Keystroke dynamics

Biometrics

Access control is a wide problem area in computer science in which access to information or resources has to be restricted to specific users. Traditional methods include

- token based identification systems. The user provides sensitive information in order to get a token which can then be used to gain access to a resource without further exchange of the original sensitive information. Many times this access is time limited. For example, you go to the dmv and exchange sensitive information (like your social security number) for a token (a driver's license) that you can then use for a period of 5 years to gain access to a resource (the privilege of driving a car on the road)
- knowledge based identification systems. The user and resource share some information that is then used to confirm the identity of the user. The assumption being that only the user would have this information and it is difficult to fake. A prime example of this is a password or pin.

Unfortunately, these access controls fall short many times in cyberspace where traditional costs that typically inhibit the abuse of these systems are significantly lower. For example, you can easily write a script to figure out the 4 digit pin on a phone whereas manually trying out all the combinations on a 4 digit padlock is a very daunting task.

However, the speed of computers doesn't just make some of these approaches redundant in cyberspace; it also opens up a lot of other authentication/identification areas that would be infeasible outside of cyberspace, and one of those areas is biometrics.

Before we proceed, we need to discuss the subtle but important difference between authentication and identification. Many times the terms are used interchangeably but they are actually different.
Identification is the process of identifying the user of a system. For example a system that correctly says that person X (out of all possible people) is the person walking through the door right now is an identification system.

On the other hand, authentication is just a yes/no question. Are you who you claim to be? A system that correctly answers that question for person X is an authentication system.
The difference is subtle but important. It is typically easier to authenticate than identify since an authentication system only needs to know what person X looks like. An identification system on the other hand would need to know what "everyone" looked

like in order to say that this is person Y and not person X.

Biometrics, as the name suggests, is a way to identify/authenticate users of a system using measurements and calculations derived from the users' bodies. Typically these measurements fall into two categories i.e. physiological (derived from the physical attributes) and behavioural (derived from the users actions). The most well known of the physiological biometrics is finger prints.

Finger prints can be used as another example of authentication vs identification. If finger prints are found at the scene of a crime, identification is attempted first. It is only successful if the system already has a finger print template that matches i.e. if the criminal was finger printed before. If that fails, then the finger prints can only be used for authentication i.e. checking against the fingerprints of potential suspects, one template at a time, as they are obtained.

A good biometric system should be

- Universal: Everyone should have the features necessary to make a profile.
- Unique: The features of one user can be easily distinguished from another.
- Permanent: Those features shouldn't easily change over time
- Collectible: Gaining access to and processing those features should be easy and cheap.
- Fast: The system should be effective, accurate, and fast.
- Accepted: Users of the system should be on board with using it.
- Robust: Features should not easily be faked.

Unfortunately, there isn't any system that ticks all these boxes. For example, even though finger prints are universally accepted, it is not a fast system, and is in fact prohibitively expensive (otherwise every building would have one).

Additionally, some of the features mentioned above are subjective and tradeoffs might have to be made. For example, if you want a cheaper system than fingerprints, then you might have to pay for it with lower accuracy. But perhaps that lower level of accuracy is high enough for the uses you have planned for the system.

Accuracy

So how does one measure the accuracy of an authentication system? Without going into too much detail, there is a need to talk about a few of measures that are used to determine how well an authentication system is working.

False Accept Rate: This is the chance that the system will incorrectly accept a template that is wrong i.e. the probability that it will let an impostor into the system. This is also referred to as type 1 error and is comparable to false positives. For example, if the system allows 12 out of the 100 illegitimate attempts, then that system has an FAR or 0.12.

False Reject Rate: This is the chance that the system will incorrectly reject a template that is correct i.e. the probability that it will stop a legitimate user from accessing the system. This is also referred to as type 2 error and is comparable to false negatives. For example, if the system rejects a legitimate user 5 times out of 100 attempts to login, then that system has an FRR of 0.05

True Accept Rate: This is the chance that the system will correctly accept a legitimate user.

True Reject Rate: This is the chance that the system will correctly reject an illegitimate user.

All of these measures depend on the choice of threshold. The threshold is an answer to the question “How similar to the stored template should the user be?”. In the example of finger prints, imagine that we have the finger prints for a serial murderer that we haven’t yet apprehended. We run the fingerprints for a suspect and find that his/her fingerprints are a 98% match for the murderer. Should we conclude that this suspect is in fact the murderer? What is our threshold? Is anything above 95% good enough? Or is 99% a better threshold? Understandably choosing a threshold is dependent on the system and how similar or disimilar different users’ templates are.
Nonetheless, one thing to note is that the higher the threshold value eventually chosen, the lower the FAR and the higher the FRR. i.e. choosing 99% as our cut off point will reduce the likelihood of putting the wrong person in jail just because their fingerprint is very similar to the serial killer. However, it also increases the likelihood of overlooking the serial killer just because his fingerprint has changed (perhaps over time or there was some issue with the fingerprint collection process). Depending on the situation, perhaps its better to have a lower threshold, and in other cases its better to have a higher threshold.

There are other measures used but they are beyond the scope of this lesson. If you are interested, feel free to look into ROC curves, and EER.

KeyStrokes

Keystrokes are a behavioural biometric that are based on the premise that everyone’s typing pattern is unique to that person and can therefore be used to authenticate and/or identify them. As far as biometric systems go, it is very cheap, collectible, universal and fast. Its accuracy isn’t as high as fingerprints or other physiological biometrics but it finds wide application because of its price. Also it works really well in combination with other verification systems (as is the case in multimodal biometric systems). For example, if combined with a simple password system, one can be sure that a user - who types a unique password, while at the same time demonstrating the typing patterns of its associated user - is indeed the user in question.

It has its origins in the mid 19th century when people sending morse code over telegraphs realised that many of the operators had their own “signature” way of typing out the dots and dashes. Some words were easier for them to string

together, timing between specific characters was predictable, etc.

In this day and age, you only have to type out a long chat message to realise that people type differently too. Some people find going from e to r very easy and fast, while others dont. Some people use the backspace key too frequently while others dont. Some use the shift key to make letters upper case while others use the Caps Lock key predominantly. Some use the left shift key, while others use the right shift key.

During the course of this class, we shall discuss the features required to build a keystroke template for a user, discuss the process of comparing different templates (in an attempt to authenticate), and then build a tool(s) that attempts to mimic another user for a simplistic single word password assuming that we have access to their keystroke template.

Weaknesses of Keystrokes

For those who are paying attention, you might have noticed that we never mentioned uniqueness, permanence, and robustness as characteristics for keystroke systems. That is because the weaknesses of a keystroke system are found in these areas.

We don’t type the same way every day of our lives. In fact, we don’t type the same way in the same day or even the same hour. The way we type depends on factors like cognitive load (are we thinking while typing, or just copying some text), how long is the text that we’re typing, how familiar are we with the language being typed, how familiar are we with the keyboard being used, are we being distracted by external factors while we type, etc.

Additionally, there is a significant probability that another user could type in a manner similar to us. Perhaps not over a long time or long text, but the likelihood of a similarity will increase with a short text.

Furthermore, there is research into how easy it is to fake the typing patterns of someone and that research has shown promising results especially if the attacker has access to average population templates.

While these weaknesses render keystroke dynamics as an insufficient access control system on its own, its cheap price and ease of implementation make it a perfect system to use in combination with other systems as in a multimodal authentication system.

Keystroke features

So we have mentioned the idea of a user template i.e. some data that can be used to identify or authenticate a user. Question is what kind of data is in that template?

Many different keystroke systems use different pieces of data, but they are typically derived from two basic measurements.

- **Key hold time (KHT)**: This is the amount of time that elapses between the pressing of a specific key on the keyboard, and the releasing of that key. The

longer that a user presses a specific key, the longer the keyhold time.

- **Key interval time (KIT)**: This is the amount of time that elapses between the releasing of a given key, and the pressing of the next key. The longer the amount of time taken between typing successive keys, the longer the key interval time. Note that it is possible for the key interval time to be zero or even negative. This happens if the user presses the second key before completely releasing the first key.

One can already see that even with just two measurements, the template of a user can have a daunting number of features. Even if we just restricted it to lower case letters, each user would have a template that consisted of 26 key hold times, and 676 key interval times (for a total of 702 features). If we allowed all keys on a typical keyboard, we are looking at 58 keyhold times and 3364 key interval times (for a total of 3422 features). Imagine that: 3422 columns for each user (row) in a table.

This is a LOT of data, and there is ongoing research on how to make this problem more manageable. One of the common ways of making this more manageable is restricting the data collection to fixed text (which works really well with passwords). Instead of collecting data for every possible KHT and KIT, just collect data for the KITs and KHTs during the typing of a simple short password.

For example, if the password was “abc”, the system could store keyhold times for the three characters, and key interval times for the transitions i.e. KHT_a, KHT_b, KHT_c, KIT_{ab}, and KIT_{bc}. A password of 10 characters would have 10 KHTs and 9 KITs. This is significantly smaller than the 3422 columns we would have to keep track of for free text.

Note that while this adjustment may make the computational cost cheaper, it sacrifices accuracy for it. It is more difficult to differentiate users based on how they type a short predictable string.

Another thing to note that instead of storing an absolute measurement that the user has to replicate every time they type, it is more prudent to store an average of multiple attempts that we know were typed by the legitimate user. It is even more helpful to store some measurement of how much that user deviates from the average for each feature. Of course each of these adds another layer of complexity to the process.

The legitimate user would be required to *train* the system i.e. type their text multiple times in order to extract their feature information. Once set up, the system could then be *tested* using both legitimate and illegitimate users to see how successful it is in allowing the legitimate user and rejecting the illegitimate user. These phases are referred to as the training and testing phase respectively.

Classification

To answer the question of authenticating a test template, we shall go to the field of data science. For the purposes of this problem, and this class, we shall only talk about it briefly. However, there are multiple different types of problems and solutions in the field that I would highly encourage you to look over for more than keystroke authentication.

The problem we are trying to solve is this. Given that a vector is a list of values (think a row of values in a spreadsheet), How can we differentiate between two vectors? If one vector is the training template of true/legitimate user and the other is the testing template of someone claiming to be the true user, how can we tell that both templates belong to the same person or to two different people? A common approach is to use the distance between the vectors. If the distance between the two vectors is short, we could interpret that to mean that they both belong to the same user, and if the distance is long, then we can assume that the vectors belong to different people.

Obviously short and long distances are subjective and part of setting up the system for a user is figuring out what threshold value you will use to differentiate between a short and long distance. However, the choice of this threshold value is not as straightforward as you might think.

You may find that the threshold value that provides a good performance for one user is different from the threshold value that would provide similar performance for another user. Furthermore, our measurement of the performance doesn't just depend on the user we are trying to authenticate, but also the type of people our system is trying to keep out. If we have test samples that are very similar to the user, then we have to make our threshold more stringent but then run the risk of mis-classifying the legitimate user (and increasing the false reject rate). Conversely, if the type of illegitimate users we are using to test our system are completely different from the legitimate user, then we can afford to make the threshold less stringent and still have good performance.

Euclidean verifier

One of the most intuitive ways to solve this problem is using a distance metric very similar to the distance metric you used in algebra to find the distance between two points on a graph.
Recall that the distance between two points can be given by the formula

$$dist = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

This formula can be extended from just two features to any number of features as shown below.

$$euclidean\,distance = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

Where n is the number of features, a_i is the value of the i^{th} feature in the test vector, and b_i is the value of the i^{th} feature in the training vector.

After calculating the euclidean distances of a given number of vectors that all belong to the same user, we can then make a decision on the threshold value that would best serve our purpose of accepting the user, and rejecting any impostors.

Scaled Manhattan verifier

Another distance measure that we shall use for comparison is the scalled manhattan verifier. This particular verifier doesn't just rely on the average value of a feature in the training process but also on the absolute average deviation. The absolute average deviation is a measure of how spread the training values were from the average value. Note that it is different from the more traditional standard deviation (but not by much).

A scaled manhattan score can be derived from the formula

$$scaled\ manhattan\ score = \sum_{i=1}^{n} \frac{|a_i - b_i|}{y_i}$$

Where n is the number of features, a_i is the value of the i^{th} feature in the test vector, b_i is the value of the i^{th} feature in the training vector, and y_i is the absolute average deviation of the i^{th} feature in the training vector. y_i is calculated from the formula

$$absolute\ average\ deviation = \frac{1}{m} \sum_{j=1}^{m} |x_j - \bar{x}|$$

where m is the number of measurements of the x^{th} feature in the training set, x_j is the j^{th} measurement of that feature, and $\bar{x}$ is the average of that feature over the training set.

The benefit of a scaled distance score like scaled manhattan is that is takes into consideration the variation in a user's feature. For some people, the keyhold time of letter 'a' will vary a lot, and for others it will be very constant. Using the deviation allows us to capture that variability and use that to make our test decisions.