Locomotion at ITU - Pervasive Computing - Mandatory Assignment 3

David Thomas
dtho@itu.dk

Egil Hansen
ekri@itu.dk

Jonas Rune Jensen
jruj@itu.dk

INTRODUCTION

The work in this assignment is based on recorded sequences of accelerometer data from an Android phone. Using Weka[1] – a data processing and machine learning tool – we try to recognize activities, e.g. walking, that were performed during such recordings.

We call these activities locomotion.

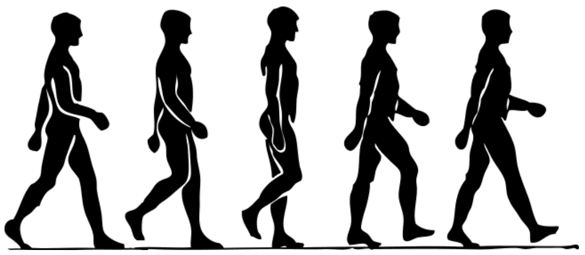

Figure 1. Locomotion: *"The movement of an organism from one place to another, oftenby the action of appendages such as flagella, limbs, or wings. Insome animals, such as fish, locomotion results from a wave-likeseries of muscle contractions."* – The American Heritage Science Dictionary. Image source: wpclipart.com.

Our source code is available at https://github.com/jeffton/Pervasive-Assignment-3.

BACKGROUND

Sensor data is often relevant in ubicomp applications as it deals with implicit input. Context-aware applications can use sensors to detect ongoing activities and automatically adjust their behavior to fit the current situation. Sensors typically provide large amounts of data, sometimes with inaccuracies or noise, and usually with a level of indirection towards what the application actually needs to know. As an example, motion detectors and microphones can be used to try to determine whether or not someone is in a room, even if this true-or-false-statement is not directly measured by the sensors. Data processing techniques enter the picture as a means to make sense of such sensor data. The data analysis in this assignment is a case of supervised learning: classification of training data lets us predict the class of new data.

[1]http://sourceforge.net/projects/weka/

THE SYSTEM

The system, consisting of the Android app and a Google App Engine (GAE) service, is composed of three packages, `app`, `shared`, and `service`, all located in the `dk.itu.spct.locomotion` namespace. The `shared` package holds the two classes `LocomotionData` and `DataPoint`, which are used to store and serialize the recorded motion data in. The `LocomotionData` class is used to represent a single recording, and the `DataPoint` class is used to represent a single event in a recording. See figure 2 for details.

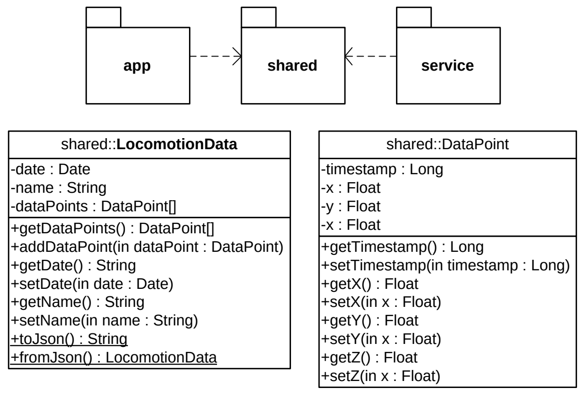

Figure 2. System packaged diagram and class diagram for the shared classes.

The Android App

The Android app consists of a single activity, `RecordingActivity`, that allows the user to record and upload a set of accelerometer data. The recording itself is handled by our `AccelerometerRecorder` class which uses the accelerometer of the Android phone via `SensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)`. After pressing the record button, the app waits a couple of seconds before actually starting the recording to allow the user to get ready. The phone vibrates before and after recording to inform the user.

Once the recording is complete, the user is prompted to name and save the recording. If the prompt is confirmed, the recording, now including the provided name and the current time, is serialized to JSON using google-gson[2] and sent to the service using our `DataUploader` class.

Service on Google App Engine

The service consists of two servlets, one that handles upload of locomotion data, i.e. `UploadServlet`, and one that serves the data back again to the user, i.e. `GetDataServlet`. The service also has a simple HTML-based front-end[3] component that enables users to perform test uploads and download already uploaded locomotion data. Upon upload, the raw locomotion data stored in `LocomotionData` object is converted to a CSV file and then stored in GAEs Blobstore. We use the Blobstore since it allows us to serve the saved CSV files directly with a simple API call, keeping the code very simple.

CAPTURING TRAINING DATA

We captured data for three different activities: walking, sitting, and ascending stairs. Two people, each using their own phone, performed each activity five times. This gave us a total of 30 recordings, each with a duration of 10 seconds with 20 samples per second. The phone was placed in a pants pocket while recording.

Table 1 shows a snippet of one recording. The timestamps are in nanoseconds and are relative to the beginning of the recording. Note how some preprocessing is already taken care of while recording: besides saving relative timestamps, the app makes it easy to carry out the recorded activities during the entire recording. This removes the need to cut the beginning and end of the recordings afterwards.

DATA PROCESSING WITH WEKA

In order to use our data in Weka we needed to preprocess it into a suitable format. The first part was already done in our Android application where we changed the timestamps of the data points to be relative to the beginning of the recording rather than absolute timestamps. The next preprocessing we performed was to manually label the recordings as walking, sitting or ascending stairs. As our application starts after 5 seconds and runs in 10 seconds, we have decided not to remove the first and last couple of recordings.

If we make a 10 fold cross validation with our training set, we get 76% correctly classified instances and the confusion matrix in table 2.

After making the 10 recordings of each activity, we have tried to classify another recording in Weka. When using a Bayes network algorithm, we get 67,9% correctly classified instances with the confusion matrix in table 3

Here we are able to see that the algorithm categorizes all the movements as walking on stairs.

[2]http://code.google.com/p/google-gson/
[3]Available at http://locomotion-at-itu.appspot.com/.

Timestamp	X	Y	Z
0	−7.434 141	−6.092 93	−5.364 844
49377444	−2.145 937 7	−17.665 665	−0.459 843 75
98907476	−2.030 976 5	−10.844 649	−2.260 898 6
149383550	−2.567 461	−13.756 992	−0.728 086
199340826	−1.609 453 2	−11.917 618	−0.038 320 314
249328617	−1.762 734 4	−10.308 165	0.0
299774174	−2.260 898 6	−9.081 915	−0.459 843 75
349426272	0.153 281 26	−6.514 453 4	0.536 484 4
399230958	0.114 960 94	−8.047 266	0.498 164 1
449218750	−1.992 656 4	−8.277 188	0.383 203 15
502990721	−2.797 382 8	−11.726 016	−0.498 164 1
549591062	−2.644 101 6	−12.032 578	−1.034 648 5
599182127	1.226 25	−15.174 845	−2.337 539 2
653045652	−3.870 351 8	−14.944 922	−0.804 726 6
699218748	0.459 843 75	−13.488 75	−1.801 054 7
749206540	−1.111 289 1	−12.032 578	−2.912 344
799560544	−0.076 640 63	−7.702 383	−1.494 492 3
849243162	−3.908 672	−4.828 359 6	−0.459 843 75
899597166	−0.651 445 3	−4.330 195 4	−1.072 968 8
949584959	−0.843 046 9	−3.333 867 3	−2.222 578 3

Table 1. Sample locomotion data. This is the first second of a recording where a person is ascending stairs.

a	b	c	Classified as
1018	2	897	a = stairs
1	1963	5	b = sit
474	9	1431	c = walk

Table 2. Confusion matrix for 10 fold cross validation.

a	b	c	Classified as
199	0	1	a = stairs
0	199	0	b = sit
190	0	7	c = walk

Table 3. Confusion matrix using the Bayes network algorithm.

DISCUSSION

In our training we have used two different kinds of phones and two people have been walking with each. This gives some better value to the data, as people have different ways of moving, and the phones can also have different values for the same operation. In order to gain even better results we also have to be aware that values can change if the phone is placed upside down, or with the screen towards the body or away from the body. So in order to gain a higher percentage of correctly classified instances, we need to get training data from more phones, persons and make sure that the phones is in different positions. But we will never get a 100

CONCLUSION

In this assignment, we have recorded time series of accelerometer data from an Android phone while performing various activities.

By classifying the recordings and using them as a training set in Weka, we have been able to determine the activities performed in new recordings with an accuracy of 67.9% – our main problem being that walking was incorrectly classified as the rather similar activity of ascending stairs.