Clusters and Markers for Keystroke Typing Rhythms

Shing-hon Lau
Machine Learning Department
Carnegie Mellon University

Roy Maxion
Computer Science Department
Carnegie Mellon University

Abstract

Background. People's blood comes in four types: A, B, AB and O. The markers for these blood types are the presence or absence of specific antigens. If people's typing rhythms – the unique pattern of someone's typing – can be similarly grouped into a small number of types, it could have forensic importance, allowing insider investigators to rule out a substantial fraction of suspects, just as Type-A blood rules out 60% of the population.
Aim. We aim to determine whether typing rhythms can be grouped into a small number (e.g., 3-10) of characteristic groups, and to find a marker that places a typist squarely into one group, as antigens do in blood typing.
Method. Data were 50 repetitions of a password (.tie5Roanl) from 51 typists. Agglomerative clustering elicited groupings in the data. Sparse logistic regression discovered the distinguishing characteristics of groups. A support vector machine identified specific markers.
Results. Three major groupings, or rhythm *types*, were identified, along with one singleton outlier. Preliminary work focused mainly on just one of these groups, whose members turned out to comprise all women. A Chi-Square test of independence determined that this was unlikely to have been a chance event ($\chi^2(df = 2, N = 50) = 13.1714, p < 0.005$). The singleton subject, an egregious outlier, suffered from a neurological disorder.
Conclusions. Typists can be grouped into a small number of types, as is done in blood typing. Markers can identify an individual as a member of a distinct type.

1 Introduction

Keystroke dynamics is the study of individual's typing rhythms, usually for the purpose of discriminating amongst different users. Most keystroke dynamics research has focused on the development of new classification algorithms to improve discrimination amongst users. Typically, researchers propose a new algorithm, gather a keystroke dataset, and evaluate the algorithm on this dataset. However, regardless of whether the results are better or worse than existing results in the literature, there is little understanding of *why* the algorithm performed better or worse. One reason for this dearth of understanding is the paucity of knowledge about the properties of keystroke data itself.

The field of keystroke dynamics is largely predicated on the fact that there is structure in the data; specifically, there is an assumption that different users generate different typing data. The differences can be used to distinguish between users. But this is not the only structure that exists. Users have various physiological and behavioral traits which may affect typing (e.g., gender, handedness, touch typist vs. hunt-and-peck, etc). Users with similar traits should produce similar data; such similarity may define a new aspect of structure in keystroke data.

Almost no research efforts have focused on understanding this additional structure. Bereft of this understanding, developers of new classification algorithms lack a guiding framework to improve classification accuracy. Instead of systematically exploiting this structure to improve accuracy, developers take a trial-and-error approach. This approach has not met with much success; most new algorithms are not markedly better than the simple algorithms proposed decades ago [8].

Our goal is to search for and quantify this additional structure in keystroke dynamics. We intend to establish a framework permitting future researchers to exploit this structure when developing new classification algorithms, whether for forensic, authentication, medical, or other purposes. We search a keystroke dataset for *clusters* – sub-groups of users that are distinct from the rest. We show that the clustered users share similar typing characteristics, which can be refined into *markers* for each sub-group. Markers are properties of a user's keystroke data that reliably identify the user as a member (or non-member) of a particular cluster, just as certain antigens classify a person as having a particular blood type.

2 Problem and approach

Our primary research question is: do people's typing rhythms fall into one of a few groups, or types? Ancillary questions ask whether these groups can be distinguished on the basis of their characteristic features, whether markers can be found that uniquely assign a user to a specific group, and whether any demographic traits are associated with the groupings, or types. Our approach comprises the following steps:

- **Find groupings in the data.** Groupings, or clusters, are found using agglomerative cluster analysis.
- **Discover distinguishing features.** Characteristics (features) of the data that distinguish members of one cluster from all other clusters or subjects are found with a sparse logistic regression classifier. This is an interpretable classifier which pinpoints specific characteristics unique to the users in a cluster.
- **Identify markers.** A Support Vector Machine (SVM) is used to create a table that rank-orders subjects by their median hold and latency keystroke timings. Users in the same cluster are ranked consecutively at the top of this table. A user is marked as a member of a cluster only if his rank lies below a particular threshold.
- **Associate clusters with demographic data.** User traits associated with clusters are found by matching cluster members against demographic information.

3 Related work

There are over 400 papers on keystroke dynamics (sometimes called keystroke biometrics or behavioral biometrics). None that we know of are relevant to grouping types of typing rhythms, but we can provide an overview of the field, as well as some pointed details, in the following paragraph. All of the cited survey papers generally treat keystroke dynamics as an authentication technique, but none unite the field's results into a pattern or theory of operation.

Peacock and his colleagues [11] provide a now somewhat-dated overview of the field, but one that is particularly accessible. Yampolskiy and Govindaraju [15] treat keystroke dynamics in the broader context of behavioral biometrics. Shanmugapriya and Padmavathi [13] give a short review of methods and metrics in keystroke dynamics. Karnan and his colleagues [4] give an overview of features and feature-extraction methods for keystroke dynamics, as well as a range of classification techniques. Banerjee and Woodard [1] provide a wide-ranging survey of the field, including the psychology of keystroke dynamics, data acquisition and environmental issues, and the usual list of technical approaches. Teh and his colleagues [14] provide a longer, more recent and broader-coverage review of the field, examining most of the same material as their predecessors, but including a longer and more recent list of cited papers.

Killourhy and Maxion have examined the influence of a variety of factors on the classification accuracy of keystroke dynamics classifiers, including the resolution of the clock used to perform keystroke timing [7], the definition of a successful login [9], and myriad other factors [6]. Their work parallels our present work. Whereas they focused on identifying and quantifying factors that affect classifier accuracy, we focus on finding identifying and quantifying structure in the keystroke data itself.

4 Data and data collection

We used a publicly-available dataset[1] whose key aspects are summarized below. Details of the experimental apparatus and instrumentation are given in [9]. Details of subject population, stimulus selection and experimental procedures are given in [7]. Human-subject trials were cleared by the CMU Institutional Review Board.

Subjects. 51 volunteers (30 male, 21 female), from a university population, contributed typing samples.

Apparatus. A Windows application on a PC (running the XP OS) prompted subjects to type the password. Typographical errors were discarded, and re-prompted, resulting in perfectly-typed repetitions of the password. Keystroke timings, taken with customized and calibrated hardware, were accurate to within $\pm$ 200 microseconds.

Stimuli. A 10-character string was used: .tie5Roanl followed by <return>.

Procedure. Subjects typed the 10-character string 50 times in each of 8 sessions, with at least one day between sessions. Total number of password repetitions was 400.

Demographic survey. Subjects provided demographic data, e.g., gender, age group, dominant hand, etc.

Data features. The data consist of *hold times* (the duration between pressing and releasing a key) and *latency times* (the durations between the release of a key and the depression of the next key). The hold and latency times are collectively referred to as *features*. Each repetition is treated as a *feature vector*, which consists of 11 hold times and 10 latency times.

Data set. The original data set [8] contained eight 50-repetition sessions for each of the 51 subjects. In the present work we use only the data from the eighth of 8 sessions, because that session is most representative of a subject's normal, practiced typing.

5 Step 1: Grouping the data

The first step in looking for groups in data is to cluster the data. This itself is a two-step process. First, the data need to be preprocessed into a form suitable for input to the clustering algorithm; then the clustering algorithm is run. We describe the preprocessing step first, to give readers some intuition about the form of the data; then we describe the clustering routine.

5.1 Method

We use the agnes clustering algorithm to find the groups [5]. Before applying the clustering algorithm, we preprocess each subject's multidimensional data so that they are represented as a single median vector.

Preprocessing. Before we can use a clustering algorithm, the input data must be preprocessed so that each of the 51 subjects is summarized in a single entity – a single vector. As previously mentioned, we have 50 repetitions of the password from each subject. Since we wish to consider each subject as a single item in the clustering algorithm, we summarize each subject by a single vector. To compress a subject into a single vector, we start by taking the median value, over all 50 repetitions, for each feature. This results in 11 median hold times and 10 median latency times. When concatenated, these 21 median times constitute a *median vector*. This process is repeated for each subject, resulting in 51 median vectors, one per subject. These 51 median vectors are then fed into the agnes algorithm.

Agnes clustering. Traditional clustering algorithms (e.g., *k*-means [3]) take some number of items to be clustered as input, and then output an assignment of each item to a single cluster. In our case, since we are interested in clustering subjects, the *k*-means algorithm will assign each subject to a single cluster. One major downside to these traditional clustering algorithms is that they often require the researcher to state the targeted number of clusters to be found (e.g., the k in k-means). When the number of clusters is unknown, researchers must use heuristic approaches to estimate the number of clusters that are present in the data.

Hierarchical clustering algorithms differ in that they return a hierarchy of clusters. Each subject will be a member of multiple hierarchical clusters. For example, a subject may be classified as a left-handed, female typist, as well as a left-handed typist, as well as a typist. Each successive cluster is more encompassing than its predecessor. Such hierarchical clusterings are often depicted as dendrograms, as shown in Figure 1. Unlike traditional algorithms, hierarchical clustering algorithms do not require the researcher to provide the targeted number of clusters as input; the clusters are derived automatically.

In this work, we use the agglomerative nesting clustering algorithm called Agnes [5]. This algorithm is an example of a "bottom-up" clustering, where each item is initially assigned to its own cluster. During iterations of the algorithm, clusters are merged together, until only a single, high-level cluster remains; this final cluster contains all items in the data set. More specifically, the algorithm is initialized by assigning each item to its own cluster. With each iteration of the algorithm, the two most similar clusters are merged. Similarity between two clusters is defined as the average Manhattan distance between all possible pairwise combinations of items in the two clusters. The algorithm iterates until all elements belong to the same, high-level cluster.

5.2 Results

Figure 1 shows the output of the agnes clustering algorithm, represented as a dendrogram, when applied to the 51 aforementioned median vectors. Nodes are labeled by subject number; red nodes are female.

A dendrogram visualization of a clustering is always a binary tree. Each child of the tree is either a cluster with a single subject (e.g., s002 at the far left), or a cluster that contains multiple subjects (e.g., the sub-tree containing both s002 and s020). The point at which two clusters merge is called a *junction*. For example, the first common junction of s002 and s020 represents the merger of those two subjects into a single cluster. The junction at height 38 (on the y-axis) represents the merger of two large clusters; all of the subjects are included in this cluster, excepting subject s036 at the far upper right.

The dissimilarity between any two subjects or clusters is proportional to the height (on the y-axis) of the first common junction that they share. For example, s002 and s020 (both on the left side of the dendrogram) are similar to each other since their first common junction has a low height, 16. However, s002 and s043 (on the extreme left and right sides of the dendrogram) are very distinct, because their first common junction is at height 38.

Four distinct clusters are visible in Figure 1: the "left cluster" containing s002 to s040; the "right cluster" containing s017 to s043; and the "middle cluster" containing all the rest of the subjects except s036, which is a singleton cluster at the far upper right.

In this paper we focus on only two of the clusters: the one containing only subject s036, and the one we called the "left cluster". A singleton cluster like s036 was completely unexpected. The "left cluster" happened to be all female subjects, again unexpected. For these reasons, and due to space constraints, we concentrate on these two clusters in the remainder of the paper. However, the techniques used from here on will generalize to any cluster or any individual subject.

6 Step 2: Discover distinguishing features

Having identified the four clusters into which the 51 subjects were grouped, we turn to the matter of determining which typing-rhythm features best distinguish one group (or, for that matter, one subject) from everyone else. Again we use a two-step process: the first is a data preprocessing step; the second, where the work is actually done, is sparse logistic regression.

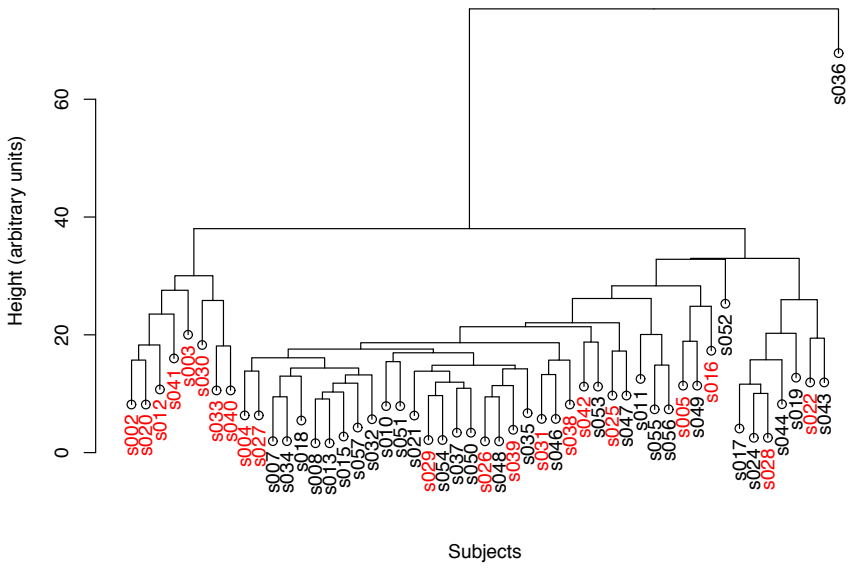

Figure 1: Dendrogram, based on session-8 median vectors, showing the hierarchical clusters of subjects produced by the agnes algorithm. Red subject numbers are female. Note the two clusters of interest: the "left cluster" of all female subjects at the far left, and the s036 singleton cluster at the far upper right.

6.1 Method

To identify the distinguishing features we use a sparse logistic regression classifier, the parameters for which are chosen via 10-fold cross validation.

Preprocessing. As a reminder, each of 51 subjects typed 50 repetitions of a password (.tie5Roanl). Each password contains ten characters plus <return>. The features are 11 hold times (the time a key is held down) and 10 latency times (the time taken to transition from one key to the next, from key-up to key-down). The typed text is preprocessed to represent the passwords as feature vectors, each of which contains the 11 hold times and 10 latency times, for a total of 21 features. The entire data set comprises 51 subjects x 50 repetitions = 2550 feature vectors. These feature vectors are provided as input to the sparse logistic regression classifier.

Sparse Logistic Regression. To identify distinguishing features that discriminate one user from others, or one cluster from the rest, we use a sparse logistic regression classifier. Since we will be directly interpreting the output of the classifier to identify distinguishing features, it is helpful to understand how the classifier operates. A typical logistic regression classifier takes n feature vectors (denoted $\mathbf{x}_i$ for $i = 1,...,n$) as input, along with a classification designation for each feature vector (denoted y_i for $i = 1,...,n$). In our case, we designate vectors belonging to subjects in the cluster with a '1' and vectors belonging to subjects not in the cluster with a '0'. The output of a logistic regression classifier is not only a classification of the feature vectors, but also a vector of weights, $\mathbf{w}$, which indicate the usefulness of each feature for the sake of accurate classification. This weight vector is the output of primary interest.

The logistic function is defined as

$$logistic(x) = \frac{1}{1+exp(-x)}$$

and the score is defined as

$$score = \mathbf{w} \cdot \mathbf{x} = \sum_{j}^{K} (\mathbf{x}_j \times \mathbf{w}_j),$$

where K is the number of features in each vector. With these definitions in mind, a logistic regression classifier assumes that the probability of a particular feature vector belonging to a subject in the cluster is:

$$P(y_i = 1|\mathbf{x}_i;\mathbf{w}) = logistic(\text{score}).$$

Of course, the classifier must actually choose $\mathbf{w}$. It does this through a maximum-likelihood approach. The likelihood function is given by:

$$\prod_i P(y = y_i|\mathbf{x}_i;\mathbf{w}).$$

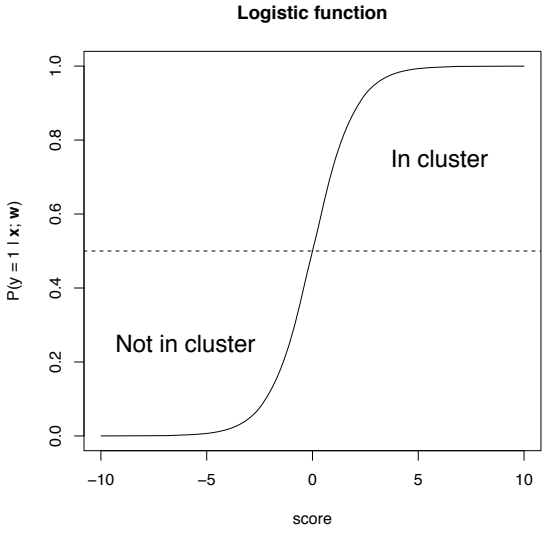

Figure 2: **Logistic function.** The logistic function converts any real-valued score into the probability that a feature vector belongs to a subject who is a member of the cluster. Vectors with a probability above 0.5 (dashed line) are classified as belonging to a subject in the cluster; points with a probability below 0.5 are classified as belonging to a subject that is not in the cluster.

Armed with the inner workings of the classifier, we can now gain an intuition about how the classifier operates. Figure 2 depicts the logistic function, which takes in any real-valued score as input and outputs a value between 0 and 1, which is interpreted as $P(y = 1|\mathbf{x};\mathbf{w})$, the probability that the feature vector $\mathbf{x}$ belongs to a subject in the cluster.

Note that the score of a feature vector is the sole determinant of whether it is labeled with a '1' or a '0'. A positive score will result in a feature vector being labeled as being from a subject who is in the cluster. Conversely, a negative score will result in a feature vector being labeled as being from a subject who is not in the cluster. The score is just the sum of terms; each term is the product between a weight and a feature value. The simple nature of the score makes it easy to interpret the information offered by the weights.

Consider any term in the score. The larger the magnitude of the term, the more influence it has on the final score and on the final label of the feature vector. If a term has a large magnitude and a positive sign, the label of the feature vector will be "pushed" towards '1'. If the term has a large magnitude and a negative sign, the label of the feature vector will be "pushed" towards '0'. A term will have a large magnitude if the weight has a large magnitude. Thus, features corresponding to high magnitude weights, regardless of sign, are important features because their terms have a large impact on the score.

At this point, it would seem that the most important features are those with the highest-magnitude weights. However, this is not necessarily the case. Keystroke features take on a wide range of values; latency times are sometimes an order of magnitude larger than hold times. Suppose that a latency time and a hold time have equal importance, and that the latency time is roughly 10 times greater than the hold time. In such a case, the hold time will receive a weight that is roughly 10 times that which was assigned to the latency time. Due to this bias, we cannot simply look for the highest-magnitude weights.

Rather, we examine the normalized weights, which are obtained by dividing each weight by the standard deviation of its associated feature. Weight normalization accounts for the bias in the weights. We can now interpret the normalized weights directly; the larger the magnitude of the normalized weight, the more important the feature.

Notice that we have been discussing a logistic regression classifier, not a *sparse* logistic regression classifier. We did this because the standard classifier is simpler to explain, and there is no difference in the interpretation of the weights generated by the two versions of the classifier. The sole difference between the two classifiers is the addition of an L1-regularization penalty to the likelihood of the sparse logistic regression classifier:

$$\prod_i P(y = y_i|\mathbf{x}_i;\mathbf{w}) - \lambda \sum_j^K |\mathbf{w}_j|,$$

L1 regularization uses a penalty term that encourages the sum of the absolute values of the parameters to be zero, making the model less complex, due to having fewer operative features. The magnitude of the penalty is controlled solely by the parameter λ. Its purpose is to cause unimportant features to have their weights set to 0, instead of a small number. The larger λ is, the more important a feature has to be to have a non-zero weight. The advantage of this penalty, and the advantage of the sparse logistic regression classifier, is that interpretation of the weights is easier since we need only focus on non-zero weights; all non-zero weights can be considered important, though their importance is still governed by the magnitude of the weight. The choice of λ is made via 10-fold cross-validation, discussed below.

For this work, we use an implementation of the sparse logistic regression classifier in the R statistical environment (version 2.15.2) [12], *glmnet* package [2].

Selection of λ via 10-fold cross-validation. As previously mentioned, the sparse logistic regression classifier has a single parameter, λ, which must be chosen. We choose this value through the use of 10-fold cross-validation. We start by selecting candidate values of λ. We use the default of the *glmnet* package, which chooses 100 candidate values of λ. The smallest candidate is

equal to 0.0001 times the smallest value of λ that would result in all weights being set to 0. Each successive candidate after the smallest is equal to 1.1 times the value of its predecessor (e.g., the second smallest candidate is 1.1 times the smallest candidate and the third smallest candidate is 1.1 times the second smallest candidate).

Next, we divide the data into 10 equally-sized partitions, called folds. For each candidate value of λ, 10 classifiers will be trained. Each classifier is trained on 9 of the 10 folds, and then tested on the remaining fold; each fold is chosen to be the test fold exactly once. The error rates of each of 10 classifiers are averaged to give an error rate for the candidate value of λ. This process is repeated for every candidate value. The chosen value of λ is the one that has the lowest average error rate.

Fitting the classifier. Prior to classification, the data are standardized so that each feature has zero mean and unit variance. The purpose of the standardization is to allow the sparse logistic regression classifier to view each feature equally.[2] Data from the cluster of interest (either s036 or "left") is labeled with a '1' and all other repetitions are labeled with a 0. Note that different classifiers are fit for the s036 and the "left" cluster. When fitting the classifier for s036, the repetitions from subjects in the "left" cluster are labeled with a 0. Similarly, when fitting the classifier for the "left" cluster, repetitions from s036 and other clusters would be labeled with a '0'. After the standardization and labeling process, cross-validation is used to select a value of λ. Finally, a classifier is fit with the chosen value of λ, using all of the data.

6.2 Results

We present results for two clusters: the s036 singleton, and the "left" cluster.

s036 Cluster. Table 1 shows the non-normalized and normalized weights produced by the sparse logistic regression classifier. For reasons previously discussed in Section 6.1, we use the magnitude of the normalized weights to determine the importance of features for discriminating between s036 and the remaining subjects.

We can see that the hold time on 'shift r', 'a', and 'n' are the most important features. There is a large gap between the magnitude of the weights on these three features and the next largest magnitude. Moreover, all three weights share a negative sign, indicating that small values of the features are indicative of s036. After the gap, the next most important feature is the latency time for 'n-l'. Its positive sign indicates that large values of that feature are indicative of s036.

In fact, these 4 features are sufficient to discriminate every single repetition of s036 from every repetition for any other user. All repetitions meeting the criteria below belong to s036. Any repetition that does not meet all 4 of these criteria belongs to some other subject.

Index	Feature	Weight	Normalized Weight
1	H.Shift.r	-137.71	-3802.45
2	H.a	-80.75	-2336.48
3	H.n	-67.38	-2178.82
4	UD.n.l	14.66	114.87
5	H.o	-2.93	-102.49
6	UD.Shift.r.o	9.96	63.89
7	UD.l.Return	9.90	55.95
8	UD.o.a	4.98	55.80
9	UD.a.n	3.93	43.91
10	UD.i.e	4.52	40.95
11	UD.period.t	2.42	14.50
12	UD.e.five	0.65	3.29
13	H.l	-0.01	-0.30
14	UD.five.Shift.r	0.05	0.23
15	H.period	0.00	0.00
16	H.t	0.00	0.00
17	H.i	0.00	0.00
18	H.e	0.00	0.00
19	H.five	0.00	0.00
20	H.Return	0.00	0.00
21	UD.t.i	0.00	0.00

Table 1: **Sparse logistic regression weights: s036.** The normalized weights (right column) with the highest magnitudes correspond to the features with the most influence on correctly separating s036 from all other subjects. "H" indicates hold time; "UD" indicates key-up to key-down interkey latency time.

1. Hold time for 'R' is less than 60 milliseconds
2. Hold time for 'a' is less than 80 milliseconds
3. Hold time for 'n' is less than 60 milliseconds
4. Latency time for 'n-l' is more than 40 milliseconds

"Left" Cluster. Table 2 shows both non-normalized and normalized weights that were produced by the sparse logistic regression classifier. For reasons previously discussed in Section 6.1, we use the magnitude of the normalized weights to determine the importance of features for discriminating between subjects in the "left" cluster and the remaining subjects.

One notable difference between the weights for the "left" cluster and the weights for the s036 cluster is that all of the "left" cluster weights are non-zero. The lack of zero weights indicates that all features are of at least some importance in discriminating between members of the "left" cluster and all other subjects.

Note that the weights associated with hold times are almost always positive, indicating that longer hold times are indicative of subjects in the "left" cluster. Weights

	Feature	Weight	Normalized Weight
1	H.i	92.45	3105.03
2	H.period	94.37	2843.08
3	H.a	89.98	2603.60
4	H.o	52.59	1842.87
5	H.l	44.29	1562.09
6	H.e	50.57	1343.99
7	H.n	25.55	826.16
8	H.Return	20.89	697.19
9	H.t	9.71	325.05
10	H.five	6.69	301.88
11	H.Shift.r	-8.80	-243.03
12	UD.t.i	7.15	79.79
13	UD.a.n	6.43	71.82
14	UD.l.Return	5.66	32.00
15	UD.i.e	-3.27	-29.57
16	UD.o.a	2.22	24.86
17	UD.Shift.r.o	-2.11	-13.54
18	UD.e.five	2.61	13.21
19	UD.period.t	0.91	5.44
20	UD.n.l	-0.65	-5.12
21	UD.five.Shift.r	-0.50	-2.13

Table 2: **Sparse logistic regression weights: "Left".** The normalized weights (right column) with the highest magnitudes correspond to the features with the most influence on correctly separating the subjects in the "left" cluster from all others. "H" indicates hold time; "UD" indicates key-up to key-down interkey latency time.

associated with latency times show no tendency to be either positive or negative, so no particular conclusions can be drawn.

7 Step 3: Identify markers

Thus far, we have accomplished two of our initial aims. We have identified clusters of subjects in our data, and we have identified features that distinguish them from the remainder of the subjects. We now turn to summarizing a large group of features into a single marker.

7.1 Method

We first introduce the concepts of average hold and latency-time rankings, and then refine these rankings further with the use of a Support Vector Machine (SVM).

Average hold and latency-time rankings. In identifying the important features for discriminating between subjects in a cluster and subjects outside a cluster, we noticed that both classifiers (one for the s036 cluster and the other for the "left" cluster) identified important hold-time and latency-time features. In an attempt to refine these features down to a marker, we determined that a rank-ordering of the subjects, from longest hold/latency times to shortest hold/latency times, might allow us to define a simple marker.

We start our discussion with a rank-ordering of subjects by their hold times. For each subject, we compute the *average hold-time ranking* as follows:

1. Choose a single hold time for a single feature.
2. For each subject, compute the median value for that hold time for that subject.
3. Rank the subjects from the largest hold time (rank 1) to the smallest (rank 51).
4. Repeat this process for each hold time, producing 11 rankings for each subject.
5. Average the 11 rankings to produce the average hold-time ranking.

A subject that always has the longest hold time would have an average rank of 1, while a subject that always has the shortest hold time would have an average rank of 51 (as we have 51 subjects). An analogous process produces the *average latency-time ranking*; the difference is that there are 10 latency times as compared to 11 hold times. Tables 3 and 4 show a portion of the average hold-time ranking and the average latency-time ranking tables.

Support Vector Machine (SVM). In an ideal world, either the average hold-time rankings or the average latency-time rankings would suffice as a marker. One would hope that all subjects inside a cluster would be consecutively listed on at least one of the two tables. In such a scenario, the marker for a cluster would be an average hold-time (or latency-time) ranking between some upper and lower limits. Unfortunately, this is not always the case. Sometimes neither table lists cluster members consecutively. In such a case, we wish to form a combined ranking that *does* create such a consecutive list.

To ensure that the results are still easily interpretable, we would like to create a combined ranking using a linear combination of the two rankings: combined ranking = A $\times$ average hold-time ranking + B $\times$ average latency-time ranking. Having decided on a linear combination, the only task left is to choose the two-element coefficient vector which is comprised of A and B. So far, our only stated goal is to have all subjects in a cluster listed consecutively when sorted by the combined ranking. To simplify this further, we can insist that all subjects inside the cluster must have a combined ranking below some threshold, while all other subjects must have a combined ranking above that threshold. That is, when using this combined ranking, we ask that subjects inside the cluster are separated from subjects outside the cluster.

There may be many coefficient vectors, however, that produce a combined ranking with the desired separation property. Therefore, we need to have some criterion as

a basis for deciding which vector to pick. A natural approach would be to select a vector that separates subjects inside and outside the cluster by the largest possible margin. The margin is the minimum distance between the combined score of any subject inside the cluster and the combined score of any subject outside the cluster. This approach, though, is still not perfect. Suppose that some pair of A and B creates a margin of 1; then the pair 10A and 10B creates a margin of 10. To get around this scaling issue, we add one final condition: the coefficient vector must have unit length.

This problem formulation is the same formulation that underlies a Support Vector Machine (SVM) using a linear kernel. In particular, an SVM will take two items as input. The first is the average hold-time ranking and average latency-time ranking for each subject. The second is a label indicating whether each subject is inside the cluster or not; subjects inside the cluster will be labeled with a '1' and subjects outside the cluster will be labeled with a '0'. The SVM will output two coefficients, A and B, that provide the largest margin of separation.

7.2 Results

Tables 3 and 4 show a portion of the average hold-time ranking and the average latency-time ranking tables, respectively. The first column in each table contains the position of each subject (2nd column) according to the average hold/latency-time ranking (3rd column). The fourth column shows the median hold/latency time in milliseconds, to provide the reader some context of the time scales at play. Some subjects omitted for brevity.

s036 cluster. Subject s036 stands out quite clearly in both the average hold-time and latency-time ranking tables. He has the lowest average hold-time ranking out of all the subjects, indicating that he has consistently short hold times overall. He also has the highest average latency-time ranking, so he has consistently long overall latency times, too. Since he is already well separated from all other subjects, there is no need to apply an SVM.

There are several possible markers that we can choose for s036. One marker is that the subject must have an average hold-time ranking above 48. Another would be that the subject must have an average latency-time ranking below 3. A third marker would be to insist that both these criteria hold simultaneously. Any of these choices of marker would cleanly separate s036 from all other subjects or clusters. Note that in actually using the marker, no additional classification needs to be done. The mere presence of, say, an average hold-time ranking above 48 is enough to separate s036 from everyone else. That's the benefit of having a marker.

"Left" cluster. The subjects in the "left" cluster all have a high average hold-time ranking. However, this separation is not perfect. As can be seen in Table 3, subject s011 has a ranking of 12.82. This is between the rankings of subjects s040 (rank 11.64) and s030 (rank 13.09). Unfortunately, s011 is not in the "left" cluster while subjects s040 and s030 *are* in the cluster. Looking at Table 4, we can see that the subjects in the "left" cluster have wildly varying latency-time rankings.

Position	Subject	Hold-time Ranking	Median hold time (ms)
1	**s041**	**3.09**	**157.40**
2	**s012**	**4.64**	**135.25**
3	**s003**	**4.82**	**143.50**
4	**s033**	**6.55**	**132.05**
5	**s002**	**8.00**	**124.75**
6	**s020**	**8.82**	**122.50**
7	**s040**	**11.64**	**117.50**
8	s011	12.82	114.05
9	**s030**	**13.09**	**113.00**
10	s056	15.82	104.75
...	...	...	....
...	...	...	....
50	s024	46.55	57.60
51	*s036*	*49.55*	*50.70*

Table 3: Average hold-time ranking for each subject. Subjects in the "left" cluster are in boldface. Because s011 (not in "left" cluster) is ranked higher than s030 (in "left" cluster), the table shows that average hold-time ranking is an imperfect marker for membership in the "left" cluster. Subjects in positions 11-49, not relevant here, are omitted for brevity.

Position	Subject	Latency Ranking	Median latency time (ms)
1	*s036*	*2.00*	*437.40*
2	s022	5.00	255.65
...	...	...	...
...	...	...	...
7	**s030**	**12.30**	**143.60**
11	**s033**	**16.40**	**159.35**
13	**s040**	**17.00**	**173.65**
25	**s002**	**24.90**	**63.10**
33	**s020**	**28.80**	**70.00**
38	**s041**	**36.50**	**52.35**
48	**s003**	**43.50**	**3.90**
51	**s012**	**44.60**	**11.05**

Table 4: Average latency ranking for each subject. Subjects in the "left" cluster are presented in boldface. The subjects in the "left" cluster are not grouped by position in the table; hence latency-time ranking is not a good marker by itself. Many subjects are omitted for brevity.

Position	Subject	Hold Ranking	Latency Ranking	In cluster?	Combined Ranking
1	**s033**	**6.55**	**16.40**	**1**	**10.20**
2	**s041**	**3.09**	**36.50**	**1**	**11.55**
3	**s002**	**8.00**	**24.90**	**1**	**13.61**
4	**s003**	**4.82**	**43.50**	**1**	**14.87**
5	**s012**	**4.64**	**44.60**	**1**	**14.95**
6	**s040**	**11.64**	**17.00**	**1**	**15.29**
7	**s020**	**8.82**	**28.80**	**1**	**15.32**
8	**s030**	**13.09**	**12.30**	**1**	**15.61**
9	s005	16.36	15.30	0	19.49
10	s016	20.91	10.30	0	22.74
11	s011	12.82	44.30	0	22.83

Table 5: SVM Ranking. Subjects in the "left" cluster (bold) are perfectly separated from the rest. Subjects are sorted by the combined ranking, which is computed as Combined Ranking (e.g., 10.20 in row 1) = 0.9722 × Hold Ranking + 0.2341 × Latency Ranking. Subjects after position 11 are omitted for brevity.

Since our goal is to obtain a marker that perfectly separates the "left" cluster from all other subjects, we take the SVM approach. Table 5 shows the results. The SVM chose 0.9722 as the coefficient for the average hold-time ranking and 0.2341 for the average latency-time ranking. Using these coefficients, the combined ranking can be computed (right-most column of Table 5). This combined ranking can be used as a marker to perfectly discriminate between members of the "left" cluster and all other subjects. All members of the "left" cluster have a combined ranking of less than 17, while all other subjects have a combined ranking of at least 17.

8 Step 4: Demographic association

After identifying two noteworthy clusters in the data, a natural question to ask is whether the subjects in these clusters share common characteristics.

8.1 Method

We did a simple correlation between a modest data base of demographic characteristics (gender, handedness, age, special conditions) and the subjects in the clusters. Two associations were found: gender and special conditions.

8.2 Results

s036 cluster. This subject suffered from temporal lobe epilepsy, a neurological disorder. Medication for epilepsy can affect fine motor control, which may explain the very short hold times and very long latency times observed in the s036 data.

"Left" cluster. The only association between demographic data and the subjects in the "left" cluster was gender. All the subjects in the "left" cluster were women.

A Chi-square test of independence was performed to examine the relation between the "left" cluster and gender. The relation between these variables was significant, $\chi^2(df = 2, N = 50) = 13.1714, p < 0.005$. That all the subjects in the "left" cluster were female is highly unlikely to have been a chance event. A Bonferroni correction [10] for multiple comparisons is probably unnecessary, but if one were applied, given an alpha of 0.05, and four comparisons, the p-value would need to be less than 0.0125 to maintain significance, which it is. We don't know what unites these women. Perhaps it's fingernails; perhaps it's nail polish; perhaps it's hand size or hand geometry. We have neither the data nor the instrumentation to make a determination at this time.

9 Discussion

We have demonstrated that a 3-step process can reveal a small number of types into which typists can be grouped, just as there is a small number of blood types in which all people are included. Also, as in blood typing, we have shown that markers can be found that make exact assignments of typists to clusters, obviating the need for re-clustering data when new subjects appear. We found that one subject, all alone in one singleton cluster, suffered from a neurological disorder, temporal lobe epilepsy. An 8-member cluster was made up entirely of women; as yet we have no explanation for what unifies these women.

The clusters into which subjects are grouped constitute new-found, fundamental structure in keystroke data, heretofore unknown. These clusters show that there appear to be constellations of characteristics that unite subjects into some groups, while isolating them from others. Demographic correlations may shed further light on these groupings or structures in the data. Extended demographic data, information about typing postures (e.g., particular hand positions), and details regarding the tendency to strike a given key with a given finger are needed to take full advantage of the new structure. With these kinds of details we may now be able to determine more than a typical classifier can do; typical classifiers tell you that two entities are different, but they don't tell you *how*

they are different. The discovery and use of new structure is a step toward resolving that problem.

We have provided a way to rule out people on the basis of their typing rhythms. For example, if an unidentified pedophile's typing fell into a particular cluster, based on monitoring of chat-room typing, the authorities would know that they are looking for only one "type" of typist, and could rule out others. The situation would be similar in other forensic applications, such as insider threat.

Classification accuracy in keystroke biometrics may improve on the basis of the new-found structure. In keystroke research it is typical to discriminate amongst a pool of users by running a classifier over the entire pool, which produces a certain classification accuracy. If the same classifier were run over just *one cluster* of that pool, classification accuracy may well improve, because the pool is smaller; and the distinction between that pool and the rest of the users would already have been made by the clustering algorithm itself. Moreover, it's possible that different classifiers will be differentially effective when applied to one cluster rather than another.

10 Limitations

The work presented in this paper is only a preliminary investigation, leaving many stones unturned. We examined only one data set; generalization to other data sets remains to be verified. We examined in detail only one cluster ("left") and one singleton (s036), leaving out marker-finding for other clusters; this is because our work here is intended as simply a proof of concept. Exploiting the new-found structure to increase classification accuracy is left to future work. Subsequent investigations will clarify these limitations.

11 Conclusion

We have shown that typists can be grouped into a small number of types. Each type is distinguished from the rest of the population by characteristic keystroke features, which can be refined into simple markers. A user's type is determined by the presence of these markers, in the same way that blood types are determined by the presence of certain antigens. Our findings constitute an initial step toward a more fundamental understanding of the intrinsic structure in keystroke data. We hope that other investigators will continue down this path by applying our techniques to various publicly available data sets.

12 Acknowledgments

This work was supported by the National Science Foundation, grant number CNS-1319117. Data collection was supported by the National Science Foundation, grant number CNS-0716677. The authors are very appreciative of the kind and generous help they received from Professor David Banks, Department of Statistical Science, Duke University.

References

[1] Banerjee, S. P., and Woodard, D. L. Biometric authentication and identification using keystroke dynamics: A survey. *Journal of Pattern Recognition Research 7* (2012), 116–139.

[2] Friedman, J., Hastie, T., and Tibshirani, R. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software 33*, 1 (2010), 1–22.

[3] Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, New York, 2001.

[4] Karnan, M., Akila, M., and Krishnaraj, N. Biometric personal authentication using keystroke dynamics: A review. *Applied Soft Computing 11*, 2 (March 2011), 1565–1573.

[5] Kaufman, L., and Rousseeuw, P. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, New York, 1990.

[6] Killourhy, K. S. *A Scientific Understanding of Keystroke Dynamics*. PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, Pennsylvania, January 2012.

[7] Killourhy, K. S., and Maxion, R. A. The effect of clock resolution on keystroke dynamics. In *11th International Symposium on Recent Advances in Intrusion Detection (RAID 2008)* (Cambridge, MA, 15-17 September 2008), R. Lippmann, E. Kirda, and A. Trachtenberg, Eds., vol. 5230 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 331–350.

[8] Killourhy, K. S., and Maxion, R. A. Comparing anomaly detectors for keystroke dynamics. In *Proceedings of the 39th Annual International Conference on Dependable Systems and Networks (DSN-2009)* (June 29–July 2, 2009, Estoril, Lisbon, Portugal, 2009), IEEE Computer Society Press, Los Alamitos, California, pp. 125–134.

[9] Maxion, R. A., and Killourhy, K. S. Keystroke biometrics with number-pad input. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN-10)* (Los Alamitos, California, 28 June - 01 July 2010), IEEE Computer Society Press, pp. 201–210. Chicago, Illinois.

[10] Miller Jr., R. G. *Simultaneous Statistical Inference*, 2nd ed. Springer Series in Statistics. Springer-Verlag, New York, 1981.

[11] Peacock, A., Ke, X., and Wilkerson, M. Typing patterns: A key to user identification. *IEEE Security and Privacy 2*, 5 (September/October 2004), 40–47.

[12] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. ISBN 3-900051-07-0.

[13] Shanmugapriya, D., and Padmavathi, G. A survey of biometric keystroke dynamics: Approaches, security and challenges. *International Journal of Computer Science and Information Security 5*, 1 (September 2009), 115–119.

[14] Teh, P. S., Teoh, A. B. J., and Yue, S. A survey of keystroke dynamics biometrics. *The Scientific World Journal 2013* (2013). Article ID 408280.

[15] Yampolskiy, R. V., and Govindaraju, V. Behavioural biometrics: A survey and classification. *International Journal of Biometrics 1*, 1 (June 2008), 81–113.

Notes

[1] Dataset available at `http://www.cs.cmu.edu/~keystroke`

[2] We compare normalized weights when we are standardizing the input data, because the *glmnet* package returns weights on the original scale, effectively "un-normalizing" the weights.