International Journal of Advanced Smart Convergence Vol.5 No3 40-46 (2016)
http://dx.doi.org/10.7236/IJASC.2016.5.3.40

IJASC 16-1-3

Mini-Batch Ensemble Method on Keystroke Dynamics based User Authentication

Jiacang Ho[1], Dae-Ki Kang[2]

[1]*Department of Ubiquitous IT, Graduate School, Dongseo University*
[2]*Department of Computer & Information Engineering, Dongseo University*
e-mail: [1]ho_jiacang@hotmail.com, [2]dkkang@dongseo.ac.kr

Abstract

The internet allows the information to flow at anywhere in anytime easily. Unfortunately, the network also becomes a great tool for the criminals to operate cybercrimes such as identity theft. To prevent the issue, using a very complex password is not a very encouraging method. Alternatively, keystroke dynamics helps the user to solve the problem. Keystroke dynamics is the information of timing details when a user presses a key or releases a key. A machine can learn a user typing behavior from the information integrate with a proper machine learning algorithm. In this paper, we have proposed mini-batch ensemble (MIBE) method which does the preprocessing on the original dataset and then produces multiple mini batches in the end. The mini batches are then trained by a machine learning algorithm. From the experimental result, we have shown the improvement of the performance for each base algorithm.

Keywords: *Mini-batch, ensemble method, keystroke dynamics, user authentication.*

1. Introduction

We know that the more the online accounts you acquired, the more the fear of your accounts being hacked. To prevent the issue, we can integrate the login system with a biometric approach. The biometric approach consists of two folds, one is the physical-based approach and another is the behavior-based approach. The physical-based approach involves iris, fingerprint, speech, etc. The behavior-based approach, on the other hand, involves keystroke dynamics, gait analysis, etc. In this study, we have focused more on the keystroke dynamics on the keyboard. The reason we choose behavior-based approach rather than physical-based approach is because it is inexpensive, implement with no extra hardware and easy to be implemented even though the performance of the behavior-based approach might not as high as the physical-based approach. We also believe that keystroke dynamics can bring the protection to the system and it can be a common approach in the future. Due to these reasons, there have been a considerable amount of researchers performing the behavior-based research. [1 – 10].

Keystroke dynamics is the information of timing detail when a user presses a key or releases a key [11]. The two common timing details are dwell time and flight time. The dwell time (also known as a duration time [9]) is a duration between a key being pressed and released. However, the flight time (also known as interval time [9]) is a duration between a key being released and a next key being pressed. We show all of the possible timing details as following:

Manuscript Received: July 17, 2016 / Revised: July 28, 2016 / Accepted: August 05, 2016
Corresponding Author: dkkang@dongseo.ac.kr
Tel: +82-51-320-1724, Fax: +82-51-327-8955
Department of Computer & Information Engineering, Dongseo University
47 Jurye-Ro, Sasang-Gu, Busan 47011, South Korea

- Hold (H): time duration (or a dwell time) of pressing a key
    - A holding time of the first key, H1
    - A holding time of the second key, H2
- Up-Down (UD): time duration (or a flight time) between key-up of the first key and key-down of the second key.
- Down-Down (DD): time duration between key-down of the first key and key-down of the second key; it is the sum of H1 and UD.
- Up-Up (UU): time duration between key-up of the first key and key-up of the second key; it is the sum of UD and H2.
- Down-Up (DU): time duration between key-down of the first key and key-up of the second key; it is the sum of DD and H2.

With the information above, a machine can use them to learn a user typing behavior [8], emotion [12], gender [13], dominant hand [4], etc. It is difficult for any intruder intrudes a user account easily due to the fact that every user has different kind of typing style when she/he types a string on a keyboard device.

We explain our proposed method in next section (Section 2). Section 3 describes the experimental method. We show the experimental result in Section 4. Last but not least, we conclude the paper in Section 5.

2. Mini-Batch Ensemble Method

Ensemble method [14] is a meta-learning algorithm that uses multiple machine learning algorithms or multiple datasets to produce a high accuracy. With the ensemble concept, we have proposed mini-batch ensemble (MIBE) method. MIBE is a kind of preprocessing method. It produces multiple mini-batches (also refers to sub-datasets) from a single dataset. We called it as a mini-batch is because the size of each batch is small enough (i.e. within 2 to 5). Later, we use a proper machine learning algorithm to train all mini-batches and produce a model for each mini-batch. We explain the motivation and distribution of mini-batch in the following sub-sections.

Figure 1. Mini-Batch Ensemble concept

2.1 Motivation

As we know, one may take more time in the early stage of learning. After practicing for a few more time, one will act faster and faster. We called it as be accustomed to something. For example, a user is requested to type a string that is totally unfamiliar to her/him. Assume that the string is the alphabet letters from 'A' to 'Z'. In the first few trials, she/he may take several seconds (10 seconds for this example) to complete it. After practicing an hour per day for a month, she/he may take only 5 or lesser than 5 seconds to complete the string. The user can practice more to improve her/his skill. However, everything has its own limitation. In other words, we cannot improve our skill after we reach a certain level (known as realm point [15]) even though we keep practicing it every day. We also can observe this phenomenon in the dataset. We can review that most of the instances in the most of the datasets are familiar to the password, and we have either fewer instances or none of the instances that are unfamiliar to the password.

Our assumption by using MIBE is the larger the dataset, the higher the accuracy. Assume that we have produced ten with the same size of mini-bathes from a dataset. From these ten datasets, assume that first three mini-batches are unfamiliar to the password and last seven mini-batches are familiar to the password. If an imposter is unfamiliar to the password, then the distance of his/her typing speed will be closer to the first three mini-batches and farther to the last seven mini-batches. If we summing up all the distance, the imposter will have a very high distance (score) (because of the seven large distance from last seven mini-batches). However, for a genuine user who is familiar to the password will have a low distance (score) because he/she will have low distance from the seven mini-batches and only three high distance from first three mini-batches.

2.2 Distribution of mini-batch

In a benchmark dataset, timing details such as H, UD and DD are used as the attributes. The more the characters used in a password string, the more the attributes will be in a dataset. For example, a password, "admin123", consists of 8 holding key (H), 7 up-down key (UD) and 7 down-down key (DD). In total, there are 8 + 7 + 7 = 22 attributes. In the preprocessing phase, MIBE will choose one of the attributes in a dataset randomly. From the chosen attribute, MIBE distributes the original dataset into *N* mini-batches based on the chosen attribute's data values. For a toy example, given a dataset with 10 instances and three attributes (*x*, *y*, and *z*). Assume that MIBE has chosen *y* attribute by random. The data values of *y* attribute are $y_1, y_2, \dots, y_{10}$. Assume that y_1 to y_5 has been ordered in ascending order (also known as an ascending sequence) from the original dataset, y_5 to y_8 is a descending sequence, and then y_8 to y_{10} is another ascending sequence. In other words, an attribute's values, {1, 2, 5, 8, 12, 7, and 4}, can be distributed into two sequences, {1, 2, 5, 8, and 12} and {12, 7, and 4}, which are an ascending sequence and a descending sequence, respectively. After that, MIBE divides the original dataset into three mini-batches (sub-datasets) based on the *y* attribute. The first mini-batch is the top five instances (with all attributes). The second mini-batch is started from 5^{th} instance to 8^{th} instance (with all attributes). The last mini-batch, on the other hand, is the last three instances (8^{th} instance to 10^{th} instance with all attributes). In summary, MIBE uses one random attribute of the dataset as criteria to split the dataset into *N* mini-batches. We have provided Figure 1 for understanding the MIBE concept easily.

2.3 Training phase and testing phase

After we have *N* mini-batches, we create a model for each mini-batch with a proper machine learning algorithm in the training phase. During the testing phase, we test a testing data with all models. Each model produces a score for the testing data. We simply summing up the score in the end to generate a final output. The final output is then used to generate a receiver operating characteristic (ROC) curve. From the ROC curve, we can produce the equal error rate (EER).

3. Dataset

In this paper, we have used two public datasets that are CMU benchmark dataset [5] and GREYC dataset [3]. We explain the experimental settings on these two datasets in the following sub-sections.

3.1 CMU benchmark dataset

The dataset consists of 20,400 instances. There are 51 subjects have participated during the enrollment phase. Each subject is requested to input 50 instances per session. There are total eight sessions in the dataset. Hence, each subject has 400 instances. To construct the experiment, we select one of the subjects to be a genuine user. The remaining subjects are the imposters. During the training phase, we use the first 200 instances of the selected subject as the training data. During the testing phase, we use the last 200 instances of the selected subject as the testing data for the genuine user. Meanwhile, we use first five instances from the remaining subjects (except the selected subject) as the testing data for the imposter. The reason we have extracted first five instances from the imposters is because of the assumption that the imposter is unfamiliar with the password in the experiment [5].

The password used in the dataset is ".tie5Roanl[enter]" ('[enter]' is an ENTER key on the keyboard). The total attributes in the dataset are 31 (11H, 10UD and 10DD). In the experiment, we have tested all attributes by using only one attribute per experiment. In other words, we have performed 31 experiments with different attribute per time for all algorithms to obtain the minimum of the EER, the maximum of the EER and the average of the EER. The results are shown in Table 1. The reason we have tested all the attributes is to analyze the effect of each attribute to the EER.

3.2 GREYC web-based keystroke dynamics dataset

We have separated the GREYC dataset into three different datasets. The datasets include login dataset, password dataset and the combination of login and password dataset. Unfortunately, not all subjects have the equivalent instances in the dataset. After filtered the dataset based on the parameters we need, it left only 31 subjects with 90 instances per subject. We have used same experimental settings as the CMU dataset in the dataset. We choose one of the subjects as a genuine user and the rest of the subjects as the imposters. From the chosen subject, we have used the first 60 instances as the training data. The remaining 30 instances are used as testing data of the genuine user. We have extracted the first instance from the imposters to form a testing data of the imposter.

The string used in the login dataset is "laboratoire greyc". There are 49 attributes (17H, 16UD and 16DD) in the dataset. For the password dataset, the password is "sésame". There are 16 attributes (6H, 5UD and 5DD). We have performed same experimental settings as the CMU dataset to obtain the minimum of the EER, the maximum of the EER and the average of the EER. The results are shown in Table 2, 3 and 4.

4. Experimental Result

We have operated four experiments with four different datasets, which are CMU benchmark dataset, GREYC – login dataset, GREYC – password dataset, and GREYC – login and password dataset. We have tested the four datasets with five machine learning algorithms that are Euclidean distance, Manhattan distance, Mahalanobis distance, median vector proximity [1], and Manhattan (scaled) distance [5]. Table 1 shows the result for the algorithms with and without the MIBE method in the CMU dataset. However, Table 2, 3, and 4 show the results for the algorithms with and without the MIBE method in the GREYC-login, GREYC-password and GREYC-login and password, respectively.

Table 1. The average of equal error rate with their standard deviation for five algorithms on the CMU dataset. The significant improvement on the performance of the algorithm is in bold.

Algorithm	EER			
	Without MIBE method	With MIBE		
		Minimum	Maximum	Average
Euclidean	0.171 (0.095)	0.170 (0.093)	0.173 (0.100)	0.17148 (0.09607)
Manhattan	0.153 (0.092)	0.127 (0.068)	0.129 (0.071)	**0.12303 (0.06994)**
Mahalanobis	0.110 (0.065)	0.154 (0.080)	0.160 (0.094)	0.15668 (0.08671)
Median Vector Proximity	0.080 (0.062)	0.074 (0.064)	0.076 (0.067)	**0.07455 (0.06513)**
Manhattan (scaled)	0.096 (0.069)	0.090 (0.055)	0.093 (0.058)	**0.09187 (0.05677)**

Table 2. The average of equal error rate with their standard deviation for five algorithms on the GREYC-login dataset. The significant improvement on the performance of the algorithm is in bold.

Algorithm	EER			
	Without MIBE method	With MIBE		
		Minimum	Maximum	Average
Euclidean	0.167 (0.131)	0.160 (0.120)	0.169 (0.135)	0.16602 (0.12663)
Manhattan	0.098 (0.081)	0.090 (0.072)	0.098 (0.079)	**0.09429 (0.07490)**
Mahalanobis	0.099 (0.075)	0.175 (0.096)	0.192 (0.111)	0.18447 (0.10453)
Median Vector Proximity	0.065 (0.059)	0.040 (0.044)	0.047 (0.053)	**0.04349 (0.04827)**
Manhattan (scaled)	0.059 (0.052)	0.059 (0.053)	0.073 (0.070)	0.06796 (0.06149)

Table 3. The average of equal error rate with their standard deviation for five algorithms on the GREYC-password dataset. The significant improvement on the performance of the algorithm is in bold.

Algorithm	EER			
	Without MIBE method	With MIBE		
		Minimum	Maximum	Average
Euclidean	0.222 (0.115)	0.222 (0.110)	0.232 (0.124)	0.22656 (0.11763)
Manhattan	0.204 (0.119)	0.171 (0.092)	0.182 (0.101)	**0.17625 (0.09663)**
Mahalanobis	0.156 (0.074)	0.177 (0.075)	0.195 (0.089)	0.18713 (0.08181)
Median Vector Proximity	0.054 (0.056)	0.125 (0.061)	0.133 (0.068)	0.12931 (0.06388)
Manhattan (scaled)	0.148 (0.074)	0.141 (0.069)	0.156 (0.084)	0.14731 (0.07731)

Table 4. The average of equal error rate with their standard deviation for five algorithms on the GREYC-login and password dataset. The significant improvement on the performance of the algorithm is in bold.

Algorithm	EER			
	Without MIBE method	With MIBE		
		Minimum	Maximum	Average
Euclidean	0.173 (0.130)	0.170 (0.119)	0.177 (0.139)	0.17332 (0.13035)
Manhattan	0.094 (0.074)	0.087 (0.066)	0.095 (0.074)	**0.09108 (0.07085)**
Mahalanobis	0.098 (0.072)	0.186 (0.082)	0.202 (0.102)	0.19303 (0.09249)
Median Vector Proximity	0.054 (0.056)	0.028 (0.038)	0.039 (0.047)	**0.03385 (0.04305)**
Manhattan (scaled)	0.056 (0.054)	0.057 (0.047)	0.065 (0.063)	0.06089 (0.05460)

From Table 1, we can observe that Manhattan distance, median vector proximity and Manhattan (scaled) distance have significant improvement. Manhattan has reduced 3% of the EER when it is operated with MIBE method. At the same time, median vector proximity has reduced around 0.6% of the EER and Manhattan (scaled) distance has decreased around 0.4% by using MIBE method.

From Table 2, Manhattan distance and median vector proximity have shown significant improvement. Manhattan distance has improved from 0.098 to 0.09429. At the same time, median vector proximity has improved from 0.065 to 0.04349.

In Table 3, only Manhattan distance has the improvement when it is tested with the MIBE method. The EER result shows 0.17625 when it is tested with the MIBE method. In Table 4, on the other hand, Manhattan distance and median vector proximity show the improvement when they are tested with the MIBE method. Manhattan distance and median vector proximity have improved to 0.09108 and 0.03385, respectively.

In these four tables, Manhattan distance has higher performance than the other algorithms. It has shown significant improvement in all datasets.

5. Conclusion

In the study, we have proposed mini-batch ensemble (MIBE) method to perform preprocessing in the user authentication using the keystroke dynamics. With the MIBE method, some algorithms have performed significant improvement. We believe that the distribution of a dataset into an appropriate number of mini-batches can improve the performance of the algorithm.

6. Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (no. NRF-2015R1D1A1A01061328).

References

[1] M. M. Al-Jarrah, "An anomaly detector for keystroke dynamics based on medians vector proximity," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 3, no. 6, pp. 988–993, 2012.

[2] S. Cho, C. Han, D. H. Han, and H.-I. Kim, "Web-based keystroke dynamics identity verification using neural network," *Journal of organizational computing and electronic commerce*, vol. 10, no. 4, pp. 295–307, 2000.

[3] R. Giot, M. El-Abed, and C. Rosenberger, "Web-based benchmark for keystroke dynamics biometric systems: A statistical analysis," in *Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP), 2012 Eighth International Conference on*. IEEE, 2012, pp. 11–15.

[4] S. Z. S. Idrus, E. Cherrier, C. Rosenberger, and P. Bours, "Soft biometrics for keystroke dynamics," in *Image analysis and recognition*. Springer, 2013, pp. 11–18.

[5] K. S. Killourhy, R. Maxion et al., "Comparing anomaly-detection algorithms for keystroke dynamics," in *Dependable Systems & Networks, 2009. DSN'09. IEEE/IFIP International Conference on*. IEEE, 2009, pp. 125–134.

[6] J. Montalv˜ao, E. O. Freire, M. A. Bezerra Jr, and R. Garcia, "Contributions to empirical analysis of keystroke dynamics in passwords," *Pattern Recognition Letters*, vol. 52, pp. 80–86, 2015.

[7] K. Revett, "A bioinformatics based approach to user authentication via keystroke dynamics," *International Journal of Control, Automation and Systems*, vol. 7, no. 1, pp. 7–15, 2009.

[8] Z. Syed, S. Banerjee, and B. Cukic, "Leveraging variations in event sequences in keystroke-dynamics authentication systems," in *High- Assurance Systems Engineering (HASE), 2014 IEEE 15th International Symposium on*. IEEE, 2014, pp. 9–16.

[9] X. Wang, F. Guo, and J.-f. Ma, "User authentication via keystroke dynamics based on difference subspace and slope correlation degree," *Digital Signal Processing*, vol. 22, no. 5, pp. 707–712, 2012.

[10] E. Yu and S. Cho, "Keystroke dynamics identity verification - its problems and practical solutions," *Computers & Security*, vol. 23, no. 5, pp. 428–440, 2004.

[11] R. Moskovitch, C. Feher, A. Messerman, N. Kirschnick, T. Mustafić, A. Camtepe, B. Lőhlein, U. Heister, S. Mőller, L. Rokach et al., "Identity theft, computers and behavioral biometrics," in *Intelligence and Security Informatics, 2009. ISI'09. IEEE International Conference on*. IEEE, 2009, pp. 155–160.

[12] A. N. H. Nahin, J. M. Alam, H. Mahmud, and K. Hasan, "Identifying emotion by keystroke dynamics and text pattern analysis," *Behaviour & Information Technology*, vol. 33, no. 9, pp. 987–996, 2014.

[13] R. Giot and C. Rosenberger, "A new soft biometric approach for keystroke dynamics based on gender recognition," *International Journal of Information Technology and Management*, vol. 11, no. 1-2, pp. 35–49, 2012.

[14] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple classifier systems*. Springer, 2000, pp. 1–15.

[15] J. Ho and D.-K. Kang, "Sequence alignment with dynamic divisor generation for keystroke dynamics based user authentication," *Journal of Sensors*, vol. 2015, 2015.