CIS4930 - Project 2
Python Implementation of $1 Recognizer Extended ($1e)

Athish Rahul Rao,
University of Florida
Email: athish.rao@ufl.edu

25 April 2019

Abstract

This project implements an extension to the $1 algorithm for gesture recognition. The original algorithm proposed in the Wobbrock et al[1] paper, was converted into a deployable software written in python and javascript using primitive geometry and trigonometry. The original implementation used a recursive algorithm called the GSS. In this implementation, the idea is to do away with GSS and base the whole algorithm with centroids and vectors as the corner stone. The dataset used was the same as the one used in the original paper - Unistroke dataset. The accuracy was almost comparable to the results obtained with the original algorithm - over 96% accuracy when trained with more than 5 template of each gesture type. There was a significant drop in the computation time of the algorithm. The graphs comparing the performance can be seen in the results section.

1 Introduction

This project implements an extension to the $1 algorithm for gesture recognition. The extension proposed, was converted into a deployable software written in python and javascript. There are two parts of this project, Offline Recognition and Online Recognition. The former deals with the formal correctness verification of the developed model by splitting the available dataset into testing-training sets and then performing validation tests to check accuracy. In this paper, the performance of the model has been elaborately explained and some aspects have been compared to the results as obtained by the original algorithm. The online recognition is deployable part of the project, where one can articulate a single stroke gesture on the developed webpage and get the results as predicted by the model. The dataset used was the same as the one used in the original paper - Unistroke dataset.

2 Dataset Details

The dataset used for this project was the Unistroke Dataset used in Wobbrock et al[1]. Ten subjects were recruited. Five were students. Eight were female. Three had technical degrees in science, engineering, or computing. The average age was 26.1 (SD=6.4). For each of the 16 gesture types from Figure 1, subjects entered one practice gesture before beginning three sets of 10 entries at slow, medium, and fast speeds. Messages were presented between each block of slow, medium, and fast gestures to remind subjects of the speed they should use. Each stroke was stored as an XML file with the coordinates and timestamp.

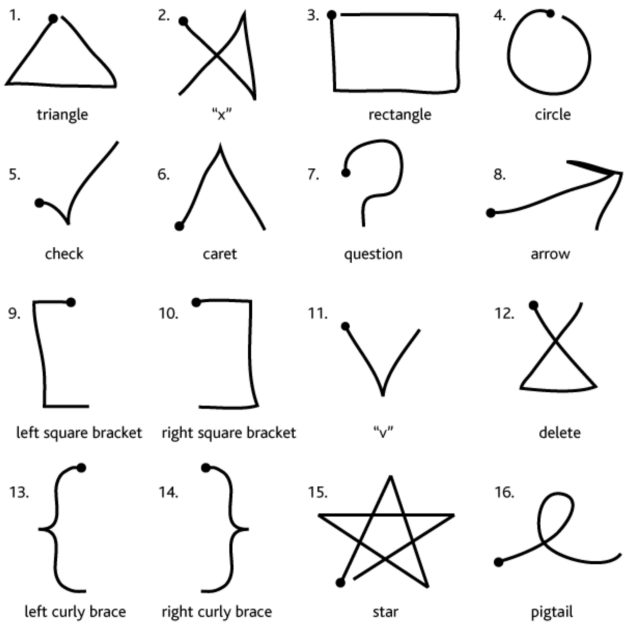

Figure 1: The set of 16 was used in our study of $1 algorithm

3 Implementation Details

3.1 System Details

The offline tests were run on a MacBook Pro 15" Late 2017 model - 2.8GHz quad-core Intel Core i7 processor and 16GB of 2133MHz LPDDR3 onboard memory. The online recognition is a web software that is deployed on a server. The server is a Google Cloud VM that runs on low specifications as the load on it would be small. The VM instance is a dual-core processor with 4GB DDR3 memory.

3.2 Extension Details

The motivation for the extension was to do away with the recursive Golden Section Search(GSS) algorithm which was used to match a candidate with a template by perturbing it by a certain degree. This being a recursive algorithm doesn't guarantee a fixed execution time for a given input. Also from a programming point of view it's better to avoid recursive algorithms. Now that we wanted to do away with GSS, we needed a substitute to do the same or more. The idea proposed for this was to use the centroid as the cornerstone for the recognition algorithm. After resampling the candidate, the centroid of the resampled points was computed. After that, the structure of the resampled points were changed - they were modified to represent their position relative to the centroid. This means that the structure would be innately position invariant. Then the scaling was done according to the original algorithm. For comparison, we no longer have the GSS, but we do have an array of vectors that are of the same dimension. The easiest way to compare them would be to subtract the vectors and then compute an L2 norm. This acted like the input difference to the score metric that was computed in the original algorithm. The next part of the algorithm is unchanged where we loop over the entirety of the template list and find the best score and then report the result.

3.3 Programs

This project is comprised of 5 programming files. Let's see each of them and their purpose in detail.

3.3.1 DollarOne.py

This is the crux of the project. This file implements the extended algorithm as defined the Extension Details section. Figure 2 elucidates the different classes that have been defined to structurally approach the implementation. Each black block represents a class. The hyphenated bullets are attributes of the class and the circular bullets are the methods defined in that class. The arrow represents the hierarchy of classes and flow of information. They do not represent any form of inheritance.
The most basic class is points that gets hold the x and y coordinates of a point. The stroke class is a set of points that form a stroke. The method names are consistent with the names used in the pseudocode defined in the Wobbrock et al[1] paper.

3.3.2 recognizeOffline.py

A symbol is a set of strokes of one type. The total class is a culmination of all symbols defined by $1. This file forms the cornerstone for verifying the correctness of our algorithm and how it fares against the original implementation and results. We perform **user-dependent** experiment. In this kind of experimentation, we iterate through the strokes made by every user and split the data into training and testing for each of those users. The splitting and choosing of data is further explored in this file. We vary the number of training samples(E) from 1 through 9 for every gesture articulated and always choose one of each as the testing dataset. The process of choosing these strokes can be biased and hence give unrealistic results. To normalize this behavior, we run the template and candidate choosing 100 times for every value of E. We collect two set of results here. One is how the overall user-dependent accuracy, averaged over all users, varies with the number of samples chosen for training. The second set of results corresponds to the accuracy or its complement - error percentage for every user. This is shows in Section 4.

3.3.3 recognizeOnline.py

This file is responsible for invoking the oneDollar.py when an online stroke is made and the request for recognition is made. The structure is very similar to the offline recognition, except for the fact that the number of templates used. Atleast 5 training example for every stroke is maintained and used for online recognition. This number was derived from the results that were obtained during the offline tests. But this is a pool of data that doesn't categorize strokes based on user.

3.3.4 index.html

This is the webpage that is used for online recognition. The webpage has a canvas that lets users to draw only one gesture at a time. On hitting the *recognize* button, the recognizeOnline.py is run in the background and the prediction is displayed on the screen. The points that are drawn by the user are recorded in a javascript array. This array, on hitting *recognize*, is stringifyied and sent via a POST request to the server hosting this webpage

through a jQuery and waits for a response. On getting a response, it displays it on the screen.

3.3.5 app.py

This file is responsible for hosting the webpage. On running this, a flask server is initiated and this sets up a communication between the different components of the software and configures an IP and port from which these contents can be accessed. This file is also responsible for redirecting requests made to the server. On receiving a POST request from the javascript file, the server reads the arrays, formats them into python arrays and send them into the recognize method of recognizeOnline.py for recognition. On receiving the response from recognizeOnline.py, the server sends the necessary information to the javascript file as a POST response.

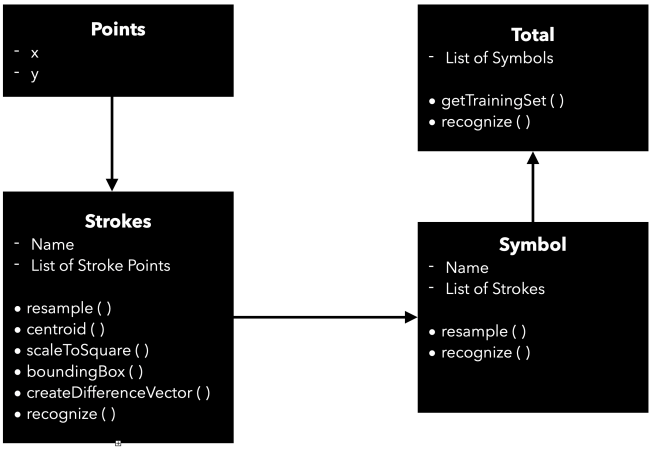

Figure 2: Code Structure - Classes with their attributes and methods

4 Results

4.1 Accuracy and Time Calculation

Individual Accuracy was calculated when the user-dependent testing was performed. The accuracy, in this experimentation is a metric of correctness. Each time, a prediction for a template was made and it was in correspondence with its original label, it was counted towards the correct-tally. After all the loops for a user was run, the ratio of correct-tally was taken against the total number of test cases run through the loop. This number, is what has been referred to as accuracy in this paper.
The next metric is the time-elapsed one. This was calculated using the "time" module of python. These time measurements are specific to my machine, whose specifications have been mentioned in section 3.1. The timer started when the random-100 loop was initialized and ended after the random-100 loop ended. This difference, is what has been referred to as time in this paper.

4.2 Analysis

We plotted two graphs that give us information regarding the accuracy of the $1e algorithm on running user-dependent experiments. The first graphs - Figure 3 is the accuracy of the strokes, averaged across all users, versus the number of training samples used by the users for those specific strokes. Figure 4 is the graph obtained when the accuracy results of the strokes, averaged across all users, from the original implementation were plotted. We see that the graph in Figure 3 takes a sharp improvement in accuracy after the introduction of the 5th template and doesn't change after that. This was the reason why five templates of each gesture was chosen as the base condition for the online recognizer. On comparison with the original implementation, we can see that the accuracy has taken a small hit. To investigate further, the analysis for time-elapsed was done. This can be seen in Figure 5 and 6. Figure 5 is the graph plotted for the original $1 implementation, while Figure 6 is plotted for the new algorithm proposed. There is a stark difference in the time taken. The time taken by the original to compare against two templates is less than the time taken by the $1e to run the dataset against nine template samples. The range of accuracy - [96.5, 98.8] reiterates the belief that people are consistent with their strokes. When we tested on the model trained with the same user's data, there is very strong correlation.

Figure 6, shoes us the per-user accuracy that was obtained when the new algorithm was run on each user. The average metric has been discussed above. The accuracy doesn't reach 100%. On checking the python output carefully, this seems to be an issue of rounding off. The accuracy values of over 99.7 have been rounded off to a 100.

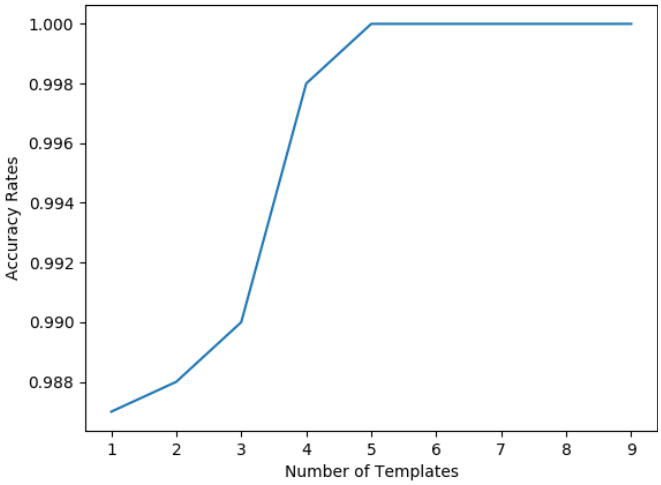

Figure 3: Overall Average - $1 Extended
(X, Y) = (Number of Training Sample, Accuracy Percentage)

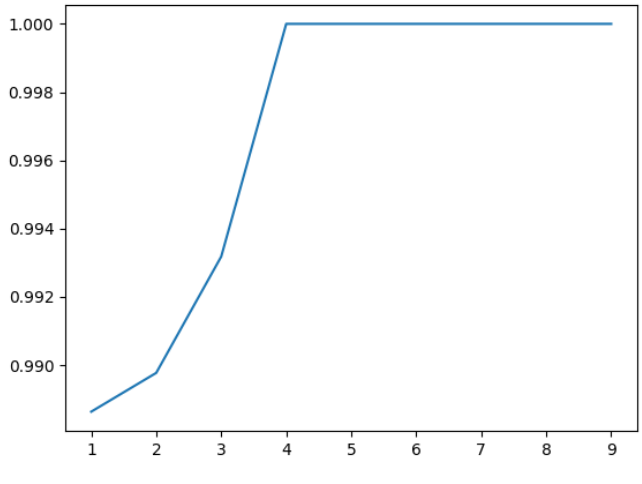

Figure 4: Overall Average - $1 Original
(X, Y) = (Number of Training Sample, Accuracy Percentage)

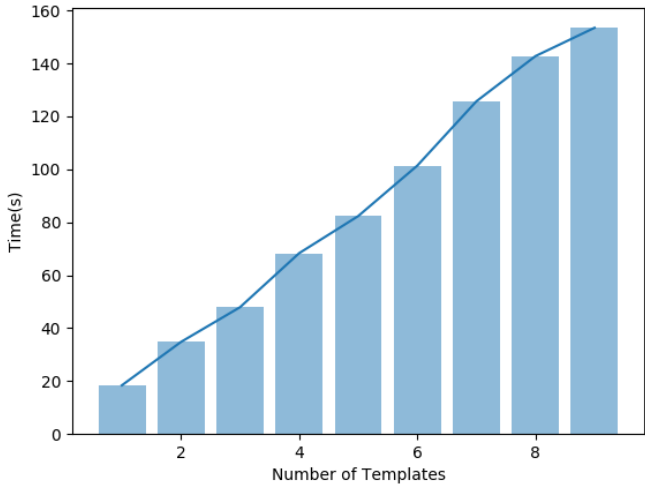

Figure 5: Original $1 Recognizer
(X, Y) = (Number of Training Sample,Time in s)

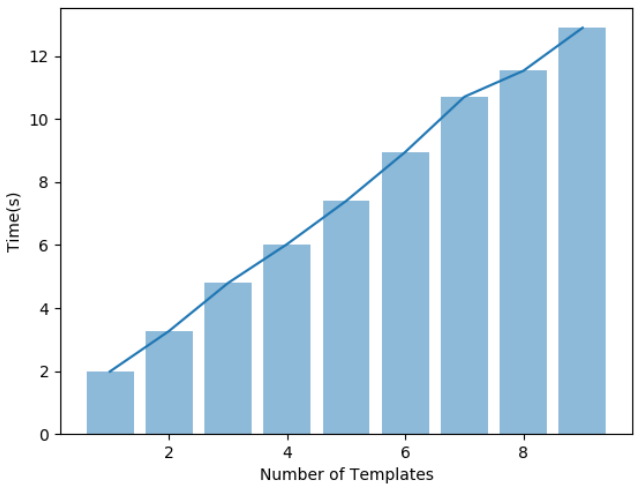

Figure 6: $1e Recognizer
(X, Y) = (Number of Training Sample,Time in s)

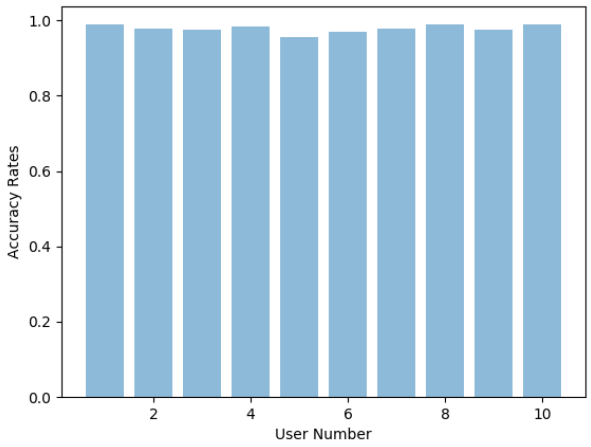

Figure 7: Accuracy of Each User
(X, Y) = (User Number, Accuracy Percentage)

References

[1] Wobbrock, J.O., Wilson, A.D. and Li, Y. (2007)
Gestures without libraries, toolkits or training:
A $1 recognizer for user interface prototypes
Proc. UIST '07. New York: ACM Press, 159-168.

A
Welch's t-test or Unequal Variances t-test

The t-test is any statistical hypothesis test in which the test statistic follows a Student's t-distribution under the null hypothesis. A null hypothesis is a general statement or default position that there is no relationship between two measured phenomena, or no association among groups. Testing the null hypothesis—and thus concluding that there are or are not grounds for believing that there is a relationship between two phenomena. Let us now apply this to the two set of accuracy values that we have obtained. We first assume that the variances of the two are different. This points us to Welch's t-test. There are always two set of hypotheses one has to present while performing a t-test. The first is a null hypothesis while second is an alternate hypothesis. We also have to set a significance level(α). Let us assume that the significance level, $\alpha = 0.05$. Now we move to the null hypothesis. Our null hypothesis is that the accuracy values have the same mean. $H_0 : \mu_{original} = \mu_{new}$, while the alternative hypothesis is that $H_a : \mu_{original} \neq \mu_{new}$. Now to find the t-value, we deploy the following formula:

$$t = \frac{\bar{X1} + \bar{X2}}{\sqrt{\frac{s_1^2}{N} + \frac{s_2^2}{N}}}$$

where $\bar{X1}$ and $\bar{X2}$ are the two means, s_1 and s_2 are the two SD's and N is the length of the two accuracy value vectors.
After setting up for the test, we calculate t.
For our case, the value of $\mathbf{t = 1.1038}$. We now need to evalute the p-value. The p-value answers the question, what is the probability from the t-distribution of getting something that is atleast this extreme. This means that we look at the two tail values of the t distribution and look for the probability that the value of t would like on either side of -1.1038 and 1.1038 in this curve. The way to do this is to use the cumulatively distribution function of the t-distribution and set the lower limit as negative infinity and the upper limit as -1.1038 and multiply the result

by 2 in order to account for symmetry on the other side of the x-axis. We should also keep in mind that the CDF needs the value of the degree of freedom. The df value here is assumed to be the most commonly form of df used, which the one less than the value of the number of elements = N - 1. P-value can be calculates using the following formula:

$$p - value = probability(|t| \geq 1.1038)$$

The **p-value = 0.1508**. This value is higher than our significance level $\alpha = 0.05$. This means that the null hypothesis holds. And we can discard our alternate hypothesis. This would mean that the means of the two accuracy from two varies tests are significantly related to one another. This reiterates the fact that this new algorithm can be used in the place of the original $1 algorithm for unistroke gesture recognition.