William Briguglio
104 204 372

June 2nd, 2019

Assignment 1

1. Dataset Insights:

The dataset used for this assignment consists of 20,400 records of password typing data from 51 subjects each contributing 200 records. The 200 records were captured in batches of 50 across 8 sessions per subject. Already it is questionable weather this data is realistic as 50 records were captured all at once without the users breaking from the task to "reset". This means the users would probably be typing much quicker, more consistently, and with a different rhythm than they otherwise would be in a real-world scenario due to muscle memory. Thus, the last 40 records from each session are unlikely to be representative of the actual typing behaviour of the subjects compared to the first 10 from each session. The last 40 records are also probably much more consistent and closer together then the first 10.

Looking at the data itself we can see that the values range from -0.2358seconds to 8.3702seconds with over 100 values greater or equal to 3 seconds. This seems likely to be due to the subjects not having the password memorized beforehand which is another problem as this is not indicative of a real-life scenario. Another reason for the long typing times could be because some of the users were inexperienced or ineffective with a keyboard. This is not an issue in and of itself but since the authors did not record the experience or skill levels of the subjects, it is difficult to determine whether this dataset is representative of a typical population, or of a population that is likely to use keystroke biometrics. Therefore, results and conclusions drawn from this dataset are spurious.

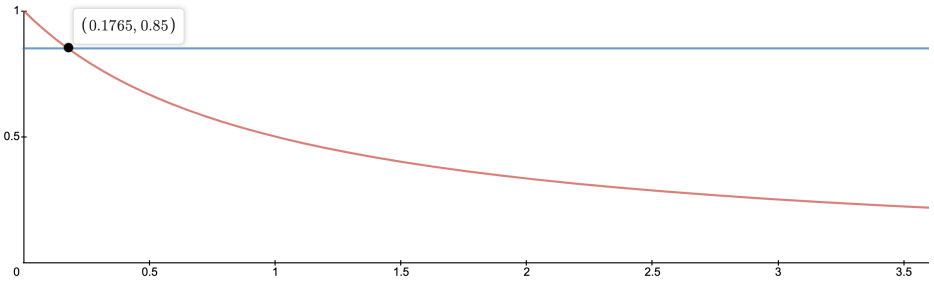
An inherit drawback of keystroke biometrics is the typing ability of the user. Users with disabilities or who have poor typing skills may give very inconsistent results making it difficult for classifiers to distinguish their typing behaviour from others. Furthermore, if a user has an injury or other temporary condition which may affect their typing behaviour, then they may fail to be recognized by the biometrics systems.

2. Keystroke Detector:

The keystroke detector uses the first 10 records from the first session of each of the subjects in the training set. This was because the later samples were likely to be influenced by muscle memory. For each subject, a model was generated by averaging out a subject's 10 timing vectors. The max distances (Euclidean, Mahalanobis, and Manhattan) between any one of these vectors and the average was also recorded, along with the covariance matrix of the subjects 10 timing vectors. These 3 components, the average timing vector, max distances (henceforth referred to as *threshold distances*), and covariance matrix made up the subject's 'model'. The threshold distances are used by the scoring function to determine what distance corresponds to a score of 0.85, i.e. the minimal score needed to match, for a particular subject.

The keystroke detector implementation has two main components, the distance function, which measures the difference between two keystroke timing vectors, and the scoring function, which converts a distance in the range $[0,\infty)$ to a score in the range $(0,1]$.

a. Scoring Function: The scoring function $s(d)$ had several constraints. First it needed to convert an input in the range $[0,\infty)$ to an output in the range $(0,1]$ in a manner which preserved order. Another constraint is that $s(d) = 1$, when $d = 0$, and as d increases, $s(d)$ approaches 0. Thus the function $s(d) = \frac{1}{1+d}$ was used.



As shown in the graph above, $s(d) \geq 0.85$ when $d \geq 0.1765$. Since a score ≥ 0.85 was to indicate a match, the inputs would have to be scaled individually for each subject so that the threshold distance for each subject would receive a score of 0.85. Thus the scoring function in effect became: $s(d_s) = \frac{1}{1+d_s}$, where $d_s = d\left(\frac{0.1765}{Threshold\ distance}\right)$.

b. Distance Function: The chosen distance function was Manhattan distance as it preformed best compare to Euclidean and Mahalanobis distance. Distance functions were compared by scoring their given distances for each combination of model and timing record. If the model and timing record belonged to the same subject and received a score lower than 0.85, this was counted as a false negative. If they did not belong to the same subject but received a score greater than 0.85, this was counted as a false positive. The false negative and false positive rates were used to determine the weighted accuracy of the classifier with the given distance function. In total there were 10 timing records for 52 subjects giving 520 total timing records which were each compared to 52 models giving 27,040 total comparisons. 520 had true class "match" and 26,520 had true class "non-match". The results of the distance function comparisons can be seen in table I. A tolerance parameter for the scoring function was tuned to find the highest weighted accuracy and that was what was recorded in table I.

Feature Set: The feature set for this detector consists only of the 11 H.[*key*] features since these features preformed the best as shown below in table I. The feature sets were compared in the same way as the distance functions.

Table I:

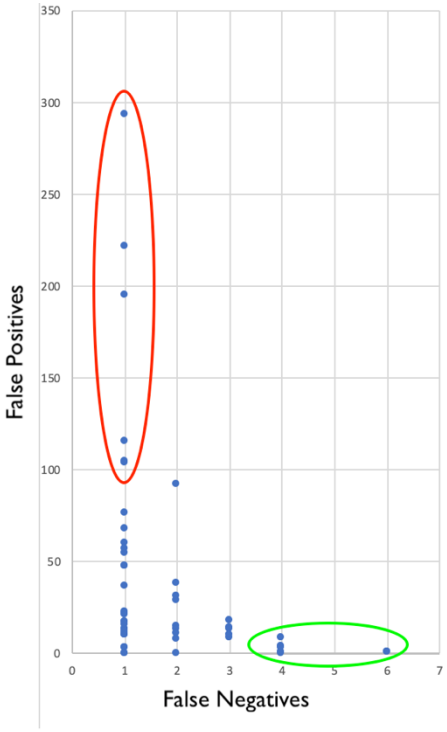
	DD+UD+H	DD+UD	DD+H	UD+H	DD	UD	H
Manhattan	74.8529	72.8078	76.8902	76.449	72.9235	72.6567	93.6373
Mahalanobis	56.6156	56.6412	56.6255	56.5098	56.6882	56.6824	68.2058
Euclidean	67.3412	67.1098	67.3961	67.5569	67.0941	67.3058	90.8294

3. Users Set Description:

In this experiment we define **Sheep** as subjects whose model has low false native (FN) and false positive (FP) rates, **Goats** as subjects whose model has high FN rates, **Lambs** as subjects whose model has high FP rates, and **Wolves** as subjects whose samples can cause a lot of FPs. The key stroke detector used to determine these rates was the same used by the GA described below. The FN and FP rates for each model was determined by counting the number of samples which wrongfully did not get a matching score, and which wrongfully did get a matching score respectively. Show below is each model's FN and FP rates plotted against each other.

William Briguglio
104 204 372

June 2nd, 2019

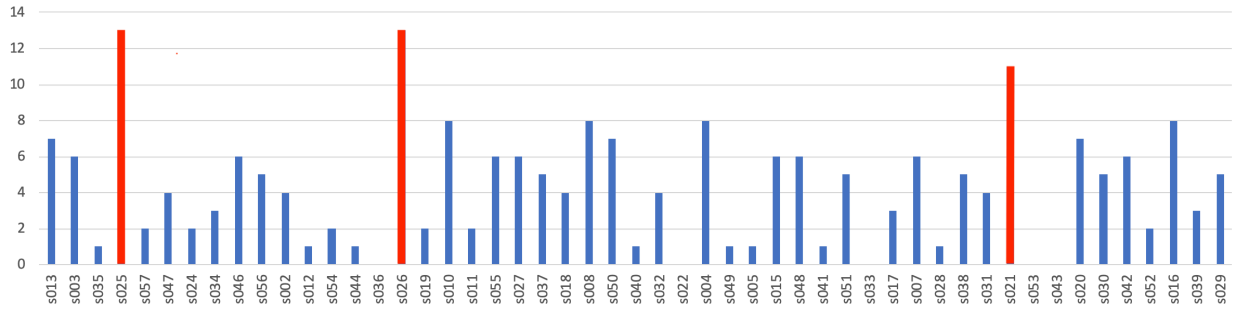


As we can see, there are 3 to 7 models which have a lot of FPs. If we make our lamb definition more specific and define lambs as any model with greater then 100 FPs, then there are 6 **lambs** in our dataset (circled in red). These are s027, s033, s018, s046, s054, and s034 in ascending order of FPs.

We can also see there are 2 to 7 models with high FNs. Refining our definition of goat to be any model with 4 or more false negatives, we can determine there are 7 **goats** (circled in green), s035, s038, s049, s055, s025, s052, s005 in ascending order of FNs.

There are no models which are both a goat and a lamb since that would correspond to the upper right region in the graph. The remaining models that are not circled, in the lower left region, are our **sheep.**

To determine the number of wolves, we look at the number of FPs the samples supplied by each subject are responsible for. If 4 or more of the 10 samples supplied by a subject create a false positive with the same model, then we can say that that subject successfully imitated that model. In the graph below we can see the number models each subject's 10 samples successfully imitated. If we refine our definition of **wolf** to be any subject who can successfully imitate 9 or more models, then we have 3 wolves in our dataset, highlighted with red bars, s021, s026, and s025. Thus, s025 is both a wolf and a goat while s021 and s026 are both wolves and sheep.



4. Biometric Adversary Framework:

The biometric adversarial framework was implemented as a Genetic Algorithm (GA) using a more strict version of the keystroke detector described above. The keystroke detector still used the H.[*key*] features with Manhattan distance but with a lowered tolerance which increased the false negative rate and decreased the false positive rate. The tolerance was chosen so that it would be the minimum which causes two subjects to have 5 or more of their own samples classified as no-match. This was done as a subjective measure to ensure the GA was tested against a keystroke detector which is not too easy to beat (i.e. a lot of false positives). This keystroke detector had a total of 95 false negatives and 1996 false positives for a weighted accuracy of 86.735% and accuracy of 92.267%. The subject chosen for the GA to imitate was that with the highest number of false negatives, s005 with 6, as this would be the most difficult subject to match and serve as a good benchmark for the GA. The genetic algorithm consisted of several main components:

a. Initialization: The population was initialized as an array of vectors of length 11, which served as the chromosomes, with random values between 0 and 1. Although the dataset consisted of values between -0.2358 and 8.3702, values between 0 and 1 were used because the majority of values in the dataset were within this range as well as the all the values making up the average timing vectors of the subjects' models. Furthermore, I considered it likely that someone attacking a keystroke detector in a real-world scenario would have knowledge of average typing timing values. Various population sizes were tested however 10,000 was found to work best.

b: Fitness: The fitness function simply returned the score the keystroke detector returned for a given chromosome against the model for subject 's005'

c. Crossover: Crossover was done by ordering the chromosomes according to their fitness score. Then, the first and second highest scoring chromosomes were mated, the second and third, third and fourth, and so on, each producing two children until the required number of offspring were produced. Initially, the highest scoring chromosome produced offspring with more than one chromosome however this caused the population to lose diversity too quickly causing the GA to not converge at an optimum. The number of offspring was calculated as the total population size minus the size of the carry over population and the mutant population, this kept the population size consistent across generations.

> Mating: Mating was done by generating an array of random indices between 0 and 10 and swapping the values at these indices between two parents generating two offspring. The number of indices swapped was proportional to the score of the parents, ranging from 5 for parents with below 0.5 score, to 1 for parents with over 0.8 score.

d. Mutation: Mutation was done by ordering the chromosomes according to their fitness score and then, mutating the first, the second, and so on until the required number of mutants were produced. The number of mutants was determined by a parameter set before runtime. For each chromosome a list of random indices between 0 and 10 was generated and some value was added or subtracted from each of these indices in the chromosome. The number of indices to mutate and the value the indices were changed by was proportional to the score of the chromosome being mutated, with 10 indices being changed by ± 0.15 for scores below 0.5 and 1 index being changed by ± 0.02 for scores above 0.80. The chances of addition vs subtraction were 50/50.

e. Selection: Selection was done by concatenating the list of offspring and the list of mutants to the list of carry over chromosomes, which was the highest scoring chromosomes of the previous generation. The size of the carry over population was determined by a parameter set before run time. Mutation and child population were kept separate so that if either of them was worse than their parent population then the GA would be able to try another round of mating and mutating on the parents in the next generation when they are carried over. This allows the GA to randomly explore the feature space without the fear that a bad mutation or mating could cause the GA to lose progress.

f. Termination: The termination can be set to either "match" or "best_match" mode. If match mode, then the GA terminates when the highest scoring chromosome gets a score above 0.85. If best-match mode, then the GA terminates when the specified number of generations are complete or a chromosome gets a score of $1 - e$, where e is some constant set before run time. In all cases the best scoring chromosome at the time of termination is printed to the terminal.

William Briguglio
104 204 372

June 2nd, 2019

Results:

After testing the GA to determine best parameters, the GA was able find a chromosome with a matching score (i.e. score ≥ 0.85) in as little as 9 generations (fig 1.) and was able to obtain a score as high as 0.96 in 27 generations (fig 2.).

Fig 1.

```
MACLT1:A1 william$ python3 GN.py

GENERATION:  0  MAX SCORE:  0.3849489119814451  GEN MAX SCORE: 0.3849489119814451

GENERATION:  1  MAX SCORE:  0.4607550682371634  GEN MAX SCORE: 0.4607550682371634

GENERATION:  2  MAX SCORE:  0.5322497105587094  GEN MAX SCORE: 0.5322497105587094

GENERATION:  3  MAX SCORE:  0.6147061041224979  GEN MAX SCORE: 0.6147061041224979

GENERATION:  4  MAX SCORE:  0.677177338647823   GEN MAX SCORE: 0.677177338647823

GENERATION:  5  MAX SCORE:  0.6908265801802114  GEN MAX SCORE: 0.6908265801802114

GENERATION:  6  MAX SCORE:  0.7476557838860759  GEN MAX SCORE: 0.7476557838860759

GENERATION:  7  MAX SCORE:  0.7775344409059255  GEN MAX SCORE: 0.7775344409059255

GENERATION:  8  MAX SCORE:  0.7799322267219074  GEN MAX SCORE: 0.7799322267219074

GENERATION:  9  MAX SCORE:  0.8623085142630676  GEN MAX SCORE: 0.8623085142630676
BEST SCORE:  0.8623085142630676
BEST CHROMOSOME:
 [0.0815299  0.1534121  0.08303242 0.13292767 0.12953157 0.12687137
 0.08725981 0.12648511 0.08140938 0.08208375 0.1069912 ]
MACLT1:A1 william$
```

Fig 2.

```
BEST SCORE:  0.962165534737943
BEST CHROMOSOME:
 [0.07564502 0.12426005 0.09008208 0.12626796 0.10476393 0.1257298
 0.09983135 0.14989988 0.07019959 0.08458567 0.09900433]
BEST GENERATION:  27
MACLT1:A1 william$
```

The average vector for s005, i.e. what the chromosomes are supposed to imitate, is:
`[0.081, 0.1259, 0.091, 0.1257, 0.1101, 0.1305,`
`0.0945, 0.1444, 0.0686, 0.0819, 0.1]`
As you can see, the solution chromosomes are very close.