Human-Computer Interaction Group

Leibniz Universität Hannover

Mobile Interaction
Summer 2024

Prof. Dr. Michael Rohs, Shashank Ahire, M.Sc.

Assignment 12

All exercises that are not explicitly declared as group tasks must be done individually and handed in individually. Identical submissions are treated as plagiarism. Plagiarism may lead to loss of exam bonus points.

You can submit the solution to this task in English or German until Wednesday, July 3, at 23:59 via https://assignments.hci.uni-hannover.de. Create a pdf file that contains the text and images of your solution, name it "Assignment-12-<Firstname>-<Lastname>.pdf", and save it together with the exported project (Android Studio: File → Export → Export to Zip File) in a single zip file. Your submission must consist of a single zip file containing all necessary files. The name of the .zip file, as well as the names of the contained files, **must not contain any umlauts**. Therefore, please resolve umlauts in file names.

Exercise 1: Dynamic Time Warping and Logistic Regression (12 points)

a) Compare Dynamic Time Warping and Logistic Regression. List similarities and differences. **(4 points)**

b) Which algorithm is more efficient after the training phase? Justify. **(2 points)**

c) Solve the following problem with the Dynamic Time Warping algorithm with absolute costs: Template: 9, 7, 6, 5, 4, 1, 8, 11, 6, 3, 3, 1, 2. Input: 7, 7, 2, 9, 1, 1. Fill in the DTW table, then select a path with minimal costs and finally specify the minimal costs. **(6 points)**

Exercise 2: Logistic Regression (16 points)

A Java template for logistic regression can be found on Stud.IP. The template is not intended for Android, but for the Java Standard Edition. To compile it, you can use the command line:

```
cd src
javac de/luh/hci/mi/logisticregression/*.java
java de.luh.hci.mi.logisticregression.LogisticRegression
```

You may also have to adapt the basePath in the LogisticRegression class to find the data files:

```
public final static String basePath = ".." + java.io.File.separator;
```

The code is intended to process gesture data delivered by the acceleration sensor and to distinguish between two gestures. The template contains sample files with the first four motion gestures shown in Figure 3 of [1].

a) Implement the function **double h(double[] x, double[] w)** to calculate the logistic function (sigmoid function). **(2 points)**

b) Implement the function **double cost(double hx, int y)** to calculate the cost of a single training sample. **(2 points)**

c) Implement the function **double J(double[] hx, int[] y)** to calculate the overall cost over all training samples. **(4 points)**

d) Implement the function **double[] train(double[][] trainXs, int[] trainYs)** to calculate the weight vector w using the given training data. **(8 points)**

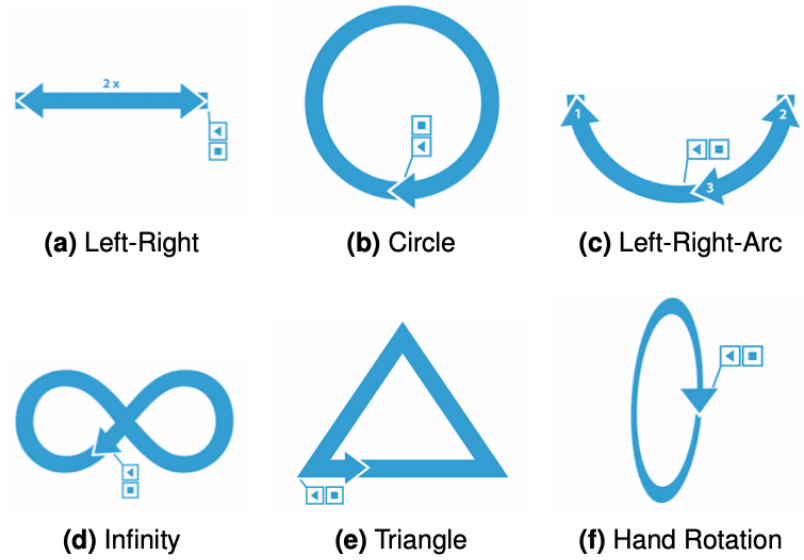

Figure 3. Visualization of the gesture classes we designed for input by the users.

[1] Sven Kratz, Michael Rohs, and Georg Essl. 2013. Combining acceleration and gyroscope data for motion gesture recognition using classifiers with dimensionality constraints. In Proc. of the 2013 international conference on Intelligent user interfaces (IUI '13). Association for Computing Machinery, New York, NY, USA, 173–178. DOI: https://doi.org/10.1145/2449396.2449419