance_x
6	Average_Pressure
7	Size_of_Signature
8	Max_y
9	Average_Speed_y
10	Average_Speed_x

C. Adaboost

For Model Adaboost, we changed the number of weak classifiers to train the model and find the best number of classifiers. The number will be chosen between 40 and 59. The results are shown in Figure 8.

Finally, the average FRR of 42 experimenters is 0.185%, the average FAR is 2.19% and the total error rate is 2.375%.

D. Random Forest

We selected CART to constitute Random forest, used the 'gini' coefficient as the evaluation parameter of CART, and elected 20 as the total amount of trees in forest. Since the total number of features is 57, the parameter "feature amount per tree" (max_features) was chosen between 4 and 19. The results are shown in Figure 9.

Finally, the average FRR of 42 experimenters is 0.34%, the average FAR is 2.44% and the total error rate is 2.78%.

Additionally, Random Forest will give a list about the importance of each feature. We put all experimenters importance list into one list and calculated the importance of all features. The top 10 important features are shown in Table I.

In summary, Random Forest Classifier and Adaboost both get satisfying results, their classification accuracy were significantly higher than SVM and Logistic Regression. Random Forest classification can make error rate as low as 2.78% and the best classifier, Adaboost, can make error rate be as low as 2.375%. For all four classifiers, FRR values are quite low. That indicates these four classifiers have high recognition rates in positive samples, and they are not likely to be misclassified as negative samples. But FAR values are higher than FRR values, and there are big fluctuations in SVM and Logistic Regression models. We figure out that these two belong to binary-class classification algorithms, and they have a poor performance when our negative samples from 4 imposters may not be merged into one class roughly. For Adaboost, by iterative empowerment, samples that are misclassified will be highly corrected in the next step. For Random Forest, each tree in forest marks a partition of sample space. Because Adaboost and Random Forest cut the sample space step by step, so they have better description of positive samples and better adaptability to negative samples, therefore perform better in our classification experiment. It is worthy to note that due to using multiple weak classifiers, the training speed of Adaboost is the slowest in four models. It might be severe when the model comes up with a large data set.

V. Conclusion and future work

Since electronic signature data is generally collected on smart device, we can get richer data via capacitive touch screen and hence improve the result of signature verification. After collecting 2,940 electronic signatures of 42 participants on smartphone, we extracted 57 representative features to construct feature vector. SVM, Logistic Regression, Adaboost and Random Forest are adopted to train and test the model respectively. Experimental results show that in our experiment settings, the classic binary algorithm SVM and Logistic Regression have, worse performance, but Adaboost and Random Forest have satisfactory results in verification, and the best one is Adaboost Classifier.

But there are still some drawbacks in this paper. First of all, the features we extracted from raw data are empirically derived, rather than select by importance of features. Among these four models, only Random Forest algorithm compared the importance of each feature. In our future study, we will try some feature extraction algorithms, such as PCA (Principal Component Analysis), to obtain some typical features. And we expect a higher classification efficiency since data and algorithms may be more reliable than human experience. Secondly, we mainly used the biometric information of the electronic signature for verification, and discarded the graphical signature features (offline signature verification), which may increase the accuracy of signature verification. We will try to make better use of all kinds of signature data and extract effective features to improve the verification accuracy as our future work.

Acknowledgment

This work was supported by National Natural Science Foundation of China [60903201, 91218301]; and the Fundamental Research Funds for the Central Universities [JBK120505, JBK140129].

References

[1] Bajaj, R., and Chaudhury, S., *Signature verification using multiple neural classifiers*, Pattern Recognition, Elsevier, Volume 30(1), 1997, pp. 1-7.

[2] Julian Fierrez-Aguilar, Stephen Krawczyk, Javier Ortega-Garcia, Anil K. Jain, *Fusion of Local and Regional Approaches for On-Line Signature Verification*, Advances in Biometric Person Authentication, IWBRS 2005, Springer Berlin Heidelberg, 2005, pp. 188-196.

[3] JGA Dolfing, EHL Aarts, *On-line Signature Verication with Hidden Markov Models*, Pattern Recognition, IEEE, Brisbane, Qld., Vol. 2, 1998, pp. 1309 1312.

[4] Edson J.R. Justinoa, Flvio Bortolozzia, Robert Sabourin, *A comparison of SVM and HMM classifiers in the off-line signature verification*, Pattern Recognition Letters, Elsevier, Volume 26(9), 2005, pp. 13771385.

[5] MENG Ming, WU Zhong-Cheng, YU Yong, GE Yun-Jian, *Online Signature Verification Based on Segment Features and HMM*, Pattem Recognition and Aitificial Intelligence, Volume 20, 2007, pp. 95-100.

[6] DZ Lejtman, SE George, *On-line handwritten signature verification using wavelets and back-propagation neural networks*, Document Analysis and Recognition, IEEE, Seattle, WA, 2001, pp. 992-996.

[7] PS Deng, HYM Liao, CW Ho, HR Tyan, *Wavelet-based off-line handwritten signature verification*, Computer vision and image, Elsevier, Volume 76, 1999, pp. 173-190.

[8] MA Hai-bao, LIU Man-dan, ZHANG cen, *A Method of On line Handwritten Signature Verification Based on Wavelet Packet Analysis and SVM*, Journal of East China University of Science and Technology (Natural Science Edition),Vol. 33, 2007, pp. 541-545.

[9] Anil K. Jain, Friederike D. Griess, Scott D. Connell, *On-line signature verification*, Pattern Recognition, Elsevier, Vol. 35, 2002, pp. 29632972.

[10] Alisher Kholmatov, Berrin Yanikoglu, *Identity authentication using improved online signature verification method*, Pattern Recognition Letters, Elsevier, Volume 26, 2005, pp. 24002408.

[11] Hairong Lv, Wenyuan Wang, Chong Wang, Qing Zhuo, *Off-line Chinese signature verification based on support vector machines*, Pattern Recognition Letters, Elsevier, Volume 26, 2005, pp. 23902399.

[12] ZHANG Da-Hai, WANG Zeng-Fu, *Feature Extraction and Personalized Feature Selection for Online Signature Verification*, Pattem Recognition and Aitificial Intelligence, Volume 22, 2009, pp. 619-623.

[13] Kai Huang, Hong Yan, *Off-line signature verification based on geometric feature extraction and neural network classification*, Pattern Recognition, Elsevier, Volume 30, 1997, pp. 917.

[14] H. Baltzakisa, N. Papamarkos, *A new signature verification technique based on a two-stage neural network classifier*, Engineering Applications of Artificial Intelligence, Elsevier, Volume 14, 2001, pp. 95103.

[15] CHENG-JIANG WANG, DI DAI, *CHINESE HANDWRITING SIGNATURE AUTHENTICATION USING DATA MINING TECHNIQUE*, 2007 International Conference on Wavelet Analysis and Pattern Recognition, Beijing, China, 2007.

[16] Dit-Yan Yeung, Hong Chang, Yimin Xiong, *SVC2004: First International Signature Verification Competition*, 2004.

[17] Kumiko Yasudaa, Daigo Muramatsub, Satoshi Shiratoa, Takashi Matsumotoa, *Visual-based online signature verification using features extracted from video*, Journal of Network and Computer Applications, Elsevier, Volume 33, 2010, pp. 333341.

[18] S. Rashidi , A. Fallah, , F. Towhidkhah, *Feature extraction based DCT on dynamic signature verification*, Scientia Iranica, Elsevier, Volume 19, 2012, pp. 18101819.

[19] Yasmine Guerbai, Youcef Chibani , Bilal Hadjadji, *The effective use of the one-class SVM classifier for handwritten signature verification based on writer-independent parameters*, Pattern Recognition, Elsevier, Volume 48, 2015, pp. 103113.