Abstract—This paper presents a method of creating Receiver Operating Characteristic (ROC) curves from the nonparametric, \(k\)-nearest-neighbor classification procedure and illustrates the method on keystroke biometric authentication data. The long-text-input keystroke biometric system developed at Pace University has the ability to identify with a high degree of accuracy the typing characteristics that are unique to each individual. The system consists of three components: a java applet that collects raw keystroke data over the internet, a feature extractor, and a pattern classifier. The performance of the keystroke authentication baseline system was also increased in this study, first by modifying the feature extraction component and second by using a higher-level, \(k\)-nearest-neighbor procedure, and these improvements halved the error rate from 4.4\% to 2.0\%.

I. INTRODUCTION

The long-text-input keystroke biometric system developed at Pace University for internet applications has been previously described. The early work was on user identification [1]. More recently, preliminary work was conducted on user authentication [2]. This paper describes further development and evaluation of the authentication system and focuses on a method of creating Receiver Operating Characteristic (ROC) curves from the nonparametric nearest-neighbor classification procedure.

Internet applications of the keystroke biometric are of increasing importance as the population of application participants continues to grow. An example authentication application is verifying the identity of students taking online quizzes or tests, an application becoming more important with the student population of online classes increasing and instructors becoming concerned about evaluation security and academic integrity. Similarly, in a business setting employees can be required to take online examinations in their training programs and the companies would like the exam-takers authenticated.

The overall efficiency of a biometric system is best illustrated graphically through the use of an ROC curve [3]. The ROC curve delineates the usefulness of the authentication system by referencing accurate identifications or misidentifications. For parametric procedures using probability distributions, the established procedure for obtaining ROC curves is to vary a threshold to obtain the tradeoff between the False Accept Rate (FAR) and the False Reject Rate (FRR). However, when using a non-parametric technique, such as the nearest neighbor procedure used in this system, there are no established procedures for obtaining ROC curves. This paper describes a unique method of obtaining ROC curves from the nearest neighbor classification procedure and illustrates the method by applying it to the keystroke classification output data. A companion paper presents a more theoretical approach to obtaining ROC curves for multivariate biometric verification models [4].

This paper also describes how the authentication system performance was increased by modifying the feature extraction component and by using a higher-level, \(k\)-nearest-neighbor procedure to halve the error rate from 4.4\% to 2.0\%.

The sections of this paper are organized as follows. Section II describes the basic system which consists of a data-collecting Java applet, a feature extraction component, and a pattern classifier. Section III presents the method of obtaining ROC curves from the nearest neighbor classification procedure. Section IV describes the experimental results. Section V discusses the ROC curve method and results, and section VI presents the conclusions and future work.

II. KEYS TROKE BIOMETRIC SYSTEM

A. Raw Keystroke Data Capture

A Java applet collects raw keystroke data samples over the internet, and for each key struck, records the time (in milliseconds) the key was pressed and the time the key was released [1, 2].

B. Feature Extraction

Because the feature extraction component has been described in detail previously [1, 2], it will only be summarized here. The system extracts a feature vector from the information in a raw data file. The features are statistical in nature and designed to characterize an individual’s keystroke dynamics over writing samples of 200 or more characters. Most of the features are averages and standard deviations of key press duration times and of transition times between keystroke pairs. While key press duration and transition times are typically used as features in keystroke biometric studies, our use of the statistical measures of means and standard deviations of the key presses and transitions is uncommon and only practical for long text input.

A granularity of the duration and transition features is obtained through corresponding duration and transition hierarchy trees. Because the computation of means and standard deviations requires special handling when there are few samples, we also use these hierarchy trees as a “fallback” procedure similar to those used in natural language processing. Three hierarchy models have been

---


Abstract—This paper presents a method of creating Receiver Operating Characteristic (ROC) curves from the non-parametric, \(k\)-nearest-neighbor classification procedure and illustrates the method on keystroke biometric authentication data. The long-text-input keystroke biometric system developed at Pace University has the ability to identify with a high degree of accuracy the typing characteristics that are unique to each individual. The system consists of three components: a java applet that collects raw keystroke data over the internet, a feature extractor, and a pattern classifier. The performance of the keystroke authentication baseline system was also increased in this study, first by modifying the feature extraction component and second by using a higher-level, \(k\)-nearest-neighbor procedure, and these improvements halved the error rate from 4.4\% to 2.0\%.

II. KEYS TROKE BIOMETRIC SYSTEM

A. Raw Keystroke Data Capture

A Java applet collects raw keystroke data samples over the internet, and for each key struck, records the time (in milliseconds) the key was pressed and the time the key was released [1, 2].

B. Feature Extraction

Because the feature extraction component has been described in detail previously [1, 2], it will only be summarized here. The system extracts a feature vector from the information in a raw data file. The features are statistical in nature and designed to characterize an individual’s keystroke dynamics over writing samples of 200 or more characters. Most of the features are averages and standard deviations of key press duration times and of transition times between keystroke pairs. While key press duration and transition times are typically used as features in keystroke biometric studies, our use of the statistical measures of means and standard deviations of the key presses and transitions is uncommon and only practical for long text input.

A granularity of the duration and transition features is obtained through corresponding duration and transition hierarchy trees. Because the computation of means and standard deviations requires special handling when there are few samples, we also use these hierarchy trees as a “fallback” procedure similar to those used in natural language processing. Three hierarchy models have been
investigated, the first was developed in the early work [1] and the other two more recently [2]. The first, termed the “linguistic” model (although it is primarily a frequency of use model), groups the alphabet keys into the vowels and into groups of consonants according to their frequency of use; and the other keys into categories like punctuation, numbers, etc. The second, termed the “touch-type” model, groups the keys of the keyboard based on the fingers used to strike keys by touch typists – for example, left index finger, left middle finger, left ring finger, etc. The third, termed the “statistical” model, groups keys having similar key-strike statistics. These first two models yielded comparable accuracy with the linguistic model slightly better than the touch-type model; the statistical model was significantly weaker than the other two [2].

Two preprocessing steps are performed on the feature measurements, outlier removal and feature standardization [1, 2]. Outlier removal consists of removing long duration or transition times (greater than a threshold in terms of standard deviations). Outlier removal is particularly important for these features because a keyboard user could pause for a phone call, for a sip of coffee, or for numerous other reasons, and the resulting outliers could skew the feature measurements. After performing outlier removal, the measurements are standardized into the range 0-1 to give each measurement roughly equal weight.

C. Classification

Because user authentication was only recently explored in the keystroke work [2], this component of the system will be described in more detail. A vector-difference model, particularly effective for multidimensional feature-space problems, was chosen. This model transforms a multi-class (polychotomy) problem into a two-class (dichotomy) problem (Fig. 1) [5-8]. The resulting two classes are positive (“you are verified”) and negative (“you are not verified”). Other terms have been used for these two classes: same person versus different people, within-class versus between-class, and intra-class versus inter-class. We will use within-class versus between-class.

In the feature distance space we use the nearest neighbor classifier with Euclidean distance to compare a feature vector distance against those in the training (enrollment) set. The training sample having the smallest Euclidean distance to the test sample is identified, and the test sample assigned as being within-class or between-class according to the truth of that training sample. We are using a non-parametric technique applied on a multidimensional problem.

III. ROC Curve Derivation

ROC curves were obtained in this study by considering the top k nearest neighbors. Two procedures for obtaining ROC curves were created – an unweighted m-match, k-nearest-neighbor (m-kNN) procedure and a weighted m-match, k-nearest-neighbor (wm-kNN) procedure.

A. Unweighted M-match k-nearest neighbor (m-kNN)

For each Q (questioned) test sample, the m-kNN procedure examines the top k nearest-neighbor outputs and counts the number of within-class matches (Fig. 2).
If the number of within-class matches is greater or equal to a threshold \( m \), the user is authenticated (3), i.e. accepted as being \( w \) (within-class), and otherwise rejected as \( b \) (between-class).

\[
c(Q) = \begin{cases} 
  w & \text{if } \sum_{i=1}^{k} w_i(Q) \geq m \\
  b & \text{otherwise}
\end{cases} \tag{3}
\]

The ROC curve is obtained from (3) by letting \( m \) vary from 0 to \( k \) and calculating the FAR and FRR in each case. For \( m = 0 \), we authenticate a user (decide within-class) if 0 or more of the \( k \) choices are \( w \) (within-class), and clearly all users are accepted in this case, yielding FRR = 0.0 (0%) and FAR = 1.0 (100%). For \( m = 1 \), we authenticate a user if 1 or more of the \( k \) choices is \( w \) and obtain the FAR and FRR. We do the same for 2 or more and continue in this manner until all \( k \) of the \( k \) choices must be \( w \), obtaining FAR and FRR in each case. Now, plotting the \( k+1 \) (FRR, FAR) pairs yields an ROC curve (FRR on the x-axis and FAR on the y-axis). For the last point, when we require all \( k \) outputs to be \( w \) for authentication, FRR is usually large and FAR small.

B. Weighted m-match, k-nearest neighbor (wm-kNN)

It seems reasonable to weight higher choices more heavily than lower ones because the first choice should clearly be more valuable than the second, the second more valuable than the third, etc. We use a linear rank weighting, assigning the first choice a weight of \( k \), second a weight of \( k-1 \), ..., and the \( k^{th} \) choice a weight of 1. The maximum score when all choices are within-class is \( k + (k-1) + ... + 1 = \frac{k(k+1)}{2} \), and the minimum score is 0. Now, consider that we authenticate a user if the weighted-within-class choices are greater or equal to \( m \), where \( m \) varies from 0 to \( \frac{k(k+1)}{2} \), and compute the (FRR, FAR) pairs for each \( m \) to obtain an ROC curve.

IV. EXPERIMENTAL RESULTS

A. Experimental design

This study continued to use data collected in a previous study that employed two independent variables – keyboard type and input mode – to determine their effect on identification performance [1]. The keyboard types were desktop and laptop PC keyboards. The input modes were a copy task and free (arbitrary) text input. By varying the independent variables, the study determined the distinctiveness of keystroke patterns when training and testing on long-text input under ideal conditions (same input mode and keyboard type for enrollment and testing) and under non-ideal conditions (different input mode, different type of keyboard, or both, for enrollment and testing).

A more recent study developed an authentication system and used the earlier collected data in preliminary authentication experiments [2]. This study continued the development of the authentication system and performed additional experiments. The focus here was only on the ideal experimental conditions – that is, enrollment and testing under the same conditions. There are four ideal experimental conditions – desktop-copy (DeskCopy), laptop-copy (LapCopy), desktop-free (DeskFree), and laptop-free (LapFree).

B. Improved Baseline System Performance

Table I, reproduced from [2], presents preliminary authentication performance results under the four ideal conditions. Using the 36-subject data, the results were obtained by training on 18 subjects and testing on the other 18. However, rather than using the “linguistic model” feature-extraction hierarchy trees as reported in [2], these results actually came from using the “touch-type model.”

<table>
<thead>
<tr>
<th>Condition</th>
<th>Intra-Inter Class Sizes</th>
<th>FRR</th>
<th>FAR</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train</td>
<td>Test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeskCopy</td>
<td>180-3825</td>
<td>180-3825</td>
<td>11.1%</td>
<td>6.0%</td>
</tr>
<tr>
<td>LapCopy</td>
<td>180-3825</td>
<td>180-3825</td>
<td>7.8%</td>
<td>4.4%</td>
</tr>
<tr>
<td>DeskFree</td>
<td>171-3570</td>
<td>176-3740</td>
<td>28.4%</td>
<td>1.4%</td>
</tr>
<tr>
<td>LapFree</td>
<td>180-3825</td>
<td>180-3825</td>
<td>15.6%</td>
<td>3.7%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>15.7%</strong></td>
<td><strong>3.9%</strong></td>
<td><strong>95.6%</strong></td>
<td></td>
</tr>
</tbody>
</table>

Table II presents improved results obtained using the “linguistic model,” again training on 18 subjects and testing on a different 18. Average performance increased from 95.6% to 97.1%.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Intra-Inter Class Sizes</th>
<th>FRR</th>
<th>FAR</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train</td>
<td>Test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeskCopy</td>
<td>180-3825</td>
<td>180-3825</td>
<td>2.8%</td>
<td>2.1%</td>
</tr>
<tr>
<td>LapCopy</td>
<td>180-3825</td>
<td>180-3825</td>
<td>3.3%</td>
<td>4.0%</td>
</tr>
<tr>
<td>DeskFree</td>
<td>176-3576</td>
<td>165-3740</td>
<td>21.0%</td>
<td>1.1%</td>
</tr>
</tbody>
</table>
C. \textit{k-Nearest-Neighbor Performance}

The results presented above used the nearest-neighbor (i.e., 1-nearest-neighbor) procedure for classification. Using a greater number of neighbors was explored. Fig. 3 graphs the average performance of the 1, 3, 5, 7, and 9-nearest-neighbor procedures. Here, we see a significant improvement in going from 1 to 3 neighbors and slight improvements in going from 3 to 5 and to higher nearest neighbors. Table III presents the 9-nearest-neighbor results for the four ideal conditions.

Combining the improved features and the improved 9-nearest-neighbor results, the overall error rate was halved from the previously reported results, going from 4.4\% to 2.0\%.

D. \textit{ROC Curves}

For simplicity, in this section we present ROC curves only for the DeskCopy and LapFree ideal conditions. Figs. 4 and 5 present the ROC curves for the unweighted \textit{m-kNN} and the linear-weighted \textit{wm-kNN} procedures for \(k = 10, 15, \) and 20. To eliminate a stair-casing effect in the graphs, only the lowest FAR value is shown for data points having identical FRR values. As \(k \) increases, more data points are generated and the curves tend to improve (FAR decreases). For the DeskCopy condition, the Equal Error Rate (EER) can be approximated as 2.7\% from these curves. For the LapFree condition, setting the operating point at the ERR does not seem appropriate.

Plots of FAR and FRR versus the threshold \( m \) can also be obtained. Figs. 6-8 show the plots of FAR and FRR versus the threshold \( m \) for the unweighted \textit{m-kNN} procedure for \( k = 10, 15, \) and 20 for the DeskCopy and LapFree conditions. Having many more between-class samples than within-class samples (by roughly a factor of 20) likely accounts for the FAR curves being smoother than the FRR curves.
worse for the LapFree. The method appears better for the DeskCopy condition and weighted and unweighted methods since the weighted data points. However, there is no clear winner between the unweighted, the weighted method provides more coordinate LapFree conditions, respectively. Compared to the ROC curves (Fig. 7).

Figs. 9 and 10 compare the unweighted and weighted ROC curves ($k = 10$, 15, and 20) for the DeskCopy and LapFree conditions, respectively. Compared to the unweighted, the weighted method provides more coordinate data points. However, there is no clear winner between the weighted and unweighted methods since the weighted method appears better for the DeskCopy condition and worse for the LapFree.

![ROC curves for DeskCopy and LapFree](image)

**V. DISCUSSION**

We believe there is considerable value in obtaining ROC curves from the non-parametric nearest neighbor procedure. The primary value of ROC curves, of course, is being able to choose an appropriate operating point to provide the desired tradeoff between security (low FAR) and convenience (low FRR).

Obtaining ROC curves for this keystroke $k$-nearest-neighbor classifier resulted in some interesting findings. The $k$-nearest-neighbor procedure typically makes its classification decision by taking the majority of an odd number of the $k$ nearest neighbors to the questioned (unknown) sample, but this method yielded high security at the sacrifice of low convenience in the keystroke experiments. For example, the majority of 8 out of the nearest 15 neighbors yields an FAR of 1.4 and an FRR of 6.7 in the DeskCopy experiment (Fig. 7). The ROC curves, however, provide a full range of possible tradeoffs between FAR and FRR, and in this case it is interesting that an EER of approximately 2.8 is achieved by choosing a seemingly low threshold of 3 out of the nearest 15 neighbors.

Although we appreciate the value of ROC curves in this work, we agree with [9] that ROC curves should not be used exclusively. Most importantly, the ROC curve does not take into account the size of the database, and performance is known to decrease as the database is increased. Furthermore, the ROC curve cannot reflect the cost of classification, the failure-to-enroll rate, the template size, the recognition time, and the psychological factors of comfort, convenience, and acceptability.

Finally, although it is not clear what their value might be, it is even possible to compute approximate “equivalent” underlying normal probability distributions and thus their corresponding smooth ROC curves from non-parametric classification data [4].

**VI. CONCLUSIONS AND FUTURE WORK**

The main contribution here is the method of obtaining ROC curves from the $k$-nearest-neighbor classifier and the application of the method to experimental data from a keystroke biometric system. And, at least for the keystroke data, it was found that the ROC curves produced by this method tend to improve as $k$ increases – that is, as more neighbors are included in the computation.

In this study we have worked only with what we call the “baseline” 36-subject keystroke data in order to establish the procedures. We have data from over 100 subjects for most of the four ideal conditions and we plan to run larger experiments to determine the degree of degradation in performance as the number of subjects increases.

We could also use these keystroke data to explore some of the more sophisticated multivariate biometric models described in [4]. However, because we use over 200 feature measurements, we are considering the use of techniques such as principal components to first reduce the number of features to a more manageable number.
REFERENCES


