An Automated Process for Recognizing Handwriting with Accelerometers

Justin Cardona

John Abbott College

360-RES-AB Independent Research Project in Science Winter 2020

Course Coordinator: Roberta Šilerová

Research Supervisor: Chris Isaac Larnder of John Abbott College

Abstract:

The goal of this project is to use accelerometry techniques to recognize handwriting. To do this, one traces out individual block letters forming a desired phrase using an accelerometer. The user must maintain constant accelerometer orientation throughout drawing in order to maintain a constant coordinate system. The acceleration versus time data from each letter is then converted into position versus time data. This is done by performing discrete double integration on the acceleration data points. This is then used to construct images of each letter, which are then classified as letters. The accelerometry technique provides accurate visual reconstructions of writing but there is drift from the intended illustration due to lack of sensor precision. This is also because acceleration is recorded discretely but acceleration is experienced at a continuous rate in reality. By using more precise sensors with higher data sampling rates, discrete double integration can provide a more faithful depiction.

Classification was done using machine learning techniques. A machine learning model for recognizing letters of the English alphabet was trained using the EMNIST letters data set[1]. This model was then given the images produced by accelerometer writing and classifies each image as a letter. Although, after being exposed to 60 000 sample images, the most successful letter recognition model had an 90.47% accuracy in guessing generic handwritten letters, it is very poor for accelerometer-written letters (10.26% accuracy). This can be improved in the future by providing the program with a data set of accelerometer-written letters to learn. To conclude, an automated method was developed for recognizing handwriting using accelerometers.

Introduction:

The objective of this project is to develop and improve an automated method for recognizing handwriting using accelerometers. An accelerometer is a device that measures acceleration. The sensor records the force experienced by the device in 3 orthonormal directions and uses the mass of the device in order to return a value for acceleration. The device records acceleration at discrete times with a constant frequency. Each recording produces a data point that has 4 elements: time of recording and acceleration in each direction. The ith data point would be represented as:

$$p_i = \left(t_i, a_{1_i}, a_{2_i}, a_{3_i}\right)$$

To use this data in order to obtain information about the position of the device over time, one can perform discrete double integration. This process uses the definition of average velocity and acceleration in the following forms:

$$x_i = x_{i-1} + v_i(t_i - t_{i-1})$$

$$v_i = v_{i-1} + a_i(t_i - t_{i-1})$$

Assuming that initial position is at the origin and initial velocity is zero, one can use the above calculations recursively through all acceleration data points to determine the position with time.

In order to recognize handwriting, a machine learning method is used. Specifically, a Multi-Layer Perceptron neural network is implemented[2]. In the case of this project, the network learns about letters by being exposed to several samples and attempting to classify this input information as letters. The network takes information about individual letters as input neurons. For example, a 28 by 28-pixel grey scale image provides 784 neurons, each of which is assigned a grey scale value. These constitute the first layer; the last layer is of size 26 since there are 26 letters in the English alphabet. Over the span of one iteration of training, the neural network randomly assigns the input layer values to values of neurons in in the next layer. These values are then assigned to values of neurons in in the next layer. This is done until the last layer is reached and an output is produced. The output is then compared with the label of the image to determine if it was correct. Neurons are then given weights and biases in order to increase accuracy. The network then uses a testing data-set to determine its accuracy. The network repeats this process for all images in the training data set. Over many iterations of this process, the weights and biases are updated in order to improve the model's accuracy.

Materials and Methods:

A dataset was developed using accelerometer data. Using the sensors from an iPhone 7 and a Google Pixel 4 with a 30 Hz sampling rate, each letter was traced out individually and the data was saved as a ".csv" file. The alphabet was drawn and saved 15 times in this way. Discrete double integration was used to convert the acceleration data in each file to position data. The position data was graphed to produce a Gray-scale 28x28 pixel image of each letter. Another data set was then prepared to train the neural network. It was the "letters" data-set from EMNIST. The first 60 000 images were allocated to the training set and the last 10 000 images were allocated to the testing data-set. The Multi-Layer Perceptron used was that of Sci Kit Learn's Libraries. It was set to a default of 3 hidden layers of size 100, 50 maximum iterations, alpha of 0.1, the sdg solver, a tolerance of 0.0001, 1 random state, and an initial learning rate of 0.1. Beginning with the default state, the alpha was set to 0.0001, and the layer size was set to 150, the number of layers was set to 1 and training was conducted, and the model was saved. This was repeated with 1, 2, 4, 5, 6, 7, 8, and 10 layers. Using the default configuration, training and saving of a model was done with layer sizes of 10, 50, 100, 150, 200, 250, and 300. Using the default configuration, training and saving of a model was done with alpha values of 0.0001, 0.001, 0.01, 0.1, 1, and 10. Using the default configuration, training and saving of a model was done with the sgd, adam and lbfgs solvers. Using the default configuration, training and saving of a model was done with 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10 random states. Using the default configuration, training and saving of a model was done with initial learning rates of 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, and 0.8. The best performing model ran a prediction on the accelerator-drawn data-set.

Results and Analysis:

Number of Layers	Cost after 50 iteration	Training Set Score	Testing Set Score
1	0.06271	0.977	0.878
2	0.12541	0.961	0.879
3	0.13884	0.959	0.884
4	0.13112	0.962	0.894
5	0.12643	0.960	0.896
6	0.12152	0.961	0.896
7	0.12367	0.954	0.889
8	0.11711	0.964	0.896
9	0.13175	0.960	0.896
10	0.13340	0.963	0.900

Table 1: Effect of Hidden Layer Sizes on Prediction Accuracy

Size of Layers	Cost after 50 iteration	Training Set Score	Testing Set Score
10	1.05901	0.676	0.665
50	0.28282	0.913	0.860
100	0.16805	0.940	0.873
150	0.13994	0.961	0.885
200	0.10639	0.962	0.889
250	0.08865	0.968	0.891
300	0.06954	0.980	0.901

Table 2: Effect of Hidden Layer Amount on Prediction Accuracy

Alpha	Cost after 50 iteration	Training Set Score	Testing Set Score
0.0001	0.17335	0.946	0.883
0.0010	0.18441	0.939	0.870
0.0100	0.22131	0.952	0.891
0.1000	0.36840	0.941	0.902
1.0000	0.97044	0.861	0.854
10.000	2.79713	0.303	0.307

Table 3: Effect of Alpha Value on Prediction Accuracy

Solver	Cost after 50 iteration	Training Set Score	Testing Set Score
sgd	0.36840	0.942	0.902
adam	2.82995	0.120	0.120
lbfgs	0.84471	0.747	0.743

Table 4: Effect of Solver on Prediction Accuracy

Random states	Cost after 50 iteration	Training Set Score	Testing Set Score
0	0.37104	0.942	0.905
1	0.36840	0.942	0.902
2	0.37133	0.941	0.906
3	0.37227	0.936	0.899
4	0.36968	0.936	0.902
5	0.37380	0.941	0.902
6	0.37608	0.930	0.890
7	0.37038	0.935	0.901
8	0.36934	0.940	0.902
9	0.37662	0.937	0.899
10	0.37924	0.935	0.899

Table 5: Effect of the Number of Random States on Prediction Accuracy

Learning Rate	Cost after 50 iteration	Training Set Score	Testing Set Score
0.1	0.36840	0.942	0.902
0.2	0.51979	0.903	0.881
0.3	0.71986	0.864	0.851
0.4	0.94702	0.834	0.825
0.5	1.23424	0.722	0.705
0.6	3.26383	0.039	0.040
0.7	3.26564	0.039	0.040
0.8	3.26545	0.039	0.037

Table 6: Effect of the Initial Learning Rate on Prediction Accuracy

Discussion:

Increasing the number of layers increases the value of the cost function, decreases the training accuracy, and increases testing accuracy (Table 1). Increasing the size of layers decreases the value of the cost function, increases the training accuracy, and increases testing accuracy (Table 2). Increasing the value of alpha decreases the value of the cost function. A value of 0.1 produced the highest testing set score, 0.901500 (Table 3). The sgd solver performs the best when compared to the adam and lbfgs solvers (Table 4). Changing the number of random states had little effect on the accuracy of the model. The accuracy decreases slightly as the number of random states increased (Table 5). Increasing the initial learning rate increases the value of the cost as well as decreasing the training and testing scores (Table 6).

When the model that performs best in testing data sets, the zero random state model, is used to classify the accelerometer-drawn images the score is 0.102564. This poor score might be since the EMNIST letters look very different from accelerometer-drawn letters. This is caused by several factors. Acceleration is experienced at a continuous rate but is sensed discretely. In addition, sensor inaccuracy causes drift when using discrete double integration to calculate position. Of course, this could be improved by using more accurate accelerometers with higher sampling rates. However, something to be investigated in the future is whether using accelerometer data as a training/testing data set, in the form of visual reconstructions or the raw acceleration data, would result in higher accuracy for accelerometer-drawn letters.

Conclusion:

An automated method for recognizing accelerometer-drawn handwriting was developed. It demonstrated a 90.4700% accuracy for recognizing hand-drawn letters but a 10.2564% accuracy for accelerometer-drawn letters. For recognizing hand-written letters with a Multi Layer Perceptron Classifier, optimal accuracy can be obtained by using many layers of large sizes, an alpha value of 0.1, the sgd solver, 0 random states, and a low initial learning rate. It is yet to be seen when using an accelerometer drawn training and testing data set to develop a model improves the accuracy when classifying accelerometer-drawn letters.

Acknowledgements

This research would not have been possible without this course’s coordinator Roberta Šilerová, my research supervisor Chris Larnder and my research partners Jerome Chabert, Hoang Anh Nguyen and Adamo Orsini.

References:

[1]Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017). EMNIST: an extension of MNIST to handwritten letters. Retrieved fromhttp://arxiv.org/abs/1702.05373

[2]sklearn.neural_network.MLPClassifier. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html