Keystroke Behavioral Anomaly Detection

Aswath Sundar Sundaram Ramasubramanian[1], Sameer Prasad Koppolu[2]

[1]Northeastern University
[2]Northeastern University
sundaramramasubram.a@northeastern.edu, koppolu.s@northeastern.edu

Abstract

This project aims to analyze the keystroke behavior of typists typing a string over a period of time to identify the unique keystroke behavior of every typist. This can then be used to classify unlabeled data of test subjects to determine if they are an actual user or trying to mimic one of the existing labeled typists. Clustering was used to determine the fact that over time, every typist had their own unique typing behavior and would hence form their own cluster. Thereafter, each of the test subjects was classified as legitimate users or mimicking a typist based on which cluster was closest to each test subject.

Introduction

One effective technique for thwarting cybercrime is anomaly detection in user authentication. Analyzing the keystroke dynamics of people typing the same text—a seemingly routine action that has minute individual differences—is one intriguing method. This method can produce a distinct profile of each user's typing habit by recording and evaluating characteristics like dwell time, flight time, and keystroke sequence. Even in cases where the attacker has the right credentials, this profile serves as the benchmark for identifying deviations and may point to abnormalities that could be signs of attempted illegal access. This continuous, non-intrusive authentication technique has a lot of promise to improve online environment security, especially when paired with additional security measures.

While keystroke behavior can vary depending on the user, it can also vary depending on the text being typed out. This is because the sequence of characters also plays an important role along with the timing information. Hence, for this project, we consider the typing and timing information of 51 users all typing the same text 400 times across 8 sessions (50 per session). This data was collected as part of a research paper (Killourhy, Maxion, 2009) done at Carnegie Mellon University (CMU).

While the authors of the paper showcased 14 different approaches to detecting anomalous users (i.e. users who were trying to gain illegitimate access to other users' accounts by mimicking their keystroke behavior), this project aims at showcasing an additional method towards solving the same problem. The key difference is that the solution to the problem taken in this project makes use of Unsupervised Machine Learning – specifically clustering.

Various clustering methods were attempted and evaluated (both before and after dimensionality reduction) while making sure that the choice of the clustering algorithm involved the pre-determination of the number of clusters (which would be the number of unique typists/users in the data). The clustering was performed for each session individually and it was seen that as the number of sessions increased, almost every user had their own cluster – an indication of a unique keystroke behavior. The users were assigned to their final clusters based on the data of the 8th session alone. Thereafter, test observations were generated by bootstrapping the data and adding noise to it. Each test observation was assigned to its closest cluster based on which cluster centroid was closest. If it was within 2 standard deviations of the closest centroid – then the observation was deemed to be a user gaining legitimate access, and if not then deemed to be fraudulent.

Additionally, the observations in the second-to-last session were taken as test observations with their user identity as true labels. On this, the same anomaly detection algorithm was applied to determine what observations were predicted to be legitimate and what observations were fraudulent.

Background

As part of the project and as mentioned above, clustering is a key aspect of the solution. Three clustering methods were explored for this project – K-Means, Gaussian Mixture Model (GMM), and Spectral Clustering. Anomaly Detection was then applied to the results of the clustering. Apart from this, dimensionality reduction was also tested using Principal Component Analysis (PCA), and t-Stochastic Neighborhood Embedding (t-SNE) which led to the exploration of clustering both without any dimensionality reduction as well as after dimensionality reduction with t-SNE.

K-Means Clustering was first performed with the clustering done on each session's data. As part of K-Means Clustering, 51 centroids were initialized using the K-Means++ method of initialization. Thereafter, each observation was clustered to its nearest centroid one by one using the Euclidean Distance metric. After each observation was clustered, the centroid of each cluster was recalculated by taking the mean of all points in the cluster.

GMMs are a soft clustering method. Here, every observation is assumed to have been obtained from a normal distribution and hence will have a probability of belonging to each of the 51 clusters – this formulates the Expectation or E step. Thereafter, the parameters of the Gaussians are updated such that they maximize the expected likelihood. This is the Maximization or M step. The E and M steps are repeated alternatively until the parameters of the Gaussians do not change.

Spectral Clustering considers the dataset as an undirected graph with connected components. From this, the algorithm calculates a Laplacian Matrix from which eigenvectors are obtained. As we are dealing with 51 clusters, we construct another matrix using 51 eigenvectors and perform K-Means Clustering on this matrix to obtain the final clusters.

For Dimensionality reduction using PCA, a covariance matrix of the centered dataset is calculated from which eigenvectors and eigenvalues are found. Thereafter, the eigenvectors that explain 95% of the variance in the data are chosen and each observation in the dataset is projected onto the eigenspace.

For t-SNE, the similarity between pairs of observations is measured in both high and low-dimensional spaces. In high-dimensional spaces, the similarity is measured using Euclidean distance while in low dimensions, the similarity is measured using the t-distribution. The algorithm aims to optimize a cost function that minimizes the distances between pairs of points that are close to each other in high and low dimensions. This in turn helps to preserve the local neighborhood of an observation.

Anomaly detection is done by identifying the closest centroid to each test observation where the centroid is the mean point of every cluster. If the centroid is within 2 standard deviations of the mean, then the observation is not a mimicker. If this condition is violated, then the observation is a mimicker.

Related Work

As mentioned earlier, this project uses a dataset that supplements the research paper - (Killourhy, Maxion, 2009). The dataset contained the typing times of 51 users/typists who each entered a password that was 10 characters long – '.tie5Roanl'. Every user typed out the password 50 times in 8 sessions – amounting to a total of 400 times in total. For every character pressed, the hold-time of the character was measured. Along with this the down-down and up-down times of every pair of consecutive characters (digraphs) were also measured – generating 31 features in total.

As part of the paper, the authors went into some methods by which anomalies could be detected. 8 of these methods were distance-based approaches involving Euclidean Distance, Manhattan Distance, and Mahalanobis Distance. Thereafter, the authors explored 2 approaches using Neural Networks, a z-Score based Approach, using anomaly detection algorithms like One-Class Support Vector Machine (SVM), Fuzzy Logic, and K-Means. All approaches involved the division of the dataset into a training set and a test set.

In all of the Euclidean Distance based approaches – one approach used the standard Euclidean Distance while the other used a normalized form of the Euclidean Distance. The approach here was to consider each training observation as a point in a p-dimensional space and then calculate the distance of each test point to each of the mean vectors which determined the anomaly score.

The Manhattan Distance based approaches were of 3 types – vanilla, filtered, and scaled. The vanilla detector was exactly the same as the standard Euclidean Detector except that the distance metric was not Manhattan Distance. The filtered approach removed outliers from the training data was also similar to the vanilla approach except that it involved the removal of outlier points (points that were greater than 3 standard deviations from the mean vector) in the training phase. In scaled Manhattan, the only difference between it and the vanilla form of the detector was the introduction of a scaling factor that scaled the Manhattan Distance. This scaling factor was the average absolute deviation in the training set.

For Mahalanobis Distance there were 3 approaches – vanilla, normed, and nearest neighbors. The vanilla detector was again the same as the standard Euclidean but instead used Mahalanobis Distance. Similarly, the normed approach was similar to the normed Euclidean Distance approach but used a normalized version of Mahalanobis Distance. In the nearest neighbors approach, the Mahalanobis distance of the test vector was calculated with respect to each training vector and the score of the test vector was the Mahalanobis distance to the closest training vector.

The standard neural network approach involved training a neural network to produce an output of 1 for each of the training vectors. Thereafter, the anomaly score of each test vector was calculated as the output of the neural network for the test vector subtracted from 1. The second neural network approach involved the use of an "auto-associative" neural network that was trained to give back the same input vector as the output. Thereafter test vectors are fed the Euclidean Distance between the output for each test vector and the test vector itself is calculated as the Anomaly Score.

The Fuzzy Logic detector makes use of typing time ranges called fuzzy sets. The sets are fuzzy because elements can partially belong to a set. In the training phase, each feature of each vector is assigned to the set with which its membership is the strongest. The average lack of membership across all test vector timing features is used to compute the anomaly score.

The z-Score method involves calculating the mean and standard deviation of each feature of the training vectors. Thereafter, the z-Score is calculated for each feature of a test vector. Additionally, a pre-determined threshold value is set. Then the anomaly score of a test vector is calculated as the number of features of the test vector whose z-Score exceeds the threshold.

For the training vectors, a one-class SVM model is constructed using the one-class SVM technique. After that, the test vectors' anomaly scores are determined by projecting them into the same high-dimensional space as the training vectors and using the signed distance from the linear separator to compute them.

The K-Means approach involves clustering the training observations. Three clusters were created in this approach. Thereafter, the anomaly score for a test observation was determined as the Euclidean Distance to the closest cluster centre.

The approach towards anomaly detection taken in this project has been inspired by these approaches – specifically the idea of using standard deviation as well as clustering. Furthermore, it was also identified that dimensionality reduction helped improve the clustering results. It was also seen that the clusters became more well-defined over time.

Project Description

As stated before, the aim of the project was to detect mimickers and non-mimickers based on the fact that every user would have their own unique keystroke behavior. Hence, the project was divided into two parts – clustering and anomaly detection.

For clustering, our project was focused on the idea that since every user had their own unique keystroke behavior, they would have their own cluster. Hence, as the number of sessions increased, the clusters would become more and more well separated. Three different clustering methods were implemented – K-Means, GMM, and Spectral Clustering. They were chosen as they allow for the decision of the number of clusters (51 in this case as there are 51 unique users in each session). The same clustering algorithms were implemented with dimensionality reduction algorithms such as PCA and t-SNE. Prior to applying any clustering or dimensionality reduction algorithms, the data was standardized by subtracting the mean from every observation and dividing by the standard deviation.

A point to note is that the clustering is performed for every session individually. Additionally, the dimensionality reduction as well as the standardization of the data was done for every session individually.

K-Means

The algorithm was run after initializing 51 clusters using the K-Means++ method of initialization. The clustering was performed for every session individually and was evaluated for every session using internal measures such as the Silhouette Score and the Calinski-Harabasz Score, and external measures such as Homogeneity, Completeness, and V-Measure.

PCA + K-Means

Prior to applying K-Means, the dataset's dimensionality was reduced for every session by applying PCA and choosing the eigenvectors and eigenvalues that explain 95% of the variance. Thereafter, the same K-Means clustering model was applied to the dataset of reduced dimensions for every session after initializing 51 clusters using the K-Means ++ initialization method.

t-SNE + K-Means

Although t-SNE is used specifically for visualization (because t-SNE preserves the local neighborhood of observations in high and low dimensions), it was observed that the performance of clustering improves when t-SNE was applied to reduce the dataset to 2 dimensions, upon which K-Means is applied with 51 clusters and K-Means ++ initialization.

GMM

This approach involves the use of soft clustering using 51 Gaussians where every observation is assigned to the cluster with which it has the highest probability. Along with this, and alternatively with calculating the probability, the parameters of the Gaussians were recalculated to maximize the likelihood of the observations. The process is repeated until the parameters of the Gaussians remain unchanged. Like in K-Means, the clustering is evaluated using internal measures like the Silhouette Score and the Calinski-Harabasz Score, and external measures such as Homogeneity, Completeness, and V-Measure.

PCA + GMM

This approach follows the same GMM approach described previously. However, prior to the GMM, PCA is applied for dimensionality reduction to reduce the dataset to the eigenspace that explains 95% of the variance. Thereafter, the same GMM approach as described above is applied on the reduced dataset.

t-SNE + GMM

Once again, t-SNE here is used to reduce the dimensionality of the dataset to 2 dimensions. Thereafter the same standard GMM approach as described above is applied to the dataset

with 51 clusters. Like before, the GMM algorithm runs until the parameters of the Gaussians remain unchanged.

Spectral Clustering

In this approach, the dataset of every session is viewed as an undirected graph of connected components. The Laplacian of this graph is obtained upon which eigenvalue decomposition is performed. Since the requirement is to have 51 clusters, 51 eigenvectors are chosen and then K-Means is applied to cluster the data. A point to note is that since every user types the same password 50 times in a session, the KNN approach of building a weight matrix is used from which the Laplacian matrix is obtained. Here, the number of neighbors is chosen as 49 as it not only gave the best clustering results but also considered the fact that if every user was to have their own cluster then every cluster would have 50 observations and hence every observation in a cluster would have 49 neighbors. Once again the clustering results were evaluated using internal measures like the Silhouette Score and the Calinski-Harabasz Score, and external measures such as Homogeneity, Completeness, and V-Measure.

PCA + Spectral Clustering

This approach involved using PCA to reduce the dimensionality of the dataset for every session. Thereafter, the standard spectral clustering algorithm was applied to each session's data of reduced dimensionality. PCA used here was to ensure that the number of features selected explained 95% of the variance of each session's data.

t-SNE + Spectral Clustering

Again, t-SNE is used here to reduce the dimensionality of the dataset to 2 dimensions. Thereafter, the standard Spectral clustering algorithm as described above with 49 nearest neighbors is used. And like before, the dimensionality reduction and the clustering are both done individually for every session.

Anomaly Detection

Once the clustering is performed, the test observations need to be classified as to whether they are mimickers or not. As clustering with t-SNE gave the best results, the same t-SNE model used prior to clustering the data, is fit on the test observations to reduce the dimension of the test observations to 2 dimensions. After reducing the dimensions of the test observations, Anomaly Detection is performed.

First, the centroid of every cluster is calculated as the mean of all the points of that cluster. Additionally, a threshold value equal to 2 standard deviations is also calculated for each cluster. Furthermore, in each cluster, the 'subject' that has the maximum number of observations (majority class user) is determined, along with whether the 'subject' is a true majority of the cluster or not (i.e. at least 50% of the observations of the cluster belong to the majority class user). Next, for each test observation, the distance to each centroid is calculated to determine the closest centroid. Once the closest centroid is determined, it is determined whether this centroid belongs to a cluster where there is a true majority. If a test observation's closest centroid is part of a cluster that has a true majority, and if the distance between the test observation and its closest centroid is more than 2 standard deviations, then the test observation is a mimicker who is mimicking the majority class user. And if this condition is not met, then the test observation is not mimicking any other user. Additionally, to replicate a real scenario where a user may type out a password multiple times for confirmation, every test observation has 2 additional copies that have noise added to it to account for the minor fluctuations in a user's typing.

In essence, the algorithm used to build the solution is as follows:

Algorithm 1: Anomaly Detection

Input: t-SNE reduced Test Observations
Output: Whether or Not an Observation is a Mimicker
1: Calculate centroids and thresholds (2 * std. deviation) for each cluster.
2: Identify Majority Class User for Every Cluster
3: Identify whether Majority Class User is a True Majority (i.e. Majority Class User is a true majority if it accounts for >= 50% of the Cluster observations)
4: **for** each test observation $test_i$:
5: Identify closest centroid c_i to $test_i$
6: Initialize *check_successful_flag* = 0
7: **for** obs_j in [$test_i$ and its noise added copies]:
8: **if** (cluster of c_i has a true majority) & (distance(obs_j, c_i) > cluster threshold):
9: **then** *check_successful_flag* += 1
10: **if** *check_successful_flag* < 2:
11: **then** $test_i$ 'Is a Mimicker'
12: **else:**
13: $test_i$ 'Is Not Mimicker'

A point to note is that the test observations were created by bootstrapping the data of the 8th session and adding noise to each observation. Like mentioned before, each test observation had 2 additional copies created with noise added to each. The size of the test set was 20% of the size of the data in the 8th session.

Additional Analysis using labeled test data was also performed. This test data was that of the 6th session. No additional noise-boosted copies were created in this case. All observations were classified as mimickers and non-mimickers. This approach was done to check whether the anomaly detection approach identifies false negatives (i.e. legitimate users identified as mimickers) and false positives (i.e. illegitimate users identified as non-mimickers).

Empirical Results

For all of the Clustering Algorithms, the number of clusters chosen was 51. The clustering was performed for each session individually and the internal measures (Silhouette Score and Calinski-Harabasz Score) and external measures (Homogeneity, Completeness, and V-Measure) were reported.

The internal measures of the Clustering Algorithms in the 8th session were as follows.

Model	Silhouette	Calinski - Harabasz
Standard K-Means	0.15	123.27
Standard GMM	0.14	115.66
Standard Spectral Clustering	0.42	84.70
PCA + K-Means	0.17	139.74
PCA + GMM	0.13	121.78
PCA + Spectral Clustering	0.42	93.91
t-SNE + K-Means	0.61	7509.02
t-SNE + GMM	0.60	7088.12
t-SNE + Spectral Clustering	0.80	7404.45

Table 1: Internal Score Measures of each Clustering Algorithm in the 8th Session

The external measures of the Clustering Algorithms in the 8th session were as follows.

Model	Homogeneity	Completeness	V-Measure
Standard K-Means	0.68	0.75	0.72
Standard GMM	0.67	0.75	0.70
Standard Spectral Clustering	0.79	0.79	0.79
PCA + K-Means	0.66	0.74	0.7
PCA + GMM	0.65	0.72	0.68
PCA + Spectral Clustering	0.78	0.78	0.78
t-SNE + K-Means	0.82	0.83	0.83
t-SNE + GMM	0.82	0.83	0.82
t-SNE + Spectral Clustering	0.83	0.83	0.83

Table 2: External Score Measures of each Clustering Algorithm in the 8th Session

The best clustering result obtained was from t-SNE + Spectral Clustering. Thereafter, the anomaly detection was performed on unlabeled test data. The test data was generated by bootstrapping the data of the 8th session to get 510 observations (equivalent to 20% of the 8th session's data). Each test observation was then classified as 'Mimicker' or 'Not a Mimicker' as described in the Project Description section.

In the additional analysis, the data from the 6th session was taken as test data with the 'subject' field being the true label. Thereafter, each test observation was again classified as 'Mimicker' or 'Not a Mimicker'. After classification, the False Positive Rate and the False Negative Rate were calculated which in turn was used to calculate the Equal Error Rate where the Equal Error Rate is defined as the sum of the False Positive and False Negative Rates divided by 2. Here's a comparison of the anomaly detection approach implemented in this project with respect to the other approaches in the paper (Killourhy, Maxion, 2009).

Detector	Equal Error Rate
Manhattan (Scaled)	0.096
Nearest Neighbor (Mahalanobis)	0.1
Outlier Count (z-Score)	0.102
One Class SVM	0.102
Mahalanobis	0.110
Mahalanobis (normed)	0.110
Manhattan (filter)	0.136
Manhattan	0.153
Neural Network (Auto Assoc.)	0.161
Euclidean	0.171
Euclidean (normed)	0.215
Fuzzy Logic	0.221
k-Means	0.372
t-SNE + Spectral Clustering + Anomaly Detection using centroid and standard deviation	0.496
Neural Network (Standard)	0.828

Table 3: Equal Error Rates of Various Detector Models

Conclusions

It was observed from this project that users do in fact develop a unique keystroke behavior over time. This was visualized and validated using the t-SNE + Spectral Clustering method. Thereafter, anomaly detection was performed by identifying the closest centroid of a test observation and determining if the cluster of the closest centroid had a true majority class. It was observed that despite ensuring a low False Positive Rate, the detector had a high False Negative Rate.

The large False Negative Rate could be due to the absence of perfect clustering – despite clusters being well separated in the 8th session, not every 'subject' became a majority class

user that was a true majority of that cluster. Despite this, it was useful to know that over time, users develop their own unique keystroke behavior. It could be the case that if more sessions were conducted with the same users, the clusters would be more well-defined where each user was a majority class user of only 1 cluster and that cluster had a true majority.

Another factor for poor performance could be due to the choice of distance metric. In the approach for this project, the anomaly detection algorithm while identifying the closest centroid to a test observation, used Euclidean Distance. For time series and sequences, methods such as Dynamic Time Warping (DTW) could be considered here as the password being typed is a sequence of characters. With a lack of enough compute, DTW took a considerable time to perform as it involved building a matrix between the every test observation and every cluster centroid thereby making it difficult to use.

Finally, the anomaly detection uses a threshold of 2 times the standard deviation in each cluster. If this value could be adjusted to a more well-defined boundary, the equal error rate could be reduced.

References

Kevin S. Killourhy; and Roy A. Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In 2009 IEEE/IFIP International Conference on Dependable Systems & Networks.

S. Haider, A. Abbas, and A. K. Zaidi. A multi-technique approach for user identification through keystroke dynamics. IEEE International Conference on Systems, Man and Cybernetics, pages 1336-1341, 2000.

P. Kang, S. Hwang, and S. Cho. Continual retraining of keystroke dynamics based authenticator. In Proceedings of the 2nd International Conference on Biometrics (ICB'07), pages 1203-1211. Springer-Verlag Berlin Heidelberg, 2007.

Git Repository

URL: https://github.com/sameerprasadkoppolu/Keystroke-Behavioral-Anomaly-Detection