Online Signature Verifcation on Mobile Devices using Machine Learning

Akshay Malhotra **Kundan Sen** **Roshan Jha**

Department of Information Technology | IIIT Allahabad

Abstract – Biometric recognition and verification systems have existed for over a century and are extremely crucial to every industry to keep their information and data secure. These include but are not limited to fingerprint scanning, palm vein authentication, retina scanning, iris scanning, face recognition and voice analysis. With developments and advancements in technology, most of the biometric verification systems have improved however, the cost of deploying a system is extremely high, let alone conducting verification of identities in mass. Not only that but most of these systems are not portable. In-order to find a middle ground between efficiency, effectiveness in mass quantity and portability, we take a deeper look into signature verification which can be deployed using mobile devices to produce fruitful results.

Introduction

Signature verification has long been an interesting and challenging problem for biometric researchers. Signatures can be broadly categorized into two categories – Online Signatures and Offline Signatures. Offline signatures are those that take place on paper and analysis of them can be carried out only when information of a signature on paper is somehow extracted and stored digitally (high quality picture of a signature). This information can be termed as features and are used primarily to distinguish between two signatures. In a way, it would be safe to say that the features provide a unique identity to a signature.

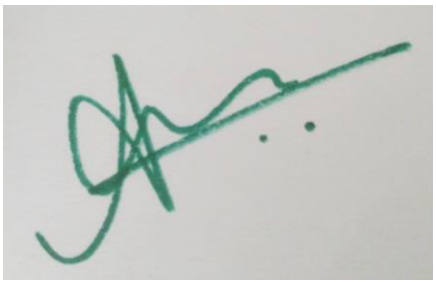


FIGURE 1 - OFFLINE SIGNATURE

Online Signatures on the other hand are signatures that take place on an electronic device that are capable of recording the movement of signature at a fixed interval of time, digitally. Therefore, x-y coordinates of the user's signatures along with attributes like pressure, time and pen up/down are also acquired. It is because of these vast range of dynamic features that online signature verification system usually achieves a better accuracy than an offline system.

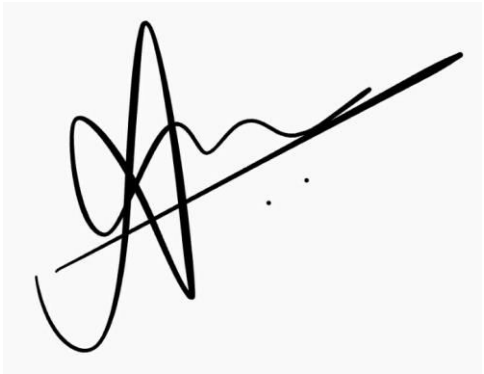


FIGURE 2 - ONLINE SIGNATURE

A FEATURE SET WILL BE EXTRACTED FROM THIS TO DIFFERENTIATE IT FROM OTHER DIGITAL SIGNATURES

Optimistic results have been obtained by previous studies and analysis on online signatures [1, 2, 3, 4] with the help of Support Vector Machines based on LCSS Kernel functions, Hybrid features, Dynamic Time Wrapping, Hidden Markov Model and Neural Networks. However, many of these are not computationally and space efficient algorithms for enrolling and verifying signatures on mobile devices. In this paper, a paramedic approach has been adopted for solving the problem in hand. What this means is that a set of parameters (features) are extracted from the digital

representation of a reference signature and test signature and are compared to decide if the test signature was authentic at all. This means that a signature is represented using a feature vector and deriving a descriptive set of features that can be used efficiently and effectively is extremely important. In 2005, Fierrez-Aguilar et al proposed a set of 100 features to represent a signature. The feature set consist of histograms that capture the distribution and values of attributes generated from a raw signature [5]. The template generator then constructs a user specific profile using the histogram based information derived from multiple instances of a user's signature. This user-specific profile is then used for testing and verifying a signature. Deployment of this method on public signature dataset (SUSIG Signature Dataset) has yielded optimal results and shows that this method is best suited for mobile devices. In-order to produce real-time, realistic results, an Android Application was developed that would take a digital signature as input and perform the above mentioned analysis effectively making it a cost-effective biometric verification system.

Methodology

Feature Extraction

(August 2016 – September 2016)

Histograms are widely used as a feature set to obtain attribute statistics when it comes to recognition/verification tasks. They have been used for object recognition and offline signature verification however at time, analysis conducted using histograms have been limited. Histograms can be used to derive a plenty of information such as but not limited to x-y trajectories, speed, angles, pressures, pen up/down and their derivatives. Time series data of the online signature is converted into a sequence of cartesian vectors and their attributes. Histograms are then extracted from these vector sequences.

Let the x-coordinates, y-coordinates and pressure coordinates of a signature with length n be sampled at times $T = \{t1, t2, \dots, tn\}$ be represented as $X = \{x1, x2, \dots, xn\}$, $Y = \{y1, y2, \dots, yn\}$ and $P = \{p1, p2, \dots, pn\}$. Since the information gathered from our exclusive android application wasn't sampled at constant time intervals, interpolation and extrapolation was carried out to generate a uniform raw data on which productive analysis could be carried out. The derivatives of the above mentioned vectors are computed and stored as X^k, Y^k and P^k.

$$X^k = \{x_i^k \mid x_i^k = x_{i+1}^{k-1} - x_i^{k-1}\}$$

$$Y^k = \{y_i^k \mid y_i^k = y_{i+1}^{k-1} - y_i^{k-1}\}$$

$$P^k = \{p_i^k \mid p_i^k = p_{i+1}^{k-1} - p_i^{k-1}\}$$

These vector capture the positional invariant features of the signature. By repeating the process of taking differences k times yields k^{th} derivative. A lower derivative value yields towards providing the global minimum whereas a higher derivative order might yield a local minimum.

With the help of this, a 5 tuple relation is created that consists of k^{th} order derivate of the cartesian, polar coordinates and pressure attributes,

$$v_i^k = (x_i^k, y_i^k, r_i^k, \theta_i^k, p_i^k).$$

This then yields to the construction of a sequence of vectors V = $\{ v_i^* \mid i = 1,2, \dots, n\}$ where each vector element $v_i^* = \{v_i^1 | v_i^2 \dots | v_i^n\}$. In the 5 tuple relation, $\theta_i^k = tan^{-1}(y_i^k / x_i^k)$ and $r_i^k = \sqrt{(y_i^k)^2 + (x_i^k)^2}$

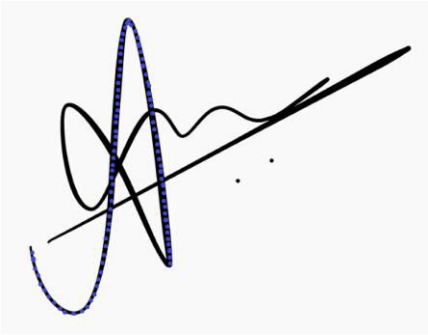


FIGURE 3 - SIGNATURE WITH N X-Y COORDINATES HIGHLIGHTED

The histograms are computed with the help of dividing the range of an attribute into equal width bin intervals and the counting the number of elements/point that fall into each bin. There are two types of histograms that can be used in this experiment, 1-D Histogram and 2-D Histogram. Taking into consideration that this project circulates around the concept of easy and mass implementation, we have taken only 1-D histograms into consideration. 1-D Histograms helps in capturing the distribution of individual attributes. Moreover, histograms of different order can provide us unique information that were usually missed out (for example, information offered by $\emptyset^1 and\ \emptyset^2$). R^1, for example, captures that speed distribution of a signature. This feature happens to be extremely unique for every user and also depends on the circumstances on which the person is signing in. The angle attribute, for example, has a range of $[-\pi, +\pi]$ and can be divided into 24 equal width bin intervals. With the help of this, each of the individual elements of θ_i^1 can be plotted in accordance to which bin they fall under. This remains mostly consistent for a user's signature, with a unique range of elements falling under a bin.

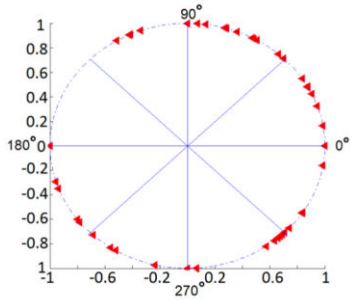


FIGURE 4 - HISTOGRAM WITH EQUAL BIN INTERVALS

A signature's feature vector is now computed with the help of histograms created with different features. This feature vector can be defined as

$$F = \{B_1 || B_2 || B_3 || \ldots . || B_k\},$$

where k is the total number of histograms and B_i is a vector of bin frequencies of an ith histogram. Therefore, an online signature is now represented by a feature vector F which consists of bin information of all the possible histograms computed.

The next step of the process involves the creation of a user template for each user for the signature matching purposes. F is used largely in this process. A feature vector for every user is generated, irrespective of whether a template is being generated for a user or a signature is being tested for verification. Let $F^{S_j} = \{ f_i^{S_j} | \ i = 1, \ldots M \}$ be the feature vector for a user u and M be the total number of features represented by the histogram. j stands for the signature instance number and can range till S for a user u. The quantization step size vector of a user, $Q^u = \{ q_i^u \mid i = 1, \ldots, M \}$ is obtained by computing the below mentioned formula

$$q_i^u = \beta \sqrt{\frac{1}{S}\sum_{j=1}^{S}\left(f_i^{s_j} - \mu_{f_i^{(u)}}\right)^2}, \quad i = 1, \ldots, M$$

where $\mu_{f_i^{(u)} = \frac{1}{S}\sum_{j=1}^{S} f_i^{s_j}}$ and β is fixed at 1.5

Using the quantized step sizes q_i^u in Q^u, the quantized feature vector for each sample *s* for a user *u* is generated $\hat{F}^{(s_j|u)} = \{\hat{f}_i^{s_j} | i = 1, \ldots, M\}$ where,

$$\hat{f}_i^{(s_j|u)} = \left[\frac{f_i^{s_j}}{q_i^u + \epsilon}\right], \quad i = 1, \ldots, M$$

Before the matching process begins, we also derive a user-specific feature vector template by averaging the quantized feature vectors for all enrolled samples of an online signature of user *u*

$$\ddot{f}_i^u = \left[\frac{\sum_{j=1}^{S} \hat{f}_i^{(s_j|u)}}{S}\right], \quad i = 1, \ldots, M$$

A tuple $(Q^u, \ddot{F}^u)$ comprising of the quantization step size vector and its associated feature vector template is stored and later used to verify a claimed signature.

For verification, the Manhattan distance between $\ddot{F}^u$ and $\hat{F}^{(t|u)}$ is calculated and depending on a predefined threshold for the score, the signature is either accepted as genuine or rejected as forgery.

Feature Extraction

(August 2016 – September 2016)

A different approach was adopted towards the second half of the project in the hope of finding productive and fruitful results. A completely new feature extraction process was implemented which includes filtering, linear interpolation, location and time normalization and size normalization in order to find the notable differences. This process yielded a fixed feature vector for every user which was used as a user's signature characteristics. Any attempt of verification with the user's signature would result in the usage and exploitation of the user's feature vector. It was also noted these steps of preprocessing yielded optimal results when a dataset was collected using an Android Application. Therefore, the model was tested on two different datasets,

1. SUSIG Online Dataset (includes information such as pressure that cannot be gathered using a mobile phone)
2. Dataset collected using an Android Application

At times, acquired signatures might not be crisp and clear due to the introductions of unwanted noise. Inorder to smoothen the signature and remove points that do not add much value to the identity of the signature, a low-pass filter was implemented. In this case, the Ramer-Douglas-Peucker algorithm was implemented to be precise. After applying the algorithm, it was noticed that there was a significant reducing and points (coordinates) and noise. The signature looked crisper and every points carried a deeper meaning with respected to the signature.

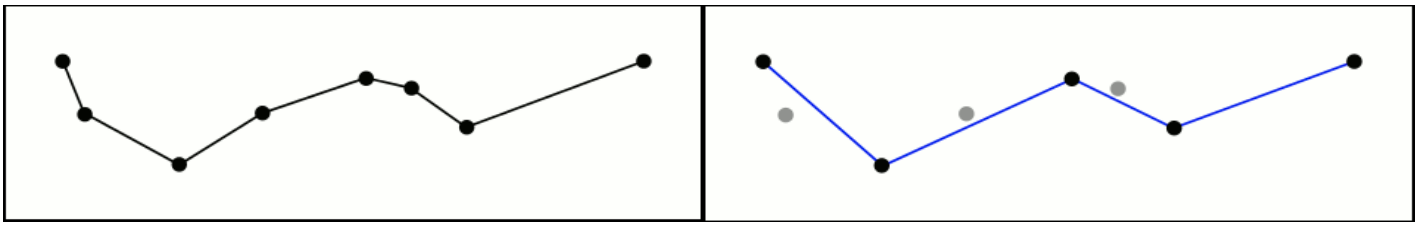


FIGURE 5 - CHANGES NOTICED AFTER APPLYING RAMER-DOUGLAS-PEUCKER ALGORITHM.
(LEFT) THE INITIAL SHAPE OF A CURVE. (RIGHT) TRANSFORMATION OF CURVE AFTER APPLYING RDP ALGORITHM

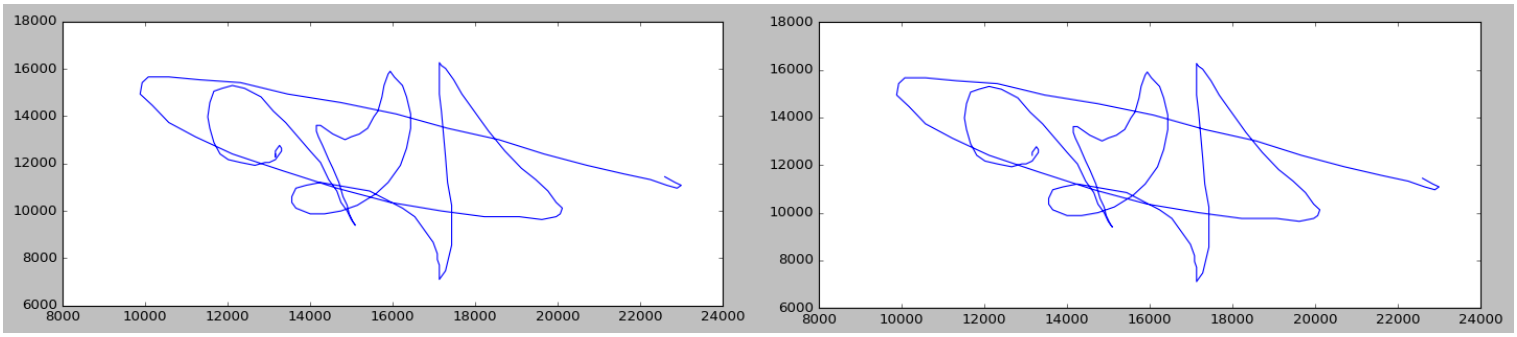


FIGURE 6 - (LEFT) SIGNATURE BEFORE APPLYING RDP ALGORITHM. (RIGHT) SIGNATURE AFTER APPLYING RDP ALGORITHM.
IT CAN BE NOTED THAT THERE ARE MINIMAL OR NO DIFFERENCES IN THE SIGNATURE

After this, linear interpolation was applied inorder to get a 256 equally spaced points for every signature. This also happened to be a part of the feature vector which ensured that the size of the feature vector for every user remained constant. The values obtained after the interpolation process were then floored and converted to integer type for convenience.

Signatures when captured, irrespective of the device, can cause a disturbing difference based on the orientation. Two signature of the same user can be treated as dissimilar with respect to the coordinate axis if the orientation differs (portrait/landscape). Inorder to solve this problem, the x-axis and y-axis temporal functions were normalized by centering the origin of the coordinates at the signature center of mass with a specific rotation. This ensured that all the 256 points of the signature would then be expressed without any error irrespective of the orientation.

The coordinates of the signature were then scaled down between the ranges of 0-1. The advantage of this step was that irrespective of how lengthy a signature might be, it would be expressed within a fixed range, thereby aiding to the verification process. All the signatures would be expressed within a fixed range and each of the 256 value would play a vital/important role.

Creation of Feature Vectors

Below is the list of features that were taken into consideration during the process of constructing a feature vector. Based on the dataset collected, its condition and scenario, some of the below mentioned featured were added/removed accordingly,

- 256 interpolated and scaled x coordinates
- 256 interpolated and scaled y coordinates
- x-y coordinates of the first pen-up (in case of no pen-up's, (0,0) was considered)
- Maximum Pressure
- Average Pressure
- Number of pen-up's.
- Time duration of signature
- Ratio of width/height of signature

When analysis was carried out on the SUSIG Online Signature Dataset, information regarding a signature's pressure was taken into account due to its availability. Therefore, 515 point feature vector was constructed for every instance of a signature that included 256 interpolated and scaled x coordinates, 256 interpolated and scaled y coordinates, x-y coordinates of the first pen-up and the maximum pressure of the signature.

Mobile devices do not have a capacitive screen and therefore, the availability of information regarding pressure is taken away. Obtaining the pressure with respect to the coordinates is extremely difficult unless a capacitive stylus is used. Since the aim of the project is to find the best biometric verification process that is not financially straining, pressure was not taken into account. An Android Application was developed that would store the coordinates of the signature and whether the pen was up/down with respect to the coordinate. Based on this information, a 515 point feature vector was constructed that included 256 interpolated and scaled x coordinates, 256 interpolated and scaled y coordinates, x-y coordinates of the first pen-up and the number of pen up's in a signature.

Verification using Support Vector Machine Classifier

Support Vector Machine is a classification algorithm which is used to find the optimal separating hyper-plane (example, a plane in a 3-dimensional space) that maximizes the margin (distance between the hyper-plane's and the closest data point) of the training data because it classifies unseen data to the best of its abilities. SVM finds a linear separating hyper-plane with the maximum margin in the higher dimensionally. It has been observed that SVM doesn't work best when classification is to be done in higher degree's, however, since the classification over here is binary (ie, forgery or genuine) SVM seems to be an ideal classifier that would fit our dataset. SVM also suits our dataset better as the features mentioned above which are a part of the feature vector as numerically less when compared to the training sample. It is also known as a black box classifier as the complex data transformations and resulting boundary plane are difficult to interpret.

Finding the optimal hyper-plane based on SVM

The equation of hyper-plane can be written as $\mathbf{w}^T\mathbf{x}=0$. Any hyper-plane can be written as the set of points $\mathbf{x}$ satisfying $\mathbf{w}\cdot\mathbf{u}+b=0$ (Decision rule : if ≥ 0, then positive samples).

Considering 2-dimentional vectors let us assume that the two hyper-planes *h1* and *h2* separate the data and have the following equations.

$$\mathbf{w}\cdot\mathbf{x}+b=1$$
$$\mathbf{w}\cdot\mathbf{x}+b=-1,$$

so that h_0 (h_0 is the optimal hyper-plane)is equidistant from $h1$ and $h2$. The hyper-planes that meet the following constraints are selected where for each vector $\mathbf{xi}$ either:

$$\mathbf{w}\cdot\mathbf{x_i} + b \geq 1 \text{ for } \mathbf{xi} \text{ having the class 1, or,}$$
$$\mathbf{w}\cdot\mathbf{x_i} + b \leq -1 \text{ for } \mathbf{xi} \text{ having the class } -1$$

Let us assume a function y_i such that

$$y_i = 1 \text{ for positive samples}$$
$$y_i = -1 \text{ for negative samples}$$

Therefore, $y_i\,(\,\mathbf{w}\cdot\mathbf{x_i} + b\,) \geq 1$ and $y_i\,(\,\mathbf{w}\cdot\mathbf{x_i} + b\,) \leq -1$. Using both the hyper-plane's equation, the margin calculated is

$$m = \frac{2}{||w||}$$

Maximizing the margin is known to be the same as minimizing the norm of $\boldsymbol{w}$.

$$minimize\ \frac{1}{2}\ ||w||^2$$

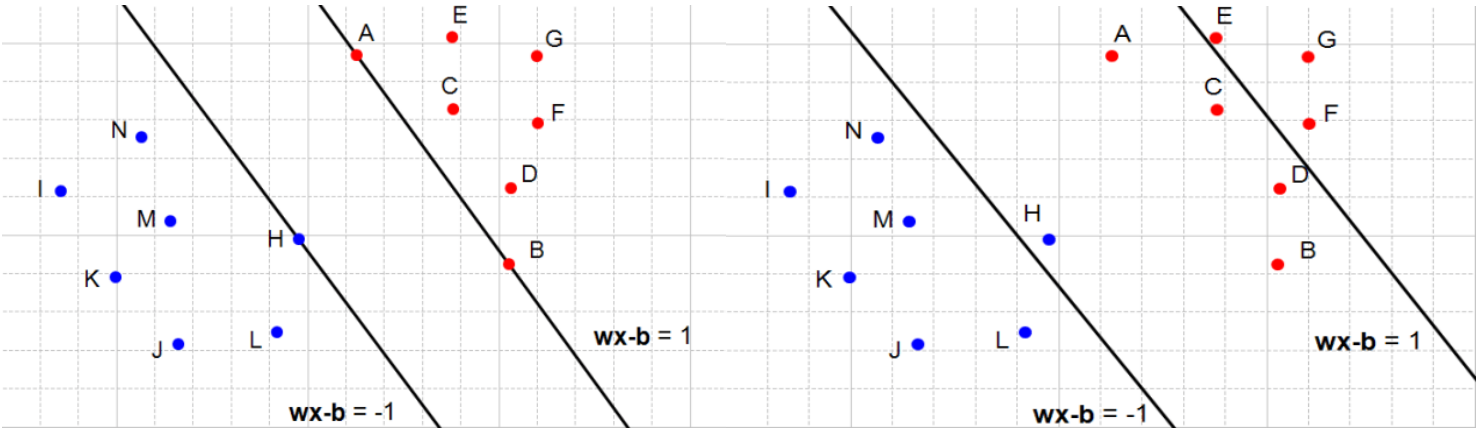


FIGURE 7 –
(LEFT) TWO HYPER-PLANES SATISFYING THE CONSTRAINTS.
(RIGHT) TWO HYPER-PLANES NOT SATISFYING THE CONSTRAINTS

If we are going to find the minimum of a function with constraints then we are going to have to use Lagrange multipliers that would give us a new expression which we can maximize or minimize without thinking of the constraints anymore.

$$L = \frac{1}{2}\ ||\mathbf{w}||^2 - \sum \propto_i\ [y_i(\boldsymbol{w} \cdot \boldsymbol{xi}\ +\ \boldsymbol{b}) - 1] \qquad -(1)$$

For the points that don't lie on the hyper-planes *h1* and *h2*, the value of $\propto$ is going to be 0. The partial derivatives are

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum \propto_i\ y_i x_i = 0$$

$$\mathbf{w} = \sum \propto_i\ y_i\ .\ x_i \qquad -(2)$$

$$\frac{\partial L}{\partial b} = -\sum \propto_i\ y_i = 0$$

$$\sum \propto_i\ y_i = 0 \qquad -(3)$$

Using equations (1), (2) and (3), we get

$$L = \sum \propto_i - \frac{1}{2}\sum_{i,j} \propto_i \propto_j\ y_i y_j x_i x_j \qquad -(4)$$

The optimization depends on the dot product of pairs in the samples, ie, $x_i.\ x_j$. Thus, if the decision rules follows the behavior-

$$\sum \propto_i\ y_i\ x_i\ .\ \boldsymbol{u}\ + b\ \geq 0 \text{ then, positive sample}$$

$$\sum \propto_i y_i x_i \,.\, \boldsymbol{u} + b < 0 \text{ then, negative sample}$$

This never gets stuck to the local maximum and keeps on evaluating its options for a better fit. Let us define a function $\varphi(\mathbf{x})$ which transforms the space to another perspective space. Support Vector Machine can also be generalized to non-linear cases. If a kernel function K exists such that $K(x_i, x_j) = \varphi(x_i).\varphi(x_j)$ then all the data can be transformed into a high dimensional space and linear algorithms can be used without knowing the transformation φ explicitily.

In the case of Online Signature Verification which is a binary classification problem (ie, forgery or genuine), radial basis function kernel (rbf) is used. This can be written as

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \gamma > 0.$$

The RBF Kernel defines a function space that is a lot larger than that of a linear kernel and polynomial kernel.

Observations, Results and Conclusion

Mentioned below are two sets of results, results obtained from the SUSIG Online Dataset and results obtained from the dataset collected for an Android Application. The SUSIG Online Dataset consisted of 20 genuine and 10 forgery signature of 115 people. Therefore, each user has 30 signatures associated with him, irrespective of genuine or forgery. In the case of the dataset collected from the Android Application, a dataset of 5 people were collected. Each person has 10 genuine signatures and 5 forgery signatures.

- ***Confusion Matrices***

  SUSIG Online Dataset (10 genuine and 10 forgery signatures)

  Below is the confusion matrix of one random user out of 115 people. Since each user had a separate classifier implemented, 115 confusion matrices were generated. Shown below is the variation of the confusion matrix with respect to the training and testing data statistics.

  In cases where either 20%, 30% or 40% of the data is used for testing and 80%, 70% and 60% of the data is used for training respectively, the below confusion matrices were generated.

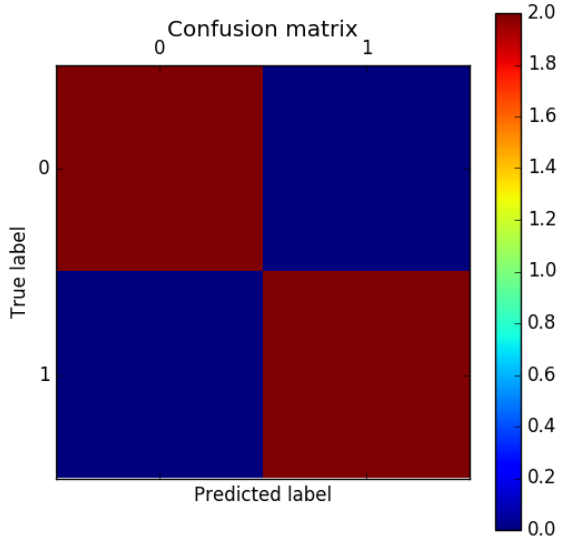


FIGURE 8

20% testing – 80% training

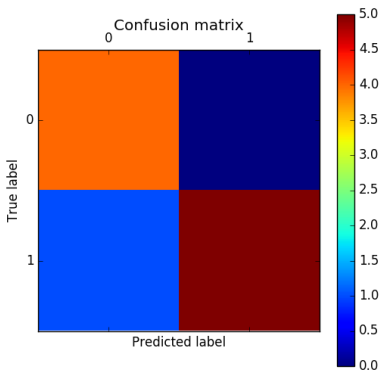
Matrix: $\begin{matrix} 2 & 0 \\ 0 & 2 \end{matrix}$

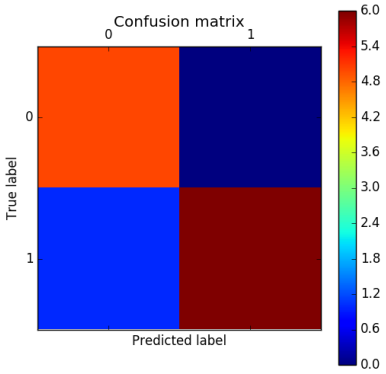
30% testing – 70% training

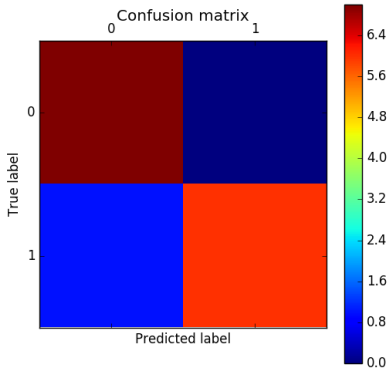
Matrix: $\begin{matrix} 3 & 0 \\ 0 & 3 \end{matrix}$

40% testing – 60% training

Matrix: $\begin{matrix} 4 & 0 \\ 0 & 5 \end{matrix}$







Left Diagram

In the case where 50% of the data was used for testing and 50% for training-

50% testing – 50% training

Matrix: $\begin{matrix} 4 & 0 \\ 1 & 5 \end{matrix}$

Center Diagram

In the case where 60% of the data was used for testing and 40% for training-

60% testing – 40% training

Matrix: $\begin{matrix} 5 & 0 \\ 1 & 6 \end{matrix}$

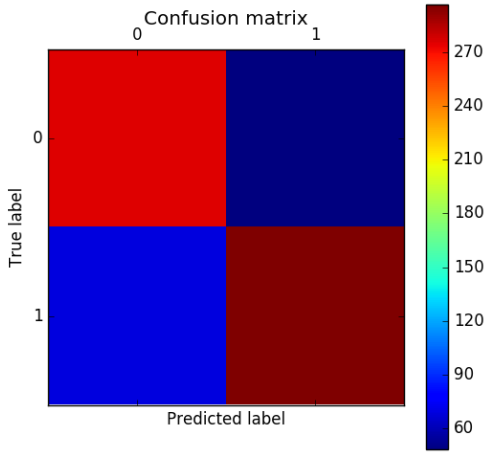
Right Diagram

In the case where 70% of the data was used for testing and 30% for training-

70% testing – 30% training

Matrix: $\begin{matrix} 7 & 0 \\ 1 & 6 \end{matrix}$

Cumulative Results

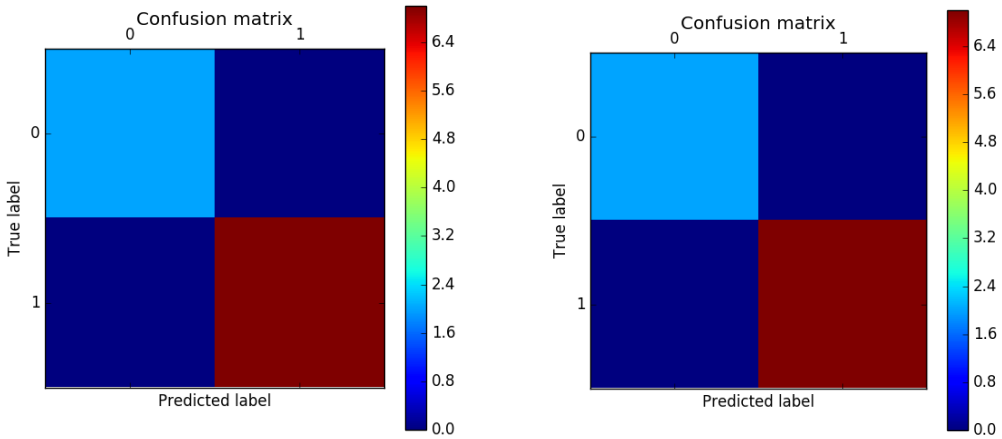


Total number of users = 115
Total number of testing files = 690
Accuracy = (573+117)/690 = 0.8304
Misclassification Rate = 0.169
True Positive Rate = 0.8147
False Positive Rate = 0.1481
Specificity = 0.8518
Precision = 0.8608
Prevalence = 0.5217

Matrix: $\begin{matrix} 276 & 48 \\ 69 & 297 \end{matrix}$

Dataset collected by Android Application (10 genuine and 5 forgery signatures)

Below is the confusion matrix of one random user out of 5 people. The dataset was accumulated using 2 Android devices. Since each user had a separate classifier implemented, 5 confusion matrices were generated. Shown below is the variation of the confusion matrix with respect to the training and testing data statistics.



Left Diagram

In the case where 30% of the data was used for testing and 70% for training-

30% testing – 70% training

Matrix: $\begin{matrix} 1 & 0 \\ 0 & 4 \end{matrix}$

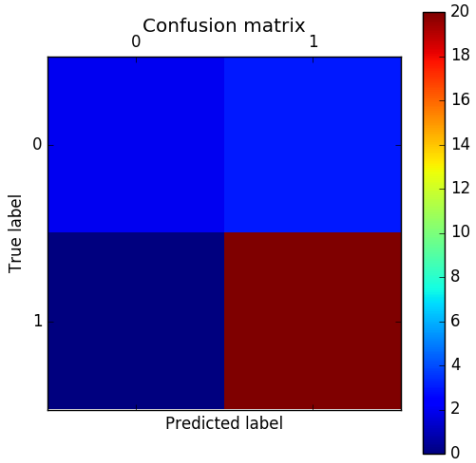
Right Diagram

In the case where 60% of the data was used for testing and 40% for training-

60% testing – 40% training

Matrix: $\begin{matrix} 2 & 0 \\ 0 & 7 \end{matrix}$

Cumulative Results



Total number of users = 5
Total number of testing files = 25
Accuracy = (2+20)/25 = 0.88
Misclassification Rate = 0.12
True Positive Rate = 1.0
False Positive Rate = 0.6
Specificity = 0.4
Precision = 0.869
Prevalence = 0.8

Matrix: $\begin{matrix} 2 & 3 \\ 0 & 20 \end{matrix}$

References

[1] H. Feng and C. C. Wah, “Online signature verification using a new extreme points warping technique,” Pattern Recognit. Lett., vol. 24, no. 16, pp. 2943–2951, 2003.

[2] A. Kholmatov and B. Yanikoglu, “SUSIG: An on-line signature data-base, associated protocols and benchmark results,” Pattern Anal. Appl.,vol. 12, no. 3, pp. 227–236, 2008.

[3] M. Faundez-Zanuy, “On-line signature recognition based on VQ-DTW, ”Pattern Recognit., vol. 40, no. 3, pp. 981–992, 2007.

[4] E. Maiorana, P. Campisi, and A. Neri, “Template protection for dynamictime warping based biometric signature authentication,” in Proc. 16th Int. Conf. Digital Signal Process., Jul. 2009, pp. 1–6.

[5] J. Fierrez-Aguilar, L. Nanni, J. Lopez-Peñalba, J. Ortega-Garcia, and D. Maltoni, “An on-line signature verification system based on fusion of local and global information,” in Audio- and Video-Based Biometric Person Authentication (Lecture Notes in Computer Science), vol. 3546. Berlin, Germany: Springer-Verlag, 2005, pp. 627–656.