Biometric Classification with Eye and Mouse movement data using Decision Trees

Jamison Rose
Western Washington University
rosej22@students.wwu.edu

ABSTRACT
This paper introduces a novel data set for classification using contemporary data mining techniques. The structure of the dataset, the technique used, and the parameters tuned will be explained in detail.

1. INTRODUCTION
The task presented in this paper is to uniquely identify an individual based on data gathered about how they look at a computer and how they operate a mouse. This is a form of biometric classification. Biometric classification has been used in the past for security purposes, but historically most biometrics have measured physical characteristics of a user which can be easily faked or coerced from a user. By using metrics gathered by observing subconscious actions or other behavioural biometrics of a user it is possible to reduce the risk of spoofing a users information. Neural networks for mouse-only biometric authentication are able to achieve 2% False Acceptance Rate (FAR) and 2% False Rejection Rate (FRR)\[1\]. If similar or better numbers can be achieved with Decision Trees augmented with eye-tracking information then the technique can be considered further.

2. METHODOLOGY

2.1 Dataset
That data set used was generated based on the actions of 41 users. Each user has 100 recorded actions. To create the actions a user was instructed to click a series of buttons in sequential order. In the center of the screen was a start button where each action was to start and around the edges of the screen were buttons labelled 1 through 8 evenly spaced so that there was one in the middle-top of the screen, upper right corner, middle right side, etc as seen in figure a. Each action created a feature vector that contained 8 datapoints. These included the speed of mouse movement, the speed of eye moment, the ratio of these speeds, the average angle of deviation from horizontal of the mouse movement, the average angle of deviation from horizontal of the eye movement, the ratio of these 2 deviations, the delay before a user started the action after pressing start, and the direction which they were travelling to do the action. These actions can be seen visualised in figure b. The red lines represent mouse movement and the blue lines represent eye movement.

While forming the data set the data for each user was cleaned and pruned. Data between the eye and mouse were aligned on timestamps after being taken in and cleaned. The parts of a sample that took place near the start and end points of a sample were removed because these parts contained little movement data due to fixation of the eye and mouse. Some samples from a user had incomplete data or data that was noisy due to factors such as the user not looking at the screen or the eye tracker not fully recording during a sample period. Data after cleaning can be seen visualized in figure c.

2.2 Libraries
Scikit-Learn was the main library used to prepare the decision trees used to evaluate this dataset. It’s implementation is in python and relies on the numpy and scipy libraries to do its mathematical processing.

2.3 Algorithms
Sci-kit learns implementation of the decision tree classification uses an optimised version of the CART algorithm. This algorithm is best suited for this application because it is fast enough to be able to run through the dataset and allows for not only classification of ordinal values, but also classification of continuous values. Both are needed because...
the angles and speed are both continuous, but the direction is ordinal.

2.4 Training Procedure

2.4.1 Data Pruning
Some users data were sparse after the cleaning and pruning procedure done when creating the dataset. Because of this tests were also done using modified datasets that removed sparse users and another set that used only the users with the most complete data. The dataset that used only the most complete data was the only one that did remotely well so the results of it's tests are the focus in the rest of this paper.

2.4.2 Data Normalization
Each field of a users input other then direction varied quite a lot from each other so tests were done with the data centered and normalized to see what effect that would have on results. These tests ended up making the fields harder to distinguish for the system so it was not used in the majority of tests.

2.4.3 Data Preprocessing
The initial results of training showed that the system performed very poorly. Because of this several techniques were used to preprocess the data to get better results.

One-Hot Encoding.
The first technique used was one-hot encoding. One hot encoding works by taking categorical data and adding a new dimension to the data vector for each value of the category. For a given data point the dimension that represents that data points category is then labelled 1 and all other dimensions for the field are labelled 0. This helps the decision tree distinguish between the categories easier which helped the overall performance of the system.

Data Batching.
The second technique tested was batching the data into sessions of eight actions. Each session contained a data point for each direction. This helps because some users have similar data in a single direction, but often do not have similar data in all directions. This caused the training data to increase in dimensionality which affected training time, but it also reduced the amount of training data by a factor of eight which caused there to be little data available to train on. Batches of this type were tested by creating batches of just eye/mouse speed and also batches using all data available.

Data Permutation.
To counteract the data loss of batching the data, the data was permuted so that more batches could be trained on. To do this many batches were created by randomly selecting a data point from the pool of data points in a direction for each direction in the batch. This allowed an arbitrary number of batches to be able to be specified for a user which also allowed for each user to have the same number of batches.

2.4.4 Decision Tree Tuning
Sci-kit learn has a variety of tuning parameters that can be changed when creating the decision trees. Each of these were tuned individually to get the best result.

Criteria.
The default split criteria that the CART algorithm implemented by Sci-Kit learn in the gini value. This value makes the most sense for the tree because it represents inequality well which makes sense when trying to compare people. The other possible split criteria is entropy which helps see the marginal benefit of a split, but does not make as much sense in this case to use. Regardless both methods were tested.

Max Depth.
Initial tests of the decision tree showed that the system created a very deep tree. It seemed like much of the depth did not add much to the system so limiting the max depth was tested. Only the results of the best tuned depth are reported.

Min/Max Features.
By default Sci-kit learn allows splits to happen by comparing any number of features at the same time up to the max being all of the features at once. By considering all features at once the split might be too general. On the other hand if to few features are considered the relation between features might be lost. The basic assumption of the dataset is that eye and mouse movement is somehow related so having independent features should be discouraged. After testing it was found that limiting the features never had a positive effect on the system. This makes sense if the data is well correlated.

2.5 Evaluation Procedure
Once trained the decision trees need to be evaluated to see how well they can recognize data. To do this 40% of the data from users was removed from the training set before the system was trained. This data was put aside to be used only for testing. To test the tree each member of the testing data set was run through the tree and it’s most likely class was predicted. These predictions were stored in a table. The metrics used to evaluate the system were precision, recall, and f-score. Each metric was calculated for each class individually, then the values of the metrics were averaged over all classes. Precision is calculated as the number of items correctly labelled as its class divided by the total number of items that should have belonged to that class. Precision was calculated as the number of items correctly labelled as a class divided by the number of items labelled as that class. Finally, f-score was calculated in equation 1.

\[
\text{F-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]
3. CONCLUSIONS

3.1 Results

<table>
<thead>
<tr>
<th></th>
<th>F-Score</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive</td>
<td>.078</td>
<td>.080</td>
<td>.076</td>
</tr>
<tr>
<td>Naive \ Normalization</td>
<td>.0009</td>
<td>.0004</td>
<td>.024</td>
</tr>
<tr>
<td>Pruned</td>
<td>.155</td>
<td>.158</td>
<td>.153</td>
</tr>
<tr>
<td>One Hot</td>
<td>.171</td>
<td>.172</td>
<td>.170</td>
</tr>
<tr>
<td>Speed Permute</td>
<td>.317</td>
<td>.353</td>
<td>.287</td>
</tr>
<tr>
<td>All Permute</td>
<td>.397</td>
<td>.407</td>
<td>.387</td>
</tr>
<tr>
<td>Entropy</td>
<td>.294</td>
<td>.301</td>
<td>.287</td>
</tr>
<tr>
<td>Depth Tuned (7)</td>
<td>.483</td>
<td>.546</td>
<td>.433</td>
</tr>
</tbody>
</table>

The table shows the results of each test. The naive approach used the raw data without any preprocessing after its initial compilation. As seen in the table it performed very poorly and did even worse when the data was normalized. By pruning the number of users tests in the results were significantly better, but still not good. Encoding the categorical direction field using one hot encoding marginally improved the performance. Batching several actions in unique directions gave the system the greatest performance boost. As seen in the table using only the speed of the actions and permuting the data to get more of it did significantly better than the previous tests that used only one direction at a time. By further incorporating all of the data from each direction the value f-score increased further. Switching the system to evaluate based on entropy hurt performance so it was not used in further tests. The best results came when the depth of the system was tuned to a maximum depth of seven. This f-score is still less than .5. The result is better then randomly trying to identify a person as one of fifteen people, but it is not nearly as good as the previous work using Neural Networks using only mouse dynamics. Based on these findings it is not recommended to use decision trees for data in this domain. A figure d shows a visualization of the best tree that was trained.

3.2 Future directions

In the future several things could be changed to try and get better results. Ski-kit learn only uses the CART algorithm so using a different library might help if other decision tree algorithms are more well suited to the task. During the evaluation procedure F-score, precision, and recall were used in hopes of making the results be comparable to classmates doing classification because they are a commonly used metric for general classification. To make the work more comparable to other biometric works the FAR and FRR metric should instead be used. During the evaluation procedure rather than having a single training and testing data set it would probably be more accurate to test the system by separating the testing data at random several times and recording the results several times to avoid a biased split of the data. The technique that would probably increase the ability of the system the most would probably be to train on larger amounts of data and incorporate more then one sample in each direction into the data batching technique. With these changes the system will most likely improve, but probably not enough to bring it in line with techniques other then decision trees. Decision trees might work better if the features extracted from users were different.

4. REFERENCES