Understanding human sleep behaviour by machine learning

Antonino Crivello[12], Filippo Palumbo[1], Paolo Barsocchi[1], Davide La Rosa[1], Franco Scarselli[2], Monica Bianchini[2]

Abstract Long term sleep quality assessment is essential to diagnose sleep disorders and to continuously monitor the health status. However, traditional polysomnography techniques are not suitable for long–term monitoring, whereas, methods able to continuously monitor the sleep pattern in an unobtrusive way are needed. In this paper, we present a general purpose sleep monitoring system that can be used for the pressure ulcer risk assessment, to monitor bed exits, and to observe the influence of medication on the sleep behaviour. Moreover, we compare several supervised learning algorithms in order to determine the most suitable in this context. Experimental results obtained by comparing the selected supervised algorithms show that we can accurately infer sleep duration, sleep positions, and routines with a completely unobtrusive approach.

1 Introduction

Recently, sleep assessment and evaluation has gained considerable attention and prominence among sleep–related researchers and clinicians [1]. The goal of obtaining sleep data on large–scale studies is a challenging task and has witnessed an increasing interest over the last few years [2]. These studies demonstrate that it is possible to identify sleep disorders by means of a fine–grained sleep history log, including timing and regularity of bed time, onset of sleep, night time awakenings, time of waking up in the morning, day time naps, and day time sleepiness [3].

Sleep disorders can be categorized into two broad groups: primary and secondary sleep disorders. Primary sleep disorders include: Sleep Disordered Breathing

1 - Information Science and Technologies Institute - National Research Council of Italy,
e-mail: {name.surname}@isti.cnr.it
2 - University of Siena - Department of Information Engineering and Mathematics,
e-mail: {name}@diism.unisi.it

(SDB), REM sleep Behaviour Disorder (RBD), Restless Leg Syndrome (RLS), and Periodic Limb Movement in Sleep (PLMS). Secondary sleep disorders are due to diseases with chronic pain and discomfort, frequent micturition during night, dyspnoea and / or medications interfering with sleep [4]. Psychiatric disorders also have a close link with insomnia. It is also known that elderly subjects with persistent insomnia are at a greater risk for the development of depression [5]. Furthermore, changes in the user's life–style due to retirement, bereavement, reduced social interactions, or environmental changes, such as moving into a new house or to a elderly home placement, can result in a change in the sleep patterns [6].

The above mentioned clinician challenges can be faced adopting emerging ICT solutions. In fact, "Humans and ICT interaction", as shown in [7, 8], can lead to important goals, such as to understand how cognitive processes can evolve with artificial devices. Finally, this knowledge can be useful for extending human brain capabilities through such devices and for studying the interaction with the capabilities of any artificial cognitive system.

In this paper, we propose a system able to capture the movement of the patient, as well as the bed posture, in an unobtrusive way. The proposed system is composed by a grid of forty–eight Force Sensing Resistor (FSR) sensor nodes, placed on the slats of the bed. This grid is physically connected to a single–board computer, which is able to send the collected data, using a middleware layer, to a main server. It is worth noting that this solution is able to overcome the weakness of classic actigraphy–based systems, using an actimetry sensor usually embedded in a wrist–watch–like package and worn on the wrist by the user, since it is extremely easier to deploy and it is based on inexpensive technology. With respect to actigraphy–based systems, the proposed system is also able to detect the bed posture, that is crucial to support pressure ulcer prevention (i.e. bedsores). In particular, elderly people are often unable to make the desirable bodily movements and repositioning, that are critical for blood circulation and relieving of prolonged pressure over the body. This critical condition, namely bedsores [9], commonly occurs among elderly persons due to the lack of desirable nursing care and immediate attention. Therefore, a continuous observation of the patients is necessary in order to prevent the above mentioned adverse effects. Position recognition for elderly people can support pressure ulcer prevention in two ways. Firstly, self–movements can be monitored in order to support risk assessment, which may be useful to make prognostications for bedsores. Secondly, it can help the caregiver to decide the care program for the elderly patients since, choosing the right frequency for the posture changes, and assessing the need of the care accurately, decreases the burden of the caregiver in preventing bedsores [10]. This work analyses several machine learning techniques and their potentialities in inferring users' sleep positions, without an a priori model of the user (e.g., vital and physiological parameters).

The rest of the paper is organized as follows. Section 2 presents the state of the art on long–term sleep monitoring systems. In Section 3, the proposed monitoring system is illustrated. Section 4 describes the methodology adopted to recognize the bed postures, while Section 5 presents the experimental results obtained by testing

the proposed solution with different classification algorithms. Finally, in Section 6, we discuss the results and draw some conclusions.

2 Related Work

In 1995, the Standards of Practice Committee of the American Sleep Disorders Association (ASDA) commissioned a task force to evaluate the role of actigraphy in sleep medicine. The term actigraphy refers to methods using wristband–like devices to monitor and collect data generated by movements. Some devices exploit a piezo–electric mechanism to detect movements, along two or three axes, and to digitally count the accumulated movements across pre–designed epoch intervals (e.g. 1 min), storing them in an internal memory.

ASDA's effort on actigraphy led to a review paper on the topic [11] and a set of guidelines [12]. The acknowledge for actigraphy as a valid tool by ASDA was an important landmark in its acceptance by sleep–related researchers and clinicians. The use of actigraphy is continuously rising in sleep research and medicine, as demonstrated by the increasing number of publications over the years [13].

Despite the main strength of actigraphy lies in the ability of monitoring sleep behaviour and inferring sleepwake patterns over long periods of time at home, actigraphy also has several weaknesses. In [14], the authors report that up to 28% of weekly recordings of children and adolescents were insufficient for the sleep analysis. The main reasons for data loss included patient non–compliance to the pre–defined protocol (inability to complete the diary or log and misplacement of the wearable actigraph device), illness, and technical problems. Indeed, detailed patient logs are essential for accurate scoring of records. Showers (with the actigraph off), just before bedtime or after risetime, can be easily confused with sleep activity. Conversely, activity of co–sleeping of bed partners or sleep during car rides may be scored as waking. For these reasons, the log should contain information about bedtimes, risetimes, times when the actigraph is not worn, and times of external motion or unusual events. When the actigraph data are retrieved, patients should be queried about moments when the log and the actigraph records are incoherent. Moreover, children and adolescents are remarkably capable of bending metal parts, dislodging event buttons and otherwise damaging the instruments. Data loss may also occur when curious wearers of any age remove the battery cover to see what's inside. Finally, instruments may lose calibration and fail in many other ways. Unlike laboratory studies, where technical problems and artefacts are recognized quickly and either resolved or thoroughly documented, problems occurring over long periods of home recording often lead to a complete loss of data.

In [9, 15], the use of wearable general purpose sensor technologies to monitor the bed posture of patients is proposed. In [16], an unobtrusive system able to infer the bed posture and the breathing signal is presented. The system is based on an expensive technology which employs a sensor, called Kinotex, that was developed by the Canadian Space Agency for tactile robotic sensing. Finally, in [17], an inexpensive

system based on placing above the mattress a capacity textile sensing technology is described. However, the authors noticed problems on the reproducibility of the experiments, due to the movement of the textile system, which necessitates a new calibration phase each time.

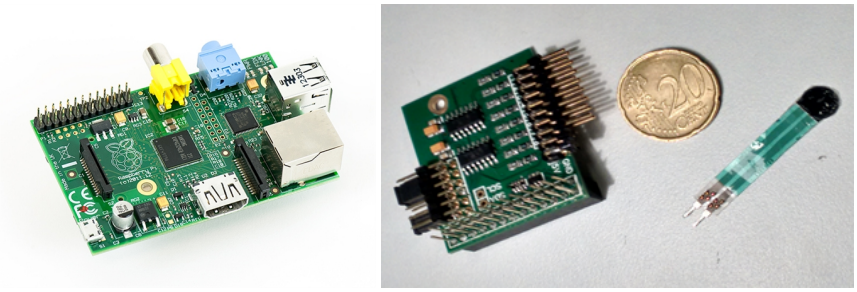
Vice versa, the proposed system is able to merge the inexpensive feature of [16] and the unobtrusive feature of [17], just placing, under the mattress, several force sensor resistors (FSRs), able to report the force pressure generated by the patient over the mattress.

3 The Sleep Monitoring System

In this section, we describe the developed platform in terms of needed hardware and software artefacts. The proposed system has been designed in order to provide an effective solution both from a cost and deployment point of view. It allows to unobtrusively provide data to the application layer and to be easily integrated in different pervasive computing scenarios, exploiting the presence of an open source middleware infrastructure.

3.1 Hardware components

DAVIDE The proposed hardware system is based on the widespread Raspberry Pi (Figure 1.a) single–board computer, equipped with a 700 Mhz ARMv6 processor, 512MB of RAM and several I/O peripherals. The board is running Raspbian OS, a platform–optimized Linux distribution. On the top of the board, additional shields can be mounted through the 26–pin expansion header. The sensors used to collect



(a) (b)

Fig. 1: The Raspberry Pi board (a) and the used ADC shield with a Force Sensing Resistor (b).

the bed pressure distribution are called Force Sensing Resistors (FSRs) and consist of a conductive polymer which changes its resistance proportionally with the force applied on the sensor surface. These sensors have a very low profile (less than 0.5mm), low cost and good shock resistance. In order to acquire and manipulate the weight pressure values, it is necessary to convert the analog resistance, seen as a voltage drop between the pressure sensor and a partition resistor, to a digital format. For this reason, we used several ADC shields (Figure 1.b), mounted on the top of the Raspberry Pi, to convert the raw voltage value coming from the sensors. Each of the ADC shield mounts a pair of Microchip MCP3424 Analog–to–Digital converters. The MCP3424 device features 18–bit, four channels delta–sigma ADC with differential inputs, self calibration of internal offset and gain on each conversion. It also mounts an on–board programmable gain amplifier (1x, 2x, 4x and 8x), to amplify the signal before its conversion. Each shield is therefore able to sample 8 channels (sensors). The I2C bus is used to communicate with the ADCs and their address is set by placing proper jumpers on the shields. A maximum of 4 shields can be stacked on the same Raspberry Pi board, limiting to 32 the maximum number of deployable pressure sensors.

3.2 Software architecture

DAVIDE The proposed hardware and software architecture, composing the data sensing and processing system, aims at providing high flexibility and scalability. From the software point of view, a middleware layer able to dispatch data among generic entities, called services, has been used. This interoperability layer allows the components, which are realized either as hardware devices and software modules, to interoperate seamlessly with each other by using a shared representation and communication model [18].

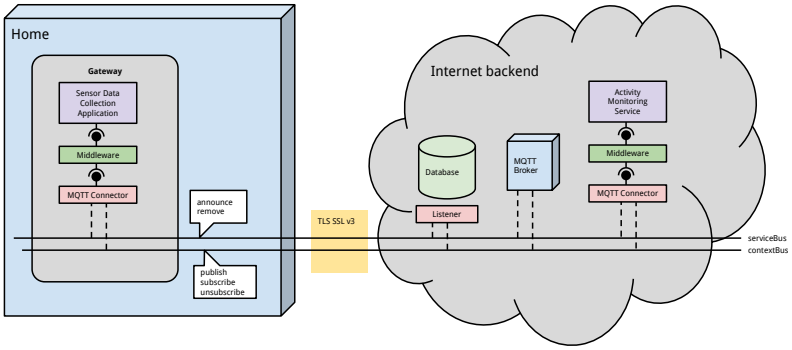


Fig. 2: The middleware architecture.

The concrete middleware architecture (Figure 2) consists of two layers: a core middleware API layer and a communication layer, that includes a publish/subscribe connector. A generic service built upon the middleware can discover both the sensors present in the environment and the other services, together with their functionalities, using methods from the middleware API layer. The underlying layer fulfils these requests exploiting the available connectors. In the communication layer, an MQTT connector is present. By means of these connectors, the middleware realizes, transparently to the services, a publish/subscribe pattern and a method description and invocation mechanism. Two buses form the heart of the proposed middleware: a context bus and a service bus. All communications between applications can happen in a round–about way via one of them, even if physically the applications are located on the same hardware node. Each of the buses handles a specific type of message/request and is realized by different kinds of topics. The aim of the middleware is to provide a publish/subscribe mechanism for accessing the context information about the physical environment. This information will be exposed as different topics: topics for device discovery and description and services that form the service bus; topics for publishing and retrieving data from devices and services that form the context bus. The middleware is in charge of presenting the available sensors and services in the system, implementing the announce mechanism on the service bus.

4 The Proposed Solution

The goal of this work is to provide a sleep monitoring system able to recognize the sleeping stages and to infer the patient's position in the bed. The proposed solution is based on an unobtrusive system, completely transparent to the user. Indeed, we suppose that the patient does not wear any wireless device able to monitor and to communicate data with a medical server.

From a technological point of view, the proposed system is composed by a grid of forty–eight Force Sensing Resistor (FSR) sensor nodes placed on the slats of the bed as shown in Figure 3. This virtual grid does not cover the entire area of the bed but, instead, it is placed at the level of the patient's chest, back, and knees.

From an algorithmic point of view, the proposed solution is based on the observation that, when a movement occurs, the pressure values change in amplitude, whereas, when the patient maintains the same position, the values of the FSRs are almost constant. The algorithm consists in two different tasks: first, the different sleep stages (begin, end, movement, limited muscle activity) are detected; then, the sleep position (supine, prone, right lateral, left lateral), corresponding to each stage, is recognized.

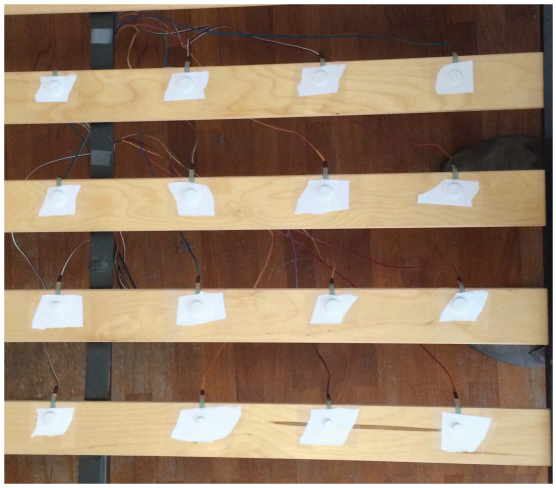


Fig. 3: Experimental setup: A grid of Force Sensing Resistor (FSR) sensor nodes placed on the slats.

4.1 Stage detection

In order to better explain how we defined the stage detection algorithm, Figure 4 shows a typical behaviour of six different FSR time series, together with the ground–truth of the sleep stages, that have been collected by a video camera inside the room. When the user get in the bed, the status of the pressed sensors drastically changes and it stabilizes at a new high pressure value, whereas, when the user changes his/her position in the bed, after a period of time, the pressure value stabilize at the original value. Summing all the pressure values, of every FSR sensor, can lead to false positives and false negatives, as shown in Figure 5.

In order to overcome this issue, a stage detection algorithm must take into account only the variation of the most stressed FSR sensors. Based on Figure 5, only if the red zone changes, the algorithm must detect a movement. Moreover, the algorithm can consider as movements also external events (for example someone who makes the bed), therefore the presence of a detection filter is needed to avoid possible misclassifications.

Relying on these considerations we propose the stage detection algorithm described in the following:

1. For each FSR pressure value p_j, where $j \in N$ (being N the set of the installed FSRs), if $\sum_{j=1}^{N} p_j^{(w)} > \gamma$, where γ is the average pressure, the presence of the patient is ascertained.
2. Only when the patient is detected, for each FSR sensor j, the mean over a W window $P_j^W = \frac{1}{W} \sum_{w=1}^{W} p_j$ is evaluated.

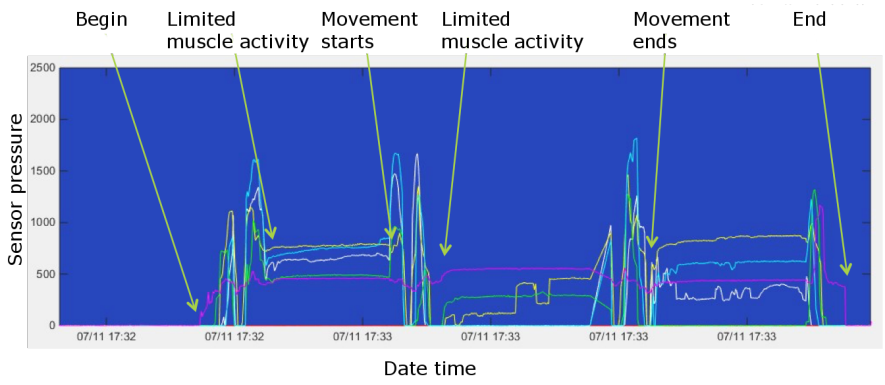


Fig. 4: An example of six FSR time series.

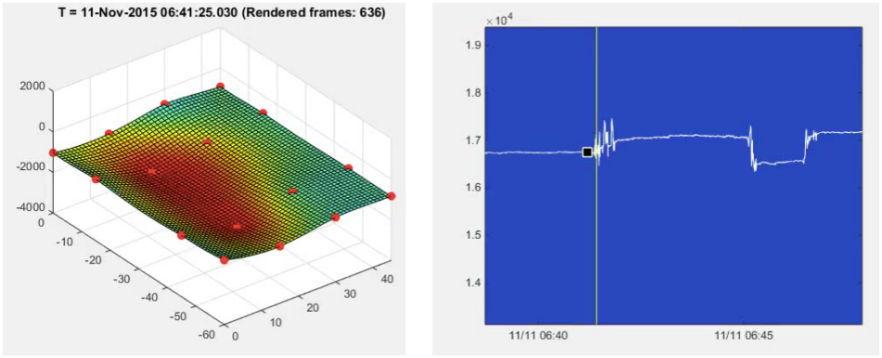


Fig. 5: An example of a false positive considering a subset of sixteen FSRs.

3. The difference between two consecutive pressure values $V_j = abs(P_j^W - P_j^{W-1})$ is calculated to find significant variations.
4. The variation values are sorted and filtered with a linear weighted filter. The obtained output is a set of sorted and weighted values V_j of significant variations in terms of pressure amplitude.
5. If $\sum_{j=1}^{N} V_j \leq \alpha$ (where α is defined as the minimum value for which the pressure variation can be considered as a real movement), the patient is not moving. The α parameter could be defined by leveraging the pressure trace of the day before or by an ad–hoc calibration procedure during the installation of the proposed system.
6. When $\sum_{j=1}^{N} V_j > \alpha$, the movement is detected and the algorithm goes back to step 1.

In particular, γ has been chosen as twice the pressure value of the empty bed, while α was fixed to the 30% percent of $\sum_{j=1}^{N} V_j$ (i.e. a significant variation).

4.2 Sleep position detection

A classification task consists in assigning a class (within a set) to a given object. In the general case, an object is defined by many characteristics (features), providing information about the object class. The information associated to a single characteristic is usually not sufficient to solve the classification problem, so that the correct class can be only inferred by combining all the features. In our case, the objects to be classified are the patient positions, the feature are the FSR signals, and four classes are considered, namely, supine, prone, right lateral, and left lateral.

Machine learning provides several techniques to solve complex classification problems [19]. In this case, a classifier model is trained on a set of examples, called the training set. After training, the classifier is able to combine the characteristics and to generalize the learned behaviour, by correctly assigning a class to unseen objects. The performance of the classifier can be evaluated by applying the trained model on a test set.

In this paper, in order to validate our system, seven different machine learning models have been applied and compared on the task of classifying FSR signals. The goal is to verify that an automatic classification of the patient bed positions is possible and to carry out a preliminary study in order to chose the best algorithm.

The considered models include statistical learning systems (Naive Bayes, Logistic Regression, IbK), ensemble methods (Bagging, HyperPipes) and rule–based learning systems (Decision trees, Decision table), . Table 1 shows the algorithms used in this work, along with a raw and short summary of their strengths and weaknesses. Some basic characteristics are investigated for each method: problem type — the method is able to face classification and/or regression tasks — training speed, prediction speed, automatically feature learning property, and if the classifier is parametric or not.

The automatic feature learning property is based on the assumption that not all the features are equal. Some features can be irrelevant and, for example, lead the algorithm to misclassification. On the other hand, some features should be much important than others. A learner can be able to perform automatically the feature selection task, using a scoring method to rank and select the features and, also, to find correlations between them.

Considering parametric models, we can identify a finite number of parameters. For example, linear models such as linear regressors have a finite number of weight coefficients. Vice versa, in non–parametric models, the complexity of the model grows with the number of training data, because the model has not a fixed structure.

In the following, the used classification algorithms are shortly introduced. In our experiments, machine learning methods used are obtained from the WEKA [20] package.

Decision tables

Decision tables are one of the simplest machine learning techniques [21]. Basically, a decision table consists of a hierarchical table in which each entry in the higher level table gets broken down by the values of a pair of additional features to form

Table 1: Comparison between the used classification methods.

Algorithm	Problem Type	Training speed	Prediction speed	Automatically feature learning	Parametric
Decision Table	Classification	Slow	Fast	No	No
G. Naive Bayes	Classification	Fast	Fast	No	Yes
Simple Logistic	Classification	Fast	Fast	No	Yes
IB*k* Lazy	Class. and Regr.	Fast	Depends on n	No	No
Hyper Pipes	Classification	Slow	Fast	No	No
Bagging	Class. and Regr.	Slow	Fast	Yes	No
Random Forest	Class. and Regr.	Slow	Moderate	Yes	No

another table. Creating a decision table might involve selecting some of the features. The problem is, of course, to decide which features to leave out without affecting the final decision. In our case, we have no a priori information about which FSR must be considered or not. In fact, each sensor, and consequently each feature, can be useful in order to identify a particular user position. Thus, a Decision table approach uses the simplest method of attribute selection: Best First. It searches the space of attributes by greedy hillclimbing, augmented with a backtracking facility.

Naive Bayes
Naive Bayes classifiers are a family of simple probabilistic tools based on applying Bayes' theorem. Naive Bayes classifiers employ the class posterior probabilities given a feature vector [22] as the discriminant function. Therefore, approximations are commonly used, such as using the simplifying assumption that features are independent given the class. This assumption of independence is certainly simplistic. However, it is largely adopted in real scenarios and it works very well in many cases, particularly when datasets are filtered with an a priori data selection, in order to avoid redundant records. The Naive Bayes method might not be the best for our scenario because it does not work when an attribute may not occur in the training set in conjunction with every class value.

Logistic regression
Logistic regression is a well–known technique based on linear regression. The idea of logistic regression is to make linear regression produce probabilities [23]. When using linear regression for binary classification, we calculate a linear function employing regression and then we apply a threshold to decide whether it is a 0 or a 1 response. Similarly, if we want to generalize to more than two classes, we can use a separate regression for each class. We set the output to 1 for instances that belong to that class, and 0 for instances belonging to all the others, thus obtaining a different regression line for each class. Given an unknown test example, the class with

the largest output must be chosen. That would give us n regressions for a problem where there are n different classes.

Coming back to the binary classification case, it is absolutely tempting to imagine that we can interpret the values produced by the linear regressor as probabilities, but this is actually incorrect. Such values are not probabilities, since the values are sometimes negative or greater than one. In order to get better probability estimates, a slightly more sophisticated technique is used. In linear regression, a linear sum is calculated. Instead, in logistic regression, we have the same linear sum, but we embed it in an exponential formula:

$$Pr[1|a_1,a_2,\ldots,a_k] = 1/(1+exp(-w_0-w_1a_1-\ldots-w_ka_k)),$$

where $a_1,\ldots,a_k$ are real input features, and $w_0,\ldots,w_k$ are the model parameters. This is called a "logit" transform. Considering the one–dimensional problem, $Pr[1|a]$ is an S–shaped curve with respect to a, that applies a softer function, i.e. a soft version of a step function that never gets below 0, never gets above 1, and has a smooth transition in between. The parameters $w_0,\ldots,w_k$ are defined by minimizing an ad–hoc error function on the training set. Working with a logit transform, instead of minimizing the squared error, it is better to choose weights to maximize a probabilistic function, called the *log–likelihood function*:

$$\mathscr{L} = \sum_{i=1}^{n}(1-x^{(i)})\log(1-Pr[1|a_1^{(i)},a_2^{(i)},\ldots,a_k^{(i)}])+ \\ x^{(i)}\log(Pr[1|a_1^{(i)},a_2^{(i)},\ldots,a_k^{(i)}]),$$

where $x^{(i)}$ and $a_1^{(i)},\ldots,a_k^{(i)}$ are the actual class and the features of the i–th pattern of the training set, respectively. We can extend this idea also to multiple classes, but in this case, a multi–response regression does not work well, because we need the probabilities to sum to 1 over the various different classes. Such a constraint introduces more computational complexity and needs to be tackled as a joint optimization problem. The result is logistic regression [24], a popular and powerful machine learning method that uses the logit transform to directly predict probabilities.

Lazy learners

Exploring different supervised approaches, it is enticing to apply a completely different point of view, using Lazy learners, also known as prototype methods. The peculiarity of this class of methods is that they are memory–based and no model is required to be fit [25]. Specifically, we consider the k–nearest neighbours(k–NN) algorithm, a classical non–parametric approach where the function is only locally approximated, whereas all the computations are deferred until classification. The principle behind k–NN is to discover the k (we consider $k = 1$) closest training examples in the feature space with respect to the new sample. The training phase of the k–NN algorithm consists in storing the features and the class label of the training objects. In the classification phase, an unlabelled object is classified by assigning the most frequent label among those of the k training samples nearest to it. During

test, new objects are classified based on a voting criteria: the k nearest objects from the training set are considered, and the new object is assigned to the class most common amongst its k nearest neighbours. A variable of this method is represented by the choice of the distance function, used to identify the nearest neighbours. Various distance metrics can be used, the Euclidean distance being the most common. In this work, considering that data were uniformly gathered, we used the most basic settings for the algorithm: Euclidean distance and k set to 1. This means that the class label chosen was the same as the one of the closest training object.

Using k–NN, the target function is approximated locally for each query to the system. These learning systems can simultaneously solve multiple problems, which constitutes, at the same time, their strength and weakness, since for a large input space, they are computationally expensive. These methods usually allow good results when there are not regular separation of the decision boundaries. Our case seems to fit perfectly with this definition.

HyperPipe
A HyperPipe is a fast classifier that is based on simple counts. During the training phase, an *n*–dimensional (parallel–)pipe is constructed for each class [26]. The pipe will contain all the feature values associated with its class. Test instances are classified according to the category that ”most contains the instance”. In this way, for each class, a pipe works as a boundary hyper–solid for each numeric feature. At prediction time, the predicted class is the one for which the greatest number of attribute values of the test instance fall within the corresponding bounds.

Bagging
Bagging is a meta–algorithm, that allows to combine and improve the results obtained by other methods. Actually, having a dataset composed by few classes and many samples for each class, classification algorithms may be affected by classical over–fitting problems. The bagging method is known for its capability of avoiding this problem [27]. Basically, the idea is that of creating a set of different training sets, by sampling them from the whole dataset, and combining the different outputs by averaging them or, in our case, voting. As a meta–algorithm, the Bagging method is based on a classification model for the classification phase. In our case, we chose a fast decision tree learner, namely REPTree. This base learner builds a decision and/ or regression tree using information gain or variance and prunes it using reduced–error pruning (with backfitting).

Considering our data, even taking into account a high number of decision trees, this approach can lead us to a bad overall accuracy. This is due to an intrinsic property of the algorithm that choose, in order to make decision trees, which variable to split in order to minimize the error. In this way, decision trees have a high correlation and a low bias in their own predictions.

Random forest
A natural step over the bagging approach is represented by the random forest (RF) algorithm. It is also based on decision trees and it is considered as an improvement

of the bagging model. Moreover, it allows a decorrelation between trees and, consequently, between their predictions [28]. The idea behind the decision tree methods is quite simple: to make a tree in which each internal node is labelled with an input feature. The arcs from a node representing a particular feature are labelled with each of the possible values of that feature. Each leaf of the tree is labelled with a class or a probability distribution over the classes.

In practice, random forest seems to fit well in our case. In fact, the random forest model is a non–parametric model and, consequently, it does not need any a priori assumption; it is able to face complex input–output relations; it is robust to errors in labels and outliers. This last property is very useful in our case since data are labelled even considering some transition phases from a position to another, in order to make a realistic training set, in comparison with a data acquisition campaign performed throughout the whole night.

5 Results

In order to evaluate the performance of the proposed unobtrusive system, two preliminary experimentations have been carried out. The former experiment aims at validating the stage detection method, and the latter is designed to asses the sleep position classification algorithm.

5.1 Experimental Setup

For the experiments on stage detection, we collected data by the proposed unobtrusive sleep monitoring system using a single bed and a 1.80 m height and 70 kg weight male. In order to have a ground–truth, we installed a video recorder with a night vision in the bedroom (Figure 6) and we synchronized it with the proposed unobtrusive system. We monitored the user for three nights.

The sampling frequency has to be set considering the computing constraints and networking overhead, which are both directly responsible for power consumption within the sensors. In this work, we have chosen a sampling rate of 10 Hz.

For the experiments on sleep position classification, a larger benchmark dataset has been constructed. A small dataset may lead to an ill posed–problem to be approached with machine learning techniques. For this reason, we prepared an ad hoc test site located in our office, with a single bed.

In particular, we carried out two hours of sleep simulation for three different users, with three different mattress thickness, in six different days. Every two days, we changed the mattress and we repeated the experiment. It is worth mentioning that the test users, inside a five minute window, permuted their postures in order to retrieve, for each class, data with small differences. More precisely, the experiment

Fig. 6: The ground–truth: video recorder with a night vision synchronized with the proposed system.

consisted in the repetition of five minute simulated sleeping, for each different class (supine, prone, left, right).

At the end of the data collection campaign, the dataset was composed by 72000 pressure samples for each user, each one labelled with the corresponding sleep position class. This approach allowed us to obtain a well–balanced benchmark. Furthermore, the three users, different by weight and height, allowed us to gather heterogeneous data in order to test the system adaptability to different users in terms of vital and physiological parameters.

5.2 Experimental Results

FILIPPO Figure 7 shows the stage detection algorithm output during an entire night. The presence of the above–described mattress filter is particularly useful to prevent a movement detection when the user was not laid down on the mattress. The figure shows that the movements are correctly detected: the pink square dots represent the time instances when the time series (sum of FSR pressures gathered) is strongly variable. Figure 7 further illustrates the mattress filter relevance. In fact, the proposed algorithm detects, approximately between 8:00 AM and 8:30 AM, that the user got out of the bed, coming back after few minutes. Moreover, at the beginning of the experiment, the user was asked to put under stress the algorithm, with frequent getting in and out of the bed. Nevertheless, all the movements were correctly detected, assessing the strong robustness of the algorithm over the three different test nights.

In order to support bedsore risk assessment, the false positive analysis is essential. In fact, if the system recognizes the immobility of the user while the user has moved, the number of the needed caregiver interventions will be overestimated. On the contrary, if the system recognizes a motion of the patient while he/she was motionless, the number of the caregiver interventions will be underestimated. The proposed algorithm shows no false positives, which is a useful result for a successive real deployment.

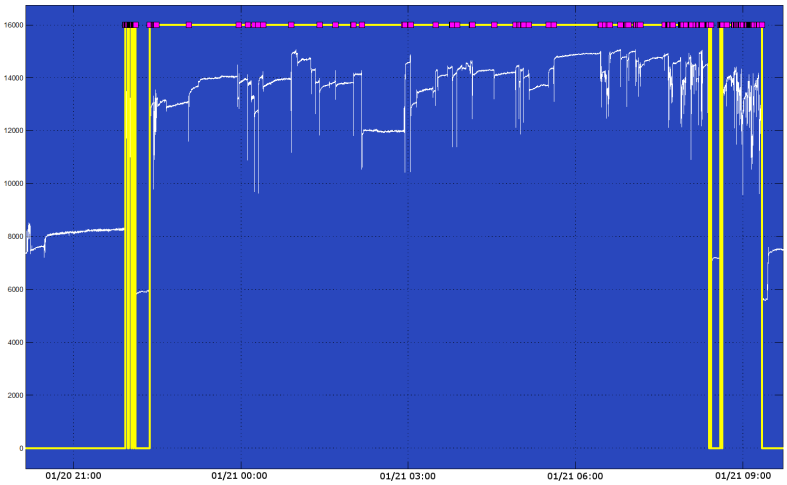


Fig. 7: The stage detection algorithm output during an entire night. No movements are detected when the bold yellow line, which represents the presence of a person on the bed, is zero. Otherwise, movements are precisely detected (represented with pink square dots).

The performance of the sleep position classification algorithms are assessed by the correctly classified instances and by confusion matrices. A confusion matrix is a compact representation describing the results of a classifier: each row of the matrix corresponds to a class and counts the patterns assigned to such class by the classifier (predicted class), while each column represents the number of pattern actually belonging to the corresponding class. A perfect classification method correctly classifies all the patterns, so that in the confusion matrix only the diagonal elements are not null. In general, the larger the diagonal elements, the better the classifier. The experimentation has also been designed to asses whether the classifier can be constructed offline, without adapting its parameters to the user under test.

In order to guarantee an unbiased estimate, training set and test set should ideally be kept separated during the model construction procedure. Successively, the test set can be used to evaluate the obtained model. In our case, the whole dataset was split into three sub–datasets, each of which is related to a single user acquisition for two hours. Then, two experiments were carried out: in the former experiment, a single

user dataset is exploited for training and another single user dataset for testing; in the latter, a single user dataset is adopted for training and the other two (merged together) for the test phase.

Tables 3,4,5,7,6,2 and 8 show the overall average percentage score for each algorithm, considering the two above mentioned configurations for training and test.

Table 2: Performance evaluation of Decision Table method.

	User1 – User2					User1 – User3					User1 – User2–3				
	Position predicted				Acc.	Position predicted				Acc.	Position predicted				Acc.
GT position	S	P	R	L	%	S	P	R	L	%	S	P	R	L	%
supine	3172	371	6	0	44.3	12100	0	1	0	41.6	15272	371	7	0	42.2
prone	2635	858	0	0		7430	354	2826	219		10065	1212	2826	219	
right	1599	0	1955	0		5738	348	5631	0		7337	348	7586	0	
left	1759	726	785	281		9189	942	470	1265		10948	1668	1255	1546	

Table 3: Performance evaluation of Naive Bayes method.

	User1 – User2					User1 – User3					User1 – User2–3				
	Position predicted				Acc.	Position predicted				Acc.	Position predicted				Acc.
GT position	S	P	R	L	%	S	P	R	L	%	S	P	R	L	%
supine	2964	0	2	583	92.0	12070	0	18	13	82.2	15034	0	20	596	84.5
prone	0	2945	540	8		0	6867	1127	2835		0	9812	1667	2843	
right	0	0	3554	0		6	1834	9285	592		6	1834	12839	592	
left	0	0	0	3551		0	1203	630	10033		0	1203	630	13584	

Table 4: Performance evaluation of Logistic Regression method.

	User1 – User2					User1 – User3					User1 – User2–3				
	Position predicted				Acc.	Position predicted				Acc.	Position predicted				Acc.
GT position	S	P	R	L	%	S	P	R	L	%	S	P	R	L	%
supine	2955	277	317	0	95.8	10889	0	1212	0	88.4	13844	277	1529	0	90.2
prone	0	3493	0	0		1719	9110	0	0		1719	12603	0	0	
right	0	0	3554	0		4	8	11113	592		4	8	14667	592	
left	0	0	0	3551		592	602	644	10028		592	602	644	13579	

As expected, different methods led us to different performances in terms of global accuracy. All the three statistical learning (SL) methods perform well on the tree different scenarios. Tables 3,4, and 5 show the performance obtained by the algorithms in the SL group with accuracies between 82.2% (worst case) and 95.9% (best case) with classes, in some cases, which are perfectly predicted. The results are promising and suggest that such methods are able to correctly classify the user's postures.

Table 5: Performance evaluation of IbK method.

	User1 – User2					User1 – User3					User1 – User2–3				
	Position predicted				Acc.	Position predicted				Acc.	Position predicted				Acc.
GT position	S	P	R	L	%	S	P	R	L	%	S	P	R	L	%
supine	2966	396	0	187	95.9	12101	0	0	0	93.3	15067	396	0	187	93.4
prone	0	3493	0	0		1109	9624	3	93		1109	13117	3	93	
right	0	0	3554	0		2	654	10469	592		2	650	14027	592	
left	0	0	1	3550		8	0	634	11224		8	0	635	14774	

Table 6: Performance evaluation of HyperPipes method.

	User1 – User2					User1 – User3					User1 – User2–3				
	Position predicted				Acc.	Position predicted				Acc.	Position predicted				Acc.
GT position	S	P	R	L	%	S	P	R	L	%	S	P	R	L	%
supine	2487	1062	0	0	83.7	10187	481	1433	0	77.3	12674	1543	1433	0	78.8
prone	0	3493	0	0		1	6755	4060	13		1	10248	4060	13	
right	4	0	3550	0		0	4	11120	593		4	4	14670	593	
left	642	265	330	2314		2849	0	1102	7915		3491	265	1432	10229	

Table 7: Performance evaluation of Bagging method.

	User1 – User2					User1 – User3					User1 – User2–3				
	Position predicted				Acc.	Position predicted				Acc.	Position predicted				Acc.
GT position	S	P	R	L	%	S	P	R	L	%	S	P	R	L	%
supine	2604	571	71	303	72.5	10691	783	18	609	68.4	13295	1354	89	912	69.3
prone	0	3493	0	0		3	7616	666	2544		3	11109	666	2544	
right	604	1186	1764	0		5	1919	9201	592		609	3105	10965	592	
left	369	792	0	2390		5619	1329	620	4298		5988	2121	620	6688	

Table 8: Performance evaluation of RF method using 100 trees and no–replacement.

	User1 – User2					User1 – User3					User1 – User2–3				
	Position predicted				Acc.	Position predicted				Acc.	Position predicted				Acc.
GT position	S	P	R	L	%	S	P	R	L	%	S	P	R	L	%
supine	2963	579	0	0	95.7	11960	0	141	0	89.6	14924	579	147	0	91.0
prone	0	3493	0	0		610	10218	1	0		610	13711	1	0	
right	0	0	3554	0		2	8	11115	592		2	8	14669	592	
left	0	0	23	3528		2844	9	635	8378		2844	9	658	11906	

Slightly worse results are obtained when considering methods of the ensemble group. In fact, as shown in Tables 7 and 6, accuracies are between 68.4% and 83.7%. The worst performance was achieved using the Decision Table method, whose accuracy ranges between 41.6 and 44.3 (see Table 2). Actually, such a result is expected, since the main advantage of approaches based on decision tables lies in their simplicity and low computational cost, whereas the classification performance is usually lower with respect to other machine learning methods.

The Random Forest (RF) method, instead, can be considered the best method in our application as shown in Table 8. In terms of global performance, it shows an accuracy between 89.6% and 95.7%.

In order to justify such a result, we need to recall some notions about random forests. The Random forest approach belongs to the class of ensemble methods, based on a combination of tree predictors. Each tree is composed using a sub–sampling of the training set. Combining outputs from each tree the algorithm is able to improve the generalization performance and avoid the over–fitting problem. Eventually, when the number of trees goes to infinity, the Strong Law of Large Numbers always guarantees that the RF accuracy converge to that of the optimal predictor.

Table 9 shows the global accuracy values, obtained by Random Forest, running the algorithm with different number of trees, and seeds fixed to 1.

Table 9: Performance evaluation of Random Forest according to different number of trees.

	User1 – User2–3 – 200 Trees					User1 – User2–3 – 500 Trees					User1 – User2–3 800 Trees				
	Position predicted				Acc.	Position predicted				Acc.	Position predicted				Acc.
GT position	S	P	R	L		S	P	R	L		S	P	R	L	
supine	14696	583	371	0	95.4	14875	583	192	0	95.7	14742	583	325	0	95.4
prone	0	14322	0	0		0	14322	0	0		0	14322	0	0	
right	0	11	14668	592		0	11	14668	592		0	11	14668	592	
left	0	14	1222	14181		0	1	1226	14190		39	0	1230	14148	

The Random Forest algorithm reaches an accuracy of 95.4% for User1–User2–User3 case, better than all the other methods previously shown. Table 10 shows how a different percentage of the original dataset, with no–replacement and fixed number of trees equal to 500, impacts in terms of accuracies. Random Forest, considering the number of features involved in our scenario, needs approximately 100 seconds for the learning phase. Instead, a real–time prediction can be performed. A downsampling strategy can be useful in the case in which the model construction is performed on–board at microcontroller–level or on some other resource–constrained device. Indeed, the overall complexity of RF, in terms of computational speed, depends on several factors, such as the number of trees, features, and instances. RF, trying to find an optimal predictor scanning several levels of possibilities, can require a good deal of computing power and memory available.

6 Conclusions

This paper presents an unobtrusive sleep monitoring system, suitable for long term monitoring, that exploits a sensing technique based on pressure sensors. The system is able to detect movements and sleep patterns.

Table 10: Classification performance of Random Forest with 500 trees, after down-sampling with no–replacement.

Sample Size %	User1–User2	User1–User3	User1–User2–3
20%	95.8	94.9	95.1%
10%	95.8	91.2	92.3%
5%	91.9	89.5	90.1%

The high versatility of the proposed system allows its use in several application scenarios, such as to assess the risk of pressure ulcer, to monitor bed exits or to observe the influence of medication on sleep behaviour.

In this paper, we have compared several supervised learning algorithms, in order to obtain the most suitable solution in this context. Comparative experimental results from seven different approaches show that we can accurately infer sleep duration, user positions, and routines in a completely unobtrusive setting by using the IbK method. Indeed, the IbK method achieves about 95 % of accuracy. A similar performance has been obtained by the Random Forest technique. However, since the Random Forest is a non–parametric model, characterised by an automatic feature learning technique, it should be preferred to detect the user's positions.

Acknowledgements

This work is supported by the INTESA "Servizi ICT integrati per il benessere di soggetti fragili" project, under the APQ MIUR MISE Regione Toscana (DGRT 758 del 16/09/2013) FAR–FAR 2014 Program.

References

1. Luci Wiggs, Paul Montgomery, and Gregory Stores. Actigraphic and parent reports of sleep patterns and sleep disorders in children with subtypes of attention-deficit hyperactivity disorder. *Sleep - New York then Westchester*, 28(11):1437, 2005.
2. Mizue Iwasaki, Sachiko Iwata, Akiko Iemura, Natsumi Yamashita, Yasushi Tomino, Tokie Anme, Zentaro Yamagata, Osuke Iwata, and Toyojiro Matsuishi. Utility of subjective sleep assessment tools for healthy preschool children: A comparative study between sleep logs, questionnaires, and actigraphy. *Journal of epidemiology*, 20(2):143–149, 2010.
3. Adriana Seelye, Nora Mattek, Diane Howieson, Thomas Riley, Katherine Wild, and Jeffrey Kaye. The impact of sleep on neuropsychological performance in cognitively intact older adults using a novel in-home sensor-based sleep assessment approach. *The Clinical neuropsychologist*, 29(1):53–66, 2015.
4. David N. Neubauer. Sleep problems in the elderly. *American Family Physician*, 59(9):2551–2558, 1999.

5. Michael L Perlis, Leisha J Smith, Jeffrey M Lyness, Sara R Matteson, Wil R Pigeon, Carla R Jungquist, and Xin Tu. Insomnia as a risk factor for onset of depression in the elderly. *Behavioral sleep medicine*, 4(2):104–113, 2006.
6. Susan K Roepke and Sonia Ancoli-Israel. Sleep disorders in the elderly. *Indian J. Med. Res.*, (131):302–310, 2010.
7. Péter Baranyi and Adam Csapo. Definition and synergies of cognitive infocommunications. *Acta Polytechnica Hungarica*, 9(1):67–83, 2012.
8. Péter Baranyi, Adam Csapo, and Gyula Sallai. *Cognitive Infocommunications (CogInfoCom)*. Springer, 2015.
9. Paolo Barsocchi. Position recognition to support bedsores prevention. *IEEE Journal of Biomedical and Health Informatics*, 17(1):53–59, 2013.
10. P. Barsocchi, M. Bianchini, A. Crivello, D. L. Rosa, F. Palumbo, and F. Scarselli. An unobtrusive sleep monitoring system for the human sleep behaviour understanding. In *2016 7th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, pages 000091–000096, Oct 2016.
11. Avi Sadeh, Peter J Hauri, Daniel F Kripke, and Peretz Lavie. The role of actigraphy in the evaluation of sleep disorders. *Sleep*, 18(4):288–302, 1995.
12. American Sleep Disorders Association et al. Practice parameters for the use of actigraphy in the clinical assessment of sleep disorders. *Sleep*, 18(4):285–287, 1995.
13. Daniel Waltisberg, Bert Arnrich, and Gerhard Tröster. Sleep quality monitoring with the smart bed. In *Pervasive Health*, pages 211–227. Springer, 2014.
14. C Acebo, A Sadeh, R Seifer, O Tzischinsky, AR Wolfson, A Hafer, and MA Carskadon. Estimating sleep patterns with activity monitoring in children and adolescents: how many nights are necessary for reliable measures? *Sleep*, 22(1):95–103, 1999.
15. Filippo Palumbo, Paolo Barsocchi, Francesco Furfari, and Erina Ferro. AAL middleware infrastructure for green bed activity monitoring. *Journal of Sensors*, 2013.
16. Daphne I Townsend, Megan Holtzman, Rafik Goubran, Monique Frize, and Frank Knoefel. Measurement of torso movement with delay mapping using an unobtrusive pressure-sensor array. *IEEE Transactions on Instrumentation and Measurement*, 60(5):1751–1760, 2011.
17. Silvia Rus, Tobias Grosse-Puppendahl, and Arjan Kuijper. Recognition of bed postures using mutual capacitance sensing. In *European Conference on Ambient Intelligence*, pages 51–66. Springer, 2014.
18. Gianluca Barbon, Michael Margolis, Filippo Palumbo, Franco Raimondi, and Nick Weldin. Taking Arduino to the Internet of Things: the ASIP programming model. *Computer Communications*, 2016.
19. Christopher M Bishop. Pattern recognition. *Machine Learning*, 128:1–58, 2006.
20. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009.
21. Ron Kohavi. The power of decision tables. In *Proceedings of the European Conference on Machine Learning*, pages 174–189. Springer Verlag, 1995.
22. George H John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 338–345. Morgan Kaufmann Publishers Inc., 1995.
23. Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.
24. Joseph F. Hair, Jr., Rolph E. Anderson, Ronald L. Tatham, and William C. Black. *Multivariate Data Analysis (4th Ed.): With Readings*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.
25. David W Aha, Dennis Kibler, and Marc K Albert. Instance-based learning algorithms. *Machine learning*, 6(1):37–66, 1991.
26. Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.
27. Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.
28. Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.