An efficient feature selection method for mobile devices with application to activity recognition

Neurocomputing 74 (2011) 3543–3552

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

An efficient feature selection method for mobile devices with application to activity recognition

Jian-Xun Peng *, Stuart Ferguson, Karen Rafferty, Paul D. Kelly

The School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Ashby Building, Stranmillis Road, Belfast BT9 5AH, UK

A R T I C L E I N F O

Article history:
Received 3 November 2010
Received in revised form
25 May 2011
Accepted 9 June 2011
Communicated by D. Zhang
Available online 3 August 2011

Keywords:
Feature selection algorithm
Data classification
Activity recognition
Mobile devices

A B S T R A C T

This paper presents a feature selection method for data classification, which combines a model-based variable selection technique and a fast two-stage subset selection algorithm. The relationship between a specified (and complete) set of candidate features and the class label is modeled using a non-linear full regression model which is linear-in-the-parameters. The performance of a sub-model measured by the sum of the squared-errors (SSE) is used to score the informativeness of the subset of features involved in the sub-model. The two-stage subset selection algorithm approaches a solution sub-model with the SSE being locally minimized. The features involved in the solution sub-model are selected as inputs to support vector machines (SVMs) for classification. The memory requirement of this algorithm is independent of the number of training patterns. This property makes this method suitable for applications executed in mobile devices where physical RAM memory is very limited.

An application was developed for activity recognition, which implements the proposed feature selection algorithm and an SVM training procedure. Experiments are carried out with the application running on a PDA for human activity recognition using accelerometer data. A comparison with an information gain-based feature selection method demonstrates the effectiveness and efficiency of the proposed algorithm.

1. Introduction

Over the past decade, there has been considerable research effort directed towards the monitoring and automatic recognition of human activity patterns from (body-fixed sensor) data. This has been motivated by a number of important health-related applications and ubiquitous computing. For example, there is growing interest in the link between physical activity levels and common health problems, such as diabetes [42], hypertension [7], depression [47], cardiovascular disease [6] and osteoporosis [38,43] with the trend towards more sedentary lifestyles especially in the developed countries. In this field, automatic activity profiling systems are beginning to play an important role in large-scale epidemiological studies as traditional measures based on self-reporting have been shown to be unreliable [2,46]. Furthermore, such systems can also be used to assess the effectiveness of different interventions aimed at increasing levels of physical activity and for motivating individuals to become more physically active.

In addition to health-related applications, activity-profiling systems could play a fundamental role in ubiquitous computing scenarios [11,45]. As a post-desktop model of human–computer interaction, ubiquitous computing is inevitably computing in context: it takes place in situations in the real world. Ubiquitous computing is also described as pervasive computing or ambient intelligence. Context awareness is a central issue in ubiquitous computing [1,12,45,11]. The opportunity to perceive the world from a user's perspective is a key benefit of ubiquitous computing systems compared to stationary, desktop-centered computers.

Human (user) activity is one of the most important ingredients of context information [51] and the recognition of human activities attracts increasing research interests [5,40,41,48–50,14]. While context information can consist of any information describing the situation of the user, for many applications the current activity and location of the user are considered to be highly important. In ubiquitous computing scenarios, activities such as walking, standing and sitting can be inferred from data provided by body-worn sensors, e.g. accelerometer [39]. One of the key difficulties in creating useful and robust ubiquitous, context-aware computer applications is developing the algorithms that can detect context from noisy sensor data.

As the popularity of portable handheld computers such as mobile phones increases and their cost decreases, opportunities for novel context-awareness applications arise into real-life. For example, mobile phones are often carried with people nearly everywhere they go, and people generally keep them functioning and charged [44]. Consequently, they can be used to gather and

* Corresponding author.
E-mail address: j.peng@qub.ac.uk (J.-X. Peng).

deliver information to the user. One important area where mobile phones and wearable accelerometers can be applied in combination is in creating valid and reliable measures of physical activity. For example, mobile phones can be used to run algorithms that automatically recognize physical activities. However in order to do this effective algorithms are also required to interpret the accelerometer data in the context of different activities [39,4].

The main contribution of this paper is the development of a two-stage feature selection algorithm for data classification. This algorithm combines a fast two-stage subset selection algorithm and a model-based variable selection technique, and is efficient and suitable for implementation on smart devices. This two-stage feature selection algorithm is used to classify physical activities of a human user when run on a smart device. Typically an activity recognition application usually requires two phases: (A) feature selection phase which is normally too resource intensive for execution on a smart device, (B) classification stage where the features are then used to classify activities. In this paper we show that it is possible to implement both phases successfully on a smart phone. This will have relevance to a range of health-related and ubiquitous computing applications.

In the next section we explore the state of the art in terms of feature selection algorithms. In Section 3, we propose a new feature selection algorithm that is efficient in memory usage and so can be implemented on a mobile device. Section 4 then explains the important steps when implementing the algorithms on a mobile device, and analyses the complexity. Testing and validation of the proposed algorithm for feature selection and activity classification are documented in Section 5. Finally we offer some conclusions on the effectiveness and accuracy of the proposed algorithm when implemented on a sample mobile device.

2. Brief state-of-the-art review

The term feature selection refers to algorithms that determine a subset from a given (complete) set of candidate features as the inputs to a data model (classifier). More generally, methods that create new features based on transformations or combinations of the original features are termed feature extraction algorithms. This paper addresses the former group, which is defined as follows: given a set of candidate features, select a subset that performs the best under some classification system in the context of activity classification.

Feature selection is of considerable importance in classification [3,15,17,18,24]. The feature subset that maximizes the classification accuracy is to be identified based on the principle of parsimony. The reason for this is twofold: to reduce the computational complexity and to improve the classifier's generalization ability. This is necessary because high-dimensional feature vectors impose a high computational cost and a high cost of data acquisition. On the other side, a low-dimensional representation reduces the risk of overfitting [13,19].

Selecting the best subset of features is in nature a combinatorial problem in the number of candidate features [15,20], and is proven to be an NP-complete problem [25]. Exhaustive search of all possible feature subsets guarantees the best solution, however this is often computationally impractical. For a data set of M features, the number of all possible feature subsets, or potential solutions, is 2^M, and this is often too large to be evaluated in practice even for modest M. A number of different techniques, of varying complexity, have been used to select appropriate features for data classification. Guyon and Elissee classify existing feature selection methods into three types: filter, wrapper and embedded methods [15].

Filter methods select features in a preprocessing step independent of the chosen classifier. Frequently used filter methods include t-test [26], chi-square test [27], Wilcoxon Mann–Whitney test [28], mutual information [29], Pearson correlation coefficients [30] and principal component analysis [31]. In a filter method, the features are scored and ranked using certain statistical criteria and those features with the highest ranking values are selected. Filter methods are generally computationally cheap, but often not very effective. In addition, it is not clear how to determine the cut-off point for rankings to select only truly important features and exclude noise.

Wrapper methods use a classifier of interest as a black box to evaluate subsets of features. An exhaustive search is not computationally feasible in practice, and as a result some learning algorithm is often employed to search for the good subset against some criterion based on the classification accuracy of the classifier. Greedy and stochastic types of search strategy have been widely employed. Currently, the most popular learning algorithm used in wrapper methods is support vector machines (SVMs). Sequential forward selection (SFS) and sequential backward selection (SBS) are the two most commonly used greedy wrapper methods using a greedy hill-climbing search strategy. Stochastic algorithms such as ant colony optimization (ACO), genetic algorithm (GA) [32], particle swarm optimization (PSO) [33] and simulated annealing (SA) [34] are at the forefront of research in feature subset selection.

Wrapper methods are computationally demanding, but often provide more accurate results than filter methods. A wrapper algorithm explores the feature space to score feature subsets (rather than individual features) according to their classification accuracy, optimizing the subsequent induction algorithm that uses the respective subset for classification. Recursive feature elimination for support vector machine (RFE-SVM) [16] is such a feature selection algorithm. It was originally formulated for binary classification problems, and then extended for multi-class problems [10,23] and some other variants were also proposed [15,17,21]. The goal of RFE-SVM is to find a subset of m from M candidate features, $m < M$, which maximizes the performance of an SVM classifier. Given that one wishes to employ only $m < M$ features for the final classifier, this method attempts to find the best subset of m features. It operates by trying to choose the m features which lead to the largest margin of class separation. This problem is based on an SBS, i.e., it removes one or more features at a time until m features remain. RFE is computationally expensive although it consistently outperforms the naive ranking methods, particularly for small feature subsets. The naive ranking and RFE are qualitatively different. The naive ranking orders feature according to their individual relevance. The RFE ranking is a feature subset ranking.

Embedded methods perform feature selection as part of the training process of the classifier. For example, Miranda et al. [22] added an extra term that penalizes the size of the selected feature subset to the standard cost function of SVM, and optimized the new objective function to select features. However these approaches are limited to linear SVMs. Another approach for feature penalization is the so-called l_0-SVM or concave feature selection [8], which minimizes the "zero norm" which is defined for a weighting vector $\mathbf{w}$ as $|\{i : w_i \neq 0\}|$, i.e., the number of nonzero components, w_i's, of the weighting vector $\mathbf{w}$. Note that this l_0-"norm" is not an ordinary norm because, unlike l_p-norms with $p > 0$, the triangle inequality does not hold here. Since the l_0-"norm" is discontinuous, it was approximated by a concave function $\sum_i (1-\exp(-a|w_i|))$, $a > 0$, that eliminates the discontinuities, where $|w_i|$ denotes the absolute value of component w_i. It is shown in [8] that for a finite value of a, which appears in the concave exponential approximation, the smooth problem generates an exact solution of the original discontinuous problem. Another example of the embedded method is feature selection via scaling factors [9].

3. A feature selection algorithm

Existing embedded and wrapper methods are based on training of the full or a series of (decreasing) SVMs. Initially all the potential features are involved in the SVM. The training of an SVM involves the determination of a scaling factor for each candidate feature and one or more features with scaling factors of zero or small magnitude are eliminated. This procedure is iterated until no irrelevant features remain. The training of the SVM involves a large-scale nonlinear programming problem, particularly for those problems with a large set of training patterns and a large number of candidate features. This means great computational resources are required, so it can be difficult to implement in mobile devices with limited computational resources, particularly physical memory.

In this paper, a feature selection algorithm is proposed which combines the model-based variable selection technique from [35] and a fast two-stage subset selection algorithm [53] with extensions for cases of multiple outputs. The relationship between the candidate features and the class label is modeled using a full regression model which is linear-in-the-parameters [35]. Each sub-model involves a subset of features. It scores this feature subset using the performance of the sub-model measured by the sum of the squared modeling errors (SSE). A sub-model of smaller SSE is assumed to include a more informative subset of features. The two-stage subset selection algorithm is employed to approach a solution sub-model by minimizing the SSE. The subset of features involved in the solution sub-model can then be used for classification. The algorithm is computationally efficient and suitable for applications on devices with limited computational resources, particularly physical memory.

This section describes in detail the design of the proposed feature selection algorithm.

To define the problem, let $\{(\mathbf{x}(k),l(k)),\ldots,k=1,\ldots,N\}$ be a set of K-class labeled patterns, referred to as the *training patterns* or the *training set*, where $l(k)\in\{L_c,c=1,\ldots,K\}$ is the label attached to the k'th pattern $\mathbf{x}(k)$ and L_c denotes the c'th class; $K>1$ and $N>1$ are the numbers of classes and patterns, respectively. The problem of data classification is to construct a discriminant function of pattern features, referred to as the *classifier*, that separates the patterns into K classes.

Suppose the patterns are of M dimensions and are represented in row vectors: $\mathbf{x}(k)=[x_1(k),\ldots,x_M(k)],k=1,\ldots,N$. Pattern components, $x_1,\ldots,x_M$, are referred to as *candidate features*, which are the candidate inputs to the classifier. Each component of a given pattern is a value of the corresponding candidate feature. In practice, there are a large number of candidate features available and it is not normally known *a priori* which features should be selected as the inputs to the final classifier. The problem of feature selection concerned in this paper is to select a subset of $m,1\le m\le M$, from the M candidate features, such that the patterns can be discriminated to a high precision. For convenience, the space spanned by all the candidate features is referred to as the *full feature space*, and the space spanned by the selected features is referred to as the *input space*.

The method proposed in this paper allows a good subset of features to be determined based on the fact that certain combinations of features are highly correlated with specific activities (the output, as identified by a label). This is revealed in some filter selection methods. Regression theory also tells us that a variable (the output label in this case) can be predicted from some correlative factors (features in this case) using a regression model. Intuitively, we can measure the correlation between the label and a subset of features using a regression model and identify the most informative subset of features.

To model the relationship between the K-class labels and the patterns in the training set using a regression model, each of the pattern labels is encoded in a binary string $\mathbf{y}(k)=[y_1(k),\cdots,y_K(k)],y_c(k)\in\{+1,-1\},c=1,\ldots,K,k=1,\ldots,N$, forming an N-by-K label matrix, which is given by

$$\mathbf{Y}=[y_{k,c}]_{N\times K},y_{k,c}=y_c(k)=\begin{cases}+1 & \text{if} \quad l(k)=L_c\\ -1 & \text{if} \quad l(k)\neq L_c\end{cases} \tag{1}$$

where $l(k)$ is the label associated to the kth pattern, L_c is the cth class label. In column c of $\mathbf{Y}$, the components corresponding to patterns belonging to class L_c are set at $+1$, while others are set at -1.

To improve the classification precision, nonlinear classifiers (SVMs) are normally employed, as the given data are often not linearly separatable in practice. Corresponding to this, the full feature space is mapped into a high-dimensional space using some nonlinear map that is given in advance. In this study, polynomials of the candidate features are employed. For example, for a three-dimensional full feature space, a pattern $[x_1,x_2,x_3]$, is mapped into a point in a 9-dimensional space using all the first- and second-order polynomial terms of the 3 candidate features, given by $[x_1,x_2,x_3,x_1x_1,x_1x_2,x_1x_3,x_2x_2,x_2x_3,x_3x_3]$. In this way, a set of terms defines the map from the full feature space into a high-dimensional space. This new high-dimensional space is simply referred to as the Φ-space, and the set of terms is referred to as the *term pool* hereafter. Each term from the term pool is a predefined linear or nonlinear function of one or a few candidate features. More generally, the term pool is denoted as $\{\phi_i(\mathbf{x}),i=1,\ldots,H\}$, where $\mathbf{x}$ denotes a full feature vector composed of all the candidate features and H is the number of predefined terms. With the term pool, the binary label string is modeled in a regression model of M inputs, $x_1,\ldots,x_M$, and K outputs, $y_1,\ldots,y_K$, which is nonlinear with regard to the features while line-in-the-parameters.

To simplify the regression model, the means of the terms and outputs over the training data are removed. In this way, a DC term does not need to be considered in the regression model. The regression model for the centered outputs and terms is thus given by

$$\mathbf{y}(k)=\mathbf{\Phi}(\mathbf{x}(k))\mathbf{W}+\mathbf{e}(k),k=1,\ldots,N \tag{2}$$

where $\mathbf{y}(k)$ denotes the kth row of column-centered $\mathbf{Y}$, $\mathbf{\Phi}(\mathbf{x}(k))$ denotes the row vector composed of the centered values of the H predefined terms evaluated for pattern k, $\mathbf{W}=[\mathbf{w}_1,\ldots,\mathbf{w}_K]$ is the model parameters to be determined and $\mathbf{e}(k)$ denotes the modeling errors of the label string code for the kth pattern, the centered outputs and terms are given by

$$\begin{cases}\mathbf{y}(k)=[y_1(k)-\overline{y}_1,\ldots,y_K(k)-\overline{y}_K]\\ \mathbf{\Phi}(\mathbf{x}(k))=[\phi_1(\mathbf{x}(k))-\overline{\phi}_1,\ldots,\phi_H(\mathbf{x}(k))-\overline{\phi}_H]\end{cases} \tag{3}$$

with the means of the outputs and terms being

$$\begin{cases}\overline{y}_c=\dfrac{1}{N}\sum\limits_{k=1}^{N}y_c(k), \quad c=1,\ldots,K\\ \overline{\phi}_h=\dfrac{1}{N}\sum\limits_{k=1}^{N}\phi_h(\mathbf{x}(k)), \quad h=1,\ldots,H\end{cases} \tag{4}$$

Note that model (2) for centered data is equivalent to a model with a DC term for un-centered data: $\mathbf{y}(k)=\mathbf{w}_0+\mathbf{\Phi}(\mathbf{x}(k))\mathbf{W}+\mathbf{e}(k)$ where $\mathbf{w}_0$ represents the DC term.

Model (2) is referred to as the full model as it involves all the candidate features. It is often too complex for practical usage. By (2), the feature selection problem is transformed into the problem of identifying a significant subset of terms. In this study, a subset selection technique is used to select a sub-model that includes only a small subset of terms from the term pool while provides satisfactory prediction performance. As each term includes one or more features, the features involved in all the selected terms are viewed as sufficiently informative so as to discriminate the output labels.

This model-based method was originally proposed for selecting inputs to single output neural networks [35], where a forward subset selection algorithm is used. It is extended in this study for cases of multiple outputs and used to select informative subset features for multi-class classification problems. A fast two-stage subset selection algorithm is used instead of the forward one. This two-stage algorithm combines both forward and backward subset selection techniques. In the first stage, an initial sub-model is selected incrementally. The second-stage is to refine the initial sub-model constructed in the first stage by reviewing each of the selected terms. The features involved in the terms of the final sub-model are then selected as inputs to a classifier.

3.1. Stage I—forward selection

In the first stage, one term is selected from the term pool in each step. For the $(s+1)$th step, suppose s terms have been selected in the previous s steps, forming an intermediate sub-model, of which the modeling errors over the training data are denoted in matrix form $\mathbf{E}_s$, and given by

$$\mathbf{E}_s=\begin{cases}\mathbf{Y}-\mathbf{P}_s\mathbf{W}_s, & s=1,2,\ldots\\ \mathbf{Y}, & s=0\end{cases} \tag{5}$$

where $\mathbf{P}_s=[\mathbf{p}_1,\cdots,\mathbf{p}_s]$ collects the s terms selected in the previous s steps, $\mathbf{W}_s$ the s-by-K model parameter matrix to be determined, $\mathbf{Y}=[\mathbf{y}_1,\cdots,\mathbf{y}_K]$ is the $N\times K$ label matrix with its columns centered. For each one from the term pool, say $\phi_h(\mathbf{x})$, a centered column vector can be computed in advance, denoted as $\mathbf{\Phi}_h=[\phi_h(\mathbf{x}(1))-\overline{\phi}_h,\ldots,\phi_h(\mathbf{x}(N))-\overline{\phi}_h]^T, h=1,\ldots,H$. Hereafter, except specifically mentioned, $\mathbf{Y}$ refers to the column-centered label matrix with its K centered columns referred to as $\mathbf{y}_1,\ldots,\mathbf{y}_K$; and $\mathbf{\Phi}_h$ refers to the centered column vector with the N components computed with term $\boldsymbol{\phi}_h$ over the N patterns, respectively.

In addition, to distinguish selected terms from candidate terms in the term pool, column vectors $\mathbf{p}_1,\ldots,\mathbf{p}_s$ refer to the centered columns generated by the s selected terms over the training data. The terms that generate $\mathbf{\Phi}_h$ and $\mathbf{p}_i$ are simply referred to as term $\mathbf{\Phi}_h$ and term $\mathbf{p}_i$, respectively.

The parameter matrix $\mathbf{W}_s$ is determined by minimizing the sum of squared errors (SSE), i.e., the sum of the squared elements of $\mathbf{E}$. Assume that $\mathbf{P}_s$ is fully column ranked. The minimized SSE for the least-squares (LS) solution to $\mathbf{W}_s$ is given by

$$V(\mathbf{P}_s)=\|\mathbf{E}_s\|_2^2=\mathrm{trace}(\mathbf{Y}^T\mathbf{R}_s\mathbf{Y}) \tag{6}$$

where $\mathbf{R}_s$ is an N-by-N matrix function of $\mathbf{P}_s$, which is given by

$$\mathbf{R}_s=\mathbf{R}(\mathbf{P}_s)=\begin{cases}\mathbf{I}-\mathbf{P}_s(\mathbf{P}_s^T\mathbf{P}_s)^{-1}\mathbf{P}_s^T, & s>0\\ \mathbf{I}, & s=0\end{cases} \tag{7}$$

where $\mathbf{I}$ is the N-by-N identity matrix.

To simplify the description, the unselected $H-s$ terms remaining in the term pool are re-sorted and numbered as $\{\mathbf{\Phi}_i(x), i=s+1,\ldots,H\}$. To select the $(s+1)$th term, for each of the $H-s$ unselected terms, say $\mathbf{\Phi}_i$, adding it into the sth intermediate model of s terms forms a temporary model with the error matrix given by $\mathbf{E}_{s+1}=\mathbf{Y}-[\mathbf{P}_s,\mathbf{\Phi}_i][{}^{\mathbf{W}_s}_{w_{s+1}}]$. The reduction in the SSE due to adding of $\mathbf{\Phi}_i$ is given by

$$\Delta V(\mathbf{P}_s,\mathbf{\Phi}_i)=V(\mathbf{P}_s)-V([\mathbf{P}_s,\mathbf{\Phi}_i])=\mathrm{tr}(\mathbf{Y}^T\Delta\mathbf{R}\mathbf{Y}) \tag{8}$$

where $\Delta\mathbf{R}=\mathbf{R}(\mathbf{P}_s)-\mathbf{R}([\mathbf{P}_s,\mathbf{\Phi}_i])$. The one that maximizes the SSE reduction (8) is selected as the $s+1$th terms:

$$\mathbf{p}_{s+1}=\arg\max\{\Delta V(\mathbf{P}_s,\mathbf{\Phi}_i),\mathbf{\Phi}_i=\mathbf{\Phi}_{s+1},\mathbf{\Phi}_{s+2},\ldots,\mathbf{\Phi}_H\} \tag{9}$$

By extending the fast forward selection algorithm proposed in [54,53] for cases of multiple outputs, the SSE reduction (8) can be computed as

$$\Delta V(\mathbf{P}_s,\mathbf{\Phi}_i)=\sum_{c=1}^{K}\frac{(r_{c,i}^{s)})^2}{d_i^{s)}},\quad i=s+1,\ldots,H \tag{10}$$

where $r_{c,i}^{s)}$ and $d_i^{s)}$ are computed iteratively as

$$\begin{cases} r_{c,i}^{s+1)}=r_{c,i}^{s)}-\dfrac{r_{c,s+1}^{s)}a_{s+1,i}}{d_{s+1}^{s)}}, & c=1,\ldots,K,\quad i=s+2,\ldots,H\\ d_i^{s+1)}=d_i^{s)}-\dfrac{a_{s+1,i}a_{s+1,i}}{d_{s+1}^{s)}}, & i=s+2,\ldots,H\\ a_{s+1,i}=\mathbf{p}_{s+1}^T\mathbf{\Phi}_i-\displaystyle\sum_{j=1}^{s}\frac{a_{j,s+1}a_{j,i}}{d_{j,j}}, & i=s+1,\ldots,H\end{cases} \tag{11}$$

with initial values $r_{c,i}^{0)}=\mathbf{y}_c^T\mathbf{\Phi}_i$ and $d_i^{0)}=\mathbf{\Phi}_i^T\mathbf{\Phi}_i$. Eqs. (10) and (11) are derived from $r_{c,i}^{s)}=\mathbf{y}_c^T\mathbf{R}_s\mathbf{\Phi}_i$, $d_i^{s)}=\mathbf{\Phi}_i^T\mathbf{R}_s\mathbf{\Phi}_i$ and $a_{s+1,i}=\mathbf{p}_{s+1}^T\mathbf{R}_s\mathbf{\Phi}_i$ using recursive equation $\mathbf{R}_{s+1}=\mathbf{R}_s-(\mathbf{R}_s\mathbf{p}_{s+1}\mathbf{p}_{s+1}^T\mathbf{R}_s)/(\mathbf{p}_{s+1}^T\mathbf{R}_s\mathbf{p}_{s+1})$. For more details, one is referred to [53,54]. It could be noted that $a_{i,i}=d_i^{i-1)}$ holds for $i=1,\ldots,m$, where m denotes the number of selected terms. Because the terms remaining in the term pool are re-sorted as mentioned before, we have $\mathbf{p}_i=\mathbf{\Phi}_i, i=1,\ldots,m$, i.e., the first m ones are the m selected terms.

Adding the selected term into the sth intermediate sub-model of s terms, results in the $s+1$th intermediate sub-model of $s+1$ terms, for which the SSE is given by $V(\mathbf{P}_{s+1})=V(\mathbf{P}_s)-\Delta V(\mathbf{P}_s,\mathbf{p}_{s+1})$. Note that initially $V(\mathbf{P}_0)=\|\mathbf{Y}\|_2^2$, i.e., the sum of the squared elements of the centered label matrix. In the first stage, this forward selection procedure is iterated for $s=0,1,2,\ldots$ until some criterion is satisfied, for example, a given number of terms or features are selected.

Suppose an initial sub-model of m terms is selected in the first stage, the model is then refined in the second stage. As the terms in the term pool $\{\mathbf{\Phi}_k, k=1,\ldots,H\}$ are normally not orthogonal with each other, the selected sub-model is often not optimal even locally. It can be further refined by replacing some of the selected model terms (especially those selected in the early stage) with terms from the term pool. This is done by testing the selected term one by one in the second stage.

3.2. Stage II—model review

In the second stage, the selected sub-model is further refined by reviewing the selected terms. For each selected term, if replacing it with another term in the term pool causes a decrease in the SSE then, swap the two terms (i.e., bring the term from the term pool into the sub-model, and put the term in the sub-model back into the term pool.) For example, for a sub-model of m terms, to test the kth selected term $\mathbf{p}_k$, denote $\mathbf{P}_m^{k)}=[\mathbf{p}_1,\ldots,\mathbf{p}_{k-1},\mathbf{p}_{k-1},\ldots,\mathbf{p}_m]$, i.e., it involves all the other $m-1$ terms. The SSE reduction (10) for $\mathbf{P}_m^{k)}$ is computed for $\mathbf{p}_k$ and all the unselected terms in the term pool, i.e., $\Delta V(\mathbf{P}_m^{k)},\mathbf{\Phi}_i)$ for $i=m+1,\ldots,H$. If

$$\Delta V(\mathbf{P}_m^{k)},\mathbf{p}_k)<\max\{\Delta V(\mathbf{P}_m^{k)},\mathbf{\Phi}_i), i=m+1,\ldots,H\} \tag{12}$$

then the term from the term pool that gives the maximum, say $\mathbf{\Phi}_j$, is selected to replace $\mathbf{p}_k$ while $\mathbf{p}_k$ is put back into the term pool. Otherwise $\mathbf{p}_k$ keeps unchanged for the sub-model.

This model review procedure is iterated until no term within the sub-model can be replaced. That is: the SSE cannot be further reduced through selecting a new candidate term to replace an existing term in the sub-model. In this sense, the refined sub-model is locally optimal, any change in one of the selected term will cause increase in the model SSE. Once the sub-model is refined, the terms presented in the final sub-model are investigated and the subset of features involved in the selected terms are thus selected for classifier design.

Note that, to check the kth term $\mathbf{p}_k$ in the second stage, there are two ways to compute the SSE reduction, $\Delta V(\mathbf{P}_m^{k)},\mathbf{\Phi}_i)$: the first one uses (11) for the re-sorted terms $\mathbf{P}_m^{k)}=[\mathbf{p}_1,\ldots,\mathbf{p}_{k-1},\mathbf{p}_{k-1},\ldots,\mathbf{p}_m]$ in order to re-construct the intermediate sub-model with the $m-1$ selected terms (without $\mathbf{p}_k$) and then re-select the mth term which can be $\mathbf{p}_k$ again or a new one from the term pool, as is done in [55]. Another one swaps $\mathbf{p}_q$ and $\mathbf{p}_{q+1}$ iteratively for $q=k,k+1,\ldots,m-1$ such that the intermediate sub-model is constructed with $\mathbf{p}_k$ being the last (mth) selected term and the reduction of SSE due to introduction of $\mathbf{p}_k$, $\Delta V(\mathbf{P}_m^{k)},\mathbf{p}_k)$, can be computed and compared with that of the remained candidate terms $\Delta V(\mathbf{P}_m^{k)},\mathbf{\Phi}_i),i=m+1,\ldots,H$, as is done in [53]. The first way is simpler when implemented as the forward selection procedure can be re-used in the second stage, but it is computationally more complex. For the first way, the computational burden measured in the number of floating-point operations (FPOs) to check each of the m selected terms is almost the same as that of the first stage. The computational burden to perform a full check loop for the sub-model (of m selected terms) is thus about m times of that of the first stage. However, for the second way, the number of FPOs to check all the m terms is only about $\frac{3}{4}$ of that of the first stage.

In this paper, the second way is employed with extension for cases of multiple outputs. One is referred to [53] for more details. Note that the operations on the vectors (including the outputs, the selected and the unselected terms, i.e., computing $\mathbf{R}_s\mathbf{y}_c$, $\mathbf{R}_s\mathbf{p}_{s+1}$ and $\mathbf{R}_s\mathbf{\Phi}_i$ for $i=s+1,\ldots,H,s=1,\ldots,m$) in [53] are unnecessary here, see the implementation of this algorithm in Section 4.

This proposed algorithm has a number of useful features when it comes to implementation as a computer program for limited resource host systems.

4. Implementation on a mobile device

The implementation of an activity classification algorithm usually requires two phases: feature selection phase determines what features are to be selected for classification and trains a classifier based on the selected features in an off-line process (normally too resource intensive for execution on a smart phone), and a classification phase where those selected features are extracted from accelerations and used to carry out the classification in real time. Our algorithm has the advantage that the classification phase can operate in real-time and even the feature selection phase is sufficiently memory efficient that it will execute on the same platform. By being able to perform both stages on a mobile device our procedure offers much greater flexibility for applications that need to adapt to widely varying operating conditions.

To demonstrate our algorithm an application program was developed for activity recognition that executed on a HP iPAQ pocket PC running Windows Mobile 6. In this application, SVMs are employed as the classifiers. This application can perform interactive activity recognition and background classifier updating. It includes five modules in additional to the user interface module:

1. Signal data buffering and windowing.
2. Feature extraction.
3. Classifier (SVM).
4. Feature selection.
5. Classifier (SVM) training.

The information flow of this application is illustrated in Fig. 1. The data source for the application can be accelerometers or a signal data file recorded on other platforms. The signal data buffering and windowing module read 256 samples each time in order to compose signal segments. The feature extracting module extracts all candidate features (for feature selection) or a specified subset of features (for activity recognition). For activity recognition, the specified features are sent to the classifier module which implements a trained SVM for real-time activity recognition.

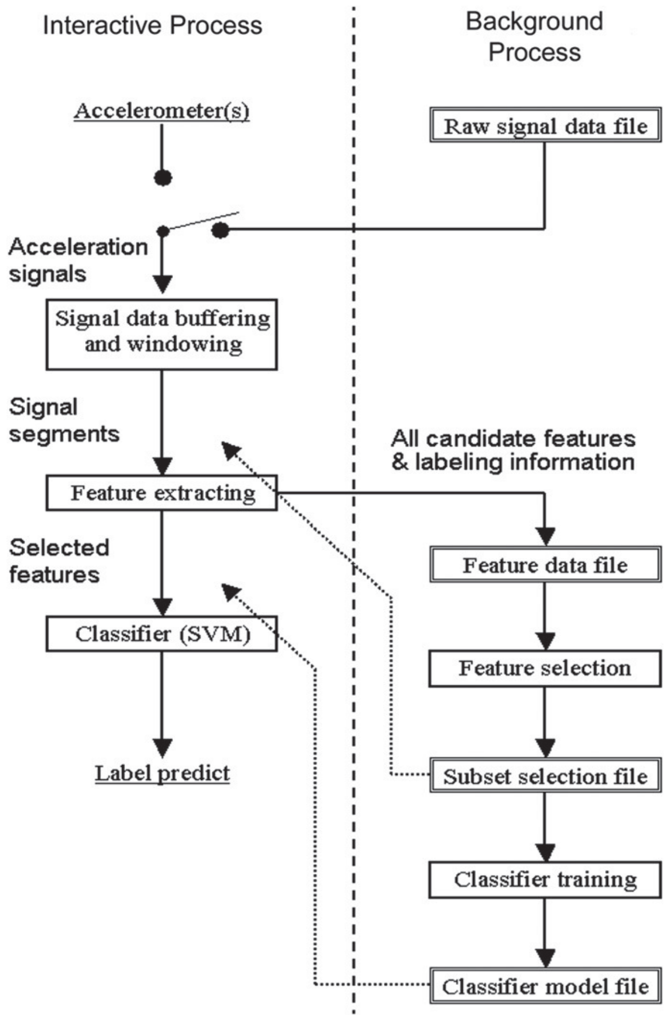

Fig. 1. Information flow of the application.

Different users (subjects) may generate different acceleration signal patterns for a particular activity, for example, the acceleration signals of disabled people could be quite different forms that of the general population. In addition to the basic thread for real-time activity recognition, a thread can be started for classifier calibration using customized acceleration data. In this classifier calibration thread, a feature data file is generated by all the candidate features for a period of accelerometer readings or a given raw signal data file, and the proposed algorithm (implemented in the feature selection module) is used to select informative features using the generated feature data file. Specifications of the selected subset of features are saved in an additional file (subset selection file), which can be used to specify the feature extracting module and to perform classifier training. An SVM training module is implemented in this application that uses the selected features and the data saved in the feature data file. The trained model is saved in a classifier model file and can be implemented in the classifier module for real-time activity recognition.

Looking at the proposed algorithm, to select m from M candidate features for a set of K-class patterns with H predefined terms, the main memory requirement is as follows: the m-by-H

matrix $\mathbf{A}=[a_{k,i}]_{m\times H}$, the K-by-H matrix $\mathbf{r}=[r_{c,i}^{s)}]_{K\times H}$ and the vector $\mathbf{d}=[d_i^{s)}]_{1\times H}$ and finally the upper-triangular part of an H-by-H symmetric matrix

$$\mathbf{C}=[c_{i,j}]_{H\times H},\quad c_{i,j}=\mathbf{\Phi}_i^T\mathbf{\Phi}_j=\sum_{k=1}^{N}\phi_i(\mathbf{x}(k))\phi_j(\mathbf{x}(k))-N\overline{\phi}_i\overline{\phi}_j \tag{13}$$

Note again that $\phi_k(\mathbf{x}),k=1,\ldots,N$ are the predefined terms and, as mentioned before, polynomial terms are employed in this study. Matrix $\mathbf{A}$, $\mathbf{r}$ and vector $\mathbf{d}$ store quantities $a_{k,i}$'s, $r_{c,i}^{s)}$'s and $d_i^{s)}$'s, respectively, that are computed using (11) in the first stage and updated corresponding to term-pair swapping during the second stage (see [53] for the details). Matrix $\mathbf{C}$ is defined to store the correlations of the centered vectors generated by all the candidate terms over the training data. It can be computed sequentially as each training pattern (with all candidate features) is obtained from the input channel (extracted from a fresh window of signals read from sensors or directly read from a given file). In this way it is not necessary to load the full training data set into the memory.

It should be noted that it is not absolutely necessary to introduce matrix $\mathbf{C}$ in order to implement the proposed feature selection algorithm, but it avoids repeating the calculation of the term correlations when updating/computing the components of matrix $\mathbf{A}$ during both the forward selection and model review stages.

For a set of N training patterns from K classes and H predefined terms, matrices $\mathbf{A}$, $\mathbf{r}$, $\mathbf{d}$ and $\mathbf{C}$ (only the upper triangular part needs to be computed and stored) are of $[m+K+(H+3)/2]H$ elements in total. Note that the memory requirement is not dependent on N, the number of patterns. This property makes this algorithm suitable for devices of limited physical memory resources, e.g., the HP iPAQ pocket PC, on which an application can only allocate about 14 MB memory.

Initially, the feature selection procedure computes N-dimensional vectors $\mathbf{\Phi}_i,i=1,\ldots,H$ over raw feature data, the upper-triangular of the correlation matrix $\mathbf{C}$ and initial values of $\mathbf{r}$ and $\mathbf{d}$. Note that the elements of matrix $\mathbf{d}$ are initially the same as the diagonals of matrix $\mathbf{C}$. Computing $\mathbf{\Phi}_i$'s involves N evaluations of each of the H predefined terms. Computing initial matrices $\mathbf{C}$ and $\mathbf{r}$ involves centering $H+K$ N-dimensional vectors and $H(H+1)/2$ and KH inner products computation of these centered vectors, respectively. Centering one N-dimensional vector and computing one inner product of two N-dimensional vectors involve $2N$ and $2N-1$ floating-point operations (FPOs, including addition/subtraction and multiplication/division and comparison). The overall computation burden in this stage includes $2N(K+H)+(2N-1)[KH+H(H+1)/2]$ FPOs and NH terms' evaluations.

In the first stage, to select the $(s+1)$th term, it needs firstly to compute SSE reduction $\Delta V(\mathbf{P}_s,\mathbf{\Phi}_i)$ for $i=s+1$ to H in (10), and to find the maximal one with the corresponding term be selected as the $(s+1)$th. This involves $4(H-s)$ FPOs. Then it needs to update matrices $\mathbf{r}$ and $\mathbf{d}$ and to compute the $(s+1)$th row of matrix $\mathbf{A}$ in (11). This involves $3(K+1)(H-s)+3s(H-s-1)$ FPOs in total. To select m terms, this selecting process is iterated for $s=0,1,\ldots,m-1$, and the computation totals up to $mH(3K+7)+\frac{1}{2}m(m-1)(3H-3K-2m-9)$ FPOs.

In the second stage, to check, say the kth, $1\le k\le m-1$ term in the model of m selected terms, it needs to re-sort the selected terms as $[\mathbf{P}_m^{k)},\mathbf{p}_k]=[\mathbf{p}_1,\ldots,\mathbf{p}_{k-1},\mathbf{p}_{k-1},\ldots,\mathbf{p}_m,\mathbf{p}_k]$ by swapping $\mathbf{p}_q$ and $\mathbf{p}_{q+1}$ iteratively for $q=k,k+1,\ldots,m-1$, and reconstruct the corresponding matrices $\mathbf{r}$, $\mathbf{d}$ and $\mathbf{A}$ for $\mathbf{P}_m^{k)}$. In this context, $\Delta V(\mathbf{P}_m^{k)},\mathbf{p}_k)$ and $\Delta V(\mathbf{P}_m^{k)},\mathbf{\Phi}_i),i=m+1,\ldots,H$ can be computed. Term $\mathbf{p}_k$ is then checked as done in (12). If (12) holds, then $\mathbf{p}_k$ is replaced and the corresponding elements of $\mathbf{r}$, $\mathbf{d}$ and $\mathbf{A}$ are updated. A model reviewing loop iterates this term checking process for $k=m-1,m-2,\ldots,1$. If a selected term is replaced during this loop, a new reviewing loop is restarted for $k=m-1,m-2,\ldots,1$ again even the current loop has not been completed. This reviewing process is iterated until no terms in the model is changed.

According to the analysis in [53], the number of FPOs for a reviewing loop is about $\frac{3}{4}$ of that of the first stage in the worst case, that is when the first term $\mathbf{p}_1$ is changed in a reviewing loop. However it should be noted that the number of iterations of this reviewing process in the second stage is not known *a priori*. Generally, the more highly the terms are correlated, the more iterations are required to approach the solution.

Normally, each evaluation of a term involves (or is equivalent to) a few FPOs. Assume that the number of model reviewing loops required in the second stage is of order $O(m)$, and that the model size m is of order $O(K)$, with $m,K\ll H$. The overall computational complexity of the algorithm measured in the number of FPOs is of order $O(NH^2+m^2HK)$.

It should also be noted that the feature selection is implemented in a batch way. That is, the feature selection can be performed only after a set of data becomes available. To confirm the uniqueness of the solution, it requires N, the number of training patterns, to be at least not smaller than the number of candidate terms H, i.e., $N\ge H$.

However this batch processing is time consuming and may cause some potential challenges in mobile devices when it is busy (engaged). In this case, an incremental version of this algorithm running asynchronously in a sample-by-sample manner is more suitable. It is not difficult to compute $\mathbf{C}$ and the initial values of $\mathbf{r}$ and $\mathbf{d}$ in an incremental way in the initialization step. After all data samples have been completed (matrix $\mathbf{C}$, and initial values of both $\mathbf{r}$ and $\mathbf{d}$ have been computed), both the second and the third steps are started to perform term selection. According to the previous complexity analysis results for the three steps, the initial step dominates the computational burden of this algorithm, given again that $m\ll H$ generally holds. But it can be noted that the full set of labeled training patterns should be ready in advance.

5. Application to activity recognition

To test the algorithm a practical scenario is considered in which information is to be presented to the user of a mobile phone in a form that depends on their activity (for example walking, running or shopping.)

In practice the activity is to be determined from data streamed by a set of accelerometers attached to the participant. For experimental purposes three subjects participated in the experiments and the set of dynamic acceleration data was collected using four Nintendo Wiimote devices. Each device contains a tri-axial accelerometer with a range of $\pm 3g$ (where g is the acceleration due to gravity) and a maximum sampling rate of 100 Hz. One Wiimote was securely attached to each lower leg just above the ankle, and to each forearm over the wrist using Velcro straps.

Care was taken to ensure that the Wiimotes were attached to each subject with the same relative position, as described above, and orientation, i.e., with the upper surface (containing the buttons) facing outward (sideways) from the body, and the infra-red sensor pointing upwards.

With the Wiimotes attached, the subjects performed three different activities: (a) walking slowly, (b) walking quickly and (c) browsing, i.e., stopping and starting, looking around, erratic movement.

The four Wiimotes were interfaced using Bluetooth to a Linux personal digital assistant (PDA) carried out in the subject's hand or pocket. The data capture program was written in C and used an early prototype version of the HaptiMap [36] toolkit together with the *Cwiid* Wiimote interface library [37]. The Cwiid library was used in callback mode: a callback is called every time an accelerometer report is received from the Wiimote (the Wiimote sends accelerometer reports asynchronously, and at a higher rate when greater acceleration is being experienced). This updates a set of shared memory variables, which are then sampled at 100 Hz in order to give a uniform sampling rate for feature derivation.

The collected signal data was then labeled manually according to the recorded time information. The feature selection and classification application were tested by passing the labeled data to the HP iPAQ PDA where all the following experiments are carried out. For this particular example there are 186 candidate features extracted from the 12 acceleration signals, including *Means* (12, with one for each of the 12 acceleration signals), *Deviations* (12), *Total Power-TD* (12), *Axes Correlation* (66, there are 66 potential pairs from the 12 acceleration signals), *FFT DC* (12), *Dominant frequency-1* (12), *Dominant frequency-2* (12), *Dominant power-1* (12), *Dominant power-2* (12), *Entropy-FD* (12), *Total Power-FD* (12), where TD and FD stand for time domain and frequency domain, respectively. The features are extracted with a sliding window of 256 signal samples. All these candidate features are defined in Table 1.

The TD features are computed using time domain quantities (raw signal samples), while the FD features are computed using the frequency–energy spectrogram composed of 129 frequency components (of frequency $\frac{100\omega}{256}$ Hz, $\omega = 0,1,\ldots,128$, respectively) obtained by a fast fourier transform (FFT) procedure on a 256-sample signal segment. In Table 1, u and v are two signal segments of 256 samples with the bracketed subscript t denoting the sample index number; $f(0)$ denotes the DC term of the FFT expression and $|f(0)|$ the absolute value; $f(\omega), \omega = 1,\ldots,128$ denote the 128 FFT coefficients for nonzero frequency components and $\overline{f(\omega)}$ the corresponding complex conjugates, $p(\omega) = f(\omega)\overline{f(\omega)}/\mathit{Total\ Power-FD}$, $\omega = 1,\ldots,128$ are the normalized frequency–energy spectrogram; $peak^{1)}$ and $peak^{2)}$ denote the two functions searching for the highest and second highest peaks, respectively, of a frequency–energy spectrogram $p(\omega)$. The magnitudes of the two peaks are respectively *Dominant power-1* and *Dominant power-2*, while the frequencies at the two peaks are respectively *Dominant frequency-1* and *Dominant frequency-2*.

From the acceleration data of the three subjects, 15 819 patterns (each of which includes 186 candidate features) are derived and labeled for use in the following tests.

Table 1
Definitions of candidate features.

	Feature name	Definition
	Mean (v)	$\frac{1}{256}\sum_{t=1}^{256} v(t)$
T	*Deviation* (v)	$\sqrt{\frac{1}{256}\sum_{t=1}^{256}(v(t)-Mean)^2}$
D	*Total Power-TD* (v)	$\frac{1}{256}\sum_{t=1}^{256} v(t)^2$
	Axes Correlation (u,v)	$\sum_{t=1}^{256} u(t)v(t)/\sqrt{\sum_{t=1}^{256} u^2(t)\sum_{t=1}^{256} v^2(t)}$
	FFT DC (v)	$\lvert f(0)\rvert$
	Dominant frequency-1 (v)	$\frac{100}{256}\omega^{1)}$, $\omega^{1)} = arg\ peak^{1)}_{\omega=1,\ldots,128}\{p(\omega)\}$
F	*Dominant frequency-2* (v)	$\frac{100}{256}\omega^{2)}$, $\omega^{2)} = arg\ peak^{2)}_{\omega=1,\ldots,128}\{p(\omega)\}$
D	*Dominant power-1*(v)	$peak^{1)}_{\omega=1,\ldots,128}\{p(\omega)\}$
	Dominant power-2(v)	$peak^{2)}_{\omega=1,\ldots,128}\{p(\omega)\}$
	Entropy-FD(v)	$-\sum_{\omega=1}^{128} p(\omega)\log_2 p(\omega)$
	Total Power-FD(v)	$\sum_{\omega=1}^{128} f(\omega)\overline{f(\omega)}$

For comparison, an information gain based feature selection method [52] has also been tested. This algorithm defines the first-order utility (FOU) as the scoring criterion which takes the first-order (pairwise) interaction information into account. This algorithm needs to store all the quantized patterns (integer valued) to generate high-dimensional joint frequency tables. Otherwise frequent data swapping between RAM and external storage (e.g. hard disk or flash memory card) of smart device is very time consuming. Therefore FOU requires much more memory than the proposed algorithm in this application. In comparison, the memory requirement of the proposed algorithm is independent of the number of samples.

Tests are performed for different sensor configurations and the results are compared in Tables 2–6, TSM indicates the results of our two-stage model-based method and FOU indicates the results based on the FOU informatics criterion. Cases 1–4 used sensors 1, 2, 3 and 4 respectively; these where mounted on the left-ankle, right-ankle, left-wrist and right-wrist. Case 5 used all the four sensors. For each of the five cases, subsets of 4, 8, 12, 16 and 20 features are selected from the involved candidates, and then SVMs are trained using the data from selected features. All the 15 819 labeled patterns are used for feature selection, while only one quarter of the patterns (i.e., 3955) are used for training SVMs (with polynomial kernels). It was failed to perform SVM training with more training patterns on the PDA because of its limited memory. The resulted SVMs are then validated over all the 15 819 labeled patterns. The precisions listed in Tables 2–6 are the percentages of the patterns successfully predicted by the trained SVMs over the 15 819 labeled patterns.

For cases 1–4, only the three acceleration components (x, y and z axes) of the involved accelerometer are used, therefore only

Table 2
Test results for case 1: using accelerometer 1 on left-ankle.

Selected feature subset		Feature selection		SVM training		Activity recognition	
		Memory used	Running time (s)	Number of SVs	Running time (s)	Precision (%)	Updating time (ms)
	4	6.4 KB	1191	687	1801	81.52	160
T	8	7.4 KB	1237	641	1912	90.26	160
S	12	8.5 KK	1265	519	2025	99.56	150
M	16	9.5 KB	1303	514	2141	99.61	150
	20	10.6 KB	1358	515	2287	99.66	150
	4	2.16 MB	312	1807	40	83.35	124
F	8	2.16 MB	416	1410	52	90.78	156
O	12	2.16 MB	604	1519	65	90.78	149
U	16	2.16 MB	910	1545	72	92.39	156
	20	2.17 MB	1775	1557	98	92.98	149

Table 3
Test results for case 2: using accelerometer 2 on right-ankle.

Selected feature subset		Feature selection		SVM training		Activity recognition	
		Memory used	Running time (s)	Number of SVs	Running time (s)	Precision (%)	Updating time (ms)
	4	6.4 KB	1201	625	1812	81.26	160
T	8	7.4 KB	1232	557	1924	90.17	160
S	12	8.5 KB	1264	467	2043	99.87	150
M	16	9.5 KB	1302	449	2127	99.87	150
	20	10.6 KB	1354	432	2267	99.88	150
	4	2.16 MB	396	2188	50	79.25	95
F	8	2.16 MB	721	1865	45	86.26	95
O	12	2.16 MB	1092	1980	58	87.28	110
U	16	2.16 MB	2450	1913	58	88.22	158
	20	2.17 MB	2704	1932	102	88.58	130

Table 4
Test results for case 3: using accelerometer 3 on left-wrist.

Selected feature subset		Feature selection		SVM training		Activity recognition	
		Memory used	Running time (s)	Number of SVs	Running time (s)	Precision (%)	Updating time (ms)
	4	6.4 KB	1211	515	1824	80.91	140
T	8	7.4 KB	1243	475	1946	89.28	140
S	12	8.5 KB	1268	388	2035	98.93	130
M	16	9.5 KB	1307	372	2135	99.15	130
	20	10.6 KB	1351	366	2264	99.17	130
	4	2.16 MB	572	2698	98	76.38	156
F	8	2.16 MB	1554	2590	110	78.43	156
O	12	2.16 MB	3224	2395	105	83.98	157
U	16	2.16 MB	3392	2191	130	87.36	169
	20	2.17 MB	3087	2134	124	89.15	162

Table 5
Test results for case 4: using accelerometer 4 on right-wrist.

Selected feature subset		Feature selection		SVM training		Activity recognition	
		Memory used	Running time (s)	Number of SVs	Running time (s)	Precision (%)	Updating time (ms)
	4	6.4 KB	1216	368	1845	80.10	110
T	8	7.4 KB	1241	327	1924	89.18	110
S	12	8.5 KB	1262	249	2035	98.98	100
M	16	9.5 KB	1304	225	2146	99.18	100
	20	10.6 KB	1353	224	2245	99.28	100
	4	2.16 MB	812	2230	102	78.88	156
F	8	2.16 MB	1482	2098	84	84.09	149
O	12	2.16 MB	2551	2144	97	85.88	143
U	16	2.16 MB	2880	2042	110	87.31	158
	20	2.17 MB	4320	2026	117	88.30	162

Table 6
Test results for case 5: using all the four accelerometers.

Selected feature subset		Feature selection		SVM training		Activity recognition	
		Memory used	Running time (s)	Number of SVs	Running time (s)	Precision (%)	Updating time (ms)
	4	149.6 KB	1413	432	1832	80.91	130
T	8	155.5 KB	1452	417	1941	90.25	130
S	12	161.5 KB	1451	331	2025	99.94	120
M	16	167.4 KB	1513	312	2147	99.95	120
	20	173.4 KB	1568	307	2289	99.98	120
	4	11.62 MB	2068	1895	59	83.16	130
F	8	11.62 MB	4498	1708	52	87.88	130
O	12	11.62 MB	6092	1695	58	90.02	131
U	16	11.62 MB	7392	1348	52	95.35	131
	20	11.62 MB	10166	1288	55	96.85	129

three axes correlation features are available, and the total number of candidate features is 33. The term pool includes all the polynomial terms of these 33 candidate features of degrees 1 and 2, resulting in term pools of size $H=594$. For the case 5, there are 186 candidate features for the 12 acceleration signals. The term pool only includes the candidate features and their squares, i.e., $x_i, x_i^2, i=1,2,\ldots,186$, forming a pool of 372 terms.

It should be noted that the FOU algorithm needs to quantize the continuous features and generate the M-dimensional joint frequency distribution table for the M candidate features and marginal frequency tables for different subsets of the M candidate features from the quantized instances. Thus, the full set of feature data with labels for feature selection needs to be loaded into the memory for: (a) the quantization, (b) the generation of pairwise joint frequency tables and (c) conditional frequency tables used during the process of selection in order to rank feature subsets in the FOU information criterion. This requires more memory than is available in the PDA, thus, the FOU feature selection algorithm has been tested in a desktop PC. In Tables 1–5 the execution times have all been normalized to apply to the ARM processor in the PDA.

It is shown in Tables 1–5 that the time taken to select a subset of 4–20 features is about 20–25 min, and to train an SVM takes about 30–40 min. The classification of incoming data is very efficient and can be accomplished in real-time, it takes the SVMs only 100–160 ms to update a prediction.

The tests also confirm that for the activities studied (walking fast, walking slow and browsing(stolling, stopping, starting)), the feature selection algorithm classifies activity with an exceptionally high success rate. It is also interesting to note that the results indicate that there is little advantage to using the data from all four sensors as opposed to a single sensor, with regards to the classification precision (the percentage of successfully predicted patterns as previously defined), given that in real-world applications we usually have only one sensor unit available.

It is also shown in Tables 1–5 that, in all the cases, the proposed algorithm produces sparser SVMs than the FOU algorithm. SVMs produced by the proposed algorithm have only $\frac{1}{3}$ to $\frac{1}{4}$ SVs of that produced by the FOU algorithm. However, SVMs produced by the proposed algorithm have significant higher prediction precision than that produced by the FOU algorithm, except very few cases, particularly for cases of more selected features.

6. Conclusions

A fast two-stage subset selection algorithm is combined with a model-based feature selection method to identify significant features to use in order to classify the activity of a human user who is carrying a suitably equipped mobile device, e.g. phone or PDA.

The proposed method allows a locally optimal subset of features (the pattern) to be determined because certain combinations of features are highly correlated with specific activities (the output, identified by a label).

A full nonlinear regression model is employed to model the relationship between the label and the pattern. A fast two-stage subset selection algorithm is employed to select the sub-model which is locally optimal. The features involved in the selected sub-model are then used as inputs to the classifier.

This feature selection method is particularly efficient in memory usage. Of special note is the fact that, the amount of required memory is independent of the number of training patterns. This property makes this method suitable for devices limited in the amount of available physical memory.

Based on this feature selection method, an application has been developed for mobile devices: PDAs and smart phones, which can perform real-time activity classification using SVMs, and background classier configuration, including feature selection and SVM training. The application has been tested on a HP iPAQ pocket PDA.

Using this case study, the method presented in this paper has been evaluated and compared against an information-gain-based feature selection method from the literature. The test results demonstrated the significant effectiveness and high degree of efficiency of the proposed feature selection method. In particular, since an SVM training algorithm is implemented in this application, it is possible for the user of the mobile device to train the classifier using their own specific movement data. This will have the added advantage of also enhancing the performance and accuracy of the activity classifier since it can be trained specifically for a user's needs.

Compared with that produced by the information-based feature selection algorithm, the SVMs (with polynomial kernels) produced by the proposed fast two-stage feature selection algorithm is much sparser in all the test cases, while have higher prediction precision in most cases, particularly for more features.

The only major drawback is that selection of the features cannot be accomplished in real-time. However this is not significant, since selection of features to use for activity classification are usually predefined in usual application software, to avoid having to require the user to indulge in a complex process of training, before they can use the program.

Compared with the information-based method, the proposed method not only outperforms it by producing more accurate classifications, but also much more memory and resource efficient and so is an ideal candidate for use in mobile and embedded processor applications.

Acknowledgment

This work was part-funded by the European Commission under the Seventh Framework Programme: large-scale integrating project *HaptiMap*, FP7-ICT-224675.

References

[1] G.D. Abowd, A.K. Dey, P.J. Brown, N. Davies, M. Smith, P. Steggles, Towards a better understanding of context and context-awareness, in: HUC '99: Proceedings of the First International Symposium on Handheld and Ubiquitous Computing, Springer-Verlag, London, UK, 1999, pp. 304–307.
[2] B.E. Ainsworth, D.R. Jacobs Jr., A.S. Leon, Validity and reliability of self-reported physical activity status: the Lipid Research Clinics questionnaire, Medicine and Science in Sports and Exercise 25 (1993) 92–98.
[3] A. Blum, P. Langley, Selection of relevant features and examples in machine learning, Artificial Intelligence 97 (1997) 245–271.
[4] K. Aminian, B. Najafi, Capturing human motion using body-fixed sensors: outdoor measurement and clinical applications, Computer Animation Virtual Worlds, vol. 15, 2004, pp. 79–94.
[5] L. Bao, S.S. Intille, Activity recognition from user-annotated acceleration data, in: Pervasive Computing (Lecture Notes in Computer Science, vol. 3001), Springer, Berlin, 2004, pp. 1–17.
[6] N.C. Barengo, G. Hu, T.A. Lakka, H. Pekkarinen, A. Nissinen, J. Tuomilehto, Low physical activity as a predictor for total and cardiovascular disease mortality in middle-aged men and women in Finland, European Heart Journal 25 (2004) 2204–2211.
[7] S.N. Blair, T.R. Collingwood, R. Reynolds, M. Smith, R.D. Hagan, C.L. Sterling, Health promotion for educators: impact on health behaviors, satisfaction, and general well-being, American Journal of Public Health 74 (1984) 147–149.
[8] P. Bradley, O. Mangasarian, Feature selection via concave minimization and support vector machines, in: Proceedings of the Fifteenth International Conference Machine Learning (ICML'98), Morgan Kaufmann, San Francisco, CA, 1998, pp. 82–90.
[9] O. Chapelle, S.S. Keerthi, Multi-class feature selection with support vector machines, Yahoo Research, Technical Report YR-2008-002, 2008.
[10] X. Chen, X. Zeng, D. van Alphen, Multi-class feature selection for texture classification, Pattern Recognition Letters 27 (14) (2006) 1685–1691.
[11] J. Coutaz, J.L. Crowley, S. Dobson, D. Garlan, Context is key, Communication of the ACM 48 (2005) 49–53.
[12] A. Dey, G.D. Abowd, D. Salber, A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications, Human–Computer Interaction 16 (2–4) (2001) 97–166.
[13] A. Famili, W.-M. Shen, R. Weber, E. Simoudis, Data preprocessing and intelligent data analysis, Intelligent Data Analysis 1 (1997) 3–23.
[14] A. Flanagan, J. Mantyjarvi, J. Himberg, Unsupervised clustering of symbol strings and context recognition, in: Proceedings of the IEEE International Conference on Data Mining (ICDM'02), 2002, p. 171.
[15] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, Journal of Machine Learning Research 3 (2003) 1157–1182.
[16] I. Guyon, J. Weston, S. Barnhill, V. Vapnik, Gene selection for cancer classification using support vector machines, Machine Learning 46 (1–3) (2002) 389–422.
[17] T.N. Lal, O. Chapelle, J. Weston, A. Elisseeff, Embedded methods, in: I. Guyon, S. Gunn, M. Nikravesh, L.A. Zadeh (Eds.), Feature Extraction: Foundations and Applications. Studies in Fuzziness and Soft Computing, vol. 207, Springer, Berlin, Heidelberg, 2006, pp. 137–165.
[18] S.J. Preece, J.Y. Goulermas, L.P.J. Kenney, D. Howard, A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data, IEEE Transactions on Biomedical Engineering 56 (3) (2009) 871–879.
[19] Y. Liu, Y.F. Zheng, FS-SFS: a novel feature selection method for support vector machines, Pattern Recognition 39 (2006) 1333–1345.
[20] G. Nemhauser, L. Wolsey, Integer and Combinatorial Optimization, John Wiley and Sons, New York, 1988.
[21] S. Maldonado, R. Weber, A wrapper method for feature selection using support vector machines, Information Sciences 179 (2009) 2208–2217.
[22] J. Miranda, R. Montoya, R. Weber, Linear penalization support vector machines for feature selection, in: S.K. Pal et al.(Ed.), First International Conference Pattern Recognition and Machine Intelligence (PReMI 2005), Lecture Notes in Computer Sciences, vol. 3776, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 188–192.
[23] M.-D. Shieh, C-C. Yang, Multiclass SVM-RFE for product form feature selection, Expert Systems with Applications 35 (1–2) (2008) 531–541.
[24] I. Gheyas, L. Smith, Feature subset selection in large dimensionality domains, Pattern Recognition 43 (2010) 5–13.
[25] A.A. Albrecht, Stochastic local search for the feature set problem, with applications to micro array data, Applied Mathematics and Computation 183 (2006) 1148–1164.
[26] J. Hua, W. Tembe, E.R. Dougherty, Feature selection in the classification of high-dimension data, in: IEEE International Work Shop on Genomic Signal Processing and Statistics, 2008, pp. 1–2.
[27] X. Jin, A. Xu, R. Bie, P. Guo, Machine learning techniques and chi-square feature selection for cancer classification using SAGE gene expression profiles, Lecture Notes in Computer Science 3916 (2006) 106–115.
[28] C. Liao, S. Li, Z. Luo, Gene selection using Wilcoxon rank sum test and support vector machine for cancer, Lecture Notes in Computer Science 4456 (2007) 57–66.
[29] H. Peng, F. Long, C. Ding, Feature selection based on mutualin formation criteria of max-dependency, max-relevance, and min redundancy, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (2005) 1226–1238.
[30] J. Biesiada, W. Duch, Feature selection for high-dimensional data—a Pearson redundancy based filter, Advances in Soft Computing 45 (2008) 242–249.
[31] L. Rocchi, L. Chiari, A. Cappello, Feature selection of stabilometric parameters based on principal component analysis, Medical and Biological Engineering and Computing 42 (2004) 71–79.
[32] J. Yang, V. Honavar, Feature subset selection using a genetic algorithm, IEEE Intelligent Systems and their Applications 13 (1998) 44–49.
[33] X. Wang, J. Yang, X. Teng, W. Xia, J. Richard, Feature selection based on rough sets and particle swarm optimization, Pattern Recognition Letters 28 (2007) 459–471.
[34] M. Ronen, Z. Jacob, Using simulated annealing to optimize feature selection problem in marketing applications, European Journal of Operational Research 171 (2006) 842–858.
[35] K. Li, J.-X. Peng, Neural input selection—a fast model based approach, Neurocomputing 70 (2007) 762–769.
[36] HaptiMap – Haptic, Audio and Visual Interfaces for Maps and Location-based Services, Large-scale Integrating Project, European Commission Seventh Framework Programme, ⟨http://www.haptimap.org/⟩.
[37] Cwiid—A Collection of Linux Tools Written in C for Interfacing to the Nintendo Wiimote, ⟨http://abstrakraft.org/cwiid/⟩.
[38] H. Hagendoorn, I. Vuori, P. Oja, Guidelines for the development of national policies and strategies for promoting health through physical activity, The European Network for the Promotion of Health-Enhancing Physical Activity, (2004) ⟨http://www.who.int/gb/ebwha/pdf_files/WHA57/A57_R17-en.pdf⟩.
[39] M.J. Mathie, A.C.F. Coster, N.H. Lovell, B.G. Celler, Accelerometry: providing an integrated, practical method for long-term, ambulatory monitoring of human movement, Physiological Measurement. 25 (2004) R1–R20.
[40] V.-M. Mantyla, J. Mantyjarvi, T. Seppanen, E. Tuulari, Hand gesture recognition of a mobile device user, in: Proceedings of the International IEEE Conference on Multimedia Expo (ICME), 2000, pp. 281–284.
[41] J. Mantyjarvi, J. Himberg, T. Seppanen, Recognizing human motion with multiple acceleration sensors, in: Proceedings of the International IEEE Conference on Systems, Man Cybernetics (SMC), 2001, pp. 747–752.
[42] J.E. Manson, E.B. Rimm, M.J. Stampfer, G.A. Colditz, W.C. Willett, A.S. Krolewski, B. Rosner, C.H. Hennekens, F.E. Speizer, Physical activity and incidence of non-insulin-dependent diabetes mellitus in women, Lancet 338 (1991) 774–778.
[43] R.R. Pate, M. Pratt, S.N. Blair, W.L. Haskell, C.A. Macera, C. Bouchard, D. Buchner, W. Ettinger, G.W. Health, A.C. King, A. Kriska, A.S. Leon, B.H. Marcus, J. Morris, R.S. Paffenbarger, K. Patrick, M.L. Pollock, J.M. Rippe, J. Sallis, J.H. Wilmore, Physical activity and public health: a recommendation from the centers for Disease Control and Prevention and the American College of Sports Medicine, JAMA 273 (5) (1995) 402–407.
[44] S.N. Patel, J.A. Kientz, G.R. Hayes, S. Bhat, G.D. Abowd, Farther than you may think: an empirical investigation of the proximity of users to their mobile phones, in: Proceedings of Ubicomp, Springer-Veralg, 2006, pp. 123–140.
[45] N. Streitz, P. Nixon, The disappearing computer, Communication of the ACM 48 (2005) 32–35.
[46] R.A. Washburn, H.J. Montoye, The assessment of physical activity by questionnaire, American Journal of Epidemiology 123 (1986) 563–576.

[47] A.K. Yancey, C.M. Wold, W.J. McCarthy, M.D. Weber, B. Lee, P.A. Simon, J.E. Fielding, Physical inactivity and overweight among Los Angeles County adults, American Journal of Preventive Medicine 27 (2004) 146–152.
[48] S.-W. Lee, K. Mase, Activity and location recognition using wearable sensors, Pervasive Computing 1 (3) (2002) 24–32.
[49] K. van Laerhoven, O. Cakmakci, What shall we teach our pants?, in: Proceedings of the Fourth International Symposium on Wearable Computers, 2000, pp. 77–83.
[50] P. Korpipaa, M. Koskinen, J. Peltola, S.-M. Makela, T. Seppanen, Bayesian approach to sensor-based context awareness, Personal and Ubiquitous Computing Journal 7 (4) (2003) 113–124.
[51] D. Abowd Gregory, K. Dey Anind, J. Brown Peter, Davies Nigel, Smith Mark, Steggles Pete, Towards a better understanding of context and context-awareness, in: HUC'99: Proceedings of the First International Symposium on Handheld and Ubiquitous Computing, Springer-Verlag, London, UK, 1999, pp. 304–307.
[52] G. Brown, A new perspective for information theoretic feature selection, in: Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, vol. 5, April 16–18, 2009, Clearwater Beach, Florida, USA, 2009, pp. 49-56.
[53] K. Li, J.-X. Pengm, E.-W. Bai, A two-stage algorithm for identification of nonlinear dynamic systems, Automatica 42 (7) (2006) 1189–1197.
[54] K. Li, J.-X. Peng, G. Irwin, A fast nonlinear model identification method, IEEE Transactions on Automatic Control 50 (8) (2005) 1211–1216.
[55] K.M. Adeney, M.J. Korenberg, Iterative fast orthogonal search algorithm for MDL-based training of generalized single-layer network, Neural Networks 13 (7) (2000) 787–799.

Jian-Xun Peng has over 15 years experience in avionics, nonlinear system modeling and control, artificial neural network, and computer graphics. He has authored over 40 journal and conference papers. He is currently with the School of Electronics, Electrical Engineering and Computer Science at the Queen's University of Belfast where he is researching on application context sensing and developing software for the European Commission-funded "HaptiMap" project.

Stuart Ferguson has over 25 years experience in computer graphics and software engineering. He is the author of the book "Practical Algorithms for 3D computer Graphics" (2001). He is currently a lecturer in the School of Electronics, Electrical Engineering and Computer Science at the Queen's University of Belfast where he is researching on real-time application program implementations for mobile devices.

Karen Rafferty has over 10 years experience working within the fields of image processing, computer vision, and intelligent devices that can perceive their environment and respond to it. Dr. Rafferty has authored over 30 journal and conference papers in the area of intelligent devices, computer vision, image processing and applications of virtual reality. Her first book on Virtual Reality was published in 2007.

Paul D. Kelly received the MEng and PhD degrees in Electrical & Electronic Engineering from the Queen's University of Belfast in 2000 and 2005 respectively. Since then he has developed research interests in geographical data processing and digital audio and video processing, with a particular emphasis on C programming. He is currently employed by Queen's University Belfast as a research fellow, primarily developing software for the European Commission-funded "HaptiMap" project.