Binary Classification of Hand Motions as Door Opening/Non Door Opening

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Brandon Ou

Spring, 2024

Technical Project Team Members

Matthew Hattrup

Richard Wang

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Aaron Bloomfield, Department of Computer Science

Binary Classification of Hand Motions as Door Opening/Non Door Opening

CS4991 Capstone Report, 2023

Brandon Ou
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
bro9tn@virginia.edu

ABSTRACT
With the advent of smart homes, there is a need to detect human behavior, such as door opening, to trigger various smart home features. Machine learning (ML) based on smartwatch sensor data can be leveraged to detect if a door is being opened. Developing this solution required an ML tool: WEKA, a smartwatch sensor tool: WADA, and python & java scripting skills involving data analysis and manipulation. This class project employed the aforementioned tools to develop three ML models— random forests, J48, and SVMs—to utilize smartwatch sensor data and detect door opening. Of the models built using the three methods, the random forest classifier performed the best, with a 96.1% accuracy. In the future, more and a larger breadth of smartwatch data would be useful in validating the models. Results from this experiment indicate a high probability that, in the future and with more development, live smartwatch sensor data can be used to operate smart home functionality.

1. INTRODUCTION

As smart watches become increasingly popular, there will be more demand for software to extend their functionality. Smart watches contain many types of sensors that can be used to sense their environment, such as accelerometers—sensors that can calculate the watch's acceleration in any direction – and gyroscopes – sensors that store the watch's tilt. This work used the ASUS Zenwatch 2 smartwatch and its accelerometer and gyroscope sensor data to determine whether the person wearing the watch was opening a door.

2. BACKGROUND

ML is a general purpose term used to describe many types of problem solving that involve computers. A variant of ML known as binary classification determines if some action or data is indicative of one class of data or another, such as door opening and non-door opening.

Most ML algorithms necessitate a model, which is a numerical representation of a problem that takes in numerical variables that quantify the environment, performs functions on these variables, and outputs value(s) to be interpreted by people. When using a model for classification, the model is known as a classifier. Classifiers require sample scenarios, also known as training data, to learn about the environment that they model. Computation is performed on classifiers using training data to improve the classifier performance. For example, training data for door opening may be gyroscope values over time for someone opening a door. This would help the model understand what data of door opening may look like. Two ML algorithms I used were decision trees and support vector machines (SVMs).

2.1 Decision Trees

At a high level, decision trees are a collection of hierarchical nodes. Each node compares the data with some criteria; if the criteria is met, one child path will be taken, otherwise another would be. This recursive pattern occurs until one gets to a node without children, which leads to some classification. For example, a decision tree might be given accelerometer data and determine if a door is opening based on the y-direction acceleration at some set of timestamps. Using criteria about this acceleration data, the decision tree will take some child path and use criteria about the x-direction acceleration data. This would repeat until the decision tree determines whether a door is being opened.

However, decision trees can suffer from making incorrect decisions given data, which has motivated many improvements, leading to the development of random forests and J48 decision trees. These models use more sophisticated techniques to implement the flow of decision trees. To learn more about these trees, see articles from IBM and Khanna.

2.2 SVM Classifiers

If one imagines every possible environment scenario as a point on a plot, SVMs create a boundary by which points on one side of the boundary are classified as one class, and vice versa. For example, the accelerometer data from one hand motion would correspond to some point in a high dimensional plot; an SVM would draw some boundary based on training data passed to it, and use this boundary to determine which class the data falls into.

3. RELATED WORKS

This project was inspired by the field of cyber-physical systems, large systems that require communication between many hardware or software components and interact with the environment. One recent form of a cyber-physical system is the smart home, a type of house that uses computer systems to autonomously control its environment (e.g. lighting, climate, and household appliances). Such a system would likely require functionality in a smart watch to sense a human's actions, leading to the development of smart watch applications that can detect human actions.

Similar concepts have been developed, by Kunwar, et al. (2022), for example. They developed a proof of concept for classifying general hand movements (e.g. drinking water from a cup, jogging in place) via a smartwatch's various sensors (Kunwar et al., 2022) with strong results. A study by Sehirli & Alesmaeil (2022) performed binary classification of face-touching using smartwatch sensors, also with strong results. In these cases, strong results refer to the models having low rates of error and high probabilities of correct classification.

4. PROJECT DESIGN

Before using the model, training data would need to be collected and prepared to be fed to models.

4.1 Data Collection

The best way to capture training data was to use the smart watch and perform hand gestures/motions. I collected five minutes of hand gestures that did not involve opening doors, followed by 60 separate intervals of opening doors. Data collection required the use of the WADA app, an app that records the watch's sensor data and can be exported to another device for data analysis.

4.2 Data Processing

Because door opening is a short action, data would need to be clipped into short time intervals to mirror the short time of door opening.

4.2.1 Truncation

I first began by removing the first and last 25 data points of all training data to remove a lot of noise that usually occurs in the beginning and end of recordings.

4.2.2 Sliding Window

After exporting data to my machine, I tested various clipping schemes to test their effectiveness. The training data was relatively large, with each sample holding hundreds of datum. To reduce this bulk, I used a "sliding window" to collapse the data; the sliding window would take every ten data points and collapse them into one data value, or *feature*, containing mean, median, root mean square, standard deviation, and variance of the accelerations in the x, y, z directions. I then tried using a sliding window of 20, 30, and 40 points to compare the efficacy of sliding windows for various sizes. These were tested with a random forest model; the model was likely not as important as the size of the sliding window, so any model should have been sufficient for testing sliding window size.

Once the best sliding window size was determined, the next step was to determine the best model of the scenario from the model choices: J48 Decision Tree, Random Forest, and SVM classifier.

4.2.3 Model Selection

ML often suffers from an issue of overfitting, which happens when a model draws too many conclusions from training data. This can result in the model making incorrect classifications about new data because it drew false conclusions about old training data.

To reduce the likelihood of overfitting, I constrained the number of features (mean, median, standard deviation, etc.) that a model used during classification. This was done by checking how much accuracy each feature added to the total accuracy and selecting the best. I stopped adding features once the best unused feature would only add 1% accuracy to the total. For example, if the combined usage of mean x-acceleration, standard deviation of y-acceleration, and mean z-acceleration contributed to a 90% accuracy with the best unused variable only contributing .5% accuracy, I would not let the model use that variable and settle with the original three features. Once I performed this with every model (J48, Random Forest, SVM), I selected the best model.

5. RESULTS

Testing various sliding window sizes against the random forest model yielded Table I, shown below.

Window Size (# Data Points)	10	20	30	40
Model Accuracy (%)	94.0	94.3	**95.2**	94.8

Table 1: Window Size vs. Max Random Forest Accuracy

A sliding window of 30 data points seemed to yield the best model in terms of accuracy, though future work could be done to support this with more evidence.

Once the sliding window was set, the model was to be selected while minimizing overfitting. Table 2, 3, and 4 show the three variables used by the various models to achieve their non-overfit accuracies along with the increase in accuracy that they contributed. Note that the variable names are of the acceleration in the given direction. Also note that all values are percentages. The bolded percentages are the total non-overfit accuracies.

Feature Added	Median x	Median y	Mean y
Total Accuracy	88.2	93.2	**94.9**
Change in Accuracy	88.2	5.0	1.7

Table 2: Feature Added vs J48 Accuracy

Feature Added	Median x	Mean z	Median y
Total Accuracy	83.1	93.0	**96.1**
Change in Accuracy	83.1	9.9	3.1

Table 3: Feature Added vs Random Forest Accuracy

Feature Added	Med. x	RMS y	STD y	Mean z	Mean y
Total Acc.	76.5	80.5	83.8	85.2	**86.4**
Change in Acc.	76.5	4.1	3.2	1.4	1.3

Table 4: Feature Added vs SVM Accuracy

Based on these results, the best classifier was the random forest classifier with an accuracy of 96.1% and a sliding window of size 30.

6. CONCLUSION

As the world becomes more digitized, there will be a growing demand for increased convenience. People will ask more of artificial intelligence, more of cyber-physical systems, and more of large technology systems in general. Smartwatches may be adopted to give systems real time sensor data to classify actions that wearers are performing. As a result, there is a need to develop ML models that can perform inference based on sensor data. I developed the door opening classifier that could power the functionality of automatic smart home doors to test the viability of smartwatch usage. Because of the strong results, it is highly possible for smartwatches to be used in conjunction with smart homes and other technology to classify human activity and bring about further convenience.

7. FUTURE WORK

Because the door opening classifier is more of a proof-of-concept, several steps will be required to bring this product to the market. For example, advances will need to be made to the model. While the accuracy of the watch is high, more research may be needed to ensure that the watch is more stable. Research could be done by exploring other types of sensors, aggregates of sensor data (e.g. modes, max, min) or different ML models (e.g. large neural networks, convolutional neural networks, long short-term memory). Because of the limited time given for this project, there was not too much data prepared. Being able to allot more resources for data collection would also greatly benefit the classifier's performance.

Once the product is more accurate and powerful, the model would likely need to be deployed to some cloud provider to be used in real time. Smartwatches would also require some software to be able to communicate with a model being deployed. With these changes, I am confident that the door opening classifier and similar models would be viable for use in conjunction with a smart home or some other cyber physical system.

8. ACKNOWLEDGEMENTS

I would like to thank my two team members who helped collect data, perform data preparation, data cleaning, and analysis: Matthew Hattrup and Richard Wang. I would also like to thank the professor who oversaw the development of this classifier, Dr. Jack Stankovic, who provided various code

snippets and tutorials that were crucial for data preparation and analysis.

REFERENCES

IBM. (n.d.). *What is Random Forest?*. IBM. Retrieved November 1, 2023, from https://www.ibm.com/topics/random-forest

Khanna, N. (2021, August 18). *J48 classification (C4.5 algorithm) in a Nutshell*. Medium. Retrieved November 01, 2023, from https://medium.com/@nilimakhanna1/j48-classification-c4-5-algorithm-in-a-nutshell-24c50d20658e

Kunwar, U., Borar, S., Berghofer, M., Kylmälä, J., Aslan, I., Leiva, L. A., & Oulasvirta, A. (2022). Robust and deployable gesture recognition for smartwatches. *27th International Conference on Intelligent User Interfaces*, 277–291. Retrieved October 12, 2023, from https://doi.org/10.1145/3490099.3511125

Sehirli, E., & Alesmaeil, A. (2022). Detecting Face-Touch Hand Moves Using Smartwatch Inertial Sensors and Convolutional Neural Networks. International Journal of Intelligent Systems and Applications in Engineering, 10(1), 122–128. Retrieved October 12, 2023, from https://doi.org/10.18201/ijisae.2022.275