# Chapter 6
# Signal Data Mining from Wearable Systems

**Francois G. Meyer**

## 6.1 Definition of the Subject

### *6.1.1 Introduction*

Sensors from wearable systems can be analyzed in real-time on-site, or can be transmitted to a central hub to be analyzed off-line. In both cases, the goal of the analysis is to extract from the measurements information about the state of the user, and identify anomalous behavior to alert the person. The notion of state depends obviously on the particular application, but in general characterizes a high-level function: the user is awake (as opposed to asleep), the user is falling, the user is going to have a heart attack, etc. Data analysis relies on sophisticated statistical machine learning methods to learn from existing training examples the association between sensors and high-level states [1, 2]. The first stage of the analysis consists in extracting meaningful features, and in removing confounding artifacts. This stage can be achieved using various time-frequency or multiscale methods. The second stage consists in reducing the dimensionality of the data. Indeed, we know that it becomes extremely difficult to learn a function of the data, when the data are in very high dimension. Linear methods to reduce dimensionality include principal component analysis (PCA) and independent component analysis (ICA). Recently, nonlinear methods, such as Laplacian eigenmaps, offer powerful alternatives to traditional linear methods. Finally, one is ready to learn a function of the measurements that describes the state of the person wearing the devices. The sensors only provide very coarse and indirect measurements about the state of the user. For instance, one may be interested in classifying the state of the user into "normal states" or "anomalous states" (the user fell asleep, or is going to have a heart attack). It is therefore necessary to learn the state of the user as a function of the measurements using statistical learning methods.

F.G. Meyer (✉)
University of Colorado at Boulder, Boulder CO 80309, USA
e-mail: fmeyer@colorado.edu

A. Bonfiglio and D. De Rossi (eds.), *Wearable Monitoring Systems*,
DOI 10.1007/978-1-4419-7384-9_6, 

This chapter is organized as follows. The current section provides an overview of the type of data analysis questions associated with wearable systems. Section 6.2 contains a description of the various feature extraction techniques used in wearable devices. The real-time analysis of data requires that an efficient dimension reduction be performed. Methods that can provide a faithful representation of the data with much fewer parameters are described in Sect. 6.3. Finally, statistical machine learning methods that are used to characterize the state of the user are described in Sect. 6.4. A glossary of the terms used in this chapter can be found in Sect. 6.6.

### *6.1.2 Shape of the Data*

Sensor data can be one-dimensional (e.g., accelerometer) or two-dimensional (e.g., video). In this chapter, we will model sensor data as time series of scalars or vectors. Formally, a wearable system can generate a vector $X$ of measurements in $\mathbb{R}^P$ (to simplify the notations and use the same formalism, we can think of an image as a vector properly reorganized). As time evolves, we can index each measurement with a time index, and we denote by $x_{i,\,j}$ the measurement that sensor $j$ generates at times $t_i, i = 1, \ldots, N$. Clearly, the temporal dimension plays a different role than the sensor index $j$ in this dataset, and methods of analysis usually take advantage of this distinction.

### *6.1.3 Scientific Questions*

The data generated by the sensors of a wearable system can be used to answer questions about the state of an individual or the state of a population of individuals.

#### 6.1.3.1 At the Level of the Individual

Wearable computers can provide information about the environment surrounding the user, as well as information about the state (e.g., activity and health monitoring) of the user. In both cases, the information is centered around the individual user and does not involve a network of wearables. The surrounding, or context, can be characterized by the location of the user [3], the amount of noise [4], and the output of several cameras [5]. The detection of the activity of the user [6, 7] is a process that is intrinsically dynamic and requires real-time computations. Finally, a wearable system can be used to monitor the health of the user [2, 8] and diagnose and detect anomalous events [9].

#### 6.1.3.2 At the Level of a Group of Individuals

Wearable systems can also be used to analyze social interaction between different users and monitor social networks [10].

### *6.1.4 Local vs. Remote Analysis*

The analysis of the data collected by the sensors can either be performed locally, using the limited computation and power resources available on the wearable system, or be sent to a hub, where remote computations can be performed.

#### 6.1.4.1 Local or On-Site Analysis

This type of analysis can provide an immediate feedback without requiring any communication to a central hub. An on-site analysis would, for instance, be useful for the continuous health monitoring of individuals living in remote areas without access to wireless networks [11]. Wearable devices can be designed around microcontrollers [12], DSP chips [13, 14, 15], or Field Programmable Gate Array (FPGA) [16, 17]. In all cases, the complexity of the algorithms that can be programmed is limited by the computational and battery power available on the device. Such limitations may prevent the usage of classification methods that require computationally expensive algorithms and massive amounts of training data. Finally, the degree of integration of the technology (handheld vs. wearable) may further limit the amount of computational power.

#### 6.1.4.2 Remote Analysis

The on-site analysis may be supplemented with, or replaced by, a remote analysis. In this scenario, the data harvested by the sensors may be pre-processed on site to eliminate artifacts, and then sent to a central computer, where a more complex analysis is performed. Typically, a wireless connection to a centralized computer allows the wearable device to send the data to a central hub, where the remote analysis is performed. For instance, in the case of medical monitoring, medical data can be sent to a health care center where diagnostic testing and monitoring are performed [18, 19, 2]. This type of processing makes it possible to use machine learning algorithms that are computationally intensive and require large amount of training data [20, 21]. The wireless connection can take advantage of wireless personal area networks standards such as the ZigBee specification: a suite of high level communication protocols using small, low-power digital radios based on the IEEE 802.15.4-2006 standard [20].

## 6.2 Feature Extraction

The very large size of the time series collected by wearable sensors is a basic hurdle to any attempt at analyzing the data. Consequently, the analysis of sensors is usually performed on a smaller set of features extracted from the data [22]. The extraction of

features serves two purposes: first it reduces the dimensionality by replacing the time-series with a more compact representation in a transformed domain (e.g., Fourier, or wavelet); second, it separates the artifact and noise from the signal.

### 6.2.1 Time-Frequency Analysis

Many of the transforms that are used to extract features operate in the frequency domain. This can be justified by a theoretical argument: if the signals are realizations of a wide sense stationary process, then the Fourier transform provides the optimal representation [23]. In practice, many physiological signals are oscillatory and can be decomposed as a sum of a small number of sinusoidal functions. For instance, [11] use Fourier coefficients of motion sensors to study gait.

However, in many applications the signals of interest are not stationary: they contain sudden changes and the local statistical properties of the signals vary as a function of time. The ability to detect sudden changes in the local frequency content is essential (e.g., prediction of seizures [24]). One alternative to the Fourier transform consists in dividing the time series into overlapping segments within which the signal can be expanded using a Fourier transform. This local Fourier analysis requires windowing functions to isolate the different time segments. Formally, we consider a cover of the time axis $\bigcup_{n=-\infty}^{n=+\infty}[a_n, a_{n+1})$, and we write $I_n = [a_{n-1}, a_n)$. The time intervals $I_n$ can be of fixed or adaptive sizes. To localize $f$ around the interval of interest $I_n$, one can construct a projection of $f$, $P_{[a_n,a_{n+1}]}f$ (see Fig. 6.1). The simplest example of $P_{[a_n,a_{n+1}]}f$ is obtained by multiplying $f$ by a smooth window function $\rho_n$, whose support is approximately $I_n$,

$$P_{[a_n,a_{n+1}]} : f \rightarrow \rho_n f. \tag{6.1}$$

We can then compute the Fourier transform of $P_{[a_n,a_{n+1}]}f$ using a fast Fourier transform.

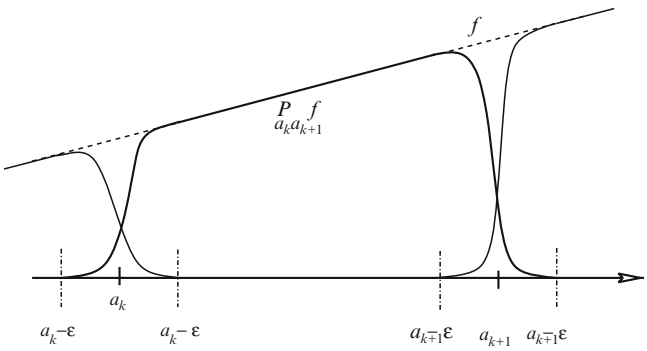


**Fig. 6.1** Localization of a time interval before computing a Fourier transform

Features are then extracted by computing the energy (estimated from the magnitude squared of the Fourier transform) present in specific frequency bands. These frequency bands are determined from prior experiments, or from a priori physiological knowledge. For instance, [25] use Fourier analysis to extract relevant features from raw tremor acceleration data around a small set of frequencies of interest (3 Hz, 4 Hz, 5 Hz, and 6 Hz).

#### 6.2.1.1 Application to Wearable Systems

The authors in [13] use a spectral representation of ECG and respiratory effort signals to estimate sleepiness and distinguish sleep from wake activity. The approach relies on the computation of the energy of the Fourier transform (spectrogram) over time intervals of length 40 s. The computation of the Fourier transform can be performed using DSP chips or FPGA, and can be performed on site [26].

#### 6.2.1.2 Available Software

- The Time-Frequency Toolbox, http://tftb.nongnu.org/, is a collection of MATLAB functions for the analysis of nonstationary signals using time-frequency distributions.
- See also WaveLab in the next section.

### *6.2.2 Multiscale Analysis*

While time-frequency methods can provide good energy compaction, they are not suited for analyzing phenomena that occur at very different scales (from a second to a day). One of the main limitations of Fourier-based algorithms is their inability to exploit the multiscale structure that most natural signals exhibit. As opposed to the short-time Fourier transform, the wavelet transform performs a multiscale, or

**Algorithm 1: Fast wavelet transform**

$$\underline{\text{Initialization}}: \quad s_k^J = x_{k+1} \quad k = 0, \ldots, N-1 \tag{6.2}$$

$$\underline{\text{Iterate}}: \text{for scale} \quad j = J \quad \text{down to} \quad 1: \tag{6.3}$$

$$\text{Low pass filter}: \quad s_k^{j-1} = \sum_n h_{n-2k}\, s_n^j \qquad k = 0, \ldots, 2^{j-1-J}N - 1 \tag{6.4}$$

$$\text{High pass filter}: \quad d_k^{j-1} = \sum_n g_{n-2k}\, s_n^j \qquad k = 0, \ldots, 2^{j-1-J}N - 1 \tag{6.5}$$

**Fig. 6.2** Fast wavelet transform

multiresolution, analysis of the signal. The wavelet transform is an orthonormal transform that provides a very efficient decorrelation of many physical signals [23]. Excellent references in the mathematical theory of wavelets and their application to image compression include [23, 27]. We describe briefly the fast wavelet transform algorithm. We consider the time series $x_n$ $n = 1, \ldots, N$ generated from one specific sensor from time $t_1$ until $t_N$. For simplicity, we assume that $N = 2^J$. The following algorithm was discovered by [23], and is called the *fast wavelet transform*. The wavelet transform of $X$ at scale $J$ is given by the vector of coefficients

$$\left[ s_0^0, d_0^0, d_0^1, d_1^1, \ldots\ldots, d_0^j, \ldots d_{2^{j-J}N-1}^j, \ldots\ldots, d_0^{J-1}, \ldots\ldots\ldots\ldots, d_{2^{-1}N-1}^{J-1} \right]. \quad (6.6)$$

The reconstruction formula is given by the following iteration:

$$s_k^{j+1} = \sum_{n \in \mathbf{Z}} h_{k-2n} s_n^j + \sum_{n \in \mathbf{Z}} g_{k-2n} d_n^j. \quad (6.7)$$

The fast wavelet transform has an overall complexity of $O(N)$ operations. As shown in Fig. 6.3, the algorithm organizes itself into a binary tree, where the shaded nodes of the tree represent the subspaces $W_j$.

One important parameter in the wavelet analysis concerns the choice of the filters $(h_n, g_n)$. In general, biorthogonal wavelet filters with linear phase introduce less distortion than orthonormal wavelets. While longer filters provide better out of band rejection with a sharp frequency cutoff, such filters may become a computational burden if a real-time on-site application is required. Finally, the number of vanishing moments of the wavelet filter controls the number of significant coefficients that one should expect when processing smooth signals. Indeed, polynomials of degree $p - 1$ will have a very sparse representation in a wavelet basis with $p$ vanishing moments: all the $d_k^j$ are equal to zero, except for the coefficients located

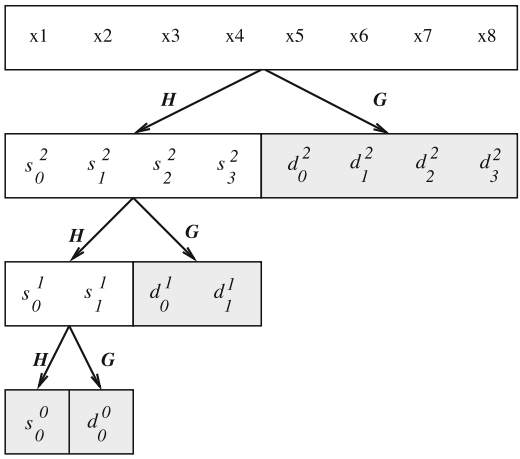


**Fig. 6.3** Pyramidal structure of the fast wavelet transform

at the border of the dyadic subdivision $(k = 0, 1, 2, 4, \cdots, 2^{J-1})$. Unfortunately, wavelet with $p$ vanishing moments have at least $2p$ coefficients, thereby increasing the computational load.

In a manner similar to Fourier analysis, one can decide a priori that certain frequency bands, which are associated with specific wavelet coefficients, should beused for the subsequent analysis of the data. Alternatively, one can remove small coefficients, which are typically generated by the noise, and keep only the largest coefficients that come from the signal. In both cases, the selected wavelet coefficients become the features representing the time series.

#### 6.2.2.1 Application to Wearable Systems

A wavelet transform is used in [28, 29] to extract important features from ECG recordings. Similarly, data from inertial sensors are processed with filter banks in [25] to quantify tremor frequency and energy. Finally, [26] use a wavelet transform to remove the noise and smooth accelerometers time series. Wavelet analysis can be performed on site, since it can be implemented efficiently on a DSP chip or an FPGA.

#### 6.2.2.2 Available Software

- WaveLab at Stanford University, http://www-stat.stanford.edu/~wavelab/, is a very comprehensive library of MATLAB software that implement wavelet transforms, wavelet packets, and other various time-frequency transforms.
- Wavelet toolbox in MATLAB.

## 6.3 Dimensionality Reduction

The extraction of features from the data effectively replaces long time series with shorter feature vectors (e.g., Fourier or wavelet coefficients). Often, this dimension reduction is not significant, and one needs to further reduce the dimension of the feature vectors. Alternatively, there are cases where standard features are not available, and one needs to apply methods to reduce dimensionality directly on the rawsensor measurements.

Methods to reduce dimension exploit the intrinsic correlations that exist in the feature vectors, or in the sensors' measurements. Principal components analysis finds the set of orthogonal components that can best explain the variance in the observations, while independent component analysis can decompose the observations into components that are statistically independent. Finally, Laplacian eigenmaps can provide a faithful parametrization of the sensor data when the data organize themselves in a nonlinear manner.

### 6.3.1 Principal Component Analysis

Instead of using a fixed transform, such as the Fourier or the wavelet transform, one can often construct a sparser representation by adaptively computing an optimal transform. Principal component analysis is one such adaptive transform.

We consider a wearable system equipped with several sensors. Let $x_{i,j}$ be the measurement that sensor $j$ generates at times $t_i$, $i = 1, \ldots, N$. We can organize the sensor measurements as an $N \times p$ matrix

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,p} \\ x_{2,1} & \cdots & x_{2,p} \\ \vdots & & \vdots \\ x_{N,1} & \cdots & x_{N,p} \end{bmatrix}. \tag{6.8}$$

We can think of the measurement generated by the sensors at time $t_i$ as a vector $X_i = [x_{i,1}, \ldots, x_{i,p}]^T$ in $\mathbb{R}^p$, and we can write the matrix $\mathbf{X}$ by stacking horizontally the time series of sensor vectors,

$$\mathbf{X} = \left[\begin{array}{c} X_1^T \\ \hline \vdots \\ \hline X_N^T \end{array}\right]. \tag{6.9}$$

As time evolves, the vectors $X_1, X_2, \ldots X_N$ form a trajectory in $\mathbb{R}^p$ (see Fig. 6.4). If the temporal sampling of the sensor is sufficiently fast, we expect that the discrete trajectory will be smooth, and that the points $X_i$ will be highly correlated. The goal of principal component analysis (PCA) is to compute a low dimensional affine approximation to the set $X_1, X_2 \ldots X_N$. First, the set of measurements is centered around the center of mass

$$\overline{X} = \frac{1}{N} \sum_{i=1}^{N} X_i.$$

Then the optimal subspace of dimension $r$ is computed from the singular value decomposition (SVD) [30] of the centered matrix

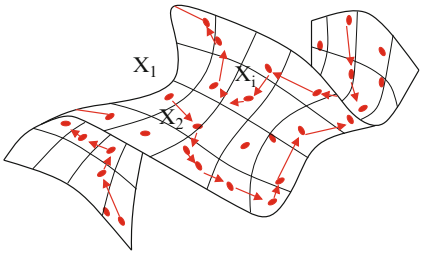


**Fig. 6.4** Trajectory of the sensor vector $X_i$ as a function of time

$$\begin{bmatrix} X_1^T - \overline{X}^T \\ \hline \vdots \\ \hline X_N^T - \overline{X}^T \end{bmatrix} = \sum_{i=1}^{p} \sigma_i U_i V_i^T, \tag{6.10}$$

where the vectors $U_1, \ldots, U_p$ (respectively $V_1, \ldots, V_p$) are mutually orthogonal [30]. In summary, the optimal affine approximation of rank $r$ is given by

$$\overline{X} + \sum_{k=1}^{r} \sigma_k U_k V_k^T. \tag{6.11}$$

This low dimensional approximation is guaranteed to maximize the dispersion (variance) of the projected points on the subspace formed by the vectors $U_1, \ldots, U_r$. Geometrically, the vectors $U_i$ will be aligned with the successive orthogonal directions along which the data changes the most (see Fig. 6.5). From a linear algebra perspective, the affine model (6.7) results in the minimum approximation error – in the mean-squared sense – among all affine models of rank $r$.

**Remark 1.** *The eigenvalues $\sigma_i$ determine the shape of the distribution of sensor measurements. If all the $\sigma_i$ are equal, then all vectors $U_i$* are equivalent. This is a degenerate case where PCA offers no gain. If the $\sigma_i$ are very different, the distribution is shaped as an ellipsoid. PCA provides a gain by aligning the coordinate axes with the main axes of the ellipsoid (see Fig. 6.5).

**Remark 2.** *If we include all the $p$* components, then we have an exact decomposition of the measurements,

$$X = \overline{X} + \sum_{k=1}^{p} \sigma_k U_k V_k^T = \overline{X} + \mathbf{U}\Sigma\mathbf{V}^T, \tag{6.12}$$

where $\mathbf{U} = [U_1, \ldots, U_p]$, $\mathbf{V} = [V_1, \ldots, V_p]$ and $\Sigma = \text{diag}(\sigma_1, \cdots, \sigma_p)$.

**Remark 3.** *PCA can also be used to "whiten" the data; a step which is often required before further analysis [14].*

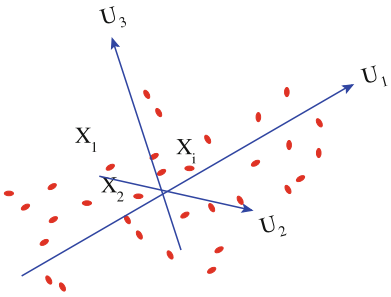


**Fig. 6.5** Principal components $U_1$, $U_2$ and $U_3$

#### 6.3.1.1 Application to Wearable Systems

PCA cannot be implemented on a DSP chip or an FPGA, but can be implemented on a small laptop, as described in [4], where a real-time analysis of the sensors using a time-varying PCA is computed. A clustering algorithm combined with self-organizing maps is also used to identify the main sensor clusters. The clusters are then updated in real-time. In [21], the authors combine feature extraction and PCA to analyze ECG recordings.

#### 6.3.1.2 Available Software

The main ingredient of the PCA algorithm is the SVD decomposition of the matrix $\mathbf{X}$. There exist several implementations of SVD in Fortran, and in MATLAB.

### *6.3.2 Independent Component Analysis*

Independent component analysis (ICA) seeks to decompose the data into a linear combination of statistically independent components [31]. The method assumes the following mixing model, where the vector of measurements $X$ generated by the sensors at any given time can be decomposed in terms of $p$ independent scalar "sources"

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,p} \\ \vdots & & \vdots \\ a_{p,1} & \cdots & a_{p,p} \end{bmatrix} \begin{bmatrix} s_1 \\ \vdots \\ s_p \end{bmatrix}, \tag{6.13}$$

or equivalently in matrix form

$$X = \mathbf{A}S. \tag{6.14}$$

Because the variance of the sources $s_1$, .., $s_p$ cannot be estimated, we can assume that the sources are decorrelated (their covariance matrix is the identity) and the matrix $\mathbf{A}$ is orthogonal. A numerical solution to the estimation of the sources and the mixing matrix $\mathbf{A}$ can be obtained by maximizing the departure from Gaussianity of the vector $S = \mathbf{A}^{-1}X = \mathbf{A}^T X$. This can be achieved by maximizing the negentropy of each coordinate $s_i$ of $S$ [32].

It has been noted [31] that in practice, if the observations are noisy, then it becomes impossible to separate the components from the noise. In fact, if the analysis is performed on real data, then the components are not even approximately independent [31]. Finally, a common problem associated with the usage of ICA is the interpretation of the components. The interpretation usually relies on post hoc heuristics, such as visual inspection of the source time-series.

#### 6.3.2.1 Application to Wearable Systems

ICA suffers from the same computational load as PCA, and cannot be easily implemented on small portable devices at the moment. Motion artifacts were identified in pulse oximeter signals using ICA [33]. But see [34] for a discussion of the validity of the assumption of independence of the motion artifact signals and the signal generated by arterial volume variations.

#### 6.3.2.2 Available Software

- The implementation of Bell and Sejnowski can be found at http://www.cnl.salk.edu/~tony/ica.html.
- A fast ICA MATLAB package is available from the Laboratory of Computer and Information Science (CIS) at the Helsinki University of Technology, http://www.cis.hut.fi/projects/ica/fastica/

### *6.3.3 Laplacian Eigenmaps*

The methods of reduction of dimensionality described in the previous sections are linear: each vector of sensor data $X_i$ is projected onto a set of components $U_k$. The resulting coefficients serve as the new coordinates in the low dimensional representation. However, in the presence of nonlinearity in the organization of the $X_i$ in $\mathbb{R}^p$ (see Fig. 6.4), a linear mapping may distort local distances. These distortions will make the analysis of the dataset more difficult. We describe in this section a method to construct a nonlinear map $\Psi$ to represent the dataset $\mathbf{X}$ in low dimensions. Because the map $\Psi$ is able to preserve the local coupling between sensors vectors, low dimensional coherent structures can easily be detected with a clustering or classification algorithm.

We describe a low dimensional embedding of the set of sensor measurements $X_1, \ldots, X_N$ into $\mathbb{R}^m$, where $m \ll p$. The embedding is constructed with the eigenfunctions of the graph Laplacian [35]. First, we represent the measurements by a graph that is constructed as follows. Each sensor vector $X_i$ becomes the node (or vertex) $i$ of the graph. Edges between vertices quantify the similarity of sensor vectors. Each node $i$ is connected to its $n_n$ nearest neighbors according to the Euclidean distance $\|X_i - X_j\|$. Finally, a weight $W_{i,j}$ on the edge $\{i, j\}$ is defined as follows,

$$W_{i,j} = \begin{cases} e^{-\|X_i - X_j\|^2/\sigma^2}, & \text{if } i \text{ is connected to } j, \\ 0 & \text{otherwise.} \end{cases} \tag{6.15}$$

The weighted graph $G$ is fully characterized by the $N^2 \times N^2$ weight matrix $\mathbf{W}$ with entries $W_{i,j}$. Let $\mathbf{D}$ be the diagonal degree matrix with entries $d_i = \sum_j W_{i,j}$.

The map $\Psi$ is designed to preserve short-range (local) distances, as measured by $W_{i,j}$. The map is constructed one coordinate at a time. Each coordinate function $\psi_k$ is the solution to the following minimization problem

$$\min_{\psi_k} \frac{\sum_{X_i \sim X_j} W_{i,j} \big(\psi_k(X_i) - \psi_k(X_j)\big)^2}{\sum_i D_{i,i} \Psi_k^2(X_i)}, \tag{6.16}$$

where $\psi_k$ is orthogonal to the previous functions $\{\psi_0, \psi_1, \cdots, \Psi_{k-1}\}$,

$$\langle \psi_k, \psi_j \rangle = \sum_{i=1}^{N} D_{i,i} \Psi_k(X_i) \Psi_j(X_i) = 0 \qquad (j = 1, \ldots, k-1). \tag{6.17}$$

The numerator of the Rayleigh ratio (6.12) is a weighted sum of the gradient of $\psi_k$ measured along the edges $\{i, j\}$ of the graph; it quantifies the average distortion introduced by the map $\Psi_k$. The denominator provides a natural normalization. The constraint of orthogonality to the previous coordinate functions (6.17) guarantees that guarantees that the coordinate $\psi_k$ describes the dataset with a finer resolution: $\Psi_k$ oscillates faster on the dataset than the previous $\Psi_j$ if $\langle \Psi_k, \Psi_j \rangle = 0$. Intuitively, $\Psi_k$ plays the role of an additional digit that describes the location of $X_i$ with more precision. It turnsout [35] that the solution of (6.12 and 6.13) is the solution to the generalized eigenvalue problem,

$$(\mathbf{D} - \mathbf{W})\psi_k = \lambda_k \mathbf{D} \psi_k, \qquad k = 0, \ldots \tag{6.18}$$

The first eigenvector $\psi_0$, associated with $\lambda_0 = 0$, is constant, $\Psi_0(X_i) = 1, i = 1, \ldots, N$; it is therefore not used. Finally, the new parametrization $\Psi$ is defined by

$$X_i \mapsto \Psi(X_i) = [\, \psi_1(X_i) \;\; \psi_2(X_i) \ldots \psi_m(X_i) \,]^T. \tag{6.19}$$

The idea of parametrizing a manifold using the eigenfunction of the Laplacian was first proposed in [36]. Recently, the same idea has been revisited in the machine-learning literature [37, 38]. The construction of the parametrization is summarized in Fig. 6.6. Unlike PCA, which yields a set of vectors on which to project each $X_i$, this nonlinear parametrization constructs the new coordinates of $X_i$ $X_i$ by concatenating the values of the $\psi_k, k = 1, \cdots, mk = 1, \cdots, m$ evaluated at $X_i$, as defined in (6.15). The embedding obtained with the Laplacian eigenmaps is in fact very similar to a parametrization of the dataset with a kernelized version of PCA, known as kernel-PCA [39].

#### 6.3.3.1 Application to Wearable Systems

The authors in [40] embed multidimensional sensor data using Laplacian eigenmaps and cluster the dataset using the new coordinates. The analysis is not performed locally, since the Laplacian eigenmaps require computationally intensive algorithms: nearest neighbor search and eigenvalue problems.

**Algorithm 2: Laplacian eigenmaps**

**Input**:

set of $N$ sensor vectors, $X_1, \ldots, X_N$,
$\sigma$: width of the kernel for the graph; $n_n$: number of nearest neighbors of each $X_i$
$m$: dimension of the embedding

**Algorithm:**

1. construct the graph defined by the $n_n$ nearest neighbors of each $X_i$
2. compute $\mathbf{W}$, $\mathbf{D}$
3. compute the $m$ eigenvectors $\psi_1, \ldots, \psi_m$ of $\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$

**Output:** $m$ coordinates $\Psi(X_i) = \left[\psi_1(X_i)\ \psi_2(X_i)\ \ldots\ \psi_m(X_i)\right]^T$.

**Fig. 6.6** Construction of the embedding

#### 6.3.3.2 Available Software

- The original MATLAB code of Beilkin is available here:http://manifold.cs.uchicago.edu/
- A suite of classification algorithms based on Laplacian eigenmaps is available here: http://manifold.cs.uchicago.edu/manifold_regularization/software.html
- A method related to Laplacian eigenmpas and known as diffusion geometry is available here: http://www.math.duke.edu/~mauro/code.html
- Finally, the Statistical Learning Toolbox of Dahua Lin available here:http://www.mathworks.com/matlabcentral/fileexchange/12333 contains code for the Laplacian eigenmaps.

## 6.4 Classification, Learning of States, and Detection of Anomalies

We are now concerned with the final stage of the analysis: the extraction of high-level information from the wearable system. This abstract information can be an alarm in the case of a natural catastrophe [41], or a diagnostic for a subject with a risk of heart disease [24]. We assume that each sensor vector is represented by a $p$-dimensional vector $X \in \mathbb{R}^p$. This vector may be composed of the raw measurements of the sensors, or may be a vector of features (see Sect. 6.2), or the outcome of a dimensionality reduction algorithm (see Sect. 6.3). At each time $t_i$, we have access to the sensor vector $X_i$ (the subscript $i$ is a time index). The question becomes: what is the state $y_i$ that characterizes the user at time $t_i$ given the sensor vector $X_i$? Depending on the application, the state $y_i$ could, for instance, encode the presence of an alarm, or the likelihood of a heart attack. There are two broad types of approaches to answer this question.

The first approach assumes that there exists a large collection of training data composed of sensors values $X_1, \ldots, X_l$ that have been carefully labeled with the

corresponding state $y_1, \ldots, y_l$ of the user. The labeling is a very time-consuming process that needs to be performed off-line by the user himself/herself, or by an expert (e.g., in the case of biomedical data). Machine-learning algorithms can use the training data to learn the association between the state $y_i$ and the sensor measurement $X_i$. Support vector machines is an important example of supervised classification methods; it is discussed in Sect. 6.4.2. This supervised approach may require significant amounts of training data to achieve good performances.

An alternative approach consists in "letting the data speak for themselves" using clustering methods that identify similarities in the sensor vectors, and group the measurements into coherent clusters. Such methods are called unsupervised and do not require any training data. While they may not provide an answer to our question, namely, "what is the state $y_i$?", unsupervised methods can rapidly organize the data into coherent states that can then be further analyzed. We described in the next section unsupervised methods.

## *6.4.1 Unsupervised Methods*

### 6.4.1.1 K-Means Clustering

The K-means clustering algorithm is a method to divide a set of vectors into homogeneous groups, within which vectors are similar to one another. The goal is to find the optimal number of clusters, $K$, the optimal cluster centroids, $C_1, \ldots, C_K$, and the optimal assignments of each vector $X_i$ to a cluster $k$ to minimize the total within cluster scatter [30]

$$\sum_{k=1}^{K} N_k \sum_{X_i \in \text{cluster k}} \|X_i - C_k\|^2, \tag{6.20}$$

where $N_k$ is the total number of vectors assigned to cluster $k$. Indeed, the term

$$\sum_{X_i \in \text{cluster k}} \|X_i - C_k\|^2,$$

quantifies the scatter of the vectors in cluster $k$ around the centroid $C_k$. Therefore, (6.20) quantifies the total amount of scatter within all clusters. Given a specified number of clusters, $K$, algorithm 3 (see Fig. 6.7) iteratively partitions the data, and computes the centroids $C_1, \ldots, C_K$ of all the clusters (see Fig. 6.8). The output of the algorithm depends on $K$, which can be further optimized [30], and depends also on the initial assignment of vectors to clusters. While the algorithm will eventually converge, there is no guarantee that it reaches the global minimum of (6.16). One should therefore repeat the algorithm of Fig. 6.7 with several different initial conditions, and retain the solution that minimizes (6.20).

**Algorithm 3: K-means clustering**

**Input**:

- $X_1, \ldots, X_N$
- $K$: number of clusters

**Algorithm:**

Repeat
1. assign each vector $X_i$ to the cluster $k$ with the closest centroid $C_k$.
2. recompute all the cluster centroids $C_1, \ldots, C_K$.

Until convergence

**Output:** The cluster label for each $X_i$

**Fig. 6.7** K-means clustering

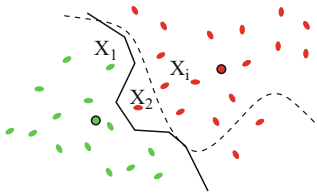


**Fig. 6.8** K-means clustering of the dataset, with $K = 2$. The cluster centroids are *circled* in *black*. The true boundary is shown as a *dashed lined*

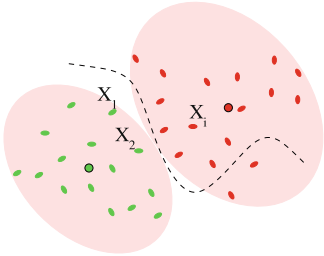


**Fig. 6.9** Mixture of Gaussian model fitted to the dataset, with $K = 2$. The mean of each Gaussian distribution is *circled* in *black*. The true boundary is shown as a *dashed lined*

The assignment of a sensor vector to a cluster results in a hard decision. A soft version of the same idea is provided by the mixture of Gaussian densities model discussed in the next section (Fig. 6.9).

### 6.4.1.2 Mixture of Gaussian Densities

The Gaussian mixture model is a generative probabilistic model that assumes that the joint distribution of the vector of sensor measurements $X$ is a finite mixture of multivariate Gaussian densities,

$$P(X) = \sum_{k=1}^{K} \pi_k \phi(X, \mu_k, \Sigma_k). \tag{6.21}$$

The mixing parameters $\pi_k$ are positive weights that add up to 1. The density $\phi$ is the $p$-multivariate normal density function. The maximum likelihood estimates $\hat{\mu}_k$, $\hat{\Sigma}_k$ and $\hat{\pi}_k$ of the mixture parameters can be computed from the measurements using the expectation minimization (EM) algorithm [42]. The estimation of the number of components $K$ (which plays the same role as the number of clusters) is often difficult [42] but can be addressed using model selection criteria [43]. We can use the estimates $\hat{\mu}_k$, $\hat{\Sigma}_k$, and $\hat{\pi}_k$ to compute the posterior probability given by

$$P(k|X) = \frac{\hat{\pi}_k \phi(X, \hat{\mu}_k, \hat{\Sigma}_k)}{\sum_{l=1}^{K} \hat{\pi}_l \phi(X, \hat{\mu}_l, \hat{\Sigma}_l),} \tag{6.22}$$

which provides an estimate of the probability that measurement $X$ be generated by component $k$.

#### 6.4.1.3 Application to Wearable Systems

Mixture of Gaussians have been used in [44] to segment sensor time series into intervals associated with distinct activities. We note that clustering algorithms can be implemented on a personal digital assistant (PDA) [45], where a clustering algorithm learns to identify the context associated with the usage of the PDA.

#### 6.4.1.4 Available Software

- The open source clustering software contains clustering routines such as k-means and k-medians, and is available at: http://bonsai.ims.u-tokyo.ac.jp/~mdehoon/software/cluster/
- PRTools is a Matlab based toolbox for pattern recognition, and is freely available at http://www.prtools.org/

### *6.4.2 Support Vector Machine*

Among all classification techniques that have been used to analyze sensor data from wearable systems, support vector machines (SVMs) is one of the most popular methods [46, 47]. SVM can construct a nonlinear separating boundary between two classes by implicitly mapping the features to a high-dimensional space, and performing a linear classification in that space. Our discussion of SVM follows [30].

We consider the problem of classifying sensor vectors into two classes defined by the labels $y = -1$ for class 0 and $y = 1$ for class 1. Extension to more than two classes can be easily obtained by using a "one-versus-all strategy," where each class is compared to the other remaining classes, and $X$ is assigned to the class that is most often selected by the different classifications.

#### 6.4.2.1 Support Vector Classifier

Given a training set composed of sensor data $X_1, \ldots, X_l$ and the associated class labels, $y_1, \ldots, y_l$, our goal is to construct a classifier $f(X)$ that assigns a label $-1$ if $X$ belongs to class 0, and 1 if $X$ belongs to class 1. We assume that the training samples are linearly separable: one can find a hyperplane that divides the training dataset according to the class membership. The classifier is defined as follows,

$$f(X) = \begin{cases} -1 & \text{if } \langle W, X\rangle + b < 0 \\ 1 & \text{if } \langle W, X\rangle + b > 0. \end{cases} \tag{6.23}$$

where $W \in \mathbb{R}^p$, $b \in \mathbb{R}$ and $\langle W, X\rangle = \sum_{i=1}^{p} x_i w_i$ is the inner product between the vectors $X$ and $W$. The hyperplane (defined by $W$ and $b$) that optimally separates the two classes can be found by maximizing the margin between the two classes for the training data (see Fig. 6.10). Specifically, we choose $W$ and $b$ such that

$$\begin{cases} \frac{2}{\|W\|}, \text{ the margin width, is maximized, and} \\ \text{for all the vectors } X_i \text{ in the training data we have, } y_i(\langle W, X_i\rangle + b) \geq 1. \end{cases}$$

This quadratic programming optimization problem can be solved using Lagrange multipliers and results in the following classifier

$$f(X) = sign\left(\sum_i \alpha_i y_i \langle X_i, X\rangle + b\right), \tag{6.24}$$

where

$$W = \sum_i \alpha_i y_i X_i, \tag{6.25}$$

and the Lagrange multipliers $\alpha_i$ are nonzero only for those vectors that lie at the boundary of the margin (see Fig. 6.10). Such training vectors are called the "support vectors". As shown in Fig. 6.10, the width of the margin is constrained by the support vectors. The support vectors all satisfy

$$y_i(\langle W, X_i\rangle + b) = 1. \tag{6.26}$$

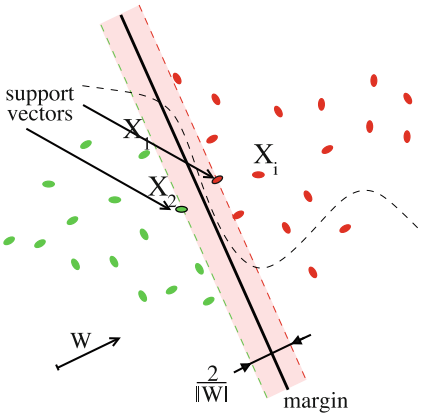


**Fig. 6.10** Maximum margin linear classifier

Finally, the offset $b$ can be computed from any such support vector $X_i$,

$$b = y_i - \langle W, X_i \rangle. \tag{6.27}$$

If the two classes cannot be linearly separated, it is still possible to apply the same classification method with the introduction of additional "slack variables". These variables allow for some training vectors to be on the wrong side of the separating hyperplane [30].

#### 6.4.2.2 Support Vector Machines

The linear classifier can be extended to a nonlinear classifier, where the separating boundary between the two classes is no longer a hyperplane, but can be a surface of arbitrary geometry. Rather than defining the surface in the original domain $\mathbb{R}^p$, the vectors are implicitly mapped to a higher dimensional space (see Fig. 6.11), where distances and inner products are measured using a kernel $\mathbf{K}$, different from the usual inner product [30]. The classifier becomes

$$f(X) = sign\left(\sum_i \alpha_i y_i \mathbf{K}(X_i, X) + b\right). \tag{6.28}$$

Popular choices for $\mathbf{K}$ include the polynomial kernel $\mathbf{K}(X_i, X_j) = \left(\langle X_i, X_j \rangle + 1\right)^d$ and the Gaussian kernel $\mathbf{K}(X_i, X_j) = \exp(-\|X_i - X_j\|^2/\sigma^2)$.

#### 6.4.2.3 Application to Wearable Systems

Dinh et al. [20] use SVM to classify various sensors measurements to detect near-fall events in older adults. In [46] the severity of tremor in patients with Parkinson

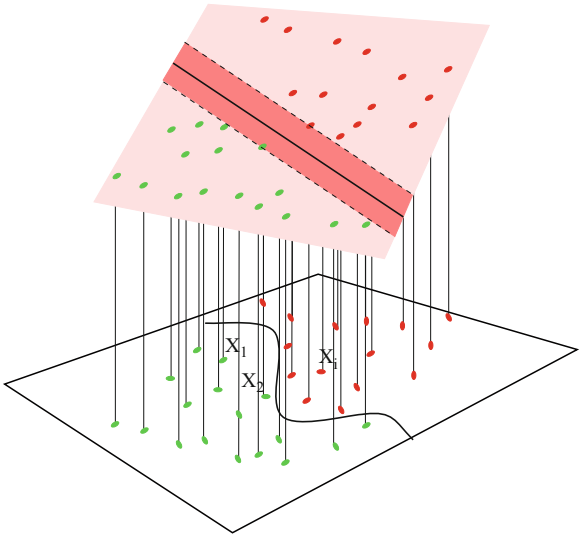


**Fig. 6.11** Classification with support vector machines: the original dataset is lifted up into a high-dimensional space, where a maximal margin linear classifier can separate the two classes

disease is quantified using a multiclass (one-versus-all) classifier based on SVM. The sensors provide accelerometer data. The computational complexity of the SVM algorithm requires that the training and the classification be performed remotely on a central computer.

#### 6.4.2.4 Available Software

Many software packages provide implementations of the SVM. Some of the main public domain packages are listed below.

- SVM light: http://svmlight.joachims.org
- LIB SVM: http://www.csie.ntu.edu.tw/~cjlin/libsvm
- Additional implementations can be found at: http://www.support-vector-machines.org/SVM_soft.html

### *6.4.3 Semi-Supervised*

An alternative to supervised learning and unsupervised learning is "semi-supervised" learning [48]. Semi-supervised can take advantage of unlabeled sensor vectors to learn the organization of the sensor data in $\mathbb{R}^p$. Most semi-supervised learning algorithm assume that the data organize themselves smoothly, and that the geometry of the data will help construct the classifier. In the case of sensors from a wearable system, it is

possible to assume that the data from a sensor lie close to a low dimensional manifold (see Sect. 6.3.3). In this case, it becomes possible to use partially labeled data to discover the geometry of the manifold, while at the same time constructing the classifier. In the context of wearable sensors, this approach is very appealing since it does not require to label a large amount of data.

#### 6.4.3.1 Application to Wearable Systems

Ali et al. [49] combine a PCA decomposition with a semi-supervised learning approach to construct a classifier that can recognize activities. Similarly, Mahdaviani and Choudhury [50] and Stikic and Schiele [7] propose an activity recognition method that only requires very few labeled data. As with SVM, the method can only be applied remotely on a computer with enough computational power.

#### 6.4.3.2 Available Software

- The group of Partha Niyogi at the University of Chicago has developed some MATLAB software to perform semi-supervised classification using manifold regularization. The software is available at: http://manifold.cs.uchicago.edu/manifold_regularization/manifold.html

## 6.5 Conclusion and Future Directions

We have reviewed in this chapter some of the dimension reduction and statistical machine-learning methods to mine and extract information from wearable systems. Many of these techniques have been borrowed from the existing statistical and machine-learning literature. In comparison with general data analysis problems, the on-site, or local analysis of wearable data has a number of well-defined constraints: limited computational power and limited bandwidth. These constraints require efficient and fast algorithms for on-line analysis or off-line transmission to a central hub. As the presence of wearables becomes ubiquitous, and as the sensors become more integrated, we expect that there will be a need for more efficient data analysis algorithms. Increasing the speed of current algorithms is clearly not the answer. Rather, we expect to see that completely new ideas will be required to tackle the amount of data generated by wearable systems. For instance, it may come as a surprise that only reading randomly a very small subset of the sensor values (the principle underpinning “compressive sampling” [51]) yields the same accuracy as uniform sampling [52], with an enormous saving in power consumption. Other directions include the development of tailored statistical models [53] that can be estimated with fewer sensors and less computations than generic probabilistic models. Clearly, the area of data analysis for wearable systems promises to be exciting and challenging.

## 6.6 Glossary

- **Independent component analysis (ICA)** A linear decomposition of the data into statistically independent sources. The sources and the mixing weights are estimated by the algorithm.
- **Karhunen-Loève transform** See Principal component analysis.
- **Kernel PCA** See Laplacian Eigenmaps.
- **Laplacian eigenmaps** A nonlinear method to parametrize a dataset using the eigenvectors of the graph Laplacian defined on the dataset. The method optimally preserves the short-range distance while assembling a global parametrization of the data.
- **Multiscale analysis** A method to decompose a dataset into signals that have well defined characteristic scales. An example is provided by a wavelet analysis.
- **Principal component analysis (PCA)** A decomposition of a dataset into linear components that best capture the variance in the data.
- **Semi-supervised learning** A classification method that combines unlabeled data with label data to construct a classifier. The method exploits the underlying smoothness (e.g., the data lie on a manifold) of the data to estimate the geometry of the data and compensate for the lack of labels.
- **Short time Fourier transform** See time-frequency analysis.
- **Singular value decomposition** See principal component analysis.
- **Support vector machine (SVM)** A classification algorithm that combines a maximum margin classifier with a measure of similarity using a kernel.
- **Time-frequency analysis** A decomposition of a signal in terms of localized oscillatory patterns with well defined frequency and position.
- **Wavelet analysis** See multiscale analysis.

## References

1. Ahola T, Korpinen P, Rakkola J, Ramo T, Salminen J, Savolainen J (2007) Wearable fpga based wireless sensor platform. In: Engineering in Medicine and Biology Society, 2007. 29th Annual International Conference of the IEEE EMBS 2007, pp 2288–2291
2. Ali A, King R, Yang G (2008) Semi-supervised segmentation for activity recognition with Multiple Eigenspaces. In: Medical Devices and Biosensors, 2008. 5th International Summer School and Symposium on ISSS-MDBS 2008, pp 314–317
3. Ali R, Atallah L, Lo B, Yang G (2009) Transitional activity recognition with manifold embedding. In: Proceedings of the 2009 Sixth International Workshop on Wearable and Implantable Body Sensor Networks, *1*, pp 98–102
4. Atallah L, Lo B, Ali R, King R, Yang G (2009) Real-time Activity Classification Using Ambient and Wearable Sensors. IEEE Transactions on Information Technology in Biomedicine
5. Baheti PK, Garudadri H (2009) An ultra low power pulse oximeter sensor based on compressed sensing. In: Proceedings – 2009 6th International Workshop on Wearable and Implantable Body Sensor Networks, BSN 2009, pp 144–148

6. Belkin M, Niyogi P (2003) Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computations 15:1373–1396
7. Bérard P, Besson G, Gallot S (1994) Embeddings Riemannian manifolds by their heat kernel. Geomet Funct Anal 4(4):373–398
8. Bonato P, Mork P, Sherrill D, Westgaard R (2003) Data mining of motor patterns recorded with wearable technology. IEEE engineering in medicine and biology magazine 22(3):110–119
9. Bonfiglio A, Carbonaro N, Chuzel C, Curone D, Dudnik G, Germagnoli F, Hatherall D, Koller J, Lanier T, Loriga G et al (2007) Managing catastrophic events by wearable mobile systems. In: Mobile Response, vol 4458/2007. Springer, Berlin, pp 95–105
10. Candes EJ, Wakin MB (2008) An introduction to compressive sampling: A sensing/sampling paradigm that goes against the common knowledge in data acquisition. IEEE Signal Process Mag 25(2):21–30
11. Chapelle O, Schölkopf B, Zien A (eds) (2006) Semi-supervised learning. MIT, MA
12. Chung F (1997) Spectral Graph Theory, *92*(92). American Mathematical Society
13. Coifman R, Lafon S (2006) Diffusion maps. Appl Comput Harmonic Anal 21:5–30
14. Davrondzhon G, Einar S (2009) Gait Recognition Using Wearable Motion Recording Sensors. EURASIP Journal on Advances in Signal Processing pp 1–16
15. Dinh A, Shi Y, Teng D, Ralhan A, Chen L, Dal Bello-Haas V, Basran J, Ko S, McCrowsky C (2009) A fall and near-fall assessment and evaluation system. Open Biomed Eng J 3:1
16. Eagle N, Pentland A (2006) Reality mining: sensing complex social systems. Personal and Ubiquitous Computing 10(4):255–268
17. Flanagan J (2005) Unsupervised clustering of context data and learning user requirements for a mobile device. Lecture Notes in Computer Science, vol 3554, pp 155–168
18. Giansanti D, Maccioni G, Cesinaro S, Benvenuti F, Macellari V (2008) Assessment of fall-risk by means of a neural network based on parameters assessed by a wearable device during posturography. Medical Engineering and Physics 30(3):367– 372
19. Glaros C, Fotiadis D (2005) Wearable Devices in Healthcare. Studies in Fuzziness and Soft Computing 184:237
20. Han D, Park S, Lee M (2008) THE-MUSS: Mobile u-health service system. In: Biomedical Engineering Systems and Technologies, vol 25. Springer, Berlin, pp 377–389
21. Hastie T, Tibshinari R, Freedman J (2009) The elements of statistical learning. Springer, Berlin
22. Hu F, Jiang M, Xiao Y (2008) Low-cost wireless sensor networks for remote cardiac patients monitoring applications. Wireless Comm Mobile Comput 8(4):513–530
23. Huynh T, Blanke U, Schiele B (2007) Scalable recognition of daily activities with wearable sensors. In: Location-and context-awareness: Third international symposium, LoCA 2007, Oberpfaffenhofen, Germany, September 20–21. 2007 Proceedings, p 50
24. Hyvärinen A (1999) Survey on independent component analysis. Neural Comput Surv 2:94–128
25. Hyvärinen A, Oja E (2000) Independent component analysis: Algorithms and applications. Neural Network 13(4–5):411–430
26. Jaffard S, Meyer Y, Ryan R (2001) Wavelets: tools for science & technology. Society for Industrial and Applied Mathematics
27. Karlen W, Mattiussi C, Floreano D (2009) Sleep and wake classification with ECG and respiratory effort signals. IEEE Trans Biomed Circ Syst 3(2):71–78
28. Ko L, Tsai I, Yang F, Chung J, Lu S, Jung T, Lin C (2009) Real-time embedded EEG-based brain-computer interface. In: Advances in neuro-information processing. Springer, Berlin, pp 1038–1045
29. Krause A, Smailagic A, Siewiorek D (2006) Context-aware mobile computing: Learning context-dependent personal preferences from a wearable sensor array. IEEE Transactions on Mobile Computing pp 113–127

30. Mahdaviani M, Choudhury T (2008) Fast and scalable training of semi-supervised crfs with application to activity recognition. In: Platt J, Koller D, Singer Y, Roweis S (eds) Advances in Neural Information Processing Systems 20,MIT Press, Cambridge, MA, pp 977–984
31. Mallat S (1999) A wavelet tour of signal processing. Academic, NY
32. Matsuyama T (2007) Ubiquitous and wearable vision systems. In: Imaging beyond the pinhole camera, Springer, pp 307–330
33. McLachlan G, Krishnan T (1997) The EM algorithm and extensions. Wiley, NY
34. Minnen D, Starner T, Essa M, Isbell C (2006) Discovering characteristic actions from on-body sensor data. In: 10th IEEE International Symposium on Wearable computers 2006, pp 11–18
35. Pantelopoulos A, Bourbakis N (2010) Design of the new prognosis wearable system-prototype for health monitoring of people at risk. In: Advances in biomedical sensing, measurements, instrumentation and systems. Springer, Berlin, pp 29–42
36. Paoletti M, Marchesi C (2006) Discovering dangerous patterns in long-term ambulatory ECG recordings using a fast QRS detection algorithm and explorative data analysis. Comput Meth Programs Biomed 82(1):20–30
37. Patel S, Lorincz K, Hughes R, Huggins N, Growdon J, Standaert D, Akay M, Dy J, Welsh M, Bonato P (2009) Monitoring motor fluctuations in patients with Parkinson's disease using wearable sensors. IEEE Trans Inform Technol Biomed 13(6):864
38. Poh MZ, Kim K, Goessling AD, Swenson NC, Picard RW (2009) Heartphones: Sensor earphones and mobile application for non-obtrusive health monitoring. Wearable Computers, IEEE International Symposium pp 153–154
39. Powell Jr H, Hanson M, Lach J (2009) On-body inertial sensing and signal processing for clinical assessment of tremor. IEEE Trans Biomed Circ Syst 3(2):108–116
40. Preece S, Goulermas J, Kenney L, Howard D (2009) A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data. Biomedical Engineering, IEEE Transactions on 56(3):871–879
41. Schölkopf B, Smola A, Müller K (1999) Kernel principal component analysis. In: Advances in kernel methods: Support vector learning. MIT, MA
42. Stetson P (2004) Independent component analysis of pulse oximetry signals based on derivative skew. Lecture notes in computer science pp 1072–1078
43. Stikic M, Schiele B (2009) Activity recognition from sparsely labeled data using multi-instance learning. In: Proceedings of the 4th International Symposium on Location and Context Awareness, Springer, p 173
44. Subramanya A, Raj A, Bilmes J, Fox D (2006) Recognizing activities and spatial context using wearable sensors. In: Proc. of the Conference on Uncertainty in Artificial Intelligence
45. Sun Z, Mao X, Tian W, Zhang X (2009) Activity classification and dead reckoning for pedestrian navigation with wearable sensors. Meas Sci Technol 20:015203
46. Tanner S, Stein C, Graves S (2009) On-board Data Mining. Scientific Data Mining and Knowledge Discovery pp 345–376
47. Tsai D, Morley J, Suaning G, Lovell N (2009) A wearable real-time image processor for a vision prosthesis. Comput Meth Program Biomed 95(3):258–269
48. Verbeek J, Vlassis N, Krose B (2003) Efficient greedy learning of Gaussian mixtures. Neural Comput 15:469–485
49. Viswanathan M (2007) Distributed data mining in a ubiquitous healthcare framework. In: Proceedings of the 20th conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence. Springer, Berlin, p 271
50. Yao J, Warren S (2005a) A short study to assess the potential of independent component analysis for motion artifact separation in wearable pulse oximeter signals. In: 27th Annual International Conference of the Engineering in Medicine and Biology Society, IEEE-EMBS, pp 3585–3588
51. Yao J, Warren S (2005b) Applying the ISO/IEEE 11073 standards to wearable home health monitoring systems. J Clin Monit Comput 19(6):427–436

52. Zinnen A, Blanke U, Schiele B (2009) An analysis of sensor-oriented vs. modelbased activity recognition. In: IEEE International Symposium onWearable Computers, pp 93–100
53. Zhang F, Lian Y (2009) QRS Detection Based on Multiscale Mathematical Morphology for Wearable ECG Devices in Body Area Networks. IEEE Transactions on Biomedical Circuits and Systems 3(4):220–228