Sound Communication: Lab III

RealTime Machine Learning Modeling (III):
Various inputs to various outputs

Sergio Giraldo
Universitat Pompeu Fabra

2017-2018

Introduction

The objective of this lab is to follow the framework presented in previous labs. The aim this time is to use different kind of input signals, such as gyroscopes, accelerometers, proximity sensors, etc. We will comment on several low cost sensors and devices available, and how the can be used withing the context of the proposed framework. However, for practical reasons we will use the smart phone as the main sensor device, which has good collection of sensors embedded on its hardware.

1 Background Theory Concepts

In this section we will briefly explore some low cost devices that provide sensors from which data can be extracted. This data can be later sent through several communication protocols to applications/devices in which we can apply machine learning techniques.

1.1 Arduino

Is an open source software/hardware company which design and manufactures computer boards for both software and hardware development. Arduino main boards make use of micro processors and micro controllers. The printed circuit boards can be extended trough extension boards also called "shields". Also several electronic devices such as sensors, actuators, leds, etc. can be connected through the input and output ports. Most of the boards can be charged and accessed through an usb port, where they can be programmed. The software of arduino consists on two main parts: an IDE (Integrated Development Environment) based on processing, and a bootloader. As mentioned before Arduino IDE's is based on processing, and can be installed in Windows Mac and Linus, also an online web IDE editor can be used.

For further information on Arduino you can check `https://www.arduino.cc/`. You can also check the Wekinator example bundles for Arduinot at `http://www.wekinator.org/examples/#Arduino`

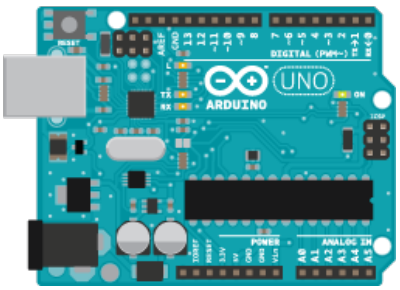

Figure 1: Arduino UNO

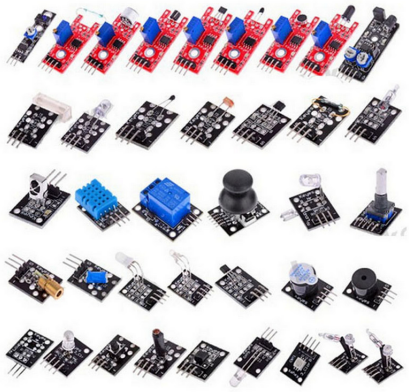

Figure 2: Arduino Sensors

1.2 MYO Gesture Control

The Myo armband is a gesture recognition device worn on the forearm and manufactured by Thalmic Labs. The Myo enables the user to control technology wirelessly using various hand motions. It uses a set of electromyographic (EMG) sensors that sense electrical activity in the forearm muscles, combined with a gyroscope, accelerometer and magnetometer to recognize gestures. The Myo can be used to control video games, presentations, music and visual entertainment. (font: Wikipedia)

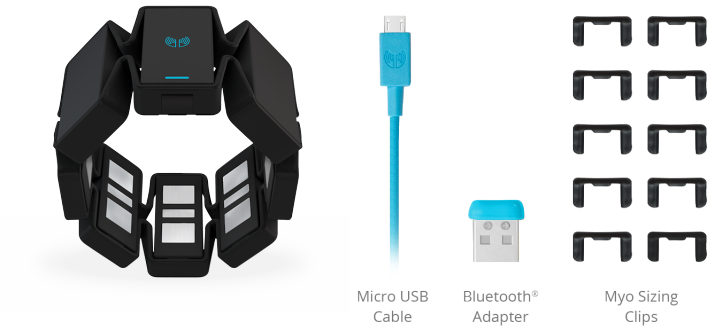

Figure 3: MYO armband

MYO sends data through bluethooth, and provides an IDE that can be installed in Windows and Mac platforms, from which raw data can be obtained.

For more information on MYO visit `https://www.myo.com/start`. You can also check the Wekinator example bundles for MYO at `http://www.wekinator.org/examples/#Myo_Armband_EMGmuscle_sensing_and_movement`

1.3 leap motion

The leap motion is a computer hardware sensor device that supports hand and finger motions as input, analogous to a mouse, but requires no hand contact or touching. It consists of a small peripheral device that designed to be placed on a surface facing up guard. By the use of two infrared cameras and tree infrared leds, the system is able to sense and hemispherical area, of about 1m of proximity. The systems connects through usb to the computer with a frame rate of 200 ms. The provided software process incoming data to infer 2D and 3D position information. However, raw data can be obtained through processing. For more information on leap motion visit `https://www.leapmotion.com/`. You can also check the Wekinator bundles for Leap motion at `http://www.wekinator.org/examples/#Leap_Motion_hardware_sensor`

1.4 Smart phone

Smart phones have become widely used in recent years, with ever growing computing networking and sensing power. Now a days a variety of sensors can be found on average smart phones despite its price. This fact has opened the doors for many interesting applications, ranging from health and fitness monitoring,

Figure 4: MYO armband

to urban computing, localization, navigation, games, etc. The sensors that can be usually find on average smart phones include (among others):

- Accelerometer.
- Gyroscope.
- Magnetometer.
- GPS.
- Barometer (in iPhone)
- Proximity sensor
- Ambient light sensor

1.5 Gesture classification using Dynamic Time Warping

Gesture classification is a wide studied problem in computer science, aiming at the recognition of gestural movements of objects in space over time. Several approaches to solve this problem have been studied in the past. In this lab we will explore an approach for gesture recognition over time by means of Dynamic Time Warping (DTW). DTW is a method that finds the minimum path by providing non-linear alignments between two time series(see Figure (). Classifiers as K-nearest neighbors can use DTW as a distance measure. This way similarity between two time series can be achieved.

2 Practical Work

In this section you will develop a system for automatic classification of hand gestures in real time, to later map the predicted class to sounds or images or a

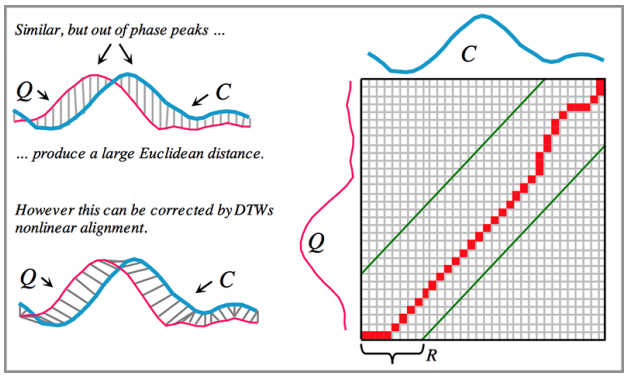

Figure 5: Linear vs DTW alignment of two time series.

combination of the two. You can choose any input device, however, for practical reasons, at this lab we will use smart phones.

2.1 materials

- Computer with sound card and built-in microphone.
- oscHook: plots smart phone sensors measurements and sends them in form of OSC messages. `https://oschook.soft112.com/`
- Wekinator. You can download wekinator from: `www.wekinator.org`
- Pure Data: A software for graphical computation. You can download from `https://puredata.info/`. You are encouraged to use *Processing or other programing language that facilitates the visualization of data.*

2.2 Framework

The framework of the system is the same as in previous labs, however the input can be any sensor from the ones presented above. Similarly, the output should be mapped to sound, image, arduino, or any other device.

2.2.1 Input

The input of the system will be the data obtained through OSC from the oscHook app. The possible inputs whit its respective OSC address are:

1. Accelerometer
   - /accelerometer/raw/x
   - /accelerometer/raw/y
   - /accelerometer/raw/z
2. Linear Acceleration
   - /accelerometer/linear/x
   - /accelerometer/linear/y

- /accelerometer/linear/z

3. Gravity
    - /accelerometer/gravity/x
    - /accelerometer/gravity/y
    - /accelerometer/gravity/z
4. Rotation vector
    - /rotation_vector/r1
    - /rotation_vector/r2
    - /rotation_vector/r3
    - /rotation_vector/r4
5. Compass
    - /orientation/azimuth
    - /orientation/roll
    - /orientation/pitch
6. Light sensor
    - /light
7. GPS Position
    - /gps/longitude
    - /gps/latitude
    - /gps/speed
    - /gps/accuracy
8. Proximity sensor
    - /proximity

The oscHook app sends OSC data to localhost, in which the port can be configured to taste (e.g. 7005). It is mandatory that both the receiving computer and the smart phone are connected to the same wireless network. On the oscHook you must provide the IP address provided by the network to the computer. (on mac and windows you can find your IP address at the network configuration panel)

2.3 Puredata OSC receiver

At the materials we provide a Pure data patch with an example where we receive from the phone through OSC at port 7005, the x, y, and z raw values of the accelerometer,(see Figure) . On the right hand of the figure, we send this data over OSC at port (6448) to Wekinator inputs as a package of three floats.

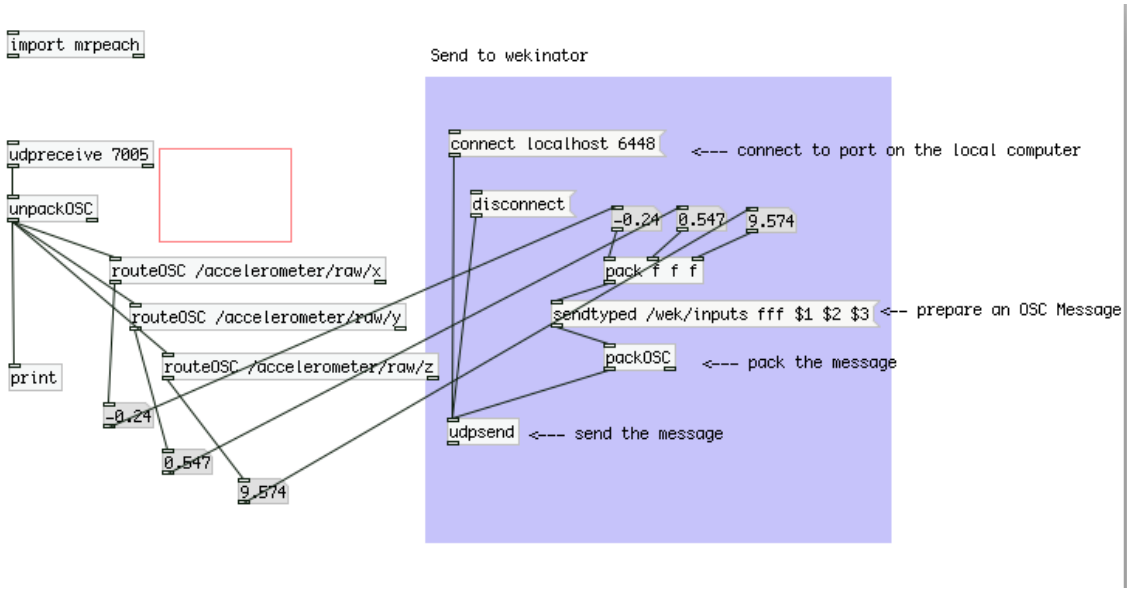

Figure 6: Pure data receiver - sender

2.4 Wekinator

Open Wekinator and start a new project. Choose the port number for the input as 6448.

Put the number of inputs wekinator is receiving from pure data patch.

Choose Dynamic Time Warping at the Outputs/type configuration pannel. Then choose number of gestures which will be equivalent to the number of outputs on previous labs. Also set the output port of wekinator to a free port (e.g 8338).

Dynamic time warping only uses one time series as a training example. You should record as many gestures as configured. When running DTW compared the received time series to the ones recored and outputs the probability of classification based on the DTW distance. For a detailed instruction on DTW in wekinator visit `http://www.wekinator.org/detailed-instructions/#Dynamic_time_warping_in_Wekinator`

2.5 output

Wekinator sends the output of each model trough OSC. You can merge the provided pure data patches of Lab 1 and Lab2 to map the

Submission

You have to deliver all the code and/or patches created for the practical work (i.e. wekinator files, and puredata files). You also have to submit as well a written report explaining the framework of your work, inputs, outputs models used. All the files should be submitted in a zip file though the Aula Global.

You will have to present and demonstrate your system at the next class.