Sensors **2014**, *14*, 20753-20778; doi:10.3390/s141120753

OPEN ACCESS
sensors
ISSN 1424-8220
www.mdpi.com/journal/sensors

Article

Adaptive Activity and Environment Recognition for Mobile Phones

Jussi Parviainen [1,*], Jayaprasad Bojja [1], Jussi Collin [1], Jussi Leppänen [2] and Antti Eronen [2]

[1] Department of Pervasive Computing, Tampere University of Technology, FI-33101 Tampere, Finland; E-Mails: jayaprasad.bojja@tut.fi (J.B.); jussi.collin@tut.fi (J.C.)

[2] Nokia Technologies, FI-33721 Tampere, Finland; E-Mails: jussi.ar.leppanen@nokia.com (J.L.); antti.eronen@nokia.com (A.E.)

* Author to whom correspondence should be addressed; E-Mail: jussi.parviainen@tut.fi; Tel.: +358-3-3115-11.

External Editor: Gianluca Paravati

Received: 10 September 2014; in revised form: 16 October 2014 / Accepted: 20 October 2014 / Published: 3 November 2014

Abstract: In this paper, an adaptive activity and environment recognition algorithm running on a mobile phone is presented. The algorithm makes inferences based on sensor and radio receiver data provided by the phone. A wide set of features that can be extracted from these data sources were investigated, and a Bayesian maximum *a posteriori* classifier was used for classifying between several user activities and environments. The accuracy of the method was evaluated on a dataset collected in a real-life trial. In addition, comparison to other state-of-the-art classifiers, namely support vector machines and decision trees, was performed. To make the system adaptive for individual user characteristics, an adaptation algorithm for context model parameters was designed. Moreover, a confidence measure for the classification correctness was designed. The proposed adaptation algorithm and confidence measure were evaluated on a second dataset obtained from another real-life trial, where the users were requested to provide binary feedback on the classification correctness. The results show that the proposed adaptation algorithm is effective at improving the classification accuracy.

Keywords: mobile sensing; classifier design and evaluation; activity recognition; environment recognition; Bayes classifier; adaptation; pervasive computing

1. Introduction

The design of novel applications for modern smartphones has boomed in the past few years. This is partly due to easy access to information sources previously unavailable, such as the Global Positioning System (GPS) and motion sensors. For application designers, the smartphone environment is very different from the earlier personal computer world. Applications are running continuously, and the environment and activity of the user changes many times a day. Context information is often required to provide only the most relevant information or services to the user. In the present paradigm, the application requires the user to provide this information, such as 'I am in a meeting'. Undoubtedly, it would be more convenient if this context information could be inferred automatically. For this reason, context recognition using the built-in sensors of modern mobile phones is an active research topic. The research problem is far from trivial, as the sensors provide only indirect information; there is no sensor for directly detecting 'meetings' or 'jogging'. Furthermore, the required accuracy for the recognition is relatively high, and misclassifications can lead to very annoying user experiences. In this paper, we introduce a framework for automatic activity and environment classification that can be easily implemented and evaluated on any modern smartphone.

In this paper, we consider activity and environment recognition from mobile phone sensor and radio receiver data. The goal of activity recognition algorithms is to output information on the activity of the user. In particular, we consider algorithms that try to classify the physical activity of a user, such as walking, running, driving a car, riding a bicycle or being still. By environment recognition, we mean the automatic recognition of user surroundings, such as whether the user is in a meeting, at the office or inside a vehicle.

Existing literature on this topic includes many interesting examples of what can be done with sensor systems carried by the user. Altun *et al.* present a study of classifying human activities using body-worn inertial and magnetic sensors [1]. The authors perform a comparative study of different methods. The results in the paper indicate that, in general, Bayesian decision making results in the highest correct classification rate with a relatively small computational cost. Gu *et al.* describe a body sensor network for activity recognition for multiple users [2]. Könönen *et al.* describe several classification and automatic feature selection algorithms, which are compared in the problem of context recognition [3]. Reddy *et al.* present a classification system that uses a mobile phone with a built-in GPS receiver and an accelerometer [4]. The system recognizes the following transportation modes: whether an individual is stationary, walking, running, biking or in motorized transport. In the UPCASEproject [5,6], multiple sensors, such as accelerometers, light, sound, humidity, temperature and GPS sensors, were connected to a mobile phone via Bluetooth. Decision trees were used to obtain the activity (walking, running, standing or lying) of the user. The system also detects whether the device is indoors or outdoors by using information on the availability of GPS signals. After recognition, the context of the user can be published on social networks, such as Twitter or Facebook. The project CenceMe [7] also describes an activity recognizer where the results can be sent to social networks. The works in [8,9] present activity-based pattern matching in different mobile environments. Bancroft *et al.* use a foot-mounted inertial measurement unit (IMU) with a GPS to detect activity and the environment of the user [10]. In [11], Pei *et al.* use location and motion tracking to build a context-aware system. Susi *et al.* present in [12] motion mode recognition for pedestrian dead reckoning. A smartphone-based lightweight hierarchical activity recognition framework is presented in Han *et al.* [13], where the recognition of 15 activities is done using the accelerometer, gyroscope, proximity sensor and GPS modules.

Our novel contribution to prior work is that we introduce an algorithm for user-specific adaptation of the context model parameters from user-friendly binary feedback. Furthermore, the introduced adaptation method is able to provide a confidence measure about the correctness of a classification for the host application.

This paper is organized as follows. In Section 2, we present the data used for training and testing the proposed classifier algorithms. Section 3 describes the features that were extracted from the collected data. Section 4 presents the feature compression algorithm used. An introduction to our classifier is presented in Section 5. In Section 6, we introduce our adaptation method. Section 7 presents the results from multiple classifiers and the results obtained with the proposed adaptation method. Finally, we conclude the paper in Section 8.

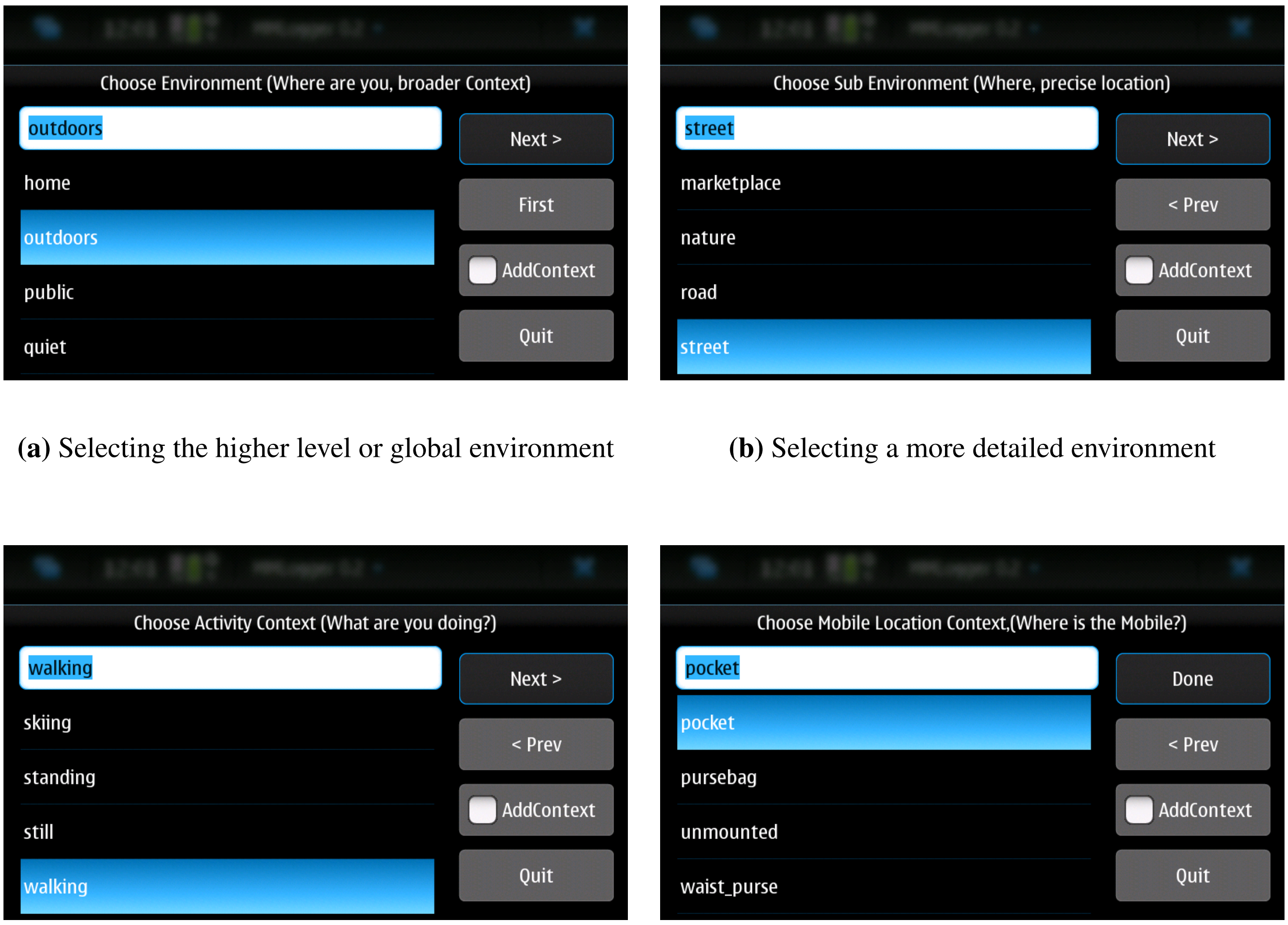
2. Data Collection

To be able to train and evaluate the classification system, a comprehensive annotated data set was collected. We chose Nokia N900 mobile phones for data collection, the main reason being easy access to sensor data via application programming interfaces and the open source nature of the Maemo platform. Two data collection campaigns were arranged. The first campaign, with 21 users, was arranged to collect a basic training set with full annotations, providing enough data for the design and validation of the feature extraction and pattern recognition algorithms. During the first campaign, the users were requested to manually input their current activity and environment from the mobile phone user interface (UI). The second campaign, with 10 users, was arranged for testing the adaptation of algorithm parameters for each individual user. The campaign was done with context recognition software running and with only binary yes, no input, indicating whether or not a recognition result was correct or not, required from the user. It should be noted that the first campaign required much more effort from the test persons since the users were providing full annotation compared to the binary yes/no answers in the second campaign.

2.1. The First Data Collection Campaign

In the first data collection campaign, the users were requested to carry the phone along with them as they normally would during three to four weeks. Our software would then ask the users to annotate their environment (e.g., outdoors, street), activity (e.g., walking) and the location of the phone (e.g., pocket) at set intervals. Once the annotation was input the phone sensors would then record data for two minutes. The default time interval between recordings was 20 min, but could be set by the user to be from a minimum of 10 min to a maximum of 60 min. The annotations were tagged to the recorded sensor data by a file name convention. The user interface used by the users to provide the annotations is depicted in Figure 1. There was a fixed set of predefined environments, activities, and phone locations available but the users could also add their own if they thought none of them matched their own situation.

Figure 1. Data collection program.



(a) Selecting the higher level or global environment **(b)** Selecting a more detailed environment

(c) Selecting the physical activity **(d)** Selecting the mobile phone location

After the user provided an annotation for the context, the sensors on the device were activated and a short period of data collection was performed. The used sensors and statistics of the obtained data are presented in Tables 1 and 2. The collected sensor data is described in more detail below.

Table 1. Data trial statistics.

Total Number of Users	21
Number of men	15
Number of women	6
Median age	31
Mean age	32.3
Total number of annotations	3478
Number of unique annotations	620

Table 2. Collected data.

Sensor	**Sampling Rate**	**Sample Length**
3-axis Accelerometer	100 Hz	1 min
GPS	1 Hz	2 min
WLAN	$\frac{1}{12}$ Hz	1 min
Bluetooth	recorded once	snapshot
GSM/3G Cell information	1 Hz	1 min
Audio	16 kHz	1 min

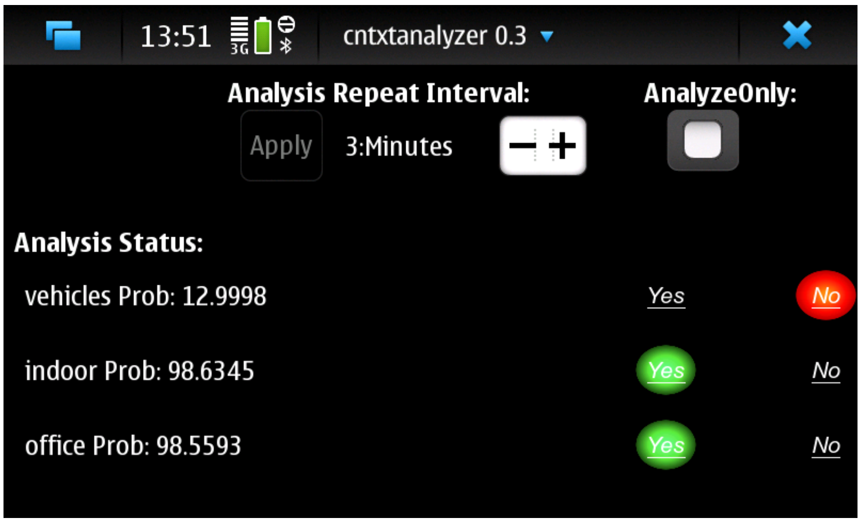
- *Accelerometer*: The sampling rate of the 3-axis accelerometer was approximately 100 Hz. Data was collected for one minute.
- *GPS*: Three minutes of GPS data was collected with a 1 Hz sampling rate. The recorded data included position, velocity, satellite elevation and azimuth angles and signal to noise ratio.
- *WLAN*: WLAN signal strengths and WLAN station identifiers (IDs) and Media Access Control (MAC) addresses were recorded. Samples were collected for one minute and the sampling interval was approximately 12 s. Thus, five scans were performed during one minute.
- *Bluetooth*: A single scan of the Bluetooth environment was performed, and the names and addresses of the visible Bluetooth devices were logged.
- *GSM/3G Cell information*: Global System for Mobile Communications (GSM) or 3rd generation mobile communications (3G) location area codes (LAC) and cell IDs were recorded
- *Audio*: One minute of audio with a single mono channel with 16-bit resolution at 16 kHz sampling rate was recorded.

2.2. Data Campaign for Adaptation

The second campaign was done using Nokia N900 devices wherein the developed classification algorithm for activity (classes: *running, bicycling, walking, standing, table, vehicles*) and environment (classes: *office, nature, street/road, home, restaurant/pub/cafe*) was running. Algorithms are described in detail in the Section 5.

The user feedback was requested by displaying a green button for "yes" and a red button for "no" right after the classification result was available. The user interface of the program is depicted in Figure 2. If the user did not provide an answer, the data was not used for adaptation. For this data collection campaign, ten users who did not participate the first campaign were selected for two week trial. In total, there were 2674 classification results with the version that asks the feedback.

Figure 2. During the second trial, user were able to give input on whether the classification was correct or incorrect.



Analysis Repeat Interval: 3:Minutes

Analysis Status:

vehicles Prob: 12.9998

indoor Prob: 98.6345

office Prob: 98.5593

3. Feature Extraction

This section presents the features extracted from the mobile phone sensor and radio receiver data. Multiple features were implemented and used, as the redundancy can be removed using a compression algorithm presented in Section 4.

3.1. GPS Features

The following features are extracted from the GPS receiver.

- The median carrier to noise ratio value of one minute of GPS data. The carrier to noise ratio is an estimate of the received power most GPS/GNSS (Global Navigation Satellite System) receivers provide for each tracked satellite. A low carrier to noise ratio indicates that there are obstacles such as building walls in the satellite-receiver path. The carrier to noise values are much lower indoors than outdoors [14].
- The minimum elevation angle value from the satellites that are used for getting the GPS position fixes. The elevation angle is the angle between the local horizontal plane and the user-satellite vector. In urban canyons satellites with a small elevation angle are rarely tracked.
- The maximum speed value from one minute of GPS data. Preferably a Doppler-based speed estimate computed by the receiver, but can also be derived from two consecutive position fixes. High speed values indicate that the device is in a moving vehicle.
- The best horizontal accuracy value of the GPS position fixes, *i.e.*, the smallest horizontal accuracy value that the GPS device outputs. Most receivers provide this kind of value, the best accuracy is obtained in rural areas with a clear sky view.
- Time to first fix (TTFF), *i.e.*, the time passed from the time of power-up to the instant when the first location measurement is obtained.

3.2. WLAN Features

The features extracted from the WLAN receiver are

- The number of unique MAC addresses.
- The number of unique station names. Usually in public areas one observes more MAC addresses than station names due to having several access points connected to the same WLAN network, whereas in private places such as homes one usually observes less MAC addresses per station name.
- The average signal strength on a scale from one to five.
- The average signal strength given on a decibel scale.
- The standard deviation of the signal strength on a scale from 1 to 5. A high standard deviation usually indicates that user is on the move.
- The standard deviation of the signal strength on a decibel scale.
- The maximum signal strength on a decibel scale. A high signal strength usually indicates that the user is indoors.
- The maximum signal strength on a scale from 1 to 5.

3.3. Bluetooth Features

A single feature related to the Bluetooth radio environment is used. The feature is the number of visible Bluetooth devices. A high number of Bluetooth devices indicates that the user is in a public place.

3.4. Cellular Network Environment Related Features

The cellular base stations to which the mobile phone connects during a minute are logged. From the logged data, the following statistics are used as features:

- The number of unique Cell IDs (base transceiver station identifiers).
- The number of unique location area codes (LAC).
- The number of Cell ID changes per minute.
- The number of LAC changes per minute.
- The standard deviation of the strength of the signal to the transceiver station the mobile phone is connected.

3.5. Accelerometer Features

To remove the effect of orientation of the mobile phone, the Euclidean norm of the three dimensional accelerometer signal is used. The following features are extracted from the accelerometer signal norm:

- The variance of acceleration.
- The value of the maximum spectrum magnitude peak.
- The bin index of the spectrum magnitude maximum value.
- The difference between the maximum and minimum spectrum magnitude value.
- The number of zero crossings.

3.6. Audio Features

For the audio features, likelihoods given by a global model trained for the audio data are used. The likelihood features are obtained as follows. First, during an offline training phase, 13 mel-frequency cepstral coefficients (MFCCs) along with their first- and second-order derivatives are calculated from audio training data [15]. The MFCCs are calculated from audio data recorded at 16 kHz sampling rate using a window of 30 ms and a frame-skip of 40 ms. After this, a single Gaussian mixture model (GMM) with 32 component densities is trained. Once the above offline training phase is complete, the likelihood features can be calculated for input audio data. This is done by, first, calculating MFCCs for the input audio data and then calculating the mixture likelihoods of the GMM for the MFCC feature vectors. The mixture likelihoods for the whole audio clip are then output as the likelihood features. Thus, for each input audio clip, we obtain a vector of 32 likelihood values. By using likelihood features we can represent a whole audio clip with a single feature vector instead of a sequence of MFCC vectors. An alternative way to represent an audio clip with a single feature vector would be to, for example, use the mean of the calculated MFCCs. Also, a covariance matrix could be calculated. However, information is lost when using only the mean of the MFCCs and for shorter audio clips, there might not be enough data to reliably calculate the variances or a covariance matrix.

4. Feature Compression

Given that a smartphone includes a multitude of relevant sensors for the classification task, the number of features is inevitably very high. To avoid numerical problems and make the adaptation more straightforward it is necessary to ignore redundant and irrelevant features. In order to reduce the dimensionality of the feature space, we use a method based on a transformation with two stages [16,17].

First, all the training features are concatenated in a $k - by - n$ matrix G, where k is the number of features and n the number of feature vectors. In the following, by covariance matrix Σ we mean the sample covariance matrix obtained from the training data G.

The averaged within-class covariance matrix is defined as

$$S_W = \frac{1}{C}\sum_{j=1}^{C}\Sigma_j \tag{1}$$

where C is the number of classes (in this work we have multiple sets of classes C, e.g., the activities and environments have separate training data sets and classifiers. Nevertheless, for notational simplicity here we only use a single set of classes C). and Σ_j is the covariance matrix of class j. The required transformation matrix R_1 is obtained using the eigenvectors (in the matrix U) and the rank of S_W (r):

$$R_1 = \Lambda_r^{-\frac{1}{2}} U_r^T \tag{2}$$

where the diagonal matrix Λ contains the r eigenvalues of S_W, and thus $R_1 S_W R_1^T = I$, an $r - by - r$ identity matrix. This transformation scales the raw features conveniently. The dimension reduction removes redundant features, for example those that are constant all the time.

The next phase is to compress the class mean information. The between-class covariance matrix is defined as

$$S_B = \frac{1}{C} \sum_{j=1}^{C} (\boldsymbol{\mu}_j - \boldsymbol{\mu})(\boldsymbol{\mu}_j - \boldsymbol{\mu})^T \tag{3}$$

where $\boldsymbol{\mu}$ is the mean of the within-class means $\boldsymbol{\mu}_j$. S_B describes how far apart the class means are from each other. In the second phase, we seek for a transformation $R_2 = V_\nu^T$ that diagonalizes S_B:

$$V_\nu^T (R_1 S_B R_1^T) V_\nu = \tilde{\Lambda}_\nu \tag{4}$$

where ν is the rank of $R_1 S_B R_1^T$ and the matrix $V_\nu = [\mathbf{v}_1, \ldots, \mathbf{v}_\nu]$ consists of ν eigenvectors of $R_1 S_B R_1^T$ (*i.e.*, eigenvectors of S_B *after* the first transformation R_1). Similarly, $\tilde{\Lambda}_\nu$ is a diagonal matrix containing ν eigenvalues.

There are at most $C - 1$ non-zero eigenvalues (after the transformation the number of features is reduced at maximum to $C - 1$) and after the transformation $R_2 R_1$ the first feature vector can be considered as the most important for the classification purposes. This geometric interpretation of linear discriminant analysis (LDA) is very useful in displaying the data, and it is straightforward to implement. The matrix $R = R_2 R_1$ can be computed off-line, and the software in the mobile phone only needs to perform a matrix-vector multiplication for the raw features. In addition, some features clearly do not follow a Gaussian distribution (number of WLAN APs, for example), but after the feature compression we can quite safely work with Gaussian models.

5. Classification

For the classification task, we use a Bayesian maximum a posteriori classifier (MAP). For comparison, the classification results are also calculated using support vector machines (SVM) and decision trees (DT). Nevertheless, the probabilistic Bayesian MAP method has the following advantages:

(1) If the features follow Gaussian distribution the classifier is straightforward to implement. Furthermore, the output is optimal in a probabilistic sense, and it is possible to derive a proper confidence measure for the classification result.

(2) Adaptation can be performed by changing the parameters of the class distribution.

(3) The Bayesian approach allows a straightforward implementation of recursive filters. For example, data from multiple users can be combined recursively from the same geolocation.

With the underlying Gaussian assumption, the pattern recognition problem simplifies to a discrimination between p multivariate normal populations. First, a training data set with known states is collected to obtain

$$\mathbf{z}_j \sim N(\boldsymbol{\mu}_j, \Sigma_j) \tag{5}$$

the distribution of the observed $q - by - 1$-vector $\mathbf{z}$ given that the observation comes from the class j. At this stage, the mean vector $\boldsymbol{\mu}_j(q \times 1)$ and the covariance matrix $\Sigma_j(q \times q)$ are here assumed to be perfectly known (very representative training data set). We also assume that for all classes $\Sigma_j > 0$. If Equation (5) holds, the density function of $\mathbf{z}_j$ is

$$f_{\mathbf{z}_j}(\mathbf{z}; \boldsymbol{\mu}_j, \Sigma_j) = \frac{1}{(2\pi)^{q/2}\sqrt{|\Sigma_j|}} \exp[-\frac{(\mathbf{z} - \boldsymbol{\mu}_j)^T \Sigma_j^{-1} (\mathbf{z} - \boldsymbol{\mu}_j)}{2}] \tag{6}$$

A new observed $\mathbf{z} = \mathbf{z}_x$ can then be classified by maximizing Equation (6) over all the classes $j = 1...p$. To assign a probability to the classification result, we need to use unconditional prior probabilities $P(C = j)$, and assume that all the possible classes are included. Then the probabilities for all the classes are obtained from the Bayes' rule:

$$P(C = j|\mathbf{z}_x) = \frac{f_{\mathbf{z}_j}(\mathbf{z}_x) P(C = j)}{f_{\mathbf{z}}(\mathbf{z}_x)} \tag{7}$$

where $f_{\mathbf{z}}$ is the unconditional density function for the observation $\mathbf{z}$. However, Equation (7) may not work well in practical classification problems as the Gaussian assumption does not necessarily hold. In addition, the class parameters $\boldsymbol{\mu}_j, \Sigma_j$ are only estimates obtained based on incomplete training data and the list of classes is not complete – we do not have data from all the possible activities or environments. For these reasons it is necessary to modify this basic model.

5.1. Combining Independent Classifiers

The information provided by all the p likelihoods $f(\mathbf{z}_x; \boldsymbol{\mu}_j, \Sigma_j)$ of the classes $j = 1...p$ becomes interesting when combining different classifiers, such as an audio-based classifier and radio receiver based classifiers. The motivation for this approach is that often there are specialized classifiers which operate on certain type of sensor data (e.g., audio or accelerometer). Different sensors have very different data rates and feature statistics, and often it is impossible to concatenate raw features to the same feature vectors to perform feature level fusion. Instead, if we have a classifier, the output of this classifier can be fed to another classifier as a feature vector. This makes it possible to roughly match the data rates of different sensors, as in the case of a radio receiver producing a set of data every one minute combined with the output of the audio classifier at the same rate. In addition, in this case the application software does not have access to the classifiers, but only obtains the result containing the likelihoods for all the possible classes, and the decision itself.

Thus it is convenient to use these likelihoods as features for another classifier that is running on a higher level in the software. By doing this we would not need to know any details of the original classifier. First we need to investigate how the $f_{\mathbf{z}_j}(\mathbf{z})$ itself is distributed, considering f as a deterministic function of a random variable $\mathbf{z}$. First, assume that the new observation $\mathbf{z} = \mathbf{z}_x$ comes from the population class $j = 1$. To simplify the computation we take the natural logarithm of the likelihood, yielding a constant term (for each class) $\ln \frac{1}{\sqrt{|2\pi\Sigma_1|}}$ plus the term

$$-\frac{(\mathbf{z}-\boldsymbol{\mu}_1)^T\Sigma_1^{-1}(\mathbf{z}-\boldsymbol{\mu}_1)}{2} \tag{8}$$

As the actual class is correct (*i.e.*, $\mathbf{z}$ and $\boldsymbol{\mu}_1$ belong to the same class), by Theorem A.85 in [18], the numerator in Equation (8) is chi-square distributed with q degrees of freedom.

Then, we need to study the distribution of incorrect class (*i.e.*, classifier parameters are wrong). The distributions of likelihoods for the incorrect classes is not as straightforward. We will have a quadratic form $(\mathbf{z}+\mathbf{a})^TC(\mathbf{z}+\mathbf{a})$, where the vector $\mathbf{a}$ and the matrix C have nothing to do with the parameters of the distribution of $\mathbf{z}$. From [19] we find that

$$P\{(\mathbf{z}+\mathbf{a})^TC(\mathbf{z}+\mathbf{a}) \le t\} = \sum_{k=0}^{\infty} c_k F(\cdot, t) \tag{9}$$

where F is a chi-square cumulative distribution function (CDF). This formula can be used to obtain a numerical approximation of the CDF for the likelihoods in the case of incorrect class. The normally distributed features produce chi-squared likelihoods when the class model is correct. We also know the shape of the distribution (Equation (9)) in the case of an incorrect class. Thus, if we leave the assumption of normally distributed features aside, we can train a new classifier using directly the likelihoods of the classifier running on a lower level. Feature compression is clearly necessary when feature vectors with a non-normal degenerate distribution are added to the classifier. We used this approach to combine an existing audio-based environment classifier with the radio-based classifier developed in this study. In our existing audio classifier we could only touch the outputs of the classifier (*i.e.*, in this case likelihoods). In addition, these likelihoods were not from the same classes that were are interested in the final classification. Thus, classical ensemble methods [20] were not valid in this case.

6. Adaptation

It is clear that one single global model for activity or environment classification will not perform well in the case of mobile phones. There are individual differences that affect the activity monitoring between users (walking pace, step impact, bicycling speed, *etc.*). The same applies for the environment classification, for example, offices tend to have different WLAN coverage and there are differences in the audio and radio environment between cities. For that reason, adaptation via user feedback is needed. To reduce the amount of work the user has to do it is preferable to ask only binary information from the user: given the global distribution model the software outputs the classification result and prompts the user to provide a *yes* or *no* answer depending on whether the classification result was correct or not. This user input is then used to adapt the distribution parameters. A similar idea is also proposed in [3], but we will add the likelihood distribution models for the adaptation process.

6.1. Adaptation Algorithm

Using likelihood distributions we can derive a new optimization criteria: the *yes* answer likelihoods should be chi-square distributed with q degrees of freedom *after* the adaptation (this was the definition of numerator in Equation (8)). In addition, to reduce the amount of incorrect classification results, the *no* answer likelihoods should be far away (towards $+\infty$) from this distribution.

In this paper we use the following optimization criterion: modify the distribution parameters so that the function

$$f = \frac{|A|}{N(yes)} + \frac{|B|}{N(no)} \tag{10}$$

is minimized. It should be noted that it is possible to use other suitable adaptation functions, for example, the one proposed in [3]. Function in the paper was chosen somewhat heuristically. In our function symbols $N(yes)$ and $N(no)$ denote the total numbers of *yes* and *no* answers and $|A|$ and $|B|$ denote the number of items in the set A and B, respectively. The sets are defined as

$$A = \{L_i(yes) | L_i(yes) > \chi_{95}\} \tag{11}$$

and

$$B = \{L_i(no) | L_i(no) < \chi_{95}\} \tag{12}$$

Here i is the index of the current class and $L_i(yes)$ and $L_i(no)$ are the set of likelihood values corresponding to observations with a *yes*-label and a *no*-label, respectively, and χ_{95} is the point where the cumulative Chi-square distribution and has a value of 0.95. The likelihood values are

$$L_i = (\mathbf{z} - \boldsymbol{\mu}_i)^T s_i \Sigma_i^{-1} (\mathbf{z} - \boldsymbol{\mu}_i) \tag{13}$$

The parameters to be adapted are the scales s_i and the class means $\boldsymbol{\mu}_i$. After the adaptation, the new covariance will be $s_i \Sigma_i$.

The motivation for the function Equation (10) is that it resembles the Neyman-Pearson type *false positive*, *false negative* terminology [21,22]. Minimization of the function f can be thought of as minimizing the false positive and false negative errors. However, the actual classification method is not taken into account here.

In essence, the adaptation algorithm attempts to minimize the number of samples with a *yes*-label that do not fit the model distribution (in Equation (6)), meaning that they fall outside the 95% threshold. Furthermore, the algorithm attempts to minimize the number of samples with *no*-label that fit the model distribution well, meaning that they fall inside the 95% threshold. To minimize the function f in Equation (10), *i.e.*, to find

$$\underset{s_i \in \mathbb{R}^+, \boldsymbol{\mu} \in \mathbb{R}^N}{\text{argmin}} \; f \tag{14}$$

We used the Matlab function *fminsearch*, which finds the minimum of the function using the simplex search method [23]. Functions f are implemented separately for activity and environment. The parameters to be adapted are the scale s, so that the new covariance will be $s\Sigma$, and the class means $\boldsymbol{\mu}$. For *fminsearch* the scale is represented as $d = s-1$, so that the adaptation starts from the zero-vector. The result of *fminsearch* then tells how the parameters need to be adapted. If the adaptation is successful, we should obtain better classification results in the future. Furthermore, the likelihood distribution should approximately follow the model we have imposed - regardless of the original assumption of a Gaussian distribution.

The information contained in the sample distribution (obtained using the Matlab function *ksdensity*, for example) of yes-likelihoods and no-likelihoods is very useful for classifier diagnostics. For example, if the real feature distribution is actually multimodal, the yes-likelihood distribution should not look like a Chi-squared distribution.

6.2. Simulated Example

To illustrate our adaptation algorithm, we present a simulated example in this section. In this simulation we assume that we have trained a MAP classifier having parameters $\hat{\mu}$ and $\hat{\Sigma}$. However, the true parameters for the individual user would be μ and Σ and there exists a nearby untrained class with parameters μ_{other} and Σ_{other}. The numerical values for the distribution parameters are given in Table 3 and the data is illustrated in Figure 3. Using Equation (9) the probability density function for the yes-likelihoods ($p_{\mathbf{L}|yes}$),

$$\begin{aligned} s_1 &= (\mathbf{z}-\hat{\boldsymbol{\mu}})^T\hat{\Sigma}^{-1}(\mathbf{z}-\hat{\boldsymbol{\mu}}) \\ \mathbf{z} &\sim N(\boldsymbol{\mu},\Sigma) \end{aligned} \tag{15}$$

and the probability density function for the no-likelihoods ($p_{\mathbf{L}|no}$),

$$\begin{aligned} s_2 &= (\mathbf{z}-\hat{\boldsymbol{\mu}})^T\hat{\Sigma}^{-1}(\mathbf{z}-\hat{\boldsymbol{\mu}}) \\ \mathbf{z} &\sim N(\boldsymbol{\mu}_{other},\Sigma_{other}) \end{aligned} \tag{16}$$

can be derived [24–27]. The CDF for the random variable s_i is given by $P\{s_i \leq t\} = \sum_{k=0}^{\infty} c_k F(\cdot, t)$. The algorithm for the coefficients ($c_0 \ldots c_N$) can be found from [19]. The likelihood distributions are shown in Figure 4 (top). The aim of the adaptation process (using Equation (10)) is to push the "no" distribution ($p_{\mathbf{L}|no}$) to the right side of the χ_{95} threshold and the "yes" distribution ($p_{\mathbf{L}|yes}$) to the left side. Assuming that the adaptation is perfect, the estimated parameters $\hat{\mu}$, $\hat{\Sigma}$ equal the true parameters μ, Σ. Thus, the probability density function for the yes-likelihoods ($p_{\mathbf{L}|yes}$) becomes

$$\begin{aligned} s_3 &= (\mathbf{z}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{z}-\boldsymbol{\mu}) \\ \mathbf{z} &\sim N(\boldsymbol{\mu},\Sigma) \end{aligned} \tag{17}$$

and the probability density function for the no-likelihoods ($p_{\mathbf{L}|no}$) become,

$$\begin{aligned} s_4 &= (\mathbf{z}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{z}-\boldsymbol{\mu}) \\ \mathbf{z} &\sim N(\boldsymbol{\mu}_{other},\Sigma_{other}) \end{aligned} \tag{18}$$

which are shown in Figure 4 (bottom). The chosen threshold is somewhat arbitrary, but it should be noted that in a real situation the parameters μ_{other} and $\Sigma other$ are not known, so it is difficult to avoid heuristics. However, the distribution for s_3 Equation (18) is known in the case of perfect adaptation. This provides means to find a stopping criterion for the adaptation. For example, when a sample distribution function

$S_n(x)$ of yes-likelihoods is sufficiently close to the optimal χ^2 distribution function the adaptation process can be stopped. The measure for closeness can be, for example, Kolmogorov's D_n,

$$D_n = \sup_x |S_n(x) - F(x)| \tag{19}$$

where F is the known χ^2 cumulative distribution function with q degrees of freedom.

Figure 3. Simulated distributions.

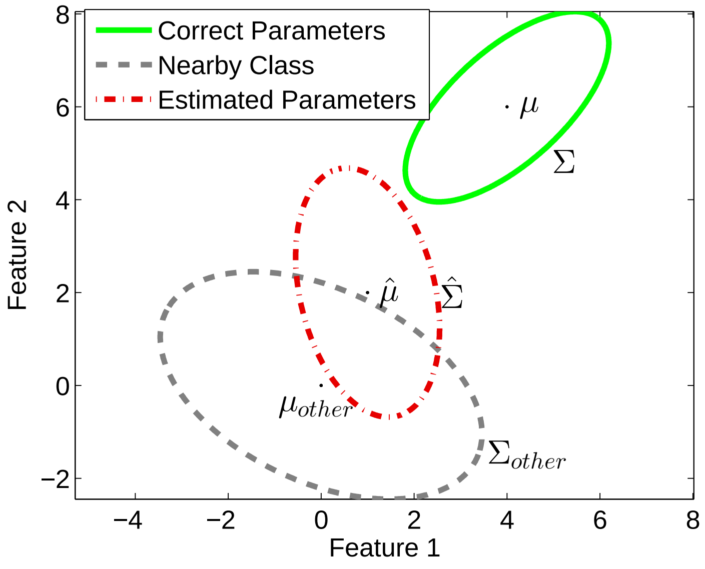
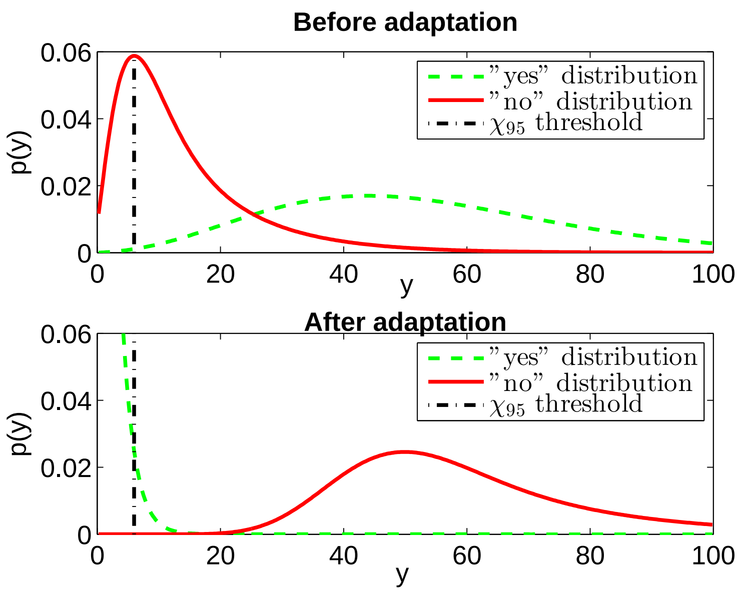


Table 3. Parameters used in the simulation.

Parameter	Value
μ	$\begin{bmatrix} 4 & 6 \end{bmatrix}$
Σ	$\begin{bmatrix} 0.8 & 0.5 \\ 0.5 & 0.7 \end{bmatrix}$
$\hat{\mu}$	$\begin{bmatrix} 1 & 2 \end{bmatrix}$
$\hat{\Sigma}$	$\begin{bmatrix} 0.4 & -0.2 \\ -0.2 & 1.2 \end{bmatrix}$
μ_{other}	$\begin{bmatrix} 0 & 0 \end{bmatrix}$
Σ_{other}	$\begin{bmatrix} 2 & -0.6 \\ -0.6 & 1 \end{bmatrix}$

Figure 4. Simulated likelihood distributions before and after perfect adaptation.



6.3. A Confidence Measure

A fundamental requirement for sensor fusion is to obtain some kind of confidence measure for the decisions made. With this kind of information sensors can be be weighted based on the accuracy, or more sensors may be turned on if, for example, the uncertainty is high. We already described a likelihood based method that can be extended to the adaptation case easily: to obtain a confidence measure for the classifications after the adaptation, we will try to predict the probability of the user answering *yes*, given the current observed features and the adaptation data. To do this we use the Bayes' theorem,

$$P_{yes|\mathbf{L}} = \frac{p_{\mathbf{L}|yes} P_{yes}}{p_{\mathbf{L}}} \tag{20}$$

where $P_{yes|\mathbf{L}}$ is the probability we are looking for, that is, the probability of the user answering *yes* given the likelihood value of the class that was selected. $p_{\mathbf{L}|yes}$ is the (empirical) conditional density of the yes-likelihood. P_{yes} is the probability of successful detection after the adaptation. $p_{\mathbf{L}}$ is the weighted combination of the *yes* and *no* likelihoods. The *no* likelihood distribution enables us to normalize the posterior probability, which would be very difficult without the adaptation process [28]. It should be noted that in the case of the user answering *no*, we do not know the actual correct class. This is a drawback of the simple binary feedback method, but our method of predicting the probability of the *yes* answers does not require the knowledge of the true class in the case of a negative answer. To avoid heavy computations on the mobile phone, the required distribution estimates can be computed on the server side. The necessary information to be transmitted to the server would be the $L_i(yes)$ and $L_i(no)$ values, and the server would return the new classifier parameters along with $p_{\mathbf{L}|yes}$,P_{yes} and $p_{\mathbf{L}}$.

7. Results and Discussion

We implemented and tested three different classifiers: an indoor/outdoor classifier, an environment classifier, and an activity classifier. In addition to the maximum a posteriori (MAP) classification method presented in Section 5 we present results using decision trees (DT) and support vector machines (SVM) for comparison. For those purposes MatLab's *ClassificationTree class* with default options and SVM functions with linear kernel function were used. In the case of SVM, one-versus-others classifiers were implemented and the class was selected based on the classifier which classified the test data with the greatest margin. Evaluation of the accuracy of the all algorithms was first done by using leave-one-user-out cross validation, where at each round all the data from a user is held out and the system is trained with the data from the remaining users. This way we can ensure that the system is not overfitting to the individual characteristics of any user. As the training data was collected in opportunistic way, the class imbalance problem [29] may occur. Thus, equal priors were given for all classes in MAP classifier as we cannot prefer any class over another.

In the second phase, to evaluate our adaptation method presented in Section 6 the second data collection campaign was arranged and the method for classifier adaptation was evaluated.

7.1. Indoor/outdoor Classification

A study to evaluate the accuracy of a simple $\{indoor, outdoor\}$ classifier was performed. In this task GPS signals are very relevant as the low power levels prevent satellite signal tracking in most buildings. The binary classification induces some problems in defining the boundaries. We defined driving a passenger car or travelling in a train as *outdoor* environment, the main reason being that radio receivers are less affected by vehicles than buildings.

The confusion matrix for the MAP classifier is presented in Table 4.

Table 4. Confusion matrix of the classes *indoor* and *outdoor* using MAP classifier. The values denote percentages.

true \ decision	**In**	**out**
In	82	18
Out	18	82

Especially good features for differentiating between these classes were the WLAN signal strengths, which are usually higher indoors than outdoors. By using only the WLAN signal strength we could obtain classification accuracies of 78% and 82% for indoor and outdoor, respectively. Nevertheless, the results may become very different if the data would be collected only from areas where the WLAN station density is low.

By using DT and SVM we could achieve higher classification rates for the *indoor* and *outdoor* class. Confusion matrices, where all the features from GPS, WLAN and Bluetooth were used, are presented in Tables 5 and 6 for DT and SVM, respectively. This shows that the decision tree gives the highest classification rate with these classes. However, to test our adaptation method we continued to use MAP algorithm also for these classes.

Table 5. Confusion matrix of the classes *indoor* and *outdoor* using decision trees. The values denote percentages.

true\decision	In	out
In	96	4
Out	6	94

Table 6. Confusion matrix of the classes *indoor* and *outdoor* using SVM. The values denote percentages.

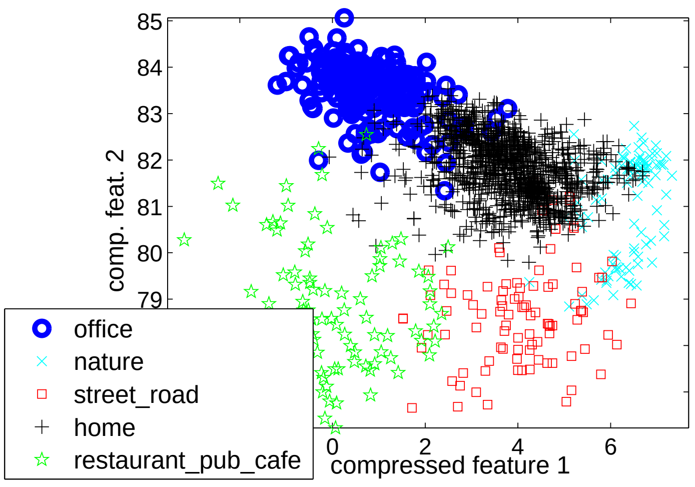
true\decision	In	out
In	86	14
Out	6	94

7.2. Environment Classifier

For environment classification we selected the set of {*restaurant/pub/cafe, office, home, street/road, nature*} as the categories to be recognized. The list includes environments that would be useful for many mobile applications, but it is short enough for reasonable accuracy analysis. In environment classification all the features from the GPS, WLAN and Bluetooth were selected. In addition, the likelihoods output by the audio classifier were used as features. Features from the accelerometer were not used in this case as they did not improve the recognition accuracy.

Figure 5 shows the scatter plot of the first two features after the compression method presented in Section 4 was done. Some of the most important features were audio features and GPS TTFF and maximum WLAN signal strength. The confusion matrix of leave-one-user-out cross-validation method is presented in Table 7.

Figure 5. Two main features after the feature compression.



DT and SVM results for the classes *office*, *nature*, *street/road*, *home* and *restaurant/pub/cafe* are presented in Table 8. As the some classes

Class average classification accuracy for MAP classifier is 70%, for DT 68% and for SVM 63%. Thus, there are no great differences between these classifiers. It should be noted that default parameters from MatLab's *ClassificationTree class* were used for DT and SVM, in addition, the class imbalance problem was not taken into account for these classifiers, thus by taking these aspects into account, accuracies may slightly increase. However, the main idea in the paper was not to show whether the Bayesian classifier can beat DT or SVM, but to show that our Bayesian classifier is not dramatically inferior to state of art methods and thus it is feasible to propose adaptation algorithm for this Bayesian approach. Comprehensive comparative between classifiers is thus left out.

Table 7. Confusion matrix of five environment classes *office*, *nature*, *street/road*, *home* and *restaurant/pub/cafe* using MAP classifier. The values denote percentages.

true\decision	Off.	Nat.	Str.	Home	Res.
Off.	75	0	2	7	17
Nat.	0	67	21	5	8
Str.	1	9	72	6	12
Home	3	5	15	62	15
Res.	7	0	21	0	73

Table 8. Confusion matrix of five environment classes *office*, *nature*, *street/road*, *home* and *restaurant/pub/cafe* using the decision tree classifier. The values denote percentages.

true\decision	Off.	Nat.	Str.	Home	Res.
Off.	70	0	1	20	10
Nat.	2	65	20	6	7
Str.	2	12	62	11	13
Home	4	2	9	77	8
Res.	2	1	26	4	68

Table 9. Confusion matrix of five environment classes *office*, *nature*, *street/road*, *home* and *restaurant/pub/cafe* using the support vector machine classifier. The values denote percentages.

true\decision	Off.	Nat.	Str.	Home	Res.
Off.	87	0	0	5	8
Nat.	2	36	26	23	13
Str	3	13	56	8	20
Home	2	5	5	73	16
Res.	18	3	8	7	65

7.3. Activity Classifier

The following classes were included in the activity recognition:

- table; phone is placed on a non-moving surface
- standing; phone is in the hand or pocket, but the user is not walking or running
- walking; the user is walking
- running; the user is running
- bicycling; the user is bicycling
- vehicles; the user is driving or traveling with a motorized vehicle

It is quite clear that the GPS speed would be a very important feature for this kind of classification. However, reliable speed information is not available all the time, so the activity classifier was implemented with two different modes, depending on the availability of the GPS [30].

In this kind of activity classifier, the classes *table*, *standing*, and *vehicles* may overlap in some cases. Reason for this is that the variance of the accelerometer data norm, which is usually the most dominant feature in the activity classifier, can be similar in some cases. For example, a car stopped in traffic lights or a person standing still can be confused as the class on *table*. Nevertheless, if we also use features from the GPS (GPS speed) and causal information based on, *e.g.*, markov chain modelling of the classification outputs ([31]), the separation of a car stopped in traffic lights and standing can usually be done.

The original data set did not contain enough data for the results with the activity classifier when the GPS speed is available. Thus, data were generated where the standard deviations (std) and means of the GPS speeds for each activity were chosen manually. This made it impossible to perform the decision tree classification but enabled us to calculate the adaptation results also for the activity classifier. The values selected are shown in Table 10. It should be noted that the values are naturally changing depending on the user. However, our adaptation algorithm can scale these values for the individual use.

Table 10. The means and standard deviations of the GPS speeds that were used in the activity classifier.

	run.	**bicyc.**	**walk.**	**stand.**	**table**	**vehic.**
Mean(km/h)	7	15	4	0.5	0.5	60
Std(km/h)	2	5	2	0.5	0.5	30

7.4. Adaptation Results

For the test persons, carrying the phones and answering yes or no to the recognition results does not require a big effort, but the results show that the information obtained is very valuable in both adaptation and evaluation of the classifier performance. As an example, a scatter plot of the two most important transformed features of the data from the class *street/road* is shown in Figure 6. In the upper panel, the original 95 percent confidence ellipse (The ellipse containing 95% of the mass of the Gaussian probability density function when using the estimated covariance calculated from the training data) of the *street/road* training data is depicted using a dashed line, and the ellipse after the adaptation in solid line. The data with *yes*-tag is plotted with small circles, and the data with *no*-tag with small crosses.

The bottom panel of Figure 6 is the same for the features three and four. Figure 7 shows the estimated probability density functions ($p(y)$) obtained using Matlab's *ksdensity* function. The figure shows the *yes*-tag likelihoods (dash-dotted) and *no*-tag likelihoods (solid line) before and after the adaptation. The dashed curve is the Chi squared probability density function which would be the optimal distribution for the *yes*-tag likelihoods. This means that if our model would be correct and our assumption of the Gaussian distributed features would be true, the dash-dotted curve ("yes" distribution) should follow the dashed curve (chi2pdf). One can clearly see that there is not much difference between the *yes*- and *no*-likelihoods before adaptation (upper plot in Figure 7). This means that we cannot assign a confidence measure based on these likelihoods alone. The reason is that the original training data was not sufficient to obtain accurate class parameters or the adaptation data is collected in an environment having different features compared to the training data. After the adaptation is completed, the bottom plot in Figure 7 shows a clear improvement: the *no*-likelihoods are visibly separated from the *yes*-likelihoods.

Figure 6. Class *street/road* features.

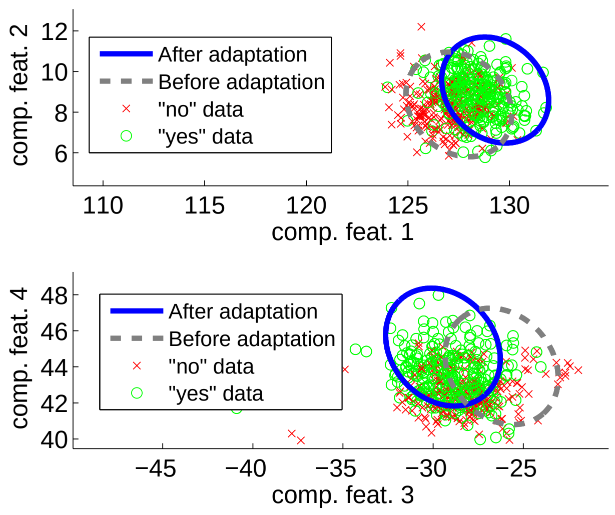
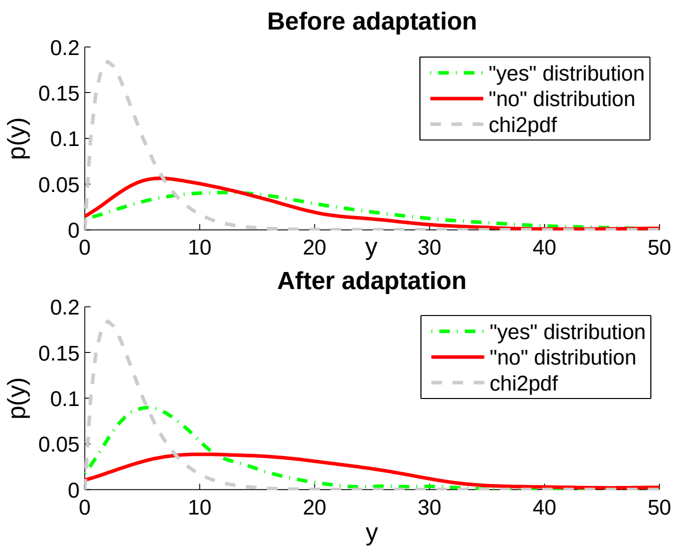


Figure 7. Class *street/road* probability distribution functions.



After the adaptation round is complete, a convenient way to estimate the accuracy of the classifiers is to use the prior probabilities, $P(user\ answers\ yes)$, using Equation (20). To evaluate accuracy of the proposed adaptation method the data from the very first data collection campaign was used (Section 2.1). Ten users with the most data were selected and the classifier was trained leaving one user out of the training data set at the time. Data from this user was then divided to half, other half containing test set for classifier and other half was used to create virtual *yes* and *no* answers using. This enabled us to know also the ground truth of the *no* answers (which was not available in the second data collection campaign presented in Section 2.2). Table 11 presents the classification data of these ten users before the adaptation was applied. Classifier precision and recall are 50% and 67% ,respectively. The results after the adaption are presented in Table 12. In this case precision is 68% and recall 70%. Naturally, the *no* answers were more valuable information for the adaptation process. It was noticed that already couple of *no* answers could improve the class parameters.

Table 11. Confusion matrix of five environment classes *office*, *nature*, *street/road*, *home* and *restaurant/pub/cafe* before adaptation.

true \ decision	Off.	Nat.	Str.	Home	Res.
Off.	76	0	2	11	11
Nat.	0	55	35	5	5
Str.	3	8	78	0	11
Home	3	1	17	65	14
Res.	11	0	30	0	59

Table 12. Confusion matrix of five environment classes *office*, *nature*, *street/road*, *home* and *restaurant/pub/cafe* after adaptation.

true \ decision	Off.	Nat.	Str.	Home	Res.
Off.	81	0	1	6	13
Nat.	0	60	20	5	15
Str.	3	5	73	5	14
Home	3	1	9	74	13
Res.	11	0	25	0	64

7.5. Confidence Measure Results

To show that the estimated probabilities of *yes*-answers actually resemble the true probabilities, a rough visualization is made for the activities in Figure 8 and for the environments in Figure 9. In the figures, all the answers from all users are sorted by the estimated probability, and the results are plotted using a circle if the user answer was *yes* and with a cross in the case of a *no* answer. To see if the prediction of probability is approximately working, we can then zoom into some probability level and see if the distribution between the circle (*yes*)) values agrees with the y-axis number. The lower box of Figure 8 shows an example where the predicted probability level of our model is 0.35 and the actual amount of yes answers in the data is 32%. Respectively, the higher box depicts a situation where our model predicts a proportion of yes answers to be 0.88, whereas the actual proportion of yes labeled samples in the data is 87%. This approximation shows that the estimation is working very well for activities, however, for environments (Figure 9) there are more differences in predicted probability compared to the estimate calculated using yes and no answers.

Figure 8. Predicted *yes*-probability for each sample in the sorted activities data. The small bar plots tell how many yes and no answers there are inside the two small rectangural areas.

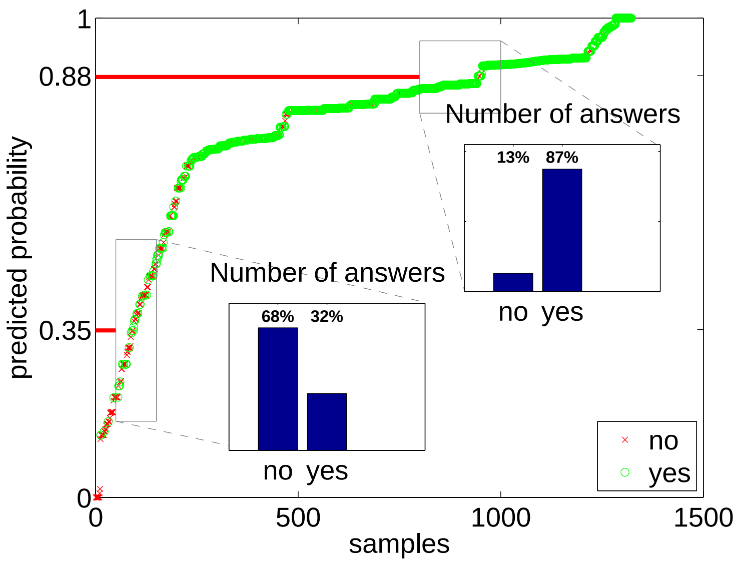
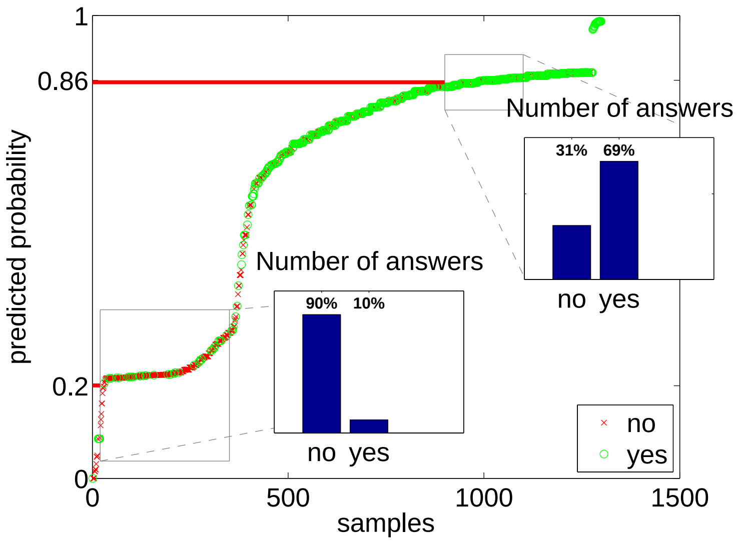


Figure 9. Predicted *yes*-probability for each sample in the sorted environment data. The small bar plots tell how many yes and no answers there are inside the two small rectangural areas.



To analyze the result more in detail, we also show the limits t_l, t_u around the mean

$$P(t_l \leq \theta < t_u) = 0.95, \tag{21}$$

with $\theta \sim Bin(N(yes) + N(no), p_m)$ for each bin to see how the samples would deviate with perfectly known p_m. The limits are shown in Figure 10 for the activities and Figure 11 for the environments, where the data is divided into 10 boxes having the middle points $p_m = 0.05 \ldots 0.95$. The height of each bin is the number of yes answers satisfying $|p_m - P_{yes|\mathbf{L}}| < 0.05$. This result shows that for the activities the estimated confidence measure is consistent. However, for the environments the confidence measure is slightly optimistic.

Figure 10. The number of samples (yes answers) at each bin centered at p_m and 95% limits from Binomial distribution (activities).

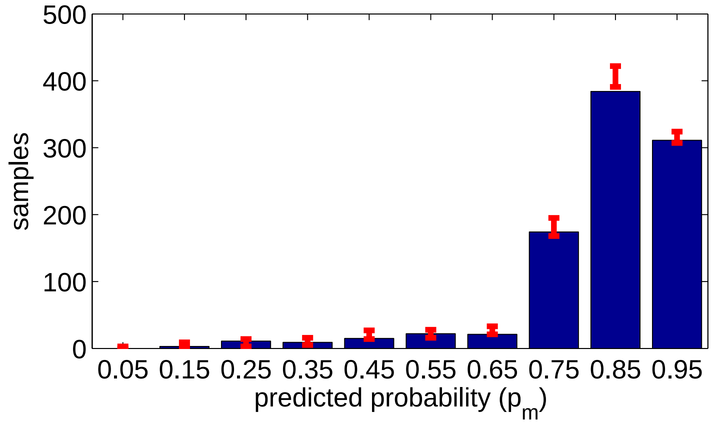
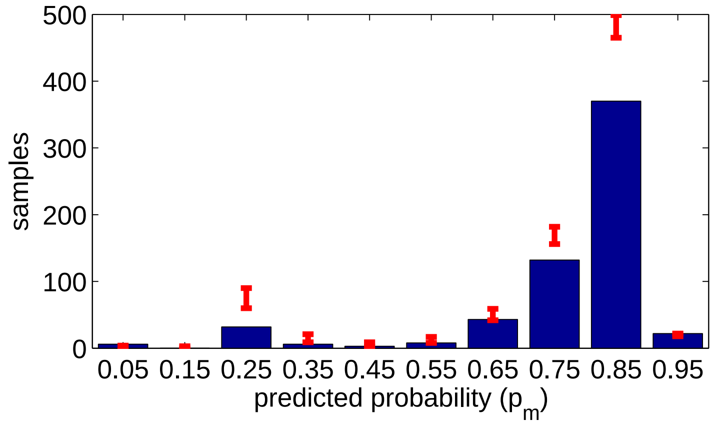


Figure 11. The number of samples (yes answers) at each bin centered at p_m and 95% limits from Binomial distribution (environments).



8. Conclusions

We have presented an activity and environment recognition implementation suitable for modern mobile phones. Two data collection campaigns were organized, the first for the background training data and the second for testing the adaptation algorithms. The initial design goal of having an adaptive system that can provide confidence measures along with the classification results was met by developing a Bayesian approach that utilizes binary user feedback.

The results show that some important environments and activities can be recognized with reasonable accuracy, but individual adaptation is very likely needed for applications requiring context information. In addition, the adaptation is needed for providing a proper confidence measure for the classification result. Furthermore, the paper showed that the confidence measure is consistent with the selected set of activities classes. For the selected environments, the confidence measure was slightly optimistic.

Our implementation enables an individual adaptation in a mobile device. To avoid heavy computations on the mobile device, the required distribution estimates can be computed on the server side. The only necessary information to be transmitted to the server would be the set of likelihood values corresponding to observations with a yes-label and a no-label.

Author Contributions

All authors in the group designed the experiments. Jussi Parviainen and Jussi Collin were the main authors who analyzed the data. Jayaprasad Bojja performed the implementation on the mobile phone. Jussi Parviainen was the main writer of the paper, and all of the other authors also edited the paper.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Altun, K.; Barshan, B.; Tunçel, O. Comparative study on classifying human activities with miniature inertial and magnetic sensors. *Pattern Recognit.* **2010**, *43*, 3605–3620.
2. Gu, T.; Wang, L.; Chen, H.; Tao, X.; Lu, J. Recognizing Multiuser Activities Using Wireless Body Sensor Networks. *IEEE Trans. Mobile Comput.* **2011**, *10*, 1618 –1631.
3. Könönen, V.; Mäntyjärvi, J.; Similä, H.; Pärkkä, J.; Ermes, M. Automatic feature selection for context recognition in mobile devices. *Pervasive Mob. Comput.* **2010**, *6*, 181–197.
4. Reddy, S.; Mun, M.; Burke, J.; Estrin, D.; Hansen, M.; Srivastava, M. Using mobile phones to determine transportation modes. *ACM Trans. Sen. Netw.* **2010**, *6*, doi:10.1145/1689239.1689243.
5. Santos, A.C.; Tarrataca, L.; Cardoso, J.M.P.; Ferreira, D.R.; Diniz, P.C.; Chainho, P. Context Inference for Mobile Applications in the UPCASE Project. In *MobileWireless Middleware, Operating Systems, and Applications*; Springer Berlin Heidelberg: Berlin, Germany, 2009; pp. 352–365.
6. Santos, A.C.; Cardoso, J.A.M.P.; Ferreira, D.R.; Diniz, P.C.; Chaínho, P. Providing user context for mobile and social networking applications. *Pervasive Mob. Comput.* **2010**, *6*, 324–341.
7. Miluzzo, E.; Lane, N.D.; Fodor, K.; Peterson, R.; Lu, H.; Musolesi, M.; Eisenman, S.B.; Zheng, X.; Campbell, A.T. Sensing meets mobile social networks: the design, implementation and evaluation of the CenceMe application. In Proceedings of the 6th ACM conference on Embedded network sensor systems, Raleigh, NC, USA, 4–7 November 2008; ACM: New York, NY, USA, 2008; pp. 337–350.
8. Wu, S.y.; Fan, H.H. Activity-Based Proactive Data Management in Mobile Environments. *IEEE Trans. Mobile Comput.* **2010**, *9*, 390 –404.
9. Lee, Y.; Lee, S.; Kim, B.; Kim, J.; Rhee, Y.; Song, J. Scalable Activity-Travel Pattern Monitoring Framework for Large-scale City Environment. *IEEE Trans. Mobile Comput.* **2011**, *11*, 644–662.

10. Bancroft, J.; Garrett, D.; Lachapelle, G. Activity and Environment Classification Using Foot Mounted Navigation Sensors. In Proceedings of International Conference on Indoor Positioning and Indoor Navigation, Sydney, Australia, 13–15 November 2012.
11. Pei, L.; Guinness, R.; Chen, R.; Liu, J.; Kuusniemi, H.; Chen, Y.; Chen, L.; Kaistinen, J. Human Behavior Cognition Using Smartphone Sensors. *Sensors* **2013**, *13*, 1402–1424.
12. Susi, M.; Renaudin, V.; Lachapelle, G. Motion mode recognition and step detection algorithms for mobile phone users. *Sensors* **2013**, *13*, 1539–1562.
13. Han, M.; Bang, J.H.; Nugent, C.; McClean, S.; Lee, S. A Lightweight Hierarchical Activity Recognition Framework Using Smartphone Sensors. *Sensors* **2014**, *14*, 16181–16195.
14. Lin, T.; ODriscoll, C.; Lachapelle, G. Development of a Context-Aware Vector-Based High-Sensitivity GNSS Software Receiver. In Proceedings of the 2011 International Technical Meeting of the Institute of Navigation, San Diego, CA, USA, 24–26 January 2011; pp. 1043–1055.
15. Davis, S.; Mermelstein, P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. Acoust. Speech Signal Process.* **1980**, *28*, 357–366.
16. Webb, A. *Statistical Pattern Recognition*, 2nd ed.; Wiley: New York, NY, USA, 2002.
17. Kittler, J.; Young, P.C. A new approach to feature selection based on the Karhunen-Loeve expansion. *Pattern Recognit.* **1973**, *5*, 335–352.
18. Rao, C.R.; Toutenburg, H. *Linear Models, Least Squares and Alternatives*; Springer: New York, NY, USA, 1999.
19. Sheil, J.; OMuircheartaigh, I. Algorithm AS 106: The Distribution of Non-Negative Quadratic Forms in Normal Variables. *J. R. Statist. Soc. Ser. C* **1977**, *26*, 92–98.
20. Dietterich, T.G. Ensemble methods in machine learning. In *Multiple Classifier Systems*; Springer: Cagliari, Italy, 2000; pp. 1–15.
21. Neyman, J.; Pearson, E.S. On the problem of the most efficient tests of statistical hypotheses. *R. Soc.* **1933**, *231*, 289–337.
22. Duda, R.O.; Hart, P.E.; Stork, D.G. *Pattern Classification*, 2nd ed.; Wiley: New York, NY, USA, 2001.
23. Lagarias, J.C.; Reeds, J.A.; Wright, M.H.; Wright, P.E. Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions. *SIAM J. Optim.* **1998**, *9*, 112–147.
24. Kuusniemi, H.; Wieser, A.; Lachapelle, G.; Takala, J. User-level reliability monitoring in urban personal satellite-navigation. *IEEE TranS. Aerosp. Electron. Syst.* **2007**, *43*, 1305–1318.
25. Ropokis, G.; Rontogiannis, A.; Mathiopoulos, P. Quadratic forms in normal RVs: Theory and applications to OSTBC over hoyt fading channels. *IEEE Trans. Wirel. Commun.* **2008**, *7*, 5009–5019.
26. Bruckner, D.; van Graas, F.; Skidmore, T. Statistical characterization of composite protection levels for GPS. *GPS Solut.* **2011**, *15*, 263–273.
27. Multivariate Normal Distribution Value for an ellipsoid Matlab algorithm. Available online: http://www.math.wsu.edu/faculty/genz/software/matlab/mvnlps.m (accessed 24 February 2014).
28. Jiang, H. Confidence measures for speech recognition: A survey. *Speech Commun.* **2005**, *45*, 455–470.

29. The class imbalance problem: A systematic study. *Intell. Data Anal.* **2002**, *6*, 429–449.
30. Kantola, J.; Perttunen, M.; Leppanen, T.; Collin, J.; Riekki, J. Context Awareness for GPS-Enabled Phones. In Proceedings of the 2010 International Technical Meeting of The Institute of Navigation, San Diego, CA, USA, 25–27 January 2010; pp. 117–124.
31. Lu, H.; Yang, J.; Liu, Z.; Lane, N.D.; Choudhury, T.; Campbell, A.T. The Jigsaw continuous sensing engine for mobile phone applications. In Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, Zurich, Switzerland, 3–5 November 2010; ACM: New York, NY, USA, 2010; pp. 71–84.