Modelling and Detection of User Activity Patterns for Energy Saving in Buildings

Jose Luis Gomez Ortega, Liangxiu Han and Nicholas Bowring

Abstract Recently, it has been noted that user behaviour can have a large impact on the final energy consumption in buildings. Through the combination of mathematical modelling and data from wireless ambient sensors, we can model human behaviour patterns and use the information to regulate building management systems (BMS) in order to achieve the best trade-off between user comfort and energy efficiency. Furthermore, streaming sensor data can be used to perform real-time classification. In this work, we have modelled user activity patterns using both offline and online learning approaches based on non-linear multi-class Support Vector Machines. We have conducted a comparison study with other machine learning approaches (i.e. Linear SVM, Hidden-Markov and K-nearest models). Experimental results show that our proposed approach outperforms the other methods for the scenarios evaluated in terms of accuracy and processing speed.

1 Introduction

Buildings are one of the main contributors to CO_2 of the atmosphere with a 40 % of the total emissions in the UK [1]. It has been noted that occupant presence and behaviour have a significant impact on building energy performance [2], therefore efficient energy management needs to be able to reduce energy consumption costs without neglecting occupant comfort which ultimately can affect productivity [3].

J.L.G. Ortega (✉) · L. Han
School of Computing, Mathematics and Digital Technology,
Manchester Metropolitan University, John Dalton Building, Manchester M1 5GD, UK
e-mail: jose.l.gomez@mmu.ac.uk; jose.l.gomez@stu.mmu.ac.uk

L. Han
e-mail: l.han@mmu.ac.uk

N. Bowring
School of Engineering, Manchester Metropolitan University,
John Dalton Building, Manchester M1 5GD, UK
e-mail: n.bowring@mmu.ac.uk

L. Chen et al. (eds.), *Emerging Trends and Advanced Technologies for Computational Intelligence*, Studies in Computational Intelligence 647,
DOI 10.1007/978-3-319-33353-3_9

Today's building management systems (BMS) are usually operated based on a fixed seasonal schedule, maximum design occupancy assumption, and pre-defined comfort levels (constant) to ensure satisfactory temperatures, ventilations and luminance at all times, but fail to capture dynamic information. This is both costly and inefficient. For example, typical public areas such as office buildings or shopping centres tend to have illuminated all the spaces yet many times all of them are empty. Moreover, depending on which activities occupants might be performing, different systems or devices must be controlled (e.g. a user performing an exhausting physical activity would want to turn off the heating, to decrease the air conditioning temperature or to open a window). This is particularly true in domestic buildings where the activities users might perform are really diverse. Therefore, it is crucial for a BMS proper operation to dynamically incorporate user occupancy and activity information. This information will provide the system with the means to establish the right balance between energy consumption and user satisfaction.

To dynamically capture information about the occupants and their surroundings, wireless sensing techniques have shown great potential, taking advance of the fact that these sensor devices are easy to install, non-intrusive and inexpensive [4]. This explains their increasing popularity of their use for activities related to human behaviour detection tasks in buildings such as energy saving, smart homes or health monitoring. Through sensed information from users and indoor conditions, it is feasible to create a mathematical model, which will process that data and will be capable of recognising human behaviour and ambient parameters [5] in order to manage BMS more effectively (Fig. 1). Machine learning (ML) approaches and probabilistic models are specially suitable for the development of human behaviour pattern modelling as they can successfully extract relevant information from datasets as well as handle human behaviour randomness.

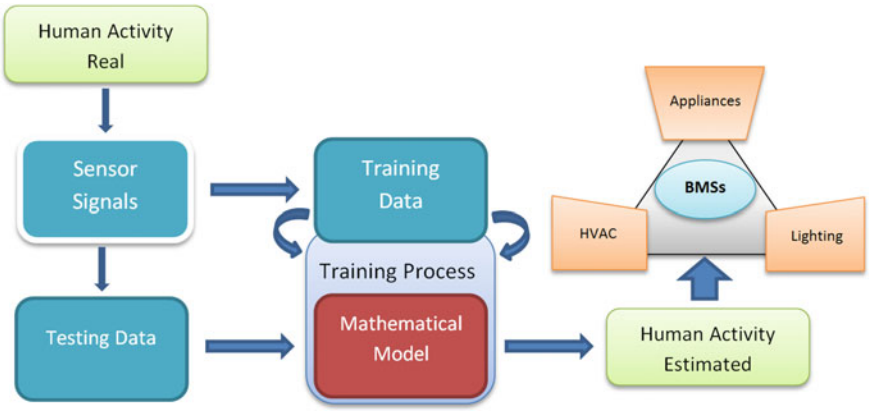

Fig. 1 Human activities trigger sensor events which are recorded and used to train a model. The same sensors also provide the data which will be for model testing and its normal performance. The information we retrieve from the model will help regulate BMSs more efficiently according to real occupant needs

In addition to explore human occupancy patterns from data, sensor information can be collected and introduced into the models as time sequences in a streaming data fashion and used to perform activity detection in real-time. To achieve this, these models need to address some specific issues [6]:

- To be able to handle the continuous flow of data and give a prediction within a certain amount of time.
- To find the best balance between performance and computational costs (time and memory).
- To update the model as new (unbounded) datapoints are being included.

In this work, we have proposed a novel non-linear multiclass Support Vector Machine (SVM) (MCSVM) to perform activities of daily living (ADL) classification in three different scenarios. We have evaluated the performance of our model based on an offline and an online approach and compared our results with other ML techniques. The rest of the paper is organised as follows: Sect. 2 outlines the related work in literature, Sect. 3 gives details about the methodologies adopted in this work, Sect. 4 explains the experiments carried out, and Sect. 5 presents the discussion based on the experimental results and we conclude in Sect. 6 where we also point future lines of research.

2 Previous Work

This section presents related previous works, which covers approaches to human behaviour patterns modelling in buildings and an overview of related works performing real-time user activities classification.

2.1 Occupancy and Activity Models Based on ML and Sensor Data

The exploitation of different ambient sensors to monitor occupancy and ambient conditions for dynamically adaptive energy control and comfort management has initially shown promising results [7, 8]. Various sensors (e.g. temperature or air quality such as CO_2, or a combination of them) were deployed in building scenarios and data was collected and processed for understanding the relationship between occupant behaviour patterns and energy consumption. With the introduction or real-time data collection techniques, new studies demonstrated how 'live' sensing approaches could be used to build data-driven models for occupancy detection [9, 10]. These models however did not consider occupant comfort levels, and could not correctly

predict occupancy (e.g. maximum of 70 % accuracy prediction in [11]). Recently, many authors have opted for SVM as the building block for their model design [9, 11, 12], showing why it is considered one of the best choices for pattern recognition applications.

Singla et al. [13] considered a hidden Markov model (HMM) based algorithm for ADL detection from data acquired by means of a combination of motion, temperature and analogue sensors for water and cooking hobs usage monitoring. The combination of sensors was intended to overcome some of the limitations when using just one sensor as also noted in the work conducted by Dong et al. [9], where another mixture of sensors including motion, temperature, humidity and CO_2 levels were used. Other contributions which addressed ADL estimation from sensors were [14, 15]. All these works pointed the significance of the activity estimation and proposed methods to achieve this by using multi-sensor data and different ML techniques based on supervised learning.

Other authors used k-Nearest Neighbour models, as in Li et al. [16], where they developed a model for occupancy detection in order to build a demand driven heating, ventilation and air conditioning (HVAC) control showing how kNN performed better than other strategies in their application. Hajj et al. [17] and Scott et al. [18] also used a kNN to model user/building interactions.

2.2 Online Modelling Approaches

Human activity data can be analysed to extract conclusions such as typical household occupancy patterns or energy consumption profiles. However, this data can also be used to make decisions in real-time. Models devoted to the improvement of energy efficiency can take advance of streaming sensor data to regulate building systems accordingly at the exact time. The aforementioned works in [9, 11] or in [8, 19] tried to present methodologies which could address this problem and use the information as streaming data to make 'live' human behaviour patterns classification.

Recently, many efforts have been devoted to the development of a kernel SVM which can be updated when partial data is included. This is the case of successful contributions as the passive-aggressive algorithm [20], the incremental SVM [21] or other similar approaches including multi incremental SVM [22]. Most of these approaches are based on the online gradient descent approach to preserve the KKT conditions while updating the model. However, in spite of all the efforts, the actual proposed methods suffer from a computational complexity making the updating process even more costly where the models were trained with the whole datasets [23], which recently made researchers to look into online solutions for less complex models as an alternative [24, 25].

3 The Proposed Methodology

Inspired by the work in [26], the aim of this research is to develop a model to monitor ADLs in buildings more accurately using the same dataset. We have developed two different approaches to perform activity identification as follows:

Offline Approach Based on Non-Linear Multiclass SVM (MCSVM):
We have developed a pattern recognition system to identify human activity based on non-linear multiclass SVM-based approach trained and tested using ambient sensor data from a real world scenario. For the purpose of the comparison study, we have also applied other methods including HMM, KNN and Linear SVM to our case and conducted experimental evaluation.

Online Approach Based on Non-Linear Multiclass SVM (MCSVM):
We have proposed a methodology to perform 'live' detection using SVM approach. This method increases the speed of the classifier training and can handle unbound streaming data more efficiently.

3.1 Data Description

Wireless ambient sensors represent a powerful means to get information from our surroundings in a simple and efficient way with no intrusive issues. Different sensors can be used to monitor various indoor parameters such as motion or temperature which are relevant for the tasks we aim to achieve. Some researchers have made their datasets publicly available (i.e. WSU Casas [15] or DomusLab [27]), which is extremely helpful in two major ways: Firstly, it allows other researchers to use the datasets to conduct their own research. Secondly, it gives researchers a ground truth to evaluate their models and compare the results. In our work we have used the dataset described in [26] for ADL recognition purposes.

3.1.1 Sensors and Labelling

The dataset consists of data divided into days and collected by sensors deployed in three different house scenarios, namely **House A**, **House B** and **House C**. The various sensors used are motion (PIR), pressure (piezo) and contact sensors. As an example, **House A** has a total of 14 sensors and the labels were annotated by the occupants using a bluetooth voice detector device rather than other scenarios in which the activities were just written down on paper. As an example, a list of the sensors used and the different labels for **House A** can be found in Table 1. For more detailed information about the datasets, please refer to: https://sites.google.com/site/tim0306/.

Table 1 Detail on the sensors used and the labels for each activity corresponding to *House A*

Activity number	Activity description	Sensor number	Sensor description
1	Idle*	1	Microwave
2	Leave house	2	Hall-Toilet door
3	Use toilet	3	Hall-Bathroom door
4	Take shower	4	Cups cupboard
5	Brush teeth	5	Fridge
6	Go to bed	6	Plates cupboard
7	Prepare breakfast	7	Frontdoor
8	Prepare dinner	8	Dishwasher
9	Get snack	9	Toilet flush
10	Get drink	10	Freezer
–	–	11	Pans cupboard
–	–	12	Washing machine
–	–	13	Groceries cupboard
–	–	14	Hall-Bedroom door

* Represents the data that is unlabelled

3.1.2 Data Pre-processing

One of the main tasks before data processing is discretisation. The dataset we are using consists of annotations on the sensors fired and the activities performed at an exact time with seconds precision. However, discretising the data in such small amounts of time is inefficient since the sensor sampling rate is of at least a few seconds (since the action that triggered the sensor happened until the moment the information is available in the dataset). Moreover, if we want to use this data for online learning we need to carefully consider the amount on time between samples as we need to perform any updating operations before the new datapoint arrives.

Based on all these reasons, the time granularity will be determined by the machine capacity and the activities we are modelling. For example, if we want to model presence of people in a room for lighting control, we don't want to wait more than an instant of time for the system to operate [5]. But for longer activities such as *sleeping* or *leave home* we can afford a more coarse-grained discretisation. A choice of 60 s per time slot has shown to be fine enough with an acceptable error rate as shown by [28, 29]. For this particular dataset, timeslices of 60 s incur in a performance error below 2 % [26], which proves to be a good trade-off between loss of information and efficiency.

The three different scenarios **House A**, **House B** and **House C**, comprise 25, 14 and 19 days of data respectively. We conducted a pre-processing step on the data in which we created two types of matrices of features. In one of them we created a binary mxn matrix in which the columns n represented each of the sensors and each row m represented each sample. For each day we had 1440 samples (60 min $\times$ 24 h)

instances per day spaced 60 s between them. A 1 in any position would represent the activation of the sensor corresponding to the row (more than one sensor might be active at any point), and the number of columns in which this value is repeated, would represent the time that sensors are activated. The label matrix consisted of an array with a numerical value k ($k = num_{activities}$) for each activity and also the 1440 timeslices per day.

3.2 SVM Based User Activity Pattern Modelling and Detection

In our work, we have proposed to apply non-linear multiclass SVM (MCSVM) for human activity identification through ambient sensor data. The main reasons include (1) the data is non-linear (2) our task is a typical multiclassification task. We have evaluated its performance against three other state-of-the-art ML algorithms namely HMM, kNN and linear SVM techniques.

3.3 Support Vector Machines

The SVM model learns by representing feature points in space and establishing a hyperplane separating the points of different classes. The most representative points in the boundaries are called Support Vectors and their position determines the final plane which will maximise the distance between the mentioned support vectors of different classes. The features will then be separated into regions by the hyperplane and the new points tested will be classified based on their region location. In order to go beyond linear classification, SVM uses the kernel trick method, which consist of modelling feature relations (from input space to feature space) instead of the features themselves, allowing the use of multidimensional hyperplanes. With these hyperplanes, SVM learners can handle multivariate data (such as from occupancy typical datasets with many sensors) which can be thus processed and classified.

We call our data $(\mathbf{x_i}, y_i)$, $i = 1, \ldots m$, where m is the total number of samples, $\mathbf{x_i} \in \chi \subseteq \mathbb{R}^d$ is the input of sensor readings and $y_i \in \{+1, -1\}$ represent each of the labels (multiclass SVM also keeps as will be explained below). This model seeks to learn from data to find a hyperplane function such as:

$$f(x) = \mathbf{w^T x} + b. \tag{1}$$

where $f(x)$ represents the plane as a function of the training samples $\mathbf{x}$, b is the bias term and $\mathbf{w^T}$ is the $weightvector$. In order to learn the model parameters b and $\mathbf{w}$, we want to optimise the function:

$$\frac{1}{2}||w||^2, \quad s.t. \quad y_i(w^T \mathbf{x_i} + b) \geq 1, \tag{2}$$

which is solved by means of a Lagrangian multiplier, yielding the optimisation expression:

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j \mathbf{x_i^T x_j} - \sum_{i=1}^{m} \alpha_i. \tag{3}$$

When (3) is solved using quadratic programming, returns a matrix of α coefficients for each $\mathbf{x_i}$. Substituting the values in α into one of the Lagrangian constraints we can solve the value for $\mathbf{w}$ and for the bias term b.

Also, the coefficients in α can be separated into zero and non-zero values. All the non-zero values in the α matrix will correspond to what we call the support vectors, which are the values in the dataset which determine the final shape of the hyperplane we are learning. For more information about SVM models, we refer the reader to the work in [30].

3.3.1 Kernel Trick and Soft Margin

What we have seen so far about SVM is the linear hard margin version of the SVM. For the soft margin version, which allows to have 'noisy' points within the marginal region, the main optimisation problem (2) changes to:

$$\frac{1}{2}||w||^2 + C \sum_{i=1}^{m} \xi_i, \quad s.t. \quad y_i(w^T \mathbf{x_i} + b) \geq 1 - \xi_i, \xi_i > 0, \tag{4}$$

where the term C regulates the penalty for each point in the margins.

Also, to handle non linear data, the SVM model can be further adapted to map the inputs x into another plane. Changing the input space into an feature space $\Phi(\mathbf{x})$ through a kernel function $K(x_i, x_j)$. So, the function that we will need to learn change to:

$$f(x) = \mathbf{w^T} \Phi(\mathbf{x}) + b. \tag{5}$$

In terms of the kernels used for the *kernel trick*, the most common functions are:

- Linear: $K(x_i, x_j) = x_i^T x_j$,
- Polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + c)^d$ for $\gamma > 0$,
- RBF: $K(x_i, x_j) = e^{(\gamma ||x_i - x_j||^2)}$ for $\gamma > 0$,
- Sigmoid: $K(x_i, x_j) = tanh \gamma x_i^T x_j + c$.

This kernel method enables the SVM to separate non-linear data as a linear model would do. Although this technique significantly increases the flexibility and the potential to successfully generalise with more complex data, it also increases the

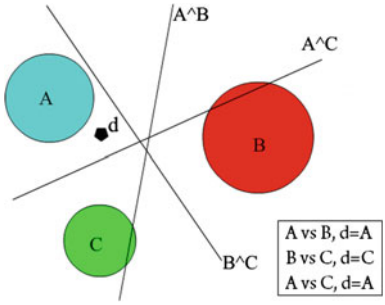

Fig. 2 One versus one multiclass SVM approach. We construct a classifier (hyperplane) for each pair of classes. For prediction, each pair is tested and the most voted option is chosen

complexity of the quadratic programming optimisation, which in turn results in a much more computational intensive task.

3.3.2 Multi-class SVM

The SVM is inherently a model to perform binary classification by nature. However, many of the real-world applications need to perform multi-class classification such as the present one. The SVM function used for our work performs the *one versus one* [31] multiclass inference approach, which has been noted as the most convenient technique for multi classification tasks. The *one versus one* involves creating a different classifier for each pair of classes, evaluating and finally selecting the most recurrent. When two or more classes are tied as the most recurrent, a random one is selected.

As shown in Fig. 2, we see an example of the *one versus one* classification method for three classes (A, B and C), which also needs to learn three SVM classifiers. For a new point d, the pair *A versus B* yields A, the pair *B versus C* gives C and the pair *A versus C* will predict class A. From the results sequence ABA, we predict the new inferred label would be A.

3.4 MCSVM Versus Alternative ML Techniques for Performance Evaluation

We have developed our MCSVM based model on MATLAB using the popular libsvm libraries [32]. All the evaluations have been conducted using a leave-one-day-out k-fold cross validation, where the k parameter corresponded to the length in days of each scenario. All the accuracies reported were calculated using the total of correct predictions over the total of predictions, from the confusion matrix generated. We tested the same data with other three state-of-the-art classification algorithms (i.e. HMM, kNN and LSVM) to compare their performance.

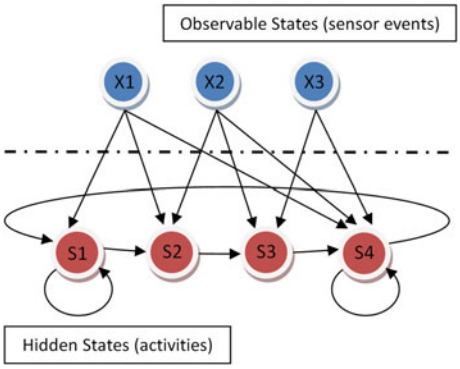

Fig. 3 HMM basic structure. The observable states are the sensor readings and the hidden states that trigger those sensors are the activities. The probability of each state depend on the observations X and the transition probability

3.4.1 Hidden Markov Model

Hidden Markov Models are generative probabilistic models based on the Markov sequences which assume interdependence between the current and the previous occurrence of classes. This usually results in a good way for modelling real events as sequences of states. However, to prevent unmanageable computing loads, this model computes only the probability of being in a determined state for consecutive times-lices (first order Markov chain). The results for this model were produced using the HMM MATLAB function included in the statistical toolbox of the product [33]. A more detailed description on this model can be found in [34]. Figure 3 shows a basic HMM structure.

3.4.2 K-Nearest Neighbours

The kNN model is a simple classification method extensively used for pattern recognition. It merely computes the distance between a new point and its closer neighbours to infer the label of the most repeated label amongst the neighbouring points. The most recurring label will be the chosen for the new point's classification. For this model we used our own kNN function based on the euclidean distance, selecting the number of neighbours which reported better accuracies. A more detailed description of this model can be found in [35].

3.4.3 Linear SVM

The technicalities of this model were covered in previous sections. This linear SVM (LSVM) is a simpler approach than the non-linear SVM as this does not apply the input into feature space mapping. This model has generally shown great potential for classification (unless the data is highly non-linear), and yields results that can challenge those obtained by other more complex models such as MCSVM, but faster

and more efficient. This algorithm is specially indicated when dealing with large quantities of sparse data with high number of features and samples. To apply this approach, we used the liblinear MATLAB libraries [36].

4 Experimental Evaluation

In this section we give details on the experiments we have conducted to evaluate the performance of our two approaches: (1) Offline MCSVM and (2) Online MCSVM.

4.1 Offline MCSVM Performance Evaluation

When using a SVM classifier, we need to select some parameters beforehand, namely a kernel (the different options were discussed above), a degree parameter in the case of polynomial and RBF, and an error coefficient C which will determine the soft margin constraint. We made preliminary experiments to see the MCSVM performance using the most commonly used kernels: linear, polynomial, radial basis and sigmoid. For parameter estimation, we used a grid search of exponential growing values. From the experimental results we can conclude that the best overall kernel to be used with this dataset was the RBF (see Fig. 4), which is also the best choice for the majority of the real world classification tasks.

In any case, MCSVM performed significantly better than the other methods (kNN, LSVM and HMM) evaluated in our experiment. For example, in the *House A*, MCSVM had 84 % of accuracy with much more consistency with some days achieving more than 90 %. Also in *House B*, this model clearly outperforms all the methods before proposed as in Table 2. In the latter scenario *House C* the values reached only

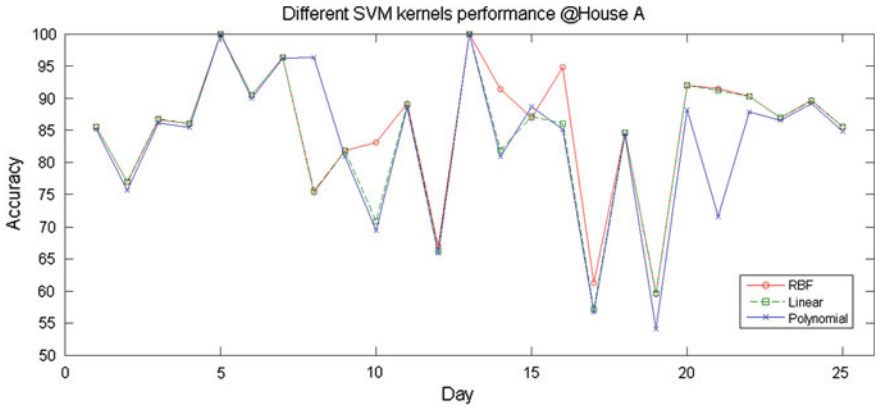

Fig. 4 Example of performance of the MCSVM model using different kernels for each day of the scenario House A

Table 2 Results from the performance evaluation experiment

	MCSVM	LSVM (%)	HMM (%)	KNN	CRF*(%)
House A	85.60	80.63	68.49	73.06	89.80
House B	84.74	73.53	63.51	65.12	78.00
House C	44.57	43.21	36.54	40.23	46.30
Average	71.64	65.79	56.21	59.47	71.37

CRF*: The last column shows the best accuracies reported for the team who made this dataset available from their own experiments

a 44.57 % accuracy, however this is still the best overall option amongst our three methods.

As an alternative for the MCSVM, LSVM obtained good accuracy results which, although still not as accurate as MCSVMs, demonstrated that this approach can have its uses when constraints such as time or memory allocation have an impact on the design of the model. With an average of 65.79 % it is the best option among the rest of the models we evaluated in this experiment.

Regarding the kNN, results were a 73 % average accuracy for *House A* and slightly lower in the other two, which is not a bad performance for such a simple classifier. We conducted the kNN inference by modifying the neighbour parameter until finding the peak performance when 87–91 neighbours were considered. Apart from the typical euclidean, other preliminary measurements for distance where evaluated (i.e. mahalanobis and correlation), yet the variations were too small to be significant and never above the values mentioned. We kept the euclidean approach. in case of the HMM, we solved the 0 emissions/transitions probability by applying a smoothing technique to give a small probability to every transmission and emission even if it never happened throughout the dataset. In our experiment we used the *Pseudoemissions* and *Pseudotransitions* option in the MATLAB's HMM function, although the results obtained using HMM is our experiments were significantly lower than other techniques.

4.2 MCSVM Online Approach

MCSVM model is complex and computationally intensive, and one of the main characteristics an online algorithm has to have is speed and efficiency. We compared the time (Machine: IOS WIN7, @2.40GH, 4GB RAM, MATLAB 2013a) each of our models took to perform the training and testing phases in our previous offline evaluation experiment (see Table 3).

Based on these times, it can be argued that the MCSVM might be too slow to perform real-time predictions.[1]

[1]Note that in this work we have discretised our data in timeslices of 60 s. Therefore, we need to develop an online approach fast enough to be able to incorporate a new point and give an estimated class prediction within 1 min.

Table 3 Time from the performance evaluation experiments

	MCSVM	LSVM	HMM	KNN
House A	420.85	20.58	0.91	36.76
House B	47.31	10.84	0.55	18.55
House C	829.21	31.34	0.744	27.79

The times are expressed in seconds. MCSVM has much higher computational times than the rest

Here, we propose a method that improves significantly the MCSVM training times by removing all non-support vectors from data, which in addition allows to integrate large quantities of new datapoints with a much more constrained growth of parameters.

4.2.1 Online MCSVM Based on Feature Reduction

As discussed in previous sections, one characteristic of the SVM algorithm is that it optimises the maximum margin distance between the hyperplane learned and the datapoints, giving the final function after training based upon just a portion of the total of samples; these are called the support vectors. The major drawback about this SVM approach is to solve the quadratic optimization, which requires a significant amount of memory and time compared to other methods, specially in the case of the non-linear SVM.

To successfully process real-time activity data, we need our algorithm to perform any model update operations to include new datapoints before the next one arrives. Moreover, if we keep on continuously introducing and memorising new points the memory necessary for the data storage will become unmanageable. The real challenge for our MCSVM approach is to be able to process and re-train the model fast and accurately enough to be ready for the next sample and to be able to handle unbounded data.

Here, we have proposed a simple feature reduction of the conventional MCSVM algorithm and we have used this approach to perform online ADL classification. To do this, we based our idea in the fact that the SVM models, once trained, use only a part of the data for their parametrisation called support vectors. As a natural characteristic of the SVM, when the training stage is finished, we can use all the non-zero coefficients from the matrix α obtained after the quadratic optimisation to identify which are the points (SVs) that are actually contributing to the final function. Once these SVs are identified, we can remove the rest of the points from the set and re-train the model with just the previous SVs as training set. The accuracy of the model after being re-trained in this fashion is exactly the same as it was when the whole dataset was used, however the processing times are largely lower (Table 4).

Table 4 Time of the evaluation model experiment for a MCSVM trained with the whole dataset against the same approach but trained with the SVs obtained from the model used with the whole dataset

	MCSVM-Full data	MCSVM-SVs	Num. Datapoints	Num. of SVs
House A	420.85	118.27	35486	9390
House B	47.31	10.14	19968	5756
House C	829.21	334.63	26236	12980

The first two columns compare the training times using the whole dataset or just the selected SVs. The other two columns indicate the number of data-points used for training in each case

4.2.2 Online MCSVM Evaluation

To validate the online approach, we have conducted an experiment in which we trained a MCSVM model using a third of the data (approximately 33 %) for the initial model training. Once the model has been initialised $t = 0$, our algorithm removes all the non-SVs from the data, makes a prediction of the $t + 1$ next label based on the sensor readings. Regardless the model is right or wrong in its prediction,[2] we include the new point in the training set and re-train the model with the previous SVs and the new point. As we have reduced dramatically the number of datapoints, the updating times are faster. Once the model has been re-trained, we test another datapoint and we train again including the new datapoint at $t + 2$ and we keep only the SVs after each re-training. This process continues throughout the test set which includes the rest 66 % on the points. In Fig. 5 we can see an graphical explanation of the process and we have included the pseudocode of our algorithm in Appendix 1.

This method was applied to the three scenarios and the predictions made step by step were compared with a batch testing of the remaining 66 % as a whole. The results of the online approach are much better than the offline evaluation as shown in Figs. 6, 7 and 8.

5 Discussion

Based on our experimental results, the offline MCSVM approach gives the more accurate performance among all the models evaluated, reaching values above 80 % accuracy in two out of three scenarios. In Figs. 9, 10 and 11 we can see how MCSVM outperforms LSVM, HMM and kNN significantly for *House A*, *House B* and *House C*. Comparing with the results reported in [26], in all scenarios SVM accuracies are better than all the previous generative models proposed (NB, HMM and HSMM); and also above the discriminative one (CRF) in the *House B* scenario. When averaging

[2] Our approach also differs from traditional online methods in which we do not apply any penalisation from a loss function, nonetheless this feature could be easily incorporated in the future versions of this algorithm.

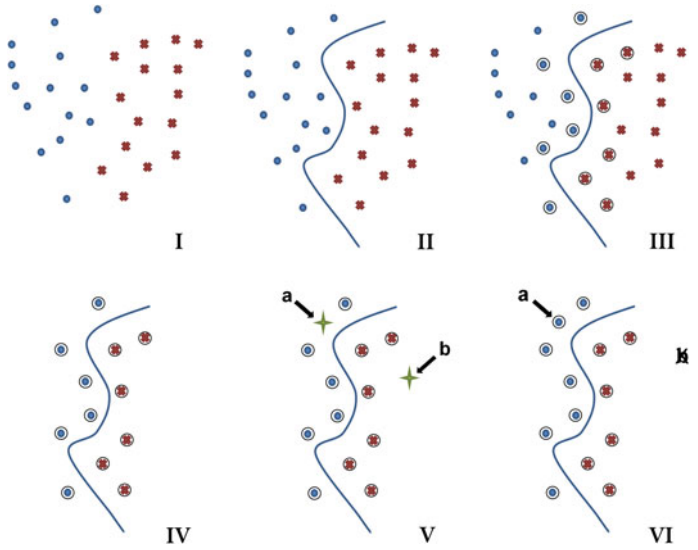

Fig. 5 Removing non-SVs online algorithm → Step I: we process the data. Step II: we train a model to separate the data. Step III: the model created identifies the SVs. Step IV: we keep only the SVs. Step V: we evaluate a new point which can be a potential new SV (**a**) or not (**b**). Step VI: if the new point is identified as a SV, we use it in the next iteration (**a**). If it is not, we discard it (**b**)

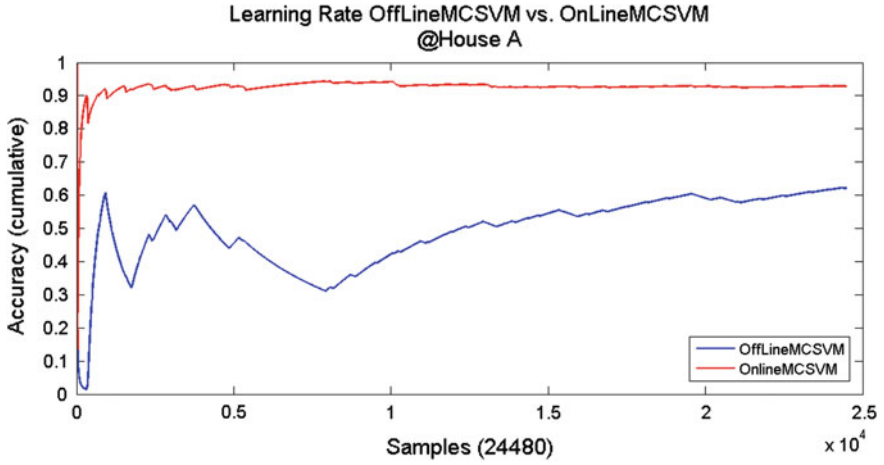

Fig. 6 The online learning versus offline learning in House A after 25000 samples/retraining (66 % of the data)

the results for the three scenarios, SVM presents the best overall result of all the previous models proposed, with a 71.64 %.

Furthermore, we have shown that this MSCSVM approach can be adapted to perform online classification by increasing the model efficiency. Results are promising

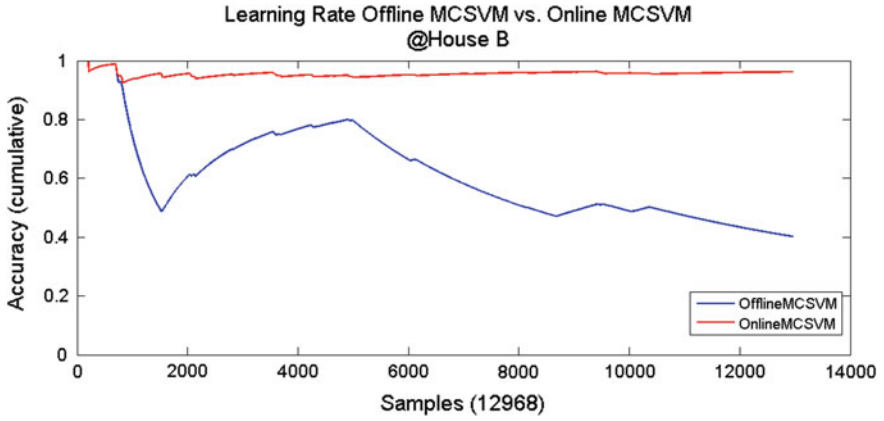

Fig. 7 The online learning versus offline learning in House B after 12000 samples/retraining (66 % of the data)

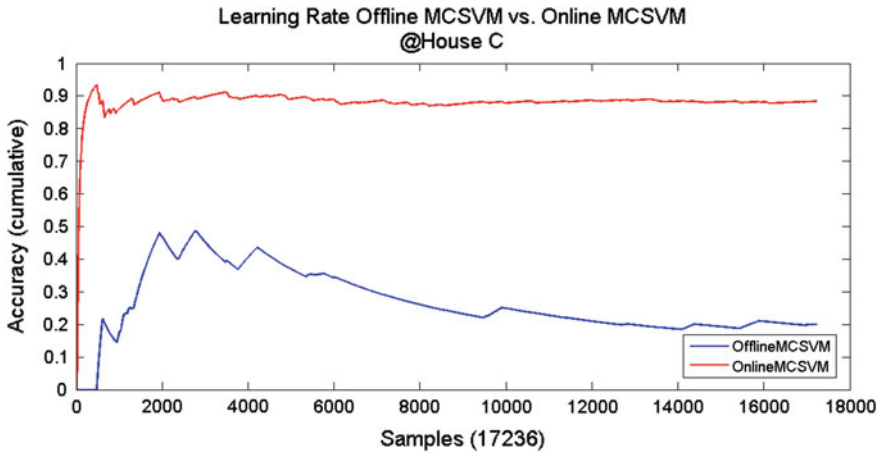

Fig. 8 The online learning versus offline learning in House B after 12000 samples/retraining (66 % of the data)

and have shown great potential, clearly improving the traditional batch approach. An specific constraint in this application was the time, but our approach made each prediction and update iteration under 2 s in House A and B, and in less than 7 s in House C; which means that, for this particular case in which our data is sampled in every 60 s, the model has plenty of time to make its updates and predictions within the time frame.

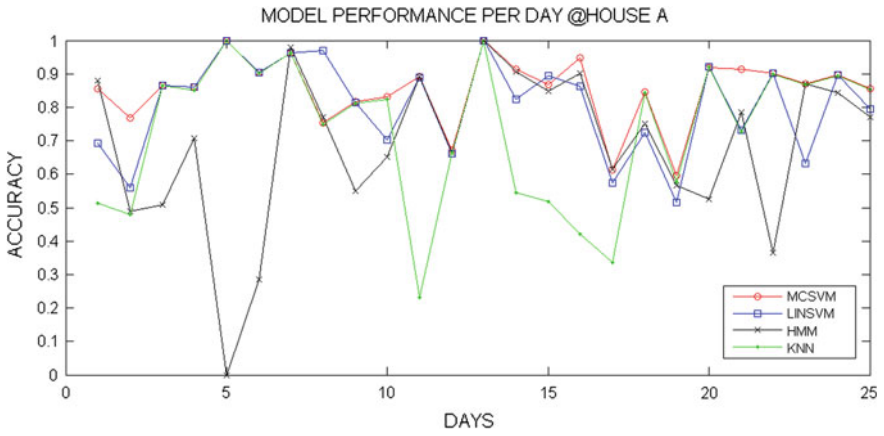

Fig. 9 Model performance @House A

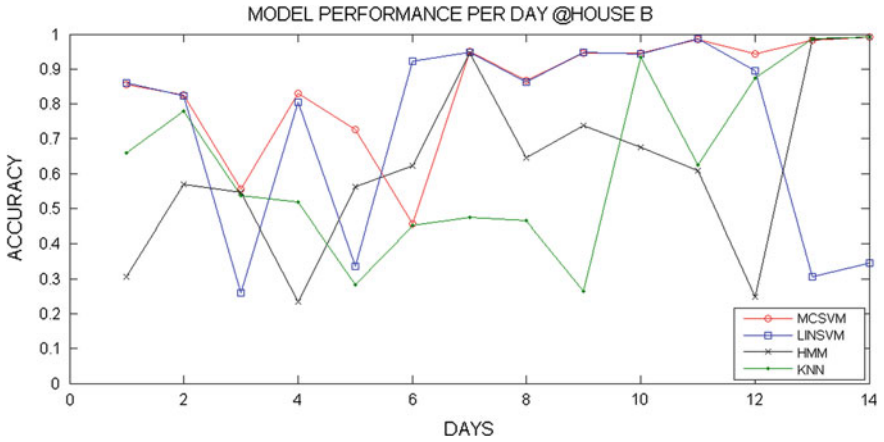

Fig. 10 Model performance @House B

6 Conclusions and Future Work

Accurate occupancy and activity pattern recognition play an important role in energy saving in buildings. This mainly include CO_2 reduction, cost savings and occupant comfort management. In this work, we have modelled user occupancy and activities based on an offline and an online machine learning approaches, using a publicly available dataset to perform activity detection more accurately. We have presented our proposed human activity pattern detection system based on a nonlinear multi-class SVM classifier (MCSVM), which has shown great potential for handling non-linear and complex features. We have conducted a multi-activity estimation based on labelled data collected by means of wireless ambient sensors using this approach. Further-

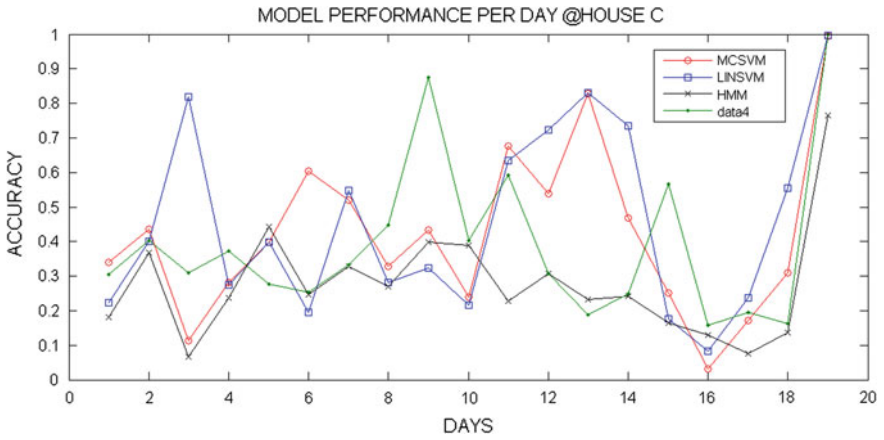

Fig. 11 Model performance @House C

more, we have compared our results with the levels of performance achieved with other state-of-the-art machine learning approaches (i.e. LSVM, HMM and kNN). The MCSVM model has achieved results over 80 % accuracy in two of the three scenarios, significantly improving classification accuracy. In our online approach, we have trained the initial model with a third of the available data, and we have made predictions for the remaining two thirds of the data both in a batch and in an online fashion. Results show that our method achieves high standards of performance compared to the batch approach. Also, each iteration of the online approach is made in a few seconds so this method can be used for fast sampling rates of data acquisition.

Further work can be done to assess and quantify how pre-processing techniques may affect the final results for a model given, and also how different sampling criteria might modify the performance and help finding the best trade-off between overall and low repetitive class prediction performance. In the case of the online approach, the training samples still tend to grow while new data is incorporated (around a 5, 10 and 16 % of the new datapoints have been added to the initial training data for each scenario) but this could be addressed by limiting the number training points if memory becomes a constraint after a certain number of iterations. Also, larger datasets could be used to study if the increasing number of suport vectors reach a saturation point when all the points available are able to represent the target function accurately.

Acknowledgments This work has formed a part of the funded project "Occupancy Pattern Detection for Energy Efficiency and Comfort Management in Buildings Based on Environmental Sensing", funded by Manchester Metropolitan University.

Appendix 1

Data is divided into 2 parts. First part is used for $trainLabels$ and $trainFeatures$.
The rest is used for $testLabels$ and $testFeatures$.
Initialisation: $SVMmodel := SVM\ training\ (trainLabels,\ trainFeatures)$.
$trainFeatures \leftarrow SVpoints, trainLabels \leftarrow SVlabels$; SVs from the previously trained model become the training set.
for i:=1 to m; where m is the number of testing samples, **do**
$activityprediction_i := SVM\ prediction\ (testLabels_i;\ testFeatures_i;\ SVMmodel)$.
$trainLabels \leftarrow trainLabels + testLabels_i$.
$trainFeatures \leftarrow trainFeatures + testestLabels_i$.
$SVMmodel := SVM\ training\ (trainLabels;\ trainData)$;
$trainFeatures \leftarrow SVpoints$ & $trainLabels \leftarrow SVlabels$, we update the datapoints with the SVs from the last SVM model.
end for

References

1. Gov.uk.: Energy performance of buildings-improving the energy efficiency of buildings and using planning to protect the environment-policies-gov.uk (2012). https://www.gov.uk/government/policies/improving-the-energy-efficiency-of-buildings-and-using-planning-to-protect-the-environment/supporting-pages/energy-performance-of-buildings
2. Nguyen, T.A., Aiello, M.: Energy intelligent buildings based on user activity: a survey. Energy Build. **56**, 244–257 (2013)
3. Jazizadeh, F., Marin, F.M., Becerik-Gerber, B.: A thermal preference scale for personalized comfort profile identification via participatory sensing. Build. Environ. **68**, 140–149 (2013)
4. Teixeira, T. Dublon, G., Savvides, A.: A survey of human-sensing: methods for detecting presence, count, location, track, and identity. ACM Comput. Surv. **V** (2010)
5. de Dear, R.J., Akimoto, T., Arens, E.A., Brager, G., Candido, C., Cheong, K.W.D., Li, B., Nishihara, N., Sekhar, S.C., Tanabe, S., Toftum, J., Zhang, H., Zhu, Y.: Progress in thermal comfort research over the last twenty years. Indoor Air **23**(6), 442–461 (2013)
6. Gaber, M.M., Zaslavsky, A., Krishnaswamy, S.: Mining data streams: a review. SIGMOD Rec. **34**(2), 18–26 (2005). http://doi.acm.org/10.1145/1083784.1083789
7. Dong, B., Andrews, B.: Sensor-based occupancy behavioral pattern recognition for energy and comfort management in intelligent buildings. In: Proceedings of Building Simulation, pp. 1444–1451 (2009)
8. Han, H., Jang, K., Han, C., Lee, J.: Occupancy estimation based on co2 concentration using dynamic neural network model. aivc.org (2013)
9. Dong, B., Andrews, B., Lam, K.P., Höynck, M., Zhang, R., Chiou, Y.-S., Benitez, D.: An information technology enabled sustainability test-bed (ITEST) for occupancy detection through an environmental sensing network. Energy Build. **42**(7), 1038–1046 (2010)
10. Ekwevugbe, T., Brown, N., Pakka, V., Fan, D.: Real-time building occupancy sensing using neural-network based sensor network. In: 2013 7th IEEE International Conference on Digital Ecosystems and Technologies (DEST), pp. 114–119 (2013)
11. Mamidi, S., Chang, Y.H., Maheswaran, R.: Improving building energy efficiency with a network of sensing, learning and prediction agents. In: International Conference on Autonomous Agents and Multiagent Systems (AAMAS) (2012)

12. Howard, J., Hoff, W.: Forecasting building occupancy using sensor network data. In: Proceedings of the 2nd International Workshop on Big Data, Streams and Heterogeneous Source Mining Algorithms, Systems, Programming Models and Applications—BigMine '13, pp. 87–94 (2013)
13. Singla, G., Cook, D.J., Schmitter-Edgecombe, M.: Recognizing independent and joint activities among multiple residents in smart environments. J. Ambient Intell. Hum. Comput. **1**(1), 57–63 (2010)
14. van Kasteren, T., Englebienne, G., Kröse, B.J.A., van Kasteren, T.L.M.: Transferring knowledge of activity recognition across sensor networks. Pervasive Comput. (2010)
15. Cook, D., Crandall, A., Thomas, B., Krishnan, N.: CASAS: A smart home in a box. Computer (2013). http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3886862/
16. Li, N., Calis, G., Becerik-Gerber, B.: Measuring and monitoring occupancy with an RFID based system for demand-driven HVAC operations. Autom. Constr. **24**, 89–99 (2012)
17. Hajj, H., El-Hajj, W., Dabbagh, M., Arabi, T.: An algorithm-centric energy aware design methodology. In: IEEE Transactions on Very Large Scale Integration (VLSI) Systems (2014)
18. Scott, J., Brush, A.B.: PreHeat: controlling home heating using occupancy prediction. In: Proceedings of the 13th International Conference on Ubiquitous Computing, pp. 281–290 (2011)
19. Krishnan, N., Cook, D.: Activity recognition on streaming sensor data. In: Pervasive and Mobile Computing (2014). http://www.sciencedirect.com/science/article/pii/S1574119212000776
20. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. J. Mach. Learn. Res. **7**, 551–585 (2006). http://dl.acm.org/citation.cfm?id=1248547.1248566
21. Cauwenberghs, G., Poggio, T.: Incremental and decremental support vector machine learning. In: Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference, vol. 13, p. 409 (2001)
22. Karasuyama, M., Takeuchi, I.: Multiple incremental decremental learning of support vector machines. Trans. Neur. Netw. **21**(7), 1048–1059 (2010). http://dx.doi.org/10.1109/TNN.2010.2048039
23. Kivinen, J., Smola, A., Williamson, R.: Online learning with kernels. Trans. Signal. Process. **52**(8), 2165–2176 (2004). http://dx.doi.org/10.1109/TSP.2004.830991
24. Tsai, C.-H., Lin, C.-Y., Lin, C.-J.: Incremental and decremental training for linear classification. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD '14, pp. 343–352. ACM, New York, NY, USA (2014). http://doi.acm.org/10.1145/2623330.2623661
25. Karampatziakis, N., Langford, J.: Importance weight aware gradient updates. CoRR abs/1011.1576 (2010). http://arxiv.org/abs/1011.1576
26. van Kasteren, T., van Kasteren, T.L.M.: Human activity recognition from wireless sensor network data: benchmark and software. Activity Recogn. Pervasive Intell. Environ. **4**, 165–186 (2011)
27. Gallissot, M., Caelen, J., Bonnefond, N., Meillon, B., Pons, S.: Using the Multicom Domus Dataset. LIG, Grenoble, France, Research Report RR-LIG-020 (2011)
28. Hoque, E., Stankovic, J.: AALO: Activity recognition in smart homes using Active Learning in the presence of Overlapped activities. In: Proceedings of the 6th International Conference on Pervasive Computing Technologies for Healthcare, pp. 139–146. IEEE (2012)
29. Alemdar, H., van Kasteren, T., Ersoy, C., van Kasteren, T.L.M.: Using active learning to allow activity recognition on a large scale. Ambient Intell., 105–114 (2011)
30. Burges, C.: A tutorial on support vector machines for pattern recognition. Data Mining Knowl. Discov. **43**, 1–43 (1998). http://link.springer.com/article/10.1023/A:1009715923555
31. Pal, M.: Multiclass approaches for support vector machine based land cover classification (2008). arXiv:0802.2411
32. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**, 27:1–27:27 (2011). http://www.csie.ntu.edu.tw/~cjlin/libsvm

33. MATLAB and Statistics Toolbox Release: The MathWorks Inc. Natick, Massachusetts, United States (2012)
34. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE **77**(2), 257–286 (1989)
35. Indyk, P., Motwani, R.: Approximate nearest neighbors: Towards removing the curse of dimensionality. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, ser. STOC '98, , pp. 604–613. ACM, New York, NY, USA (1998). http://doi.acm.org/10.1145/276698.276876
36. Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., Lin, C.-J.: LIBLINEAR: a library for large linear classification. J. Mach. Learn. Res. **9**, 1871–1874 (2008)