Detection of sensing contexts through machine learning algorithms using Participatory sensing data

Faculty of Engineering and Technology, Jadavpur University in fulfillment of the requirements for the Degree of Master of Computer Application

Submitted by

Partha pratim saha

Registration No: 137332 of 2016-2017

Class Roll No: 001610503025

Examination Roll No: MCA196021

Under the supervision of

Dr. Sarbani Roy

Associate Professor, Dept. of Computer Science & Engineering

Jadavpur University

A thesis submitted in partial fulfillment of the requirements of degree of Master of Computer Application

Department of Computer Science and Engineering

Faculty of Engineering & Technology

Jadavpur University

May 2019

<u>Declaration of Originality & Compliance of Academics Ethics</u>

I hereby declare that this thesis contains literature survey and original research work done by me, as part of my MCA studies. All information in this document have been obtained and presented in accordance with academic rules and ethical conduct.

Name	:	**PARTHA PRATIM SAHA**
Registration No	:	137332 of 2016-2017
Class Roll No	:	001610503025
Examination Roll No	:	MCA196021
Project Report Title	:	Detection of sensing contexts through machine learning algorithms using Participatory sensing data

----

Partha pratim saha

Department of Computer Science & Engineering
Faculty of Engineering & Technology
Jadavpur University

This is to certify that PARTHA PRATIM SAHA, Registration Number: 137332 of 2016-2017, Class Roll Number: 001610503025, Examination Roll Number: MCA196021, a student of MCA, from the Department of Computer Science & Engineering, under the Faculty of Engineering and Technology, Jadavpur University has done a project report under my supervision, entitled as " Detection of sensing contexts through machine learning algorithms using Participatory sensing data". The thesis is approved for submission towards the fulfillment of the requirements for the degree of Master of Computer Application, from the Department of Computer Science & Engineering, Jadavpur University for the session 2018-19.

Dr. Sarbani Roy
Associate Professor
Department of Computer Science and Engineering
Jadavpur University

Dr. Mahantapas Kundu
(Head of the Department)
Professor
Department of Computer Science and Engineering
Jadavpur University

Dr. Chiranjib Bhattacharjee
(Dean)
Professor
Faculty of Engineering and Technology
Jadavpur University

Certificate of Approval

(Only in case the project report is approved)

The forgoing thesis is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve this thesis only for the purpose for which it is submitted.

____________________________	____________________________
Signature of the Examiner	Signature of the Examiner
Date: _____________	Date: _____________

JADAVPUR UNIVERSITY

Abstract

Faculty of Engineering and Technology

Department of Computer Science and Engineering

Detection of sensing contexts through machine learning algorithms using participatory sensing data

by Partha pratim saha

Detecting of sensing context is an important part when we use raw sensor data for any work. Suppose we take some sensor data for any work. Now how we know that these data are noisy or not. For this we detect the context and see when these data were collected what is the situation of participator.

To implement this system we create an android application that take sensor data and store these data in local storage. Now we apply machine learning algorithm over these data sets and detect the context. Since our problem is a classification problem so we use decision tree and random forest to solve our problem. When we finally detect the context then we say that these data is noisy or not and after that we can use these data depend on the data is noisy or not.

<u>Acknowledgement</u>

I would like to express my deepest gratitude to my advisor and guide Dr. Sarbani Roy, for her excellent guidance, care, patience, and providing me with an excellent atmosphere for doing research. I received a lot of encouragement and inspiration from her throughout the project. I am equally grateful to Dr. Mahantapas Kundu, Head of The Department, Computer Science & Engineering, Jadavpur University, for his support towards our department. Last but not the least, I would like to thank my parents and all respected teachers for their valuable suggestions and helpful discussions.

Regards,

Partha pratim saha

Examination Roll No: MCA196021

Master of Computer Application

Jadavpur University

Contents

Chapter 1

Introduction

1.1 Overview

Participatory sensing is the concept of communities (or other groups of people) contributing sensory information to form a body of knowledge. A growth in mobile devices, for example smart phones, tablet computers or activity trackers, which have multiple sensors, has made participatory sensing viable in the large-scale. Participatory sensing can be used to retrieve information about the environment, weather, urban mobility, congestion as well as any other sensory information that collectively forms knowledge [1].

Context sensing is the technology that infers the characteristics of one or more persons, locations, objects, situations, or activities and uses that information to dynamically adapt to, synchronize, and frame situations and processes. Context sensing is a key technology for mobile systems, smart objects, and multi-device environments for extending and augmenting human-computer interactions [2].

In our system group of people are participate to contribute sensory information using their smart phones, tablet computer etc. We store the information in local storage of this device. Then apply the machine learning algorithms and sensing the context that is when the data was collected at that time the participator holds the device on his hand or put in the pocket or talking to someone etc.

1.2 Motivation

Suppose we collect the noise data. Participator can take data in any situation which we don’t know. If we take all the data from participators and use it blindly then many problems can occur. Consider three participator take data in different situation and contribute this. Suppose a participator take data when he was talking to someone, another participator take data when the device was present in his pocket and the rest one take data when he holds the device in his hand. Now out of this three participators we use the data that gives the last participator because when this participator take data that time his device did not perform any other task that make our data noisy. Second participator data was not taken because for the noise data collection if we put our device in our pocket then the noise data was not well collected. This system is used when data filtering is occur that is after taking the all participators data, the data sets were noisy or not depend on the data collection context and using this system we find the context when the data was collected then it was very helpful to decide which data sets are use and which are not use.

1.3 Contribution

One android application is created using java programming language that takes the sensor data from the mobile device in any situation and stores these data sets in local csv file. Now apply the machine learning algorithms over these data sets to predict the contexts. Since the problem is classification problem so here we use classification algorithms to solve this problem. Here we implement two classification algorithms one is Decision Tree and the other one is Random Forest. After applying these two algorithms now we find the accuracy. Finally take the better accuracy result and depend on this result take the features and the algorithm which help to give this result.

1.4 Organization

The organization of the thesis is as follows. The Background is presented in Chapter 2. The design of the android application, data collection and prediction technique are implemented in Chapter 3. The experimental setup along with results are then discuss under Result and Analysis in Chapter 4. Finally the thesis concludes in Chapter 5.

Chapter 2

Background

2.1 Participatory Sensing

Participatory sensing is the process whereby individuals and communities use ever-more-capable mobile phones and cloud services to collect and analyze systematic data for use in discovery. The convergence of technology and analytical innovation with a citizenry that is increasingly comfortable using mobile phones and online social networking sets the stage for this technology to dramatically impact many aspects of our daily lives.

2.1.1 Application and uses model

One application of participatory sensing is as a tool for health and wellness. Potential contexts include chronic-disease management and health behavior change. Communities and health professionals can also use participatory approaches to better understand the development and effective treatment of disease. The same systems can be used as tools for sustainability. In addition, participatory sensing offers a powerful “make a case” technique to support advocacy and civic engagement. It can provide a framework in which citizens can bring to light a civic bottleneck, hazard, personal-safety concern, cultural asset, or other data relevant to urban and natural-resources planning and services, all using data that are systematic and can be validated. These different applications imply several different usage models. These models range from public contribution, in which individuals collect data in response to inquiries defined by others, to personal use and reflection, in which individuals log information about themselves and use the results for personal analysis and behavior change. Yet across these varied applications and usage models, a common workflow is emerging.

2.1.2 Essential Components

Ubiquitous data capture and leveraged data processing are the enabling technical components of these emerging systems. The need for the individual to control access to the most intimate of these data streams introduces a third essential component: the personal data vault.

While empirical data can be collected in a variety of ways, mobile phones are a special and, perhaps, unprecedented tool for the job. These devices have become mobile computing, sensing, and communication platforms, complete with image, audio, video, motion, proximity, and location data capture and broadband communication, and they are capable of being programmed for manual, automatic, and con-text-aware data capture. Because of the sheer ubiquity of mobile phones and associated communication infra-structure, it is possible to include people of all backgrounds nearly everywhere in the world. Because these devices travel with us, they can help us make sustainable observations on an intimately personal level. Collectively, they provide unmatched coverage in space and time.
In some cases, the data collected with a mobile device are enough to reveal an interesting pattern on their own. However, when processed through a series of external and cross-user data sources, models, and algorithms, simple data can be used to infer complex phenomena about individuals and groups. Mapping and other interactive capabilities of today’s Web enhance the presentation and interpretation of these patterns for participants. Many applications will call for the comparison of current measures to past trends, so robust and long-term storage and management of this data is a central requirement.

Participatory-sensing systems leveraging mobile phones offer unprecedented observational capacity at the scale of the individual. At the same time, they are remarkably scalable and affordable given the wide proliferation of cellular phone infrastructure and consumer devices that incorporate location services, digital images, accelerometers, Bluetooth access to off-board sensors, and easy programmability. These systems can be leveraged by individuals and communities to address a range of civic concerns, from safety and sustainability to personal and public health. At the same time, they will push even further on our societies’ concepts of privacy and private space [7].

2.2 Importance of context sensing

Technologies that infer the characteristics of one or more persons, locations, objects, situations, or activities and use that information to dynamically adapt to, synchronize, and frame situations and processes. Context sensing is a key technology for mobile systems, smart objects, and multi-device environments for extending and augmenting human-computer interactions. Integrating contextual information helps us create and deliver content that better meets user needs.

Technical communicators need to understand how to describe these systems, known as context-aware or smart technologies, and how to use them for delivering context-aware content that can automatically infer information needs and user intentions. Context sensing approaches can be categorized by their use of predefined information, sensor-based situational information and aggregated information. Approaches that aggregate, combine, or integrate sensor information are also referred to as context detection. Approaches that can relate the contexts of different entities for identifying their situational similarity are called context matching. Context detection and matching typically operate on five groups of contextual information for enriching the user experience: individuality, activity, location, time and date, and relationships [2]. To implement our system we required context sensing because without context sensing we cannot find situation of the participator. Participatory sensing is an important part of our system because if it was not done well then noisy data was collected by the participators and our system predict wrong context.

Chapter 3

Proposed System for Context Sensing

3.1 Problem Statement

Create an android application that take the sensor value from mobile device in any environment and store these sensor data in local storage. Share this android application to all participators. Apply the machine learning over these data sets that was given by the participators and detect the context that is when the data was collected at that time participator put the mobile device in his or her pocket or talking to someone or walking etc.

3.2 Android Application Development

Android apps can be written using Kotlin, Java, and C++ languages. The Android SDK tools compile our code along with any data and resource files into an APK, an Android package, which is an archive file with an .apk suffix. One APK file contains all the contents of an Android app and is the file that Android-powered devices use to install the app. The Android operating system is a multi-user Linux system in which each app is a different user. By default, the system assigns each app a unique Linux user ID (the ID is used only by the system and is unknown to the app). The system sets permissions for all the files in an app so that only the user ID assigned to that app can access them. Each process has its own virtual machine (VM), so an app's code runs in isolation from other apps. By default, every app runs in its own Linux process. The Android system starts the process when any of the app's components need to be executed, and then shuts down the process when it's no longer needed or when the system must recover memory for other apps. The Android system implements the principle of least privilege. That is, each app, by default, has access only to the components that it requires to do its work and no more. This creates a very secure environment in which an app cannot access parts of the system for which it is not given permission. App components are the essential building blocks of an Android app. Each component is an entry point through which the system or a user can enter your app. There are four different types of app components and they are activities, services, broadcast receivers and content providers. Each type serves a distinct purpose and has a distinct lifecycle that defines how the component is created and destroyed. There are two major parts in android application development and they are design and implementation.

3.2.1 Design

The functionality of the application, the way of using the interface, and should be presented to the user in an easy-to-understand manner. This section will discuss how the program was designed, represented by the Activity Diagram that was used to design and understand the user flow of the application. The design pattern is a template that shows the interactions between the program components, such as the classes and the objects. It is used as a reusable solution for software engineering problems. The use of the MVC design pattern helped to separate the programming of the view and the model in two different classes, helping to extend and modify both much easier. The view provides a visual representation of the data model, while the controller links the model and the view, by 'listening' for events from the view, and carrying out the corresponding action on the model. The view represented in our android application is given below.

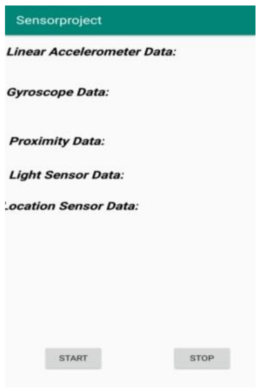


Figure 3.1: view of android application

The Activity class is a crucial component of an Android app, and the way activities are launched and put together is a fundamental part of the platform's application model. Unlike programming paradigms in which apps are launched with a main() method, the Android system initiates code in an Activity instance by invoking specific callback methods that correspond to specific stages of its lifecycle. The Activity class is designed to facilitate this paradigm. When one app invokes another, the calling app invokes an activity in the other app, rather than the app as an atomic whole. In this way, the activity serves as the entry point for an app's interaction with the user. You implement an activity as a subclass of the Activity class. An activity provides the window in which the app draws its UI. This window typically fills the screen, but may be smaller than the screen and float on top of other windows. Generally, one activity implements one screen in an app. For instance, one of an app's activities may implement a preferences screen, while another activity implements a select photo screen. Most apps contain multiple screens, which mean they comprise multiple activities. Typically, one activity in an app is specified as the main activity, which is the first screen to appear when the user launches the app. Each activity can then start another activity in order to perform different actions.

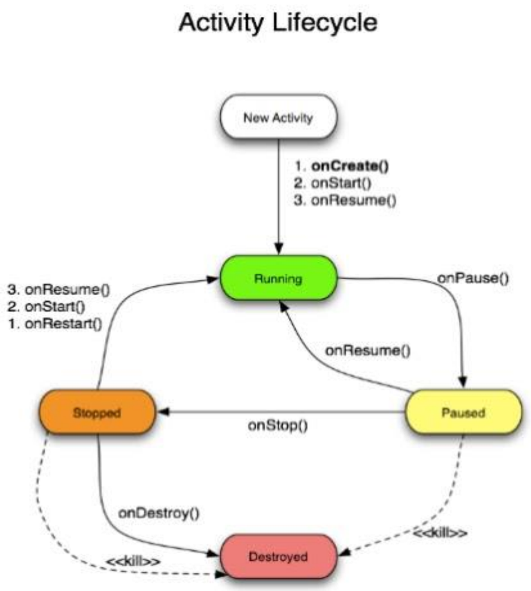


Figure 3.2: Lifecycle of an activity

In our application when the application is open the activity starts. At that time it call onCreate(), onStart() and onResume() methods consequently. Now the activity is on running state and our application take data from device sensor and store in the local storage. If participator minimize our application for any reason then onPause() and onStop() methods call and the activity goes in the stopped state. When participator go back to our application it call onRestart(), onStart() and onResume() methods and it goto the running state again. When the participator exit from our application it call onPause(), onStop() and onDestroy() methods and the activity is shutdown. The following flow chart shows how our android application is work.

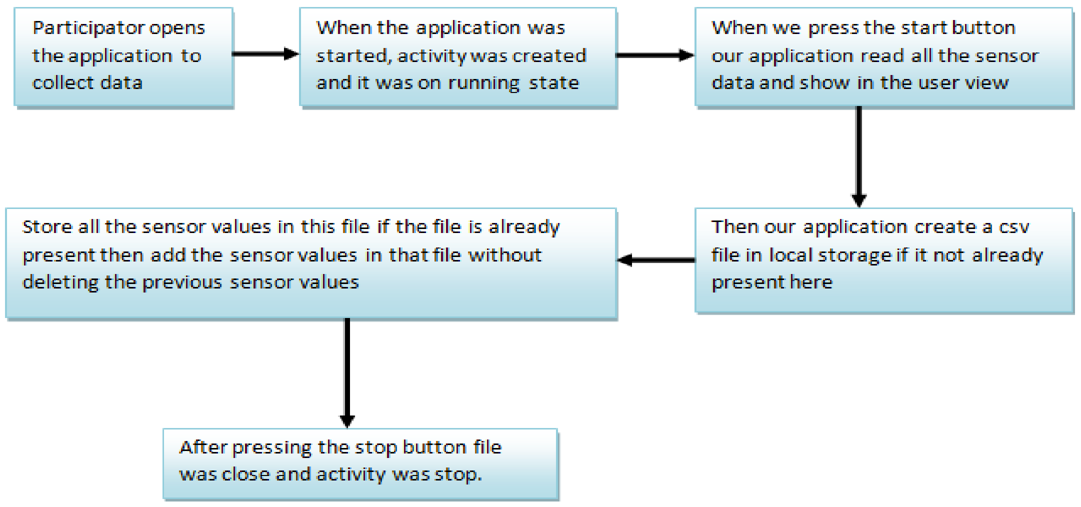
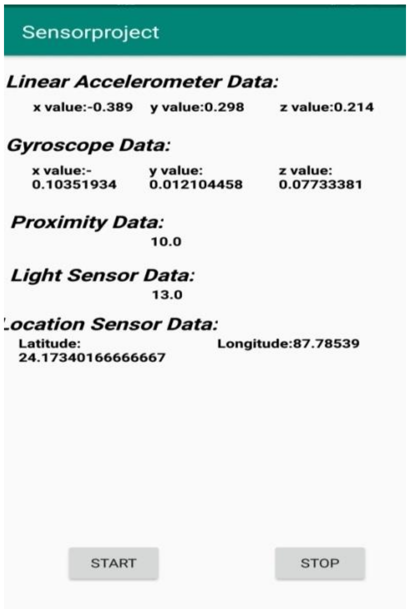

Participator opens the application to collect data
When the application was started, activity was created and it was on running state
When we press the start button our application read all the sensor data and show in the user view
Then our application create a csv file in local storage if it not already present here
Store all the sensor values in this file if the file is already present then add the sensor values in that file without deleting the previous sensor values
After pressing the stop button file was close and activity was stop.

Figure 3.3: How our android application work

3.2.2 Implementation

Our android application's view partition was implemented in activity_main.xml file. Every text view has a unique id in this file. Using this id we can access this text view in main java file. Suppose the id of one text field is value then it access from main java file by using this code *value=(TextView)findViewById(R.id.value)*. Mobile device sensors are three type motion sensor, environmental sensor and position sensor. Motion sensor includes linear accelerometer, gravity, gyroscope and relational vector sensor. Environmental sensor includes temperature, pressure and humidity. Position sensor includes latitude and longitude. You can access those sensors and acquire raw sensor data by using the android sensor framework. The sensor framework is part of the android hardware package and includes the classes and interfaces are SensorManager, Sensor, SensorEvent and SensorEventListener. The system use SensorManager class to create an instance of sensor service. This class provides various methods for accessing and listing sensor, register and unregister sensor event listener and acquiring orientation information. The system use Sensor class to create an instance of a specific sensor. The system use SensorEvent class to create a sensor event object, which provides the information about a sensor event. A sensor event object include the information are raw sensor data, the type of the sensor that generate the event, the accuracy of the data and the time stamp for the event. The system use SensorEventListener interface to create to callback method that receive notification when the sensor value change or when the sensor accuracy change.

First we write onCreate() method. In onCreate() method first we access all the text that are present in the view by their unique id and store the reference in local text view variable. Suppose we access the x value of linear accelerometer then the corresponding code is *lxValue=(TextView)findViewById(R.id.lxvalue)* where lxvalue is the id of the x value of linear accelerometer's text field. Similarly we can access all text fields by using their unique id and store the reference in different variable. This was done because when the sensor values are display in view section then using this text views variable we can do it easily. Now we can create an object of sensor manager by using this code *sensormanager=(SensorManager)getSystemService(Context.SENSOR_SERVICE)* . After that we create the object of all the sensors. For linear accelerometer the code is *linearaccelerometer=sensormanager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELEROMETER)* , where linearaccelerometer is the object of sensor class. Similarly we can do for gyroscope, proximity, light and location service and onCreate() method is ended here. Now we write onStart() method. In onStart() method we check if the sensor objects are null or not. If any sensor object is null then display the massage that this sensor is not present in this device and if it is not null then register the listener corresponding this sensor object. onStart() method is ended here. Now we write the onResume() method. In onResume() method first we check the device state then we check the external memory is available or not. Now we check the current directory and create a folder name as 'Myfile' if it is not present. Then create a csv file name as 'sensordata'. On every sensor event change there is a method is called onSensorChanged() and it accept the argument as a sensor event object. This method was automatically call when any sensor event change occur. Using the code *Sensor sensor=sensorEvent.sensor* we get the sensor from sensor event where sensorEvent is an object of sensor event. We check the type of the sensor and depend on that this sensor value is display in the corresponding text view in user view section. For location sensor we implement a method call as onLocationChanged(). To get latitude use the code *latitude=location.getLatitude()*. Similarly we get longitude also and display in the user view section.



Sensorproject

Linear Accelerometer Data:

x value:-0.389 y value:0.298 z value:0.214

Gyroscope Data:

x value:- 0.10351934 y value: 0.012104458 z value: 0.07733381

Proximity Data:

10.0

Light Sensor Data:

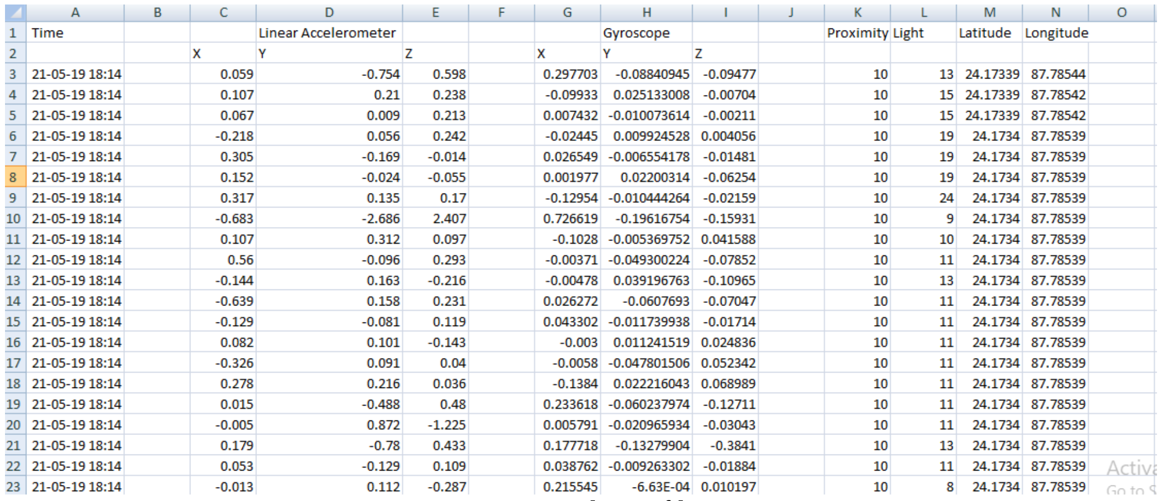
13.0

.ocation Sensor Data:

Latitude: 24.17340166666667 Longitude:87.78539

Figure 3.4: user view when data are storing in file

Now we take data after 2 second and store it in the local csv file. To do this we create a function call as showData() and call it from onSensorChanged() method after every two second. We also add the current data and time in the csv file. In onPasue() method we close the file only. In onStop() method we unregister the listener class from sensor manager by using this code *sensormanager.unregisterListener(MainActivity.this)*.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Time			Linear Accelerometer				Gyroscope			Proximity	Light	Latitude	Longitude	
2			X	Y	Z		X	Y	Z						
3	21-05-19 18:14		0.059	-0.754	0.598		0.297703	-0.08840945	-0.09477		10	13	24.17339	87.78544	
4	21-05-19 18:14		0.107	0.21	0.238		-0.09933	0.025133008	-0.00704		10	15	24.17339	87.78542	
5	21-05-19 18:14		0.067	0.009	0.213		0.007432	-0.010073614	-0.00211		10	15	24.17339	87.78542	
6	21-05-19 18:14		-0.218	0.056	0.242		-0.02445	0.009924528	0.004056		10	19	24.1734	87.78539	
7	21-05-19 18:14		0.305	-0.169	-0.014		0.026549	-0.006554178	-0.01481		10	19	24.1734	87.78539	
8	21-05-19 18:14		0.152	-0.024	-0.055		0.001977	0.02200314	-0.06254		10	19	24.1734	87.78539	
9	21-05-19 18:14		0.317	0.135	0.17		-0.12954	-0.010444264	-0.02159		10	24	24.1734	87.78539	
10	21-05-19 18:14		-0.683	-2.686	2.407		0.726619	-0.19616754	-0.15931		10	9	24.1734	87.78539	
11	21-05-19 18:14		0.107	0.312	0.097		-0.1028	-0.005369752	0.041588		10	10	24.1734	87.78539	
12	21-05-19 18:14		0.56	-0.096	0.293		-0.00371	-0.049300224	-0.07852		10	11	24.1734	87.78539	
13	21-05-19 18:14		-0.144	0.163	-0.216		-0.00478	0.039196763	-0.10965		10	13	24.1734	87.78539	
14	21-05-19 18:14		-0.639	0.158	0.231		0.026272	-0.0607693	-0.07047		10	11	24.1734	87.78539	
15	21-05-19 18:14		-0.129	-0.081	0.119		0.043302	-0.011739938	-0.01714		10	11	24.1734	87.78539	
16	21-05-19 18:14		0.082	0.101	-0.143		-0.003	0.011241519	0.024836		10	11	24.1734	87.78539	
17	21-05-19 18:14		-0.326	0.091	0.04		-0.0058	-0.047801506	0.052342		10	11	24.1734	87.78539	
18	21-05-19 18:14		0.278	0.216	0.036		-0.1384	0.022216043	0.068989		10	11	24.1734	87.78539	
19	21-05-19 18:14		0.015	-0.488	0.48		0.233618	-0.060237974	-0.12711		10	11	24.1734	87.78539	
20	21-05-19 18:14		-0.005	0.872	-1.225		0.005791	-0.020965934	-0.03043		10	11	24.1734	87.78539	
21	21-05-19 18:14		0.179	-0.78	0.433		0.177718	-0.13279904	-0.3841		10	13	24.1734	87.78539	
22	21-05-19 18:14		0.053	-0.129	0.109		0.038762	-0.009263302	-0.01884		10	11	24.1734	87.78539	
23	21-05-19 18:14		-0.013	0.112	-0.287		0.215545	-6.63E-04	0.010197		10	8	24.1734	87.78539	

Figure 3.5: sensor data in file

3.3 Data Collection

Participators collect the data in all possible situations. Possible situations are when they are collecting data is talking to someone, walking in road, travelling in any transport system, present in indoor or outdoor, put the device in his or her pocket, hold the device in hand.

Figure 3.6: Participators collect data

After completing their data collection process the file where the data store is share with the system where the machine learning algorithms are implemented. Files are transfer from the mobile device to that system through data cable and this system can access these files easily.

3.4 Prediction Techniques

Prediction technique is the machine learning algorithms that are use to solve this problem. Machine Learning (ML) can be explained as automating and improving the learning process of computers based on their experiences without being actually programmed i.e. without any human assistance. The process starts with feeding good quality data and then training our machines (computers) by building machine learning models using the data and different algorithms. The choice of algorithms depends on what type of data do we have and what kind of task we are trying to automate. Generally there are two main types of machine learning problems, supervised and unsupervised. Supervised machine learning problems are problems where we want to make predictions based on a set of examples. Unsupervised machine learning problems are problems where our data does not have a set of defined set of categories, but instead we are looking for the machine learning algorithms to help us organize the data. Within supervised machine learning we further categorize problems into two parts, one is classification problem and another one is regression problem. Unsupervised machine learning problems are problems where we have little or no idea about the results should look like. We are basically providing the machine learning algorithms with data and asking it algorithm to look for hidden features of data and cluster the data in a way that makes sense based on the data. Examples of algorithms used for unsupervised machine learning problems are K-means clustering, Neural Networks and Principal component analysis. A classification problem is a problem where we are using data to predict which category something falls into. An example of a classification problem could be analyzing a image to determine if it contains a car or a person, or analyzing medical data to determine if a certain person is in a high risk group for a certain disease or not. In other words we are trying to use data to make a prediction about a discrete set of values or categorizes. Since our problem is classification problem so we discuss only these algorithms which solve the classification problems. These algorithms are Decision Tree, Random Forest, Naive Bayes Classifier, Logistic Regression and K-Nearest Neighbors. Here we use only decision tree and random forest [4].

3.4.1 Decision Tree

A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. Though a commonly used tool in data mining for deriving a strategy to reach a particular goal, it's also widely used in machine learning, which will be the main focus of this article. For this let's consider an example that the mobile device was present in his hand or put the device in his pocket. If the device was present in his hand at that time he was walking or not. We consider all three axes of linear accelerometer and proximity as features. Now the following diagram show how decision tree work.

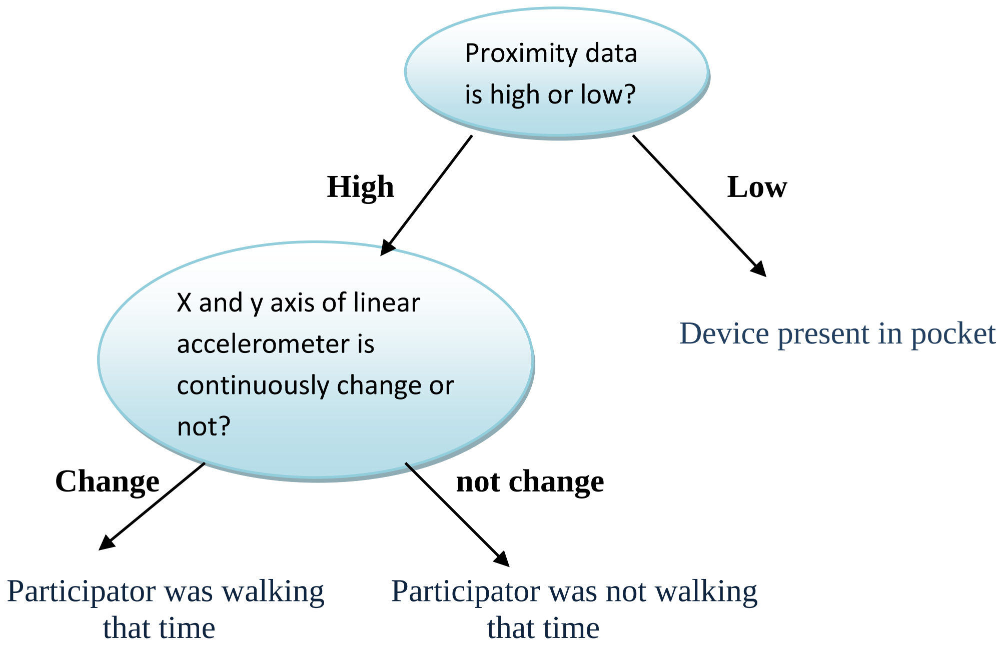

Proximity data is high or low?
High
Low
X and y axis of linear accelerometer is continuously change or not?
Device present in pocket
Change
not change
Participator was walking that time
Participator was not walking that time

Figure 3.7: Decision Tree

Although, a real dataset will have a lot more features and this will just be a branch in a much bigger tree, but you can't ignore the simplicity of this algorithm. The feature importance is clear and relations can be viewed easily. This methodology is more commonly known as learning decision tree from data and above tree is called Classification tree as the target is to classify passenger as survived or died. Regression trees are represented in the same manner, just they predict continuous values like price of a house. In general, Decision Tree algorithms are referred to as CART or Classification and Regression Trees. So, what is actually going on in the background? Growing a tree involves deciding on which features to choose and what conditions to use for splitting, along with knowing when to stop. As a tree generally grows arbitrarily, you will need to trim it down for it to look beautiful. Let's start with a common technique used for splitting. In Recursive Binary Splitting all the features are considered and different split points are tried and tested using a cost function. The split with the best cost (or lowest cost) is selected. This algorithm is recursive in nature as the groups formed can be sub-divided using same strategy. Due to this procedure, this algorithm is also known as the **greedy algorithm**, as we have an excessive desire of lowering the cost. This makes the root node as best predictor/classifier. Let's take a closer look at cost functions used for classification and regression**.** In both cases the cost functions try to find most homogeneous branches, or branches having groups with similar responses**.** This makes sense we can be surer that a test data input will follow a certain path. **Regression : sum(y—prediction)^2**

Lets say, we are predicting the price of houses. Now the decision tree will start splitting by considering each feature in training data. The mean of responses of the training data inputs of particular group is considered as prediction for that group. The above function

is applied to all data points and cost is calculated for all candidate splits. Again the split with lowest cost is chosen. **Classification : G = sum(pk * (1—pk))**
A Gini score gives an idea of how good a split is by how mixed the response classes are in the groups created by the split. Here, pk is proportion of same class inputs present in a particular group. A perfect class purity occurs when a group contains all inputs from the same class, in which case pk is either 1 or 0 and G = 0, where as a node having a 50–50 split of classes in a group has the worst purity, so for a binary classification it will have pk = 0.5 and G = 0.5. You might ask when to stop growing a tree? As a problem usually has a large set of features, it results in large number of split, which in turn gives a huge tree. Such trees are complex and can lead to over fitting. So, we need to know when to stop? One way of doing this is to set a minimum number of training inputs to use on each leaf. For example we can use a minimum of 10 passengers to reach a decision(died or survived), and ignore any leaf that takes less than 10 passengers. Another way is to set maximum depth of your model. Maximum depth refers to the the length of the longest path from a root to a leaf. The performance of a tree can be further increased by pruning. It involves removing the branches that make use of features having low importance. This way, we reduce the complexity of tree, and thus increasing its predictive power by reducing over fitting. Pruning can start at either root or the leaves. The simplest method of pruning starts at leaves and removes each node with most popular class in that leaf, this change is kept if it doesn't deteriorate accuracy. It's also called reduced error pruning. More sophisticated pruning methods can be used such as cost complexity pruning where a learning parameter (alpha) is used to weigh whether nodes can be removed based on the size of the sub-tree. This is also known as weakest link pruning**.** This is all the basic, to get you at par with decision tree learning. An improvement over decision tree learning is made using technique of boosting [5].

3.4.2 Random Forest

Random Forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because its simplicity and the fact that it can be used for both classification and regression tasks. Random Forest is a supervised learning algorithm. Like you can already see from its name, it creates a forest and makes it somehow random. The "forest" it builds, is an ensemble of Decision Trees, most of the time trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result. One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems. I will talk about random forest in classification, since classification is sometimes considered the building block of machine learning. Below you can see how a random forest would look like with two trees:

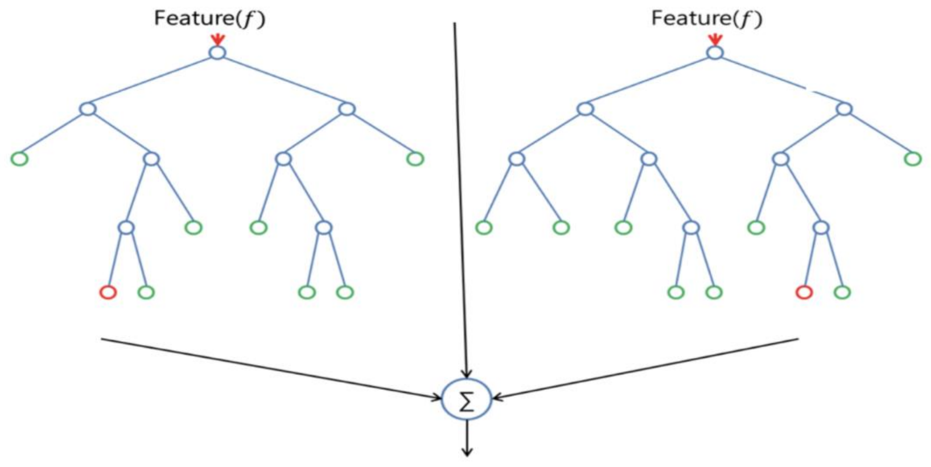


Figure 3.8: Random Forest

Random Forest has nearly the same hyper parameters as a decision tree or a bagging classifier. Fortunately, you don’t have to combine a decision tree with a bagging classifier and can just easily use the classifier-class of Random Forest. Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model. Therefore, in Random Forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node. You can even make trees more random, by additionally using random thresholds for each feature rather than searching for the best possible thresholds (like a normal decision tree does). Sklearn provides a great tool for this, that measures a features importance by looking at how much the tree nodes, which use that feature, reduce impurity across all trees in the forest. It computes this score automatically for each feature after training and scales the results, so that the sum of all importance is equal to 1. If you don’t know how a decision tree works and if you don’t know what a leaf or node is, here is a good description from Wikipedia: In a decision tree each internal node represents a “test” on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). A node that has no children is a leaf. Through looking at the feature importance, you can decide which features you may want to drop, because they don’t contribute enough or nothing to the prediction process. This is important, because a general rule in machine learning is that the more features you have, the more likely your model will suffer from over fitting and vice versa.

Random Forest is a collection of Decision Trees, but there are some differences. If we input a training dataset with features and labels into a decision tree, it will formulate some set of rules, which will be used to make the predictions. For example, if we want to predict whether a participator walking or not, we would collect the data after labeling and take some sensors as a features. If we put the features and labels into a decision tree, it will generate some rules. Then we can predict whether the advertisement will be clicked or not. In comparison, the Random Forest algorithm randomly selects observations and features to build several decision trees and then averages the results. Another difference is that "deep" decision trees might suffer from over fitting. Random Forest prevents over fitting most of the time, by creating random subsets of the features and building smaller trees using these subsets. Afterwards, it combines the sub trees. Note that this doesn't work every time and that it also makes the computation slower, depending on how many trees your random forest builds [6].

Chapter 4

Results and Analysis

After doing the usual Feature Engineering, Selection, and of course, implementing a model and getting some output in forms of a probability or a class, the next step is to find out how effective is the model based on some metric using test datasets. Different performance metrics are used to evaluate different Machine Learning Algorithms. For now, we will be focusing on the ones used for Classification problems. We can use classification performance metrics such as Log-Loss, Accuracy, AUC(Area under Curve) etc.
Accuracy in classification problems is the number of correct predictions made by the model over all kinds predictions made. In the Numerator, are our correct predictions (True positives and True Negatives)(Marked as red in the fig above) and in the denominator, are the kind of all predictions made by the algorithm(Right as well as wrong ones).

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

True Positives (TP): True positives are the cases when the actual class of the data point was 1(True) and the predicted is also 1(True)
True Negatives (TN): True negatives are the cases when the actual class of the data point was 0(False) and the predicted is also 0(False)
False Positives (FP): False positives are the cases when the actual class of the data point was 0(False) and the predicted is 1(True). False is because the model has predicted incorrectly and positive because the class predicted was a positive one. (1)
False Negatives (FN): False negatives are the cases when the actual class of the data point was 1(True) and the predicted is 0(False). False is because the model has predicted incorrectly and negative because the class predicted was a negative one. (0)
The accuracy of the proposed system has been evaluated through extensive testing and experimentation and this chapter gives the insight to our experiments. Here we use two algorithms and take feature alternatively and see which algorithm and feature combination give the better accuracy.

4.1 Decision Tree

In this section we see that using decision tree algorithm how the result differs when we take different features and the accuracy is change. We take a training data set and fit our model using decision tree algorithm. Now we take an unknown data set and predict our contexts depend on that data present in unknown data set. Since gyroscope, light and location sensor are not important for our system that is these sensor values are not more helpful for our system so we can ignore this sensor values as a features and take all three axis of linear acceleration and proximity as a features. Our below table shows that how accuracy is differing when we take various set of feature:

Use Algorithm	**Use as Features**	**Total no of correct prediction using these feathers and this algorithm**	**Accuracy**
Decision Tree	LA-x, LA-y, LA-z, proximity	329	0.951
Decision Tree	LA-x, LA-y, proximity	328	0.947
Decision Tree	LA-y, LA-z, proximity	323	0.933
Decision Tree	LA-x, LA-z, proximity	325	0.939
Decision Tree	LA-x, proximity	308	0.890
Decision Tree	LA-y, proximity	307	0.887
Decision Tree	LA-z, proximity	315	0.910

Here LA-x means x axis of linear accelerometer, LA-y means y axis of linear accelerometer, LA-z means z axis of linear accelerometer. Total number of data present in unknown data set is 346. Here we see that when we take LA-x, LA-y, LA-z and proximity as a features using decision tree algorithm we get the highest accuracy that is using this sensors as features we predict maximum number of correct context.

4.2 Random Forest

In this section we see that using random forest algorithm how the result differs when we take different features and the accuracy is change. We take a training data set and fit our model using random forest algorithm. Now we take an unknown data set and predict our contexts depend on that data present in unknown data set. As we take linear acceleration and proximity in earlier section for the same reason here we take all three axes of linear acceleration and proximity as features. Our below table shows that how accuracy is differing when we take various set of feature:

Use Algorithm	**Use as Features**	**Total no of correct prediction using these feathers and this algorithm**	**Accuracy**
Random Forest	LA-x, LA-y, LA-z, proximity	338	0.976
Random Forest	LA-x, LA-y, proximity	338	0.976
Random Forest	LA-y, LA-z, proximity	329	0.950
Random Forest	LA-x, LA-z, proximity	337	0.973
Random Forest	LA-x, proximity	308	0.890
Random Forest	LA-y, proximity	308	0.890
Random Forest	LA-z, proximity	316	0.913

LA-x means x axis of linear accelerometer, LA-y means y axis of linear accelerometer, LA-z means z axis of linear accelerometer. Total number of data present in unknown data set is 346. Here we see that when we take LA-x, LA-y, LA-z and proximity as a features or LA-x, LA-y and proximity using random forest algorithm we get the highest accuracy that is using these sensors as features we predict maximum number of correct context.

4.3 Discussion

Now if we compare these two algorithms decision tree and random forest we see that when we use random forest algorithm over the same data set it gives better result from decision tree. So we can say that using random forest algorithm and if LA-x, LA-y, LA-z and proximity or LA-x, LA-y and proximity take as a feature set then we have the highest accuracy which is 0.976. Now if we notice over these two feature set LA-x, LA-y, LA-z, proximity and LA-x, LA-y, proximity then we can see that z axis of linear accelerometer’s present or absent is not affected the accuracy. So we can say that our context detection system does not depend on the z axis of the linear accelerometer. Now if we implement our android application to take noise data and apply decision tree or random forest algorithm over this data set so we detect the context that when participator collect noise data at that time what participator do with his mobile device. After seeing the context we can decide which data is noisy or not and take those data which are not noisy and we use these accurate data in different work.

Chapter 5

Conclusion

In this work a system has been proposed in order to detect the context when the participators collect data. This system recognizes that when data was collected at that time participator talking to someone or walking or put mobile device in his or her pocket. A csv file was used to store these sensor data. A decision tree classifier and random forest classifier has been used to do a real time and aggressive event recognition. The data collection was still going on and in a few months the system would perform a lot better then as of now when we will have more data.

References

[1] Xiao-Feng Xie & Zun-Jing Wang. "An empirical study of combining participatory and physical sensing to better understand and improve urban mobility networks.". In: Transportation Research Board (TRB) Annual Meeting. Washington, DC, USA (2015).

[2] Christian Glahn. “The Language of Technical Communication”. In: XML Press that was founded in 2008 to publish books and ebooks on writing, management, content strategy, technology, and social media (2016).

[3] Bleumers, L.” Capturing context: Mobile and pervasive game-play in participatory sensing”. In: K. Mitgutsch, S. Huber, H. Rosenstingl, M. Wagner, & J. Wimmer (Eds.), Context Matters! Exploring and Reframing Games in Context (pp. 168–180). Wien, Austria: New Academic Press (2013).

[4] Samuel, Arthur. "Some Studies in Machine Learning Using the Game of Checkers". In: IBM Journal of Research and Development (1959). **3** (3): 210–229.

[5] Kamiński, B.; Jakubczyk, M.; Szufel, P.”A framework for sensitivity analysis of decision tree”. In: Central European Journal of Operations Research (2017). **26** (1): 135–159.

[6] Shi T, Seligson D, Belldegrun AS, Palotie A, Horvath S. "Tumor classification by tissue microarray profiling: random forest clustering applied to renal cell carcinoma". In: Modern Pathology (April 2005). 18 (4): 547–57.

[7] Deborah Estrin. “Participatory Sensing: Applications and Architecture”. In: IEEE Internet computing (January 2010).