User Authentication Based On Keystroke Dynamics using Artificial Neural Network

Group 8

Aryendra Singh 18EC10077

Devansh Rajgarhia 18EC10073

Awadh Kejriwal 18EC10078

Viplaw Srivastava 18EC10067

Utkarsh Patel 18EC35034

Need & Motivation

- Today most computer systems identify users by means of secret phrases known as passwords.
- However, this authentication system does nothing to protect the computer from unauthorized access once the user has started an active session.
- These limitations of password based authentication lead to the introduction of authentication techniques based on biometrics.
- Biometrics offer automated methods of identity verification or identification on the principle of measurable physiological or behavioral characteristics
- The two keylogger features for a particular user are almost unique and hence it is used to uniquely identify the user

Problem Statement

" To authenticate the user based on the keystroke logging features like hold time and latency time of the neutral, happy and sad mood data using an artificial neural network. "

1) Acquire data of mouse dynamics: The data collection process involves typing sentences using a virtual keyboard before and after watching a video to capture the different moods of the user and log the keystrokes of the user.

2) Recognise the user using mouse dynamics: Artificial Neural Network is implemented to differentiate between and authenticate the user

3) Apply five-fold validation to report results. The accuracy of correct authentication of the user by classifier is reported.

Artificial Neural Network

If we have a training instance $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, then we can use perceptron modelling to find $z = \boldsymbol{w}^T\boldsymbol{x} + b$, and use non-linearity $a = g(z)$ for classification purpose. The most common non-linearities used are: $ReLU(\cdot)$, $\tanh(\cdot)$ and $sigmoid$ function.

ANN is a feed-forward fully connected network of perceptrons or what we call neurons. It may have several hidden layers. There exists no feedback or loop in the network.

ANNs are used both in classification and regression tasks. The error function, also called loss function in this domain, is *mean-squared error* for regression and *cross-entropy* for classification. The weights and biases of all the layers are updated so as to make this loss function minimal. Gradient-based optimization is used in ANNs and the gradients are propagated using chain rule.

Forward Propagation

Let k^{th} layer has $n^{[k]}$ neurons. So, if we are considering i^{th} layer, each of its $n^{[i]}$ neurons are fully-connected to $n^{[i-1]}$ neurons of previous layer. Therefore, for $j = 1, 2, \dots, n^{[i]}$, we can model each neuron with a logit unit given as

$$z_j^{[i]} = \boldsymbol{w}_j^{[i]^T} \boldsymbol{a}^{[i-1]} + b_j^{[i]}$$
$$a_j^{[i]} = g\left(z_j^{[i]}\right)$$

Hence, the output of i^{th} layer is given as $\boldsymbol{a}^{[i]} = \left(a_1^{[i]}, a_2^{[i]}, \dots, a_{n^{[i]}}^{[i]}\right)^T$. Usually, in this notation, the input feature vector $\boldsymbol{x}$ is denoted by $\boldsymbol{a}^{[0]}$. Hence, using vectorized representation, we have,

$$\boldsymbol{z}^{[i]} = \boldsymbol{W}^{[i]} \boldsymbol{a}^{[i-1]} + \boldsymbol{b}^{[i]}$$
$$\boldsymbol{a}^{[i]} = g(\boldsymbol{z}^{[i]})$$

Here, $\boldsymbol{z}^{[i]}, \boldsymbol{a}^{[i]}, \boldsymbol{b}^{[i]} \in \mathbb{R}^{n^{[i]}}$, $\boldsymbol{a}^{[i-1]} \in \mathbb{R}^{n^{[i-1]}}$ and $\boldsymbol{W}^{[i]} \in \mathbb{R}^{n^{[i]} \times n^{[i-1]}}$.

Backward Propagation

Let the ANN of interest has l layers, with input being layer 0 and output being layer l. The loss function L, which is a function of weights and biases of all the layers is defined directly from $\boldsymbol{a}^{[l]}$, the output of layer l. In back propagation, we use $\partial\vartheta$ to represent $\frac{\partial L}{\partial \vartheta}$. Therefore, $\partial \boldsymbol{a}^{[l]}$ can be computed easily. This gradient is transferred through the layers, from output layer to input layer, using chain rule of differentiation. Consider that we computed $\partial \boldsymbol{a}^{[i]}$ for i^{th} layer, then, to compute $\partial \boldsymbol{a}^{[i-1]}$ we have,

$$\partial \boldsymbol{z}^{[i]} = \partial \boldsymbol{a}^{[i]} \cdot g^{[i]'}(\boldsymbol{z}^{[i]})$$

$$\partial \boldsymbol{W}^{[i]} = \partial \boldsymbol{z}^{[i]} \cdot \boldsymbol{a}^{[i-1]^T}$$

$$\partial \boldsymbol{b}^{[i]} = \partial \boldsymbol{z}^{[i]}$$

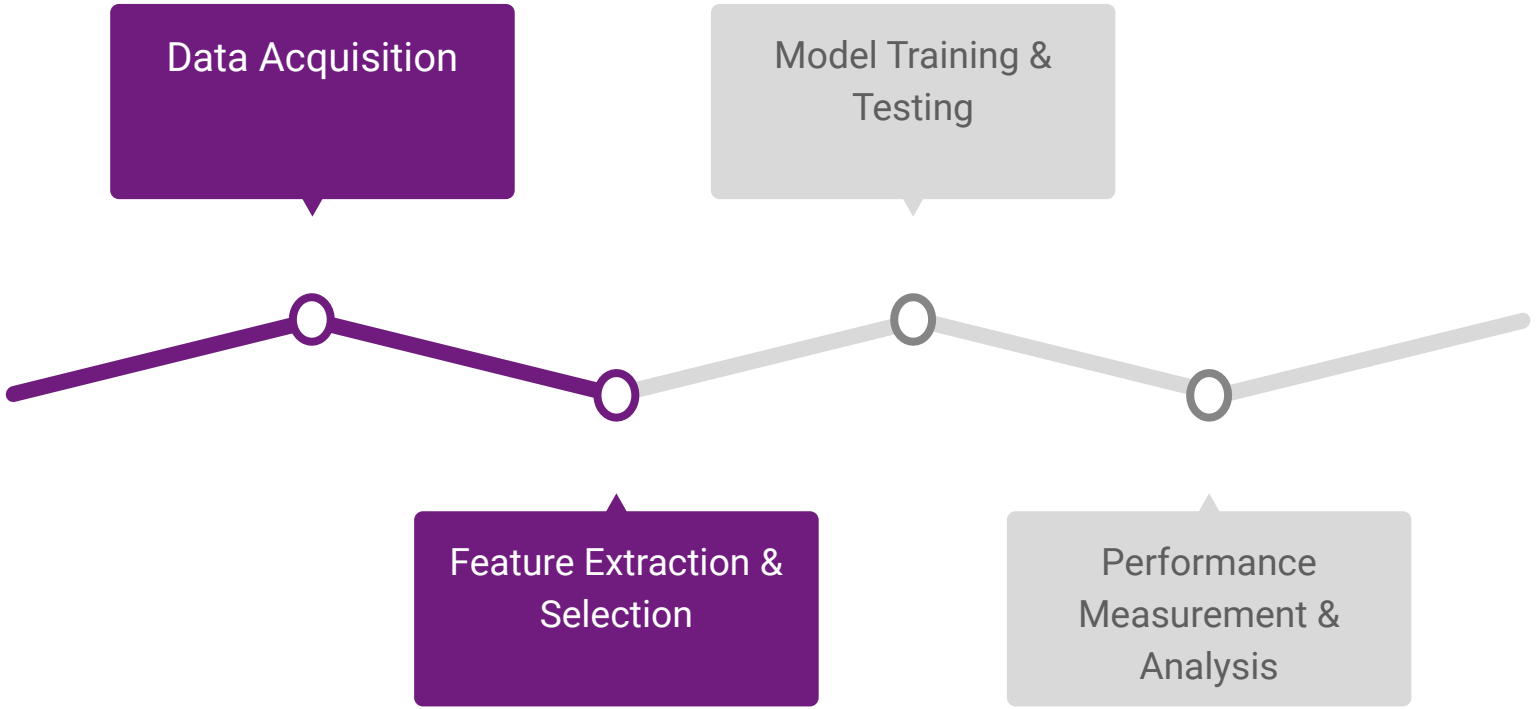
$$\partial \boldsymbol{a}^{[i-1]} = \boldsymbol{W}^{[l]^T} \cdot \partial \boldsymbol{z}^{[i]}$$

During this, the weights and biases of layer i are updated as follows

$$\boldsymbol{W}^{[i]} = \boldsymbol{W}^{[i]} - \eta \partial \boldsymbol{W}^{[i]}$$

$$\boldsymbol{b}^{[i]} = \boldsymbol{b}^{[i]} - \eta \partial \boldsymbol{b}^{[i]}$$

Architecture & Steps



Data Acquisition

```
KeyDown Key.down        01:10:2021:09:24:05:760632
KeyUp   Key.down        01:10:2021:09:24:05:923290
KeyDown Key.enter       01:10:2021:09:24:06:086599
KeyUp   Key.enter       01:10:2021:09:24:06:151741
KeyDown 'c'     01:10:2021:09:27:25:905440
KeyUp   'c'     01:10:2021:09:27:25:977029
KeyDown 'o'     01:10:2021:09:27:26:096065
KeyUp   'o'     01:10:2021:09:27:26:156241
KeyDown 'd'     01:10:2021:09:27:26:278919
KeyDown 'e'     01:10:2021:09:27:26:371860
KeyUp   'd'     01:10:2021:09:27:26:426673
KeyUp   'e'     01:10:2021:09:27:26:503062
KeyDown Key.enter       01:10:2021:09:27:26:708957
KeyUp   Key.enter       01:10:2021:09:27:26:770757
KeyDown 'n'     01:10:2021:09:27:51:787508
KeyUp   'n'     01:10:2021:09:27:51:911927
KeyDown 'e'     01:10:2021:09:27:51:949105
KeyDown 'w'     01:10:2021:09:27:52:028063
KeyUp   'e'     01:10:2021:09:27:52:079404
KeyUp   'w'     01:10:2021:09:27:52:153778
KeyDown 't'     01:10:2021:09:27:52:272753
KeyUp   't'     01:10:2021:09:27:52:362712
KeyDown 'o'     01:10:2021:09:27:52:475946
KeyUp   'o'     01:10:2021:09:27:52:526694
KeyDown 'n'     01:10:2021:09:27:52:668905
KeyUp   'n'     01:10:2021:09:27:52:738851
KeyDown Key.space       01:10:2021:09:27:52:748568
```

Fig. Raw Keystroke Data

```
('?\x16', 0.189), ('?', 0.513), ('\x16\x08', 97.591), ('\x16', 0.151), ('\x081', 0.435),
.485), ('5', 0.077), ('.S', 0.443), ('.', 0.075), ('SC', 0.2), ('S', 0.073), ('CI', 0.15
0.161), ('E', 0.076), ('NT', 0.166), ('N', 0.082), ('TI', 0.147), ('T', 0.056), ('IS', 0
', 0.071), ('T', 0.062), ('S ', 0.107), ('S', 0.097), (' H', 0.336), (' ', 0.099), ('HA'
VE', 0.123), ('V', 0.051), ('E ', 0.075), ('E', 0.065), (' D', 0.202), (' ', 0.096), ('D
 ('SC', 0.198), ('S', 0.077), ('CO', 0.202), ('C', 0.083), ('OV', 0.257), ('O', 0.062),
3), ('RE', 0.262), ('R', 0.086), ('ED', 0.217), ('E', 0.085), ('D ', 0.212), ('D', 0.089
, ('\x08', 0.743), ('\x08\x08', 0.036), ('\x08', 0.391), ('\x08\x08', 0.04), ('\x08', 0.
', 0.214), ('\x08', 0.746), ('\x08\x08', 0.165), ('\x08', 0.692), ('\x08\x08', 0.14), ('
x08\x08', 0.151), ('\x08', 1.052), ('\x08\x08', 0.823), ('\x08', 0.901), ('\x08T', 0.219
, 0.265), ('R', 0.087), ('AC', 0.135), ('A', 0.09), ('CJ', 0.243), ('C', 0.063), ('JE',
 ', 0.094), ('D', 0.109), (' \x08', 0.507), (' ', 0.069), ('\x08\x08', 0.152), ('\x08',
x08', 0.143), ('\x08', 0.222), ('\x08K', 0.482), ('\x08', 0.079), ('KE', 0.098), ('K', 0.
, 0.075), (' B', 0.25), (' ', 0.071), ('BU', 0.203), ('B', 0.059), ('UT', 0.186), ('U',
T', 0.063), ('ER', 0.227), ('E', 1.389), ('RF', 0.357), ('R', 0.076), ('FL', 0.095), ('F
 ('I', 0.087), ('ES', 0.165), ('E', 0.065), ('S ', 0.105), ('S', 0.087), (' T', 0.251),
06), ('H', 0.075), ('AT', 0.108), ('A', 0.907), ('T ', 0.137), ('T', 1.199), (' C', 0.416
092), ('A', 0.916), ('N ', 0.164), ('N', 0.066), (' T', 0.232), (' ', 2.009), ('TR', 0.01
0.111), ('A', 0.095), ('VE', 1.029), ('V', 0.075), ('EL', 0.088), ('E', 0.076), ('L ', 0
, 1.435), ('3', 0.076), (',0', 0.505), (',', 0.056), ('00', 0.164), ('0', 0.416), ('00',
 M', 0.239), (' ', 1.007), ('MI', 0.164), ('M', 0.768)]
```

Fig. Keystroke data after feature extraction

Feature Extraction

1.) We get 26 hold times for the letters A to Z
2.) We get 676 (26✖26) latency values for each pairing: AA,AB...ZY,ZZ

We take these 26 + 26✖26 = 702 values as a feature vector for one class and use it to classify from several classes. We take the average of all hold time and latency values for one particular key or one particular pair of keys as this helps in smoothing the values and returns an accurate representation of the several values.

Hold time tells us the length of the keypress or the time difference between the press time and the release time. It signifies the impulsive nature of the user.Latency tells us the time to switch from one particular key to the other. It signifies the switching and typing speed of the user.

RESULTS

Best Accuracy on the Test set was of **66.1%**, obtained when the random state was 99, maximum iterations were 500 and there were 2 hidden layers of sizes 100 and 25 respectively.

We performed hyper-parameter search over 2 parameters: Random State and Hidden Layers.

We tabulate our results in the table on the next slide.

Hidden Layers	Five Fold Test Accuracy		Random State Configuration	Five Fold Test Accuracy
100::25	**0.644**		10	0.638
200::100::25	0.616		25	0.600
200::50::25	0.611		40	0.622
200::50::50	0.627		50	0.605
300::200::50::50	0.600		76	0.622
300::200::20::20	0.616		**99**	**0.661**

REFERENCES

1. https://medium.com/machine-learning-researcher/artificial-neural-network-ann-4481fa33d85a
2. https://analyticsindiamag.com/a-beginners-guide-to-scikit-learns-mlpclassifier/
3. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

THANK YOU