Precision and Information Gain per Input Variable in Gesture Recognition with Machine Learning

Estevam Nicolas Chen, Jucimar Maia da Silva Junior, Ricardo da Silva Barboza
Universidade do Estado do Amazonas - Engenharia de Computação - UEA
Manaus, Amazonas - Brazil
enc.eng@uea.edu.br, jucimar.jr@uea.edu.br, rsbarboza@uea.edu.br

***Abstract*—A way of interacting with computers is through hand gestures, and a method of recognizing said gestures is through machine learning. However many machine learning algorithms exist. As such, this paper compares the effectiveness of various machine learning gesture recognition algorithms, as well as the information gain of each input variable. Using a previously created dataset to train various algorithms with Orange and making use of a combination of Unity, a python program and Leap Motion, we managed to reach a precision of over 99%.**

***Keywords*-Virtual Reality, Leap Motion, Motion Capture, Machine Learning**

I. Introduction

We communicate in a variety of ways both with computers and each other. As such, many forms of HCI (Human-Computer Interaction) have been developed. This paper addresses a relatively more novel HCI method: gesture recognition.

Gesture recognition has a number of applications, specially with the development of VR (Virtual Reality) and AR (Augmented Reality) technology. For example, it can be used in games and applications that use AR and VR, but it can also have other applications, such as an approach to control robots. It can also be used to identify sign languages, such as ASL (American Sign Language) and LIBRAS (Língua Brasileira de Sinais, the Brazilian Sign Language)

Hand gestures can be divided in two categories: Static and dynamic gestures. While in static gestures the position of the hand and fingers denotes the sign, a dynamic gesture is denoted by moving the hand and fingers in addition to the position. This paper, however, focuses specifically on static hand gestures.

The hand gesture recognition can also be divided in two categories: appearance-based and 3D-model-based algorithms. While the former uses the data acquired from the silhouette or contour of the input images, the latter uses volumetric or skeletal data, or a combination thereof. Leap Motion uses skeletal data, therefore the data used in this paper belongs in the aforementioned category.

The Leap Motion controller is a USB device that can be mounted onto a virtual reality headset or placed on a physical desk, facing upwards. The controller utilizes two IR (Infrared Light) cameras as well as three IR emitters [1].

Orange is a machine learning and data mining framework. It improves the workflow for machine learning researchers, which eases the process of prototyping new algorithms and experimental procedures [2].

In this work, we present a comparison between five different machine learning algorithms: Decision Tree [3], Random Forest [4], kNN (K-Nearest Neighbors algorithm), SVM (Support Vector Machines) [5] and Naive Bayes [6].

In machine learning, information gain is measured by the Kullback-Leibler divergence (also called relative entropy) [7]. It represents the amount of information gained when a random variable is sampled. This work will also make a comparison of the information gain of the variables available in the dataset.

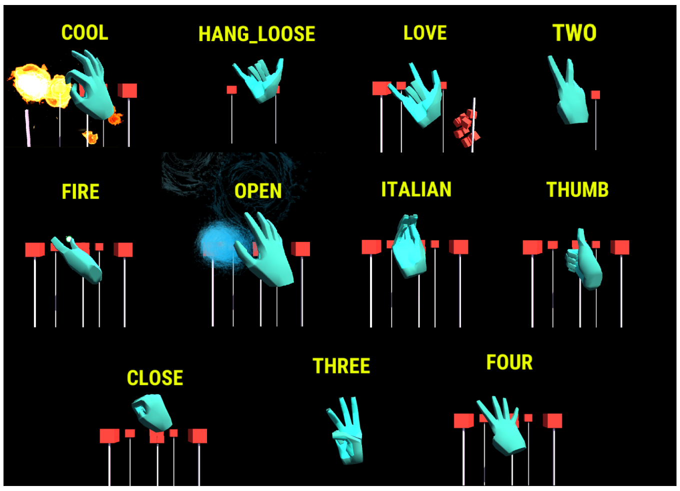

Figure 1. The 11 classes used in this work: “CLOSE”, “COOL”, “FIRE”, “FOUR”, “HANG_LOOSE”, “ITALIAN”, “LOVE”, “OPEN”, “THREE”, “THUMB”, “TWO”

A similar work can be seen in [8], which introduced an application with gestural hand control using leap motion for medical visualization. The application is based on spatial feature descriptors of the positions of fingertips and palm center. The features are extracted and fed into a support vector machine classifier to recognize the performed gestures. In this work, the experimental results demonstrated a high accuracy rate of about 81%.

II. Framework

To train the machine learning algorithms, the dataset recorded in [9] was used. However, the dataset has been updated since its creation, even including new classes. As such, the results may differ from the ones found by [9]. The data contains the normalized position of the tip of the 5 fingers, as well as the angles between adjacent fingers, for a total of 19 variables, classified in 11 classes, each representing a different hand gesture, as seen in figure 1.

The Orange machine learning and data mining toolkit [10] was used to train the various machine learning algorithms. The general configuration utilized on Orange can be seen in figure 2.

The data was loaded into a table element. From that table 70% of the samples provided data for the algorithms as input, and that data is used as training data. Having the algorithms trained, the program uses the remaining 30% as test data. The score can then be seen, as well as the information ratio of each variable.

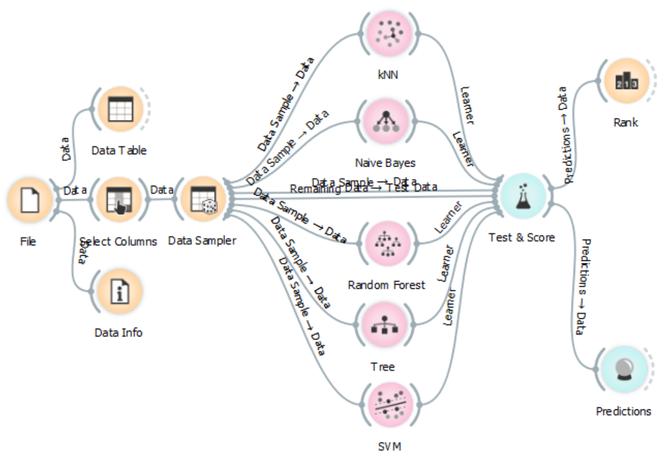

Figure 2. Orange configuration.

The kNN algorithm was configured is such way it had 5 euclidean, uniformly weighted neighbors. The decision tree algorithm was induced to generate binary trees, with a maximun tree depth of 100. The Random Forest has a total of 10 trees, and each of which are induced to be binary trees. The SVM algorithm has a cost (C) = 1,00 and a regression loss (ϵ) = 0,10.

Of the dataset, 70% was randomly sampled for the training step, while the remaining 30% was used in the testing step.

III. Experimental Results

After the algorithms are trained, the results of the testing step can be seen in figure 3. Overall, the random forest algorithm had the best results.

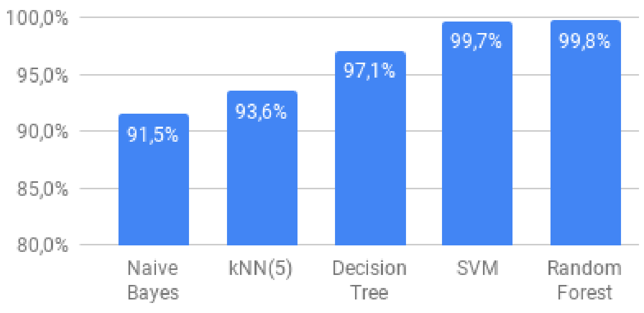

Figure 3. Average precision over algorithms.

In figure 5, the results divided per class in each algorithm can be seen. In the random forest algorithm, the “FIRE”, “COOL” and “OPEN” gestures have a slight but noticeable lower precision when compared with the average value, at 99.5%, 99.7% and 99.7% respectively.

The results of the chosen algorithm was then saved into an output file. That file is used in a data loop between a Python and a Unity program, which can be seen in figure 4. The python program first loads the model contained in the file and creates a socket running on localhost, through which the two programs can communicate with each other. The Unity project then sends the Leap Motion input to the python program, while the latter answers with a prediction based on the received data. Finally, as shown in figure 1, Unity creates effects as needed.

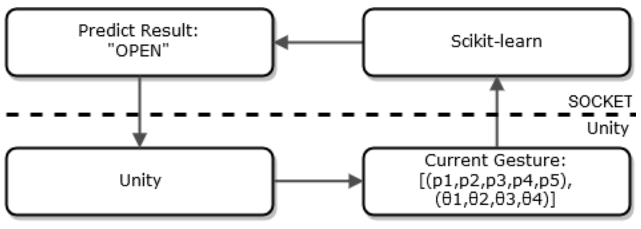

Figure 4. Data loop between Python and Unity.

It should be noted that while the results presented in [9] show that the Decision Tree algorithm has better results than the SVM, the opposite is true in this dataset. This is likely due to the fact the dataset was updated to include more classes.

From the trained algorithms, it is also possible to access the information gained from each variable. The information gain per variable can be seen in figure 6.

Noticeably, the Y position of the pinky finger holds the most information, followed by the angle between the pinky

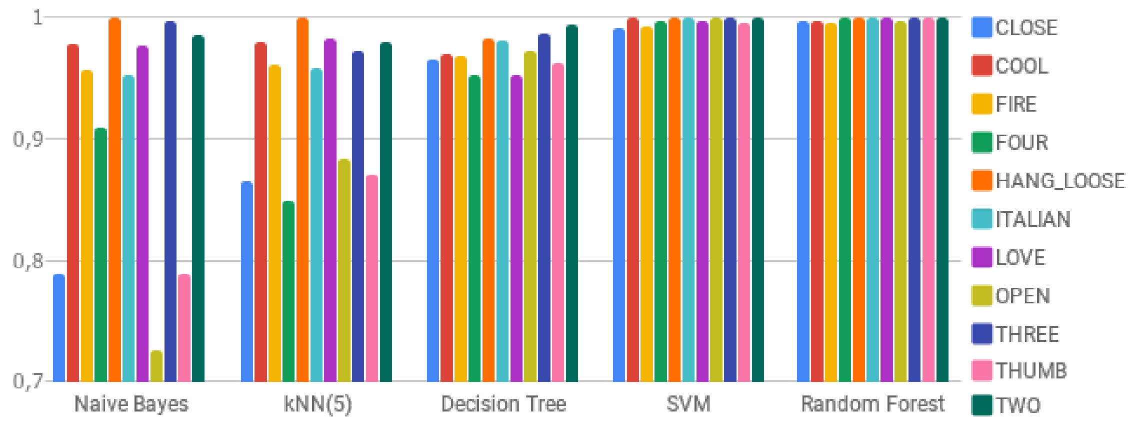

Figure 5. Average precision over algorithms per class.

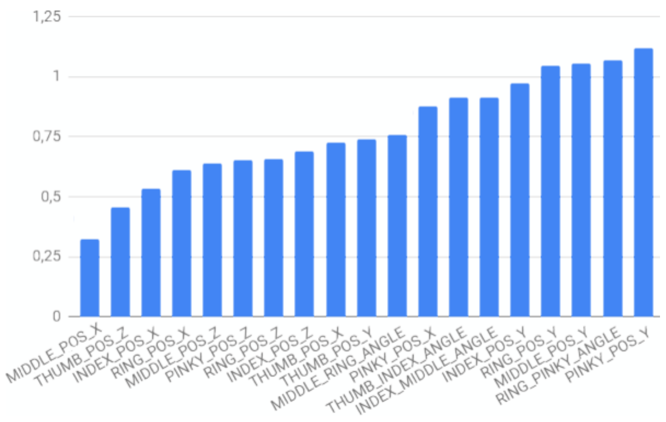

Figure 6. Information gain per input variable.

finger and the ring finger, while in third place, we have the Y position of the middle finger.

In contrast, the data that holds the least information is the middle finger X position, followed by the thumb Z position.

IV. Qualitative Testing

Having Leap Motion configured for head-mounted devices, and Unity and the python program set up, a series of qualitative tests were effected to assess the quality of the algorithms used in this paper. For the qualitative tests, 2 algorithms were compared: the one used in [9], called algorithm 1, and the Random Forest, called algorithm 2.

The testees were instructed on how Leap Motion works, including they only use the right hand and always remain in the field of view of the device.

The dataset was recorded having the Leap Motion controller strapped to the head and as such the testees were also instructed to hold the controller with the left hand in a similar position while performing the actions.

The testees were then asked to perform a series of gestures.

A total of 5 volunteers evaluated the 2 algorithms and all 5 came to the conclusion that the second was the superior algorithm, one commenting that it better identifies their intention, and another commenting that the “FOUR” gesture was better identified in the second algorithm.

However, during the tests, some inconsistencies were observed, specially in the “FIRE” and “THREE” gestures, which caused certain difficulty in performing some effects. In particular, that the “FIRE” gesture put the index finger in such a way the Leap Motion Controller cannot correctly see it. In addition to this, that was the gesture that had the lowest precision out of all the classes.

Conclusion and Future Work

In this paper, we performed a study on gesture recognition using a Leap Motion sensor and the Orange data mining toolkit.

Taking in consideration solely the final precision, the Random Forest algorithm has the overall best results, with qualitative tests agreeing with such a conclusion.

Still, some inconsistencies were observed while the tests were performed, particularly in the “FIRE” and “THREE” gestures, which may call for a more thorough dataset, or perhaps from a different angle.

Furthermore, study of how dynamic gestures behave ought to be considered, especially how they interact with normalization, as they can also be used for a variety of applications such as sign language recognition and virtual reality and augmented reality.

Acknowledgments

The authors would like to thank UEA, UEA/EST Ludus Lab and Elgin (Project Veraciti) for supporting the development of this work.

References

[1] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler, "Analysis of the accuracy and robustness of the leap motion controller," *Sensors*, vol. 13, no. 5, pp. 6380–6393, 2013.

[2] J. Demšar, B. Zupan, G. Leban, and T. Curk, "Orange: From experimental machine learning to interactive data mining," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2004, pp. 537–539.

[3] D. Yao, M. Jiang, A. Abulizi, and X. You, "Decision-tree-based algorithm for 3d sign classification," in *Signal Processing (ICSP), 2014 12th International Conference on*. IEEE, 2014, pp. 1200–1204.

[4] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[5] S. R. Gunn *et al.*, "Support vector machines for classification and regression," *ISIS technical report*, vol. 14, no. 1, pp. 5–16, 1998.

[6] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, 2001, pp. 41–46.

[7] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[8] S. Ameur, A. B. Khalifa, and M. S. Bouhlel, "A comprehensive leap motion database for hand gesture recognition," in *Sciences of Electronics, Technologies of Information and Telecommunications (SETIT), 2016 7th International Conference on*. IEEE, 2016, pp. 514–519.

[9] I. A. Stinghen Filho, B. B. Gatto, J. L. d. S. Pio, E. N. Chen, J. M. Junior, and R. Barboza, "Gesture recognition using leap motion: A machine learning-based controller interface," in *Signal Processing (ICSP), 2014 12th International Conference on*, 2018.

[10] U. of Ljubljana. (1996) Orange. [Online]. Available: https://orange.biolab.si/