Sensor-Based Activity Recognition with Dynamically Added Context

Jiahui Wen∗‡, Seng W. Loke⁎, Jadwiga Indulska∗‡, Mingyang Zhong∗‡
∗The University of Queensland, School of Information Technology and Electrical Engineering, Australia
‡National ICT Australia (NICTA)
⁎Department of Computer Science and Computer Engineering, La Trobe University, Australia
j.wen@uq.edu.au, s.loke@latrobe.edu.au, jaga@itee.uq.edu.au, m.zhong1@uq.edu.au

ABSTRACT

An activity recognition system essentially processes raw sensor data and maps them into latent activity classes. Most of the previous systems are built with supervised learning techniques and pre-defined data sources, and result in static models. However, in realistic and dynamic environments, original data sources may fail and new data sources become available, a robust activity recognition system should be able to perform evolution automatically with dynamic sensor availability in dynamic environments. In this paper, we propose methods that automatically incorporate dynamically available data sources to adapt and refine the recognition system at run-time. The system is built upon ensemble classifiers which can automatically choose the features with the most discriminative power. Extensive experimental results with publicly available datasets demonstrate the effectiveness of our methods.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

activity recognition, extra context, activity adaptation

1. INTRODUCTION

Sensor-based activity recognition has experienced its wide application in context-aware computing in the past decade, due to the important role it plays in everyday life. To name a few, recognizing human lifestyle can help to evaluate energy expenditure [1]; monitoring human activity in smart homes enables just-in-time activity guidance provisioning for elderly people and those suffering from cognitive deficiencies [4]; detecting walk and counting step can help to monitor elderly health [3].

State of the art activity recognition models usually rely on a static model, where only pre-defined data sources are considered while opportunistically available contexts which may potentially refine the systems are ignored. Here we argue that dynamically discovered context is also significant for the adaptation and refinement of activity models. For example, in [31], the authors demonstrate that additional features such as vision features can help to improve the recognition accuracy for human activities, especially for static activities (e.g. sitting). Maekawa et al. [15] show in their work that, contextual information, such as the objects that the subjects interact with and the sound during the interaction, captured by camera and microphone can help to improve activity recognition performance. Extensive works prove that additional information such as location information [17], vital signs [11], readings from thermal sensor [6] and barometer [18] can also improve activity recognition accuracy.

Note that all the aforementioned extra data sources are specific to the post-deployment environment. Therefore, considering all the contextual information at the beginning of activity modelling is infeasible, due to the problem of data sparsity and the changes in the environment during post-deployment. Another motivation for our work is that sensors deployed for activity sensing are constantly broken and updated [14], so it is extremely important that the activity monitoring system can automatically evolve with the changing environment. Our work is inspired by [7], where the authors propose an autonomic context management system which is able to populate dynamically discovered contextual information sources for automatic context provisioning. We state here that several challenges need to be addressed in order to achieve an activity recognition system that is able to incorporate dynamically discovered context. First, incorporating new data sources would change the feature dimensionality, the pre-learned activity model should be flexible enough to allow for increment and decrement of the feature dimensionality. Second, the system should be able to automatically identify the context that have the discriminative power, while ignore those with marginal discriminative power. Furthermore, as model refinement with dynamically available context usually requires the labels of the new examples to point out the direction of model adaptation, asking the user for the true labels is obtrusive. Therefore, selecting

.

MOBIQUITOUS 2015, July 22-24, Coimbra, Portugal
Copyright © 2015 ICST
DOI 10.4108/eai.22-7-2015.2260164

the most profitable and informative examples for adaptation is still challenging.

In this paper, we propose such an activity recognition system that addresses the aforementioned challenges. We practically analyze and choose a machine learning model that is flexible with the change of feature dimensionality and can automatically identify the most discriminative features. In order to retrain and adapt the activity model by incorporating the information provided by dynamically discovered data sources, we propose a method to choose the profitable examples without human intervention. Finally, we exploit temporal patterns of human behaviour and leverage graphical models to further improve the recognition performance. To conclude, this paper makes the following contributions.

1. We propose an activity recognition framework that can automatically incorporate dynamically discovered discriminative contexts, so as to improve activity recognition performance.

2. We propose a method that chooses the profitable and informative examples (incorporating discovered context) to retrain and adapt activity models without human intervention. We also propose a novel way of combining basic classifier (i.e., AdaBoost) with graphical models (i.e. Hidden Markov model and Conditional Random Field) in order to exploit the temporal information to improve the recognition accuracy.

3. We demonstrate our system with three publicly available datasets and analyze its effectiveness through comprehensive experimental and comparison studies. We also investigate the conditions under which the opportunistically discovered context is beneficial to recognition performance.

It should be noted that in this paper, we choose inertial sensor (i.e. accelerometer, gyroscope) data as an example, but our methods can be easily extended to other data sources, as we do not make assumptions on the sensor data type, so any kind of sensors (e.g. inertial sensors, binary sensors, microphone, camera) can be used because the AdaBoost approach adopted in this paper can deal with both numeric and categorical features [13]. In addition, we do not distinguish the concepts of new data sources, new features and new contexts. Since new data source and context can be seen as dynamically discovered information from the viewpoint of the whole system, while feature is from the viewpoint of the classifier. The remainder of this paper is organized as follows. In Section 2, we discuss related work. In Section 3, we briefly describe the system overview and architecture of our activity model, and detail each component in Section 4. Section 5 reports the experimental results and analysis, followed by Section 6 where we conclude this paper with a summary.

2. RELATED WORK

Activity recognition [27, 26] is not a new topic, especially with the proliferation of smartphones where on-board sensors such as GPS, camera, microphone, accelerometers and gyroscope, provide unprecedented opportunities for recognizing wide variety of human behaviours [10]. However, most of the state-of-the-art activity models are built upon static machine learning models, the reader is referred to [28] for more details.

Considering new context in dynamic environments to retrain and refine the activity model relates to model personalization and semi-supervised learning from the viewpoint of operation. Activity personalization adapts the general model to a specific user giving his/her data, while semi-supervised learning trains recognition models with labeled and unlabeled data. To name a few, Zhao et al. [32] propose a cross-people activity recognition algorithm for personalized activity-recognition model adaptation by integrating a decision tree and the k-means clustering algorithm. The predictions given by decision tree are re-organized by K-means, based on which the decisive thresholds in the tree are re-estimated. In [16], the authors train a classifier for each user. The ensemble classifiers are then weighted based on the error they make using the target user's data. While in the semi-supervised area, unlabelled examples classified with high confidence are added to the training dataset to retrain and refine the model. Examples are self-training, co-training [21] and label propagation [20]. The problem of aforementioned methods is that only high-confidence examples are considered, due to the fact that they can minimize the entropy [5]. However, high-confidence examples are less informative and make less contribution to the convergence of the model [21]. More importantly, those methods are built with statically defined input and are not suitable to cope with emerging context in dynamic environments.

Some other work leverage the knowledge-based method to deal with unseen data sources for activity recognition. For example, Tapia et al. [23] address the problem of *model incompleteness* by leveraging external knowledge base to measure the similarity between unseen features (object) and existing features, so that they are able to obtain the probability of an unseen object given the activity classes. While in [25], the authors perform activity recognition based on the object usage and human actions. With no label for the action data, they use common sense knowledge to build an activity model by jointly training Dynamic Bayesian Network and Virtual AdaBoost. Those methods, however, rely heavily on existing knowledge to activity recognition. In this light, they are not applicable in the situation that we have no prior knowledge about dynamically discovered data sources.

Other research even perform activity recognition with dynamic sensor selection. For example, in [8], the authors generate multiple processing plans for the context to be monitored. The system dynamically updates the processing plans when sensors are newly registered or de-registered. In another work, Zappi et al. [30] introduce a scheme to dynamically select the sensor set for activity recognition in order to achieve the trade-off between accuracy and power. Since those work mainly focus on the aspect of energy-efficiency, they simply train each activity with all the available sensors, so that when the sensors are registered at runtime, the system already has the knowledge of how to post-process the sensor data, hence this limits the scalability of the system.

3. FRAMEWORK

In this section, we will introduce our framework. The workflow of our system can be divided into three phases: *modelling*, *learning to adapt* and *online prediction*. In the *modelling* phase, an initial activity model is built with currently available sensor data. As new data sources become dynamically available, we perform adaptation for the activity model by considering the dynamic data sources in the *learning to adapt* phase. In the *prediction* phase, the initial model is combined with graphical models to exploit the temporal information to further improve the recognition per-

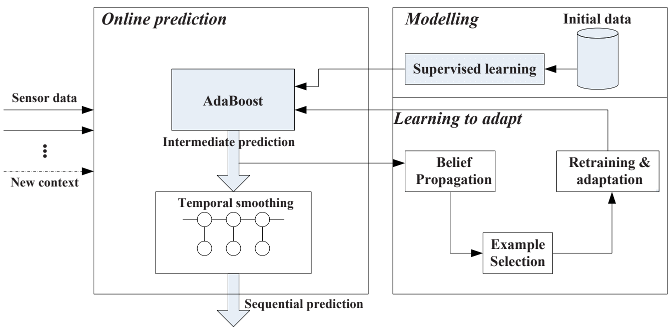

Figure 1: Top level framework.

formance. It should be noted that *prediction* is not the final stage. Instead, our system can keep looping between *learning to adapt* and *prediction* as long as discriminative context is discovered.

Modelling. We choose AdaBoost as our basic classifier, as it is lightweight enough for on-body devices and has been demonstrated to be robust for classification tasks [9]. The rationale for choosing AdaBoost also lies in the fact that it is flexible in the dimension of the feature space, and is able to automatically choose the most discriminative features in the training process.

Learning to adapt. When new data sources are dynamically discovered(the data sources can be discovered universally with sensor modelling, the reader is referred to [7] for more details), the information they provide may be beneficial to improving the recognition accuracy. The goal of this stage is to perform adaptation for the activity models, so as to incorporate the information provided by the new data source (if it is discriminative enough). To achieve this, we perform belief propagation on the predictions given by AdaBoost and choose examples for retraining. The selected examples, which contain newly discovered context, are fed into AdaBoost to retrain and adapt the classifier.

Prediction. AdaBoost makes prediction individually and assumes no dependency between the posterior probability of neighbouring examples. We combine AdaBoost with graphical models to provide sequence predictions, as those models make temporal assumptions between adjacent predictions and are able to smooth out the outliers.

4. METHODOLOGY

4.1 Basic modelling

Because of the special characteristics that meet out requirements, AdaBoost is selected as our basic classifier. The core of AdaBoost is to train an ensemble of weak classifiers and combine them to form a more robust and accurate classifier. Each weak classifier makes decision based on a single feature and needs only be slightly better than random guessing. The final classifier is a linear combination of the weak classifiers, with each classifier weighted by the error it makes during the training process; more weight is given to the classifier that makes fewer errors.

As AdaBoost incrementally builds weak classifiers on the training dataset, it is more flexible in the dimensional changes of the feature space. When discriminative context is detected during the *learning to adapt* phase, all AdaBoost has to do is training a weak learner on the context and add it to the ensemble along with its weight, without the necessity to change the feature space and retrain the whole model. Also, in each iteration, AdaBoost only chooses the weak learner with minimum training error. In this light, it presents an effective and tractable way to automatically select the features with maximum discriminative power [12]. Therefore, it is not necessary to evaluate the discrimination of the new context manually.

As depicted in Algorithm 1, the AdaBoost learning algorithm takes as input the examples, the initial example weights and maximum iterations. The training of AdaBoost follows an iterative process. In each iteration, each weak learner is fitted to training dataset, and the one with the minimum weighted error is chosen (step 2). After that, the example weights are updated, so that more weights are given to the misclassified examples (step 4). During the next iteration, the weak classifiers will focus more on those problematic examples. The output of the training process is an ensemble of weak learners (step 6). Notice that in step 2, it trains a weak learner for each dimension of the feature space, but only selects the one with minimum weighted error. In this paper, we adopt decision stump as the weak learner, and then training weak learner $h_t^k(x)$ for dimension k is equivalent to finding the threshold θ_k in that dimension to minimize the weighted error such that $h_t^k(x_i) = h_t^k(x_i^k) = 1$ if $x_i^k > \theta_k$ and $h_t^k(x_i) = -1$ otherwise, where x_i^k is the value of k^{th} dimension of example x_i.

Algorithm 1 AdaBoost.

```
Input:
    Examples $(x_1, y_1), \cdots, (x_n, y_n)$ where $x_i \in \Re^k$ a is k-dimension feature vector, $y_i \in \{+1, -1\}$ ;
    Initial weight of n examples $D_0(i) = 1/n$ for $i = 1, \cdots, n$;
    Weak learners $h(x) \in \{+1, -1\}$;
    Max iterations $T$;
Output:
    Ensemble of weak learners;
1: for $t = 1$ to $T$ do
2:     Find weak learner $h_t(x)$ that minimizes the weighted error:
       $h_t(x) = argmin_{h_t^k(x)} \sum_{i=1}^n D_t(i) I[h_t^k(x_i) \neq y_i]$
       $\epsilon_t = \sum_{i=1}^n D_t(i) I[h_t(x_i) \neq y_i]$ ;
3:     Compute the weight for the weak learner $h_t(x)$: $\alpha_t = \frac{1}{2} ln(\frac{1-\epsilon_t}{\epsilon_t})$;
4:     Update the weight of examples: $D_{t+1}(i) = \frac{D_t(i) exp(-\alpha y_i h_t(x_i))}{\sum_i D_t(i) exp(-\alpha y_i h_t(x_i))}$ for $i = 1, \cdots, n$;
5: end for
6: return $H(x) = sign(\sum_{t=1}^T \alpha_t h_t(x))$;
```

AdaBoost is a discriminative classifier, and it performs classification by giving the definitive decision. This approach has a potential problem that even if the classifier is uncertain with the class of the example, it chooses the class against which the example has the maximum evidence as the prediction. We argue that the posterior probability of an example is much more helpful, since it reflects the confidence in that prediction. This is important to the later stages such as the stage of *learning to adapt.* To this end, we calculate the posterior probability for examples using the method from [12].

$$P(y_i|x_i) = \begin{cases} \frac{e^{\psi m(x)}}{e^{\psi m(x)}+1} & \text{if } y_i = +1 \\ \frac{e^{-\psi m(x)}}{e^{-\psi m(x)}+1} & \text{if } y_i = -1 \end{cases} \quad (1)$$

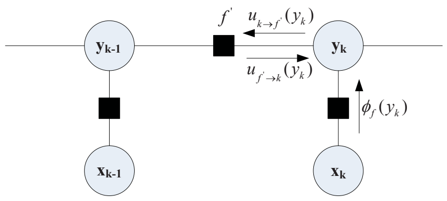

Figure 2: Belief propagation between hidden variable

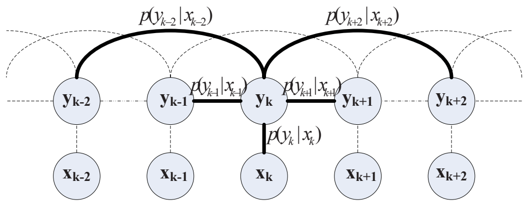

Figure 3: Belief propagation in our scenario. The solid lines show the messages received by node k from neighbouring four nodes.

where ψ is a constant and $m(x) = \frac{\sum_{t=1}^{T}\alpha_t h_t(x)}{\sum_{t=1}^{T}\alpha_t}$. $P(y_i|x_i)$ is thus regarded as the posterior distribution of example x_i. Notice that the binary AdaBoost can be easily extended to multi-class classifiers by training a set of weak learners for each activity class i to separate itself from others:

$$H_i(x) = \sum_{t=1}^{T} \alpha_t^i h_t^i(x) \tag{2}$$

Accordingly, the prediction is made by $argmax_i(H^i(x))$ for a given example x.

4.2 Belief propagation

As new sensors are dynamically discovered, we need to select examples that contain the new sensor data to adapt AdaBoost. The aim in this stage is to leverage belief propagation to smooth the outliers and rectify the results produced by AdaBoost, so as to choose the most profitable and informative examples to learn the new context and adapt the activity model.

Due to the temporal characteristic of human behaviours, the current activity is more likely to be continued in the next time point than a new one. Therefore, there are strong correlations among the sequential predictions of the examples. It is apparent that AdaBoost makes no use of the temporal information, since it assumes no dependencies among the examples, and performs classifications based solely on the local features. As a result, sensor noises or temporary interruption of the activities would certainly result in misclassification.

Belief propagation is mainly performed for inference in graphical models, and in the form of message passing between the nodes. The passing messages among the nodes are actually exerting influence from one variable to the others. In this light, the belief propagation is to send messages to the connected node and tell it what it should believe [29], and the hidden state of a node depends on not only local observations, but also the product of all incoming messages from locally connected nodes.Upon convergence, the marginal distribution of the variable nodes can be approximated with:

$$p(y_k|\boldsymbol{X}) = \frac{\phi_f(y_k)\prod_{f'\in N(k)\setminus f}\mu_{f'\to k}(y_k)}{\sum_{y'_k}\phi_f(y'_k)\prod_{f'\in N(k)\setminus f}\mu_{f'\to k}(y'_k)} \tag{3}$$

where $\phi_f(y_k)$ is the local evidence, and $\mu_{f'\to k}(y_k)$ is the message from neighbouring factor nodes for node k, as shown in Figure 2.

In our scenario, the belief propagation is performed among the observation nodes and hidden nodes. The observation node at time t is the feature vector collected from the sensor data while the hidden node is the latent activity. Since the latent activity is unknown, the latent variable y_k is represented in the form of a multinomial distribution over all the activities. The multinomial distribution is iteratively updated by incorporating the messages from not only local observations, but also adjacent nodes.

In our system, we only consider pairwise connections (Figure 3) between the hidden nodes when performing belief propagation. Therefore, the messages that a node receives are the posterior probabilities of its neighbouring nodes based on their own local observations, as shown in (4)

$$p(y_k|\boldsymbol{X}) = \frac{p(y_k|x_k)\prod_{i\in N(k)\setminus i:y_i=y_k}p(y_i|x_i)}{\sum_{y'_k}p(y'_k|x_k)\prod_{i\in N(k)\setminus i:y_i=y'_k}p(y_i|x_i)} \tag{4}$$

Therefore, belief propagation is performed with an inference step and followed by several iterative update steps. In the inference step, for each observation, AdaBoost generates a posterior probability distribution over the hidden activities using Eq.(1). In the propagation step, those initial estimations of posterior probabilities are propagated to neighbouring nodes. Those recipient nodes k then combine the received probability distribution over y_i together with its local evidence given by AdaBoost and convert them into a distribution over y_k, using Eq.(4). The iterative process can be repeated until convergence. In our experiment, we found that running belief propagation for only one iteration is sufficient to converge the posterior distribution.

The belief propagation is slightly modified in our implementation. As the examples classified with high confidence usually tend to be the correct classification, we do not update the posterior distribution for those high-confidence examples during the iterative process of belief propagation, so that their beliefs can be propagated to the uncertain examples.

4.3 Examples selection

In this subsection, we introduce the method to select the examples for classifier retraining and adaptation. The examples contain dynamically discovered context, and AdaBoost is able to automatically incorporate the new context if it is discriminative enough. In this way, AdaBoost can be self-adapted or -refined. We perform examples selection after the belief propagation for the sake of selecting the informative and profitable examples to quickly converge the classifier without human intervention.

4.3.1 Measurements

First of all, we introduce the measurements that can evaluate the profitability of an example (data point), so that based on those quantitative criteria, the examples can be selected to adapt the model. The first metric we consider is the "drift" in the posterior distribution before and after the belief propagation. Belief propagation is able to smooth out the outliers by exploiting the temporal information. Those examples that experience huge "drift" in their posterior distributions are much more valuable, since they are not modelled by the initial activity model and have a greater chance of residing near the classification boundaries. Jensen-Shannon divergence can be used to measure the "drift", as it has been proved to be efficient to measure the distance between two distributions in previous work [22]. Supposing p_i and q_i are the posterior distributions of example i before and after belief propagation respectively, and then the JS-divergence is:

$$JS(p_i, q_i) = \frac{1}{2}D_{KL}(p_i||m) + \frac{1}{2}D_{KL}(q_i||m) \tag{5}$$

where $m = \frac{1}{2}(p_i + q_i)$ and $D_{KL}(p_i||m) = \sum_j p_{ij} log \frac{p_{ij}}{m_j}$ is the Kullback-Leibler divergence between two distributions. Therefore, we derive the first measurement as:

$$score_{i1} = \frac{JS(p_i, q_i) - JS_{min}(p, q)}{JS_{max}(p, q) - JS_{min}(p, q)} \tag{6}$$

we normalize the JS-divergence, so that the measurement based on the posterior distribution "drift" is always ranged in [0,1], in this way it is able to cater for characteristics of different activity data set.

As for the second measurement of profitability, we consider the number of consecutive neighbouring examples that have the same predicted results.

$$\begin{aligned} N_i &= min(N_i^{forward}, N_i^{backward}) \\ score_{i2} &= \frac{N_i - min(N)}{max(N) - min(N)} \end{aligned} \tag{7}$$

where $N_i^{forward}$ and $N_i^{backward}$ are the number of consecutive neighbouring observations that have the same predictions along the two directions of time series, from current observation i. It is normalized due to same reason as $score_{i1}$. This measurement shows the extent to which the neighbouring nodes have the consensus predictions, and the higher the number, the more likely that the prediction is correct. Obviously, $score_{i2}$ is proposed based on the temporal characteristic of human behaviour. One extreme condition is that the observation happens to be in the middle of an ongoing activity, and the $score_{i2}$ tends to be large and it is more confident about the prediction.

Finally, we consider the confidence of the examples after the belief propagation. The posterior distribution itself provides the information about the confidence of an example. Adding the examples with the highest confidence is equivalent to locating the class center, which in turn also helps to adapt the model to some extent, even though those examples are less informative. Therefore, the third measurement is formulated as $score_{i3} = max(p(y_i|x_i))$ (Eq.(4)).

To decide which example is more profitable, we need to take into account all the aforementioned metrics. Therefore, we determine the final score for the profitability of an example based on the corresponding scores for each of the metrics. The combined score is defined as follows:

$$\begin{aligned} score_i = \alpha_1 score_{i1} + \alpha_2 score_{i2} + \alpha_3 score_{i3} \\ s.t. \sum_{i=1}^{3} \alpha_i = 1 \end{aligned} \tag{8}$$

where the weights α_i is manually given. In our method, we evenly distribute the importance to the three metrics by setting $\alpha_1 = \alpha_2 = \alpha_3$. However, by giving different weights, the model may present different characteristics. For example, by increasing α_3 we give more weight to the high-confidence examples, and then the model adapts conservatively and the convergence is quite slow. By contrast, when we put more weight to $score_{i1}$, the model only takes those examples whose posterior distribution changes dramatically before and after belief propagation, and then the adaptation is performed aggressively. There is a danger that noisy data may be added and the model is jeopardised.

4.3.2 Retraining

Upon selecting the examples for model adaptation, AdaBoost can automatically determine the discriminative power of the new context (if there is any) in the example, and dynamically incorporate them for classification if they are discriminative enough. In this way, the model is adapted to new coming data.

One issue should be addressed when selecting the examples, this is the amount of retraining data among different activity classes should be balanced during the adaptation process. During the experiment we found that for activity class with small training dataset, the iterative process of training weak learners is unexpectedly terminated earlier. As a result, the trained ensemble of classifiers for that class overfit the small amount of data. That is the reason AdaBoost focuses more on training activities with unevenly large dataset [9]. Therefore, in this paper, we accumulate for each activity class the same amount of dataset before retraining.

4.4 Sequential prediction

When the adapted AdaBoost is deployed for online prediction, we combine it with graphical models to further smooth outliers. Even though the basic idea behind this stage and belief propagation are both to exploit the temporal information among the activity data, belief propagation is deployed for offline data analysis, sufficient data should be accumulated and analyzed for model adaptation (second stage in Figure 1), while graphical models cater for online lightweight predictions (third stage in Figure 1). Furthermore, belief propagation requires the posterior distribution to evaluate the profitability of the examples.

In this section, we introduce the methods of combining AdaBoost with Conditional Random Field, referred to as BoostCRF. It should be noted that, hybrid classifiers are not new topics, in [12, 15, 31] the authors used the posterior probabilities from discriminative classifier as new input features to train HMM or CRF. However, the modelling of discriminative classifier is dissociated from the modelling of structured classifier. Therefore, the two classifiers are trained independently, using the output of one classifier as input for another. Moreover, they train HMM for each activity class separately, and during the inference phase for all the examples in a sequence, they produce the same label

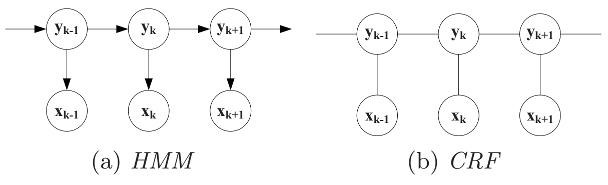

Figure 4: Graphical model of HMM and CRF.

that has maximum likelihood. Therefore, they do not model the transitions among different classes.

4.4.1 BoostHMM

In Hidden Markov models, the variables include hidden states and observations. As shown in Figure 4(a), it models the joint distribution of those variables by making Markov assumptions that current latent activity y_k only depends on previous latent activity y_{k-1}, while current observation x_k depends on current latent activity, formulated as follows:

$$p(\boldsymbol{x}, \boldsymbol{y}) = p(y_1)p(x_1|y_1)\prod_{k=2}^{K} p(y_k|y_{k-1})p(x_k|y_k) \tag{9}$$

where the emission probability $p(x_k|y_k)$ can approximated with the posterior distribution $p(y_k|x_k)$ given by AdaBoost (Eq.(1)) using Bayes' rule:

$$p(x_k|y_k) = \frac{p(y_k|x_k)p(x_k)}{p(y_k)} \propto p(y_k|x_k) \tag{10}$$

where prior knowledge $p(y_k)$ is identical for different activities because we balance the training data over all the activity classes. For a variable x_k that is observed at time k, $p(x_k)$ is a constant when calculating its evidence against different classes. Therefore, the emission probability is proportional to the posterior probability given by AdaBoost, and the joint distribution can be re-formulated as follows:

$$p(\boldsymbol{x}, \boldsymbol{y}) \propto p(y_1)p(y_1|x_1)\prod_{k=2}^{K} p(y_k|y_{k-1})p(y_k|x_k) \tag{11}$$

As for transition probability, we manually set the self-transition probabilities to be large to temporally smooth out the activities, and encourage them to continue unless observable evidence strongly suggests a different activity [25], denoted as follows:

$$p(y_k|y_{k-1}) = \begin{cases} 1-\epsilon & y_k = y_{k-1} \\ \epsilon & \text{otherwise} \end{cases} \tag{12}$$

we experimentally set ϵ to be 0.1 in our system, as it is demonstrated to be effective enough to achieve reasonable accuracy. Inferring the hidden states is equivalent to finding the sequences that maximize the joint probability depicted in Eq.(11), which can be performed by the Viterbi algorithm.

4.4.2 BoostCRF

In Conditional Random Field (CRF), the connections between the variables denote the potentials between them, and the potential functions map those potentials into real numbers. Due to the flexible definition of the potential functions, CRF has various structures. In our system, we only consider linear-chain CRF (Figure 4(b)). Therefore, we need to define local potential functions between observation and hidden node at each time step, and pairwise potential functions between consecutive hidden nodes. The conditional distribution can be formulated as:

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{x}) &= \frac{1}{Z(x)} exp\left(\sum_{k=1}^{K} \lambda^T f(y_k, y_{k-1}, x_k)\right) \\ &= \frac{1}{Z(x)} exp\left(\sum_{k=1}^{K} \left(\lambda_s^T f_s(y_k, y_{k-1}) + \lambda_j^T f_j(y_k, x_k)\right)\right) \end{aligned} \tag{13}$$

where $f_j(y_k, x_k)$ and $f_s(y_k, y_{k-1})$ are the local and pairwise potential functions at time k. λ_s and λ_j are the corresponding weights. $Z(x)$ is the normalization factor, formulated as $\sum_{\boldsymbol{y}} exp\left(\sum_{k=1}^{K} \lambda_k f_k(y_k, y_{k-1}, x_k)\right)$.

Inspired by [13], we map the weak learners trained in AdaBoost to the local potential functions in CRF, while the weights of the potential functions are mapped to the weights of the weak learners. This is reasonable since more weights are given to the potential functions that can better explain the data, whereas weak learners with less error rate have a larger weight. Using Eq.(2), the weighted sum of local potential functions against activity class i is:

$$\lambda_j^T f_j(y_k, x_k) = \sum_{t=1}^{T} \alpha_t^i h_t^i(x_k) \tag{14}$$

However, mapping the weight of pairwise potential function is non-trivial. To deal with this, we define pairwise potential function that characterize the temporal transition between activities:

$$f_{ij}(y_k, y_{k-1}) = \begin{cases} 1 & y_k = i, y_{k-1} = j \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

where potential function f_{ij} characterize the transition from activity j to activity i. Assume that there is a weak learner $h^i(y_k = i, y_{k-1} = j)$ in AdaBoost that can be mapped to the potential function f_{ij}. Obviously, the error rate of the weak learner can be estimated from the training dataset by frequency counting:

$$\epsilon_{ij} = 1 - \frac{\text{expected number of transitions from j to i}}{\text{expected number of transitions out of j}} \tag{16}$$

then according to Algorithm 1, the weight of the weak learner can be approximated as:

$$\alpha_{ij} = \frac{1}{2} ln(\frac{1-\epsilon_{ij}}{\epsilon_{ij}}) \tag{17}$$

the weight of weak learner $h^i(y_k = i, y_{k-1} = j)$, α_{ij}, is mapped to the weight of the pairwise potential function f_{ij} in CRF. Once we have the parameters, the inference process can be carried by loopy belief propagation to find the most likely assignment of the latent activities. Notice that, we have T local potential functions, but only 1 pairwise potential function, thus the temporal evidence weighs less when compared with local evidence. Therefore, we multiply the pairwise potential functions with a constant (average number of weak learners of the activity classes), so that the inferred results do not overfit the local evidences.

5. EXPERIMENT

In this section, we will validate our methods introduced in the previous sections. We firstly introduce the datasets, and then specify the method to evaluate our approach.

5.1 Datasets

Smartphone dataset (SD) [19]: Activity data was collected from accelerometer, gyroscope and magnetometer on an Android device worn in different body position (arm, belt, waist and pocket), when the subject performs standing, walking, upstairs, sitting, running and downstairs. The sample rate is set to be 50Hz. We compute time domain features such as mean, standard deviation, median, zero crossing rate, variance, root mean square for each axis of the sensors with a 2 sec sliding window and 50% overlap.

Sensors activity dataset (SAD) [19]: Sensor data was collected when the 10 volunteers perform standing, walking, upstairs, sitting, downstairs, jogging and biking. We extract the same features as the first dataset.

UCI HAR dataset [2]: The dataset was collected with accelerometer and gyroscope from a Samsung Galaxy SII smartphone worn by 30 volunteers. The smartphone was fixed on the waist when the subjects perform six activities (walking, walking_upstairs, walking_downstairs, sitting, standing, laying). The 561 features were computed based the sliding window of 2.56 sec and 50% overlap. In our experiment, we only consider time domain features, as it is computationally expensive to compute the frequency domain features on the mobile phone during online prediction. Therefore, we have 80 features from the gyroscope and 120 features from the accelerometer.

5.2 Set up

To validate our system, each of the datasets is divided into three portions, in accordance with the three stages in Figure 1. Specifically, we train the activity model with the first part of the dataset that contains **only** gyroscope data at the first stage. At the second stage, the activity model is used to classify the second part of the dataset which contains **both** accelerometer and gyroscope data, and after offline data analysis we choose the profitable examples to retain the activity model, and features from the accelerometer are automatically incorporated into AdaBoost if they are discriminative. In the final stage, we classify the third part of the dataset with the adapted model and compare the results with ground truth.

The first dataset is personalized, we evenly partition the dataset into three parts and perform 6-cross validation. While the latter two datasets involve multiple volunteers, so we perform the leave-one-out classification. Data from all the persons except one is used to create the initial model. Whereas the data from the testing people is divided into two parts: one for *learning to adapt* and the other one for validation. This process is repeated for all the users.

In what follows, we will validate the effectiveness of our system in terms of several aspects, especially the ability to incorporate new context, the importance of belief propagation and examples selection, the benefit of combining AdaBoost with graphical models. Finally, we investigate the conditions under which our methods provide a marginal improvement or even jeopardise the initial model.

5.3 Incorporating new context

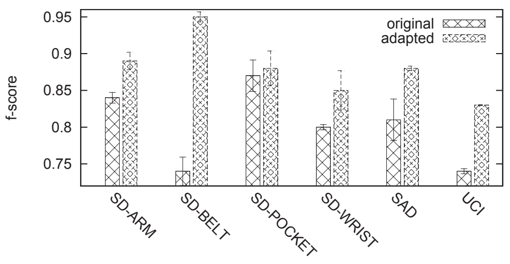

Figure 5: F-score improvement by dynamically and automatically incorporating accelerometer data

In this section, we validate our method by building activity model with gyroscope data, and dynamically incorporating accelerometer data to refine the model. 300 weak learners are trained for each activity and the *score* threshold is set to be 0.7 to select examples for retraining, as it is low enough to select sufficient training data and high enough to exclude the noisy examples. We do not perform the iterative process to select the examples and retrain the model, as we found that additional iterations do not provide significant accuracy improvement according to our experiments. On the other hand, repeatedly retraining the model is expensive. For all the experiments, we compare the recognition performance in terms of **f-score**($f - score = \frac{2*precision*recall}{precision+recall}$).

In Figure 5, we can see that, our method (*adapted*) can improve the recognition accuracy to some extent across the datasets, especially for the dataset that the user fixes the smartphone on the belt. Because it is difficult to distinguish standing and sitting with gyroscope when the device is put on the belt. However, as belief propagation is able to correct most of the uncertainties, and then the retraining examples would help to refine the initial model. Furthermore, the f-score improvement in SD-POCKET setting is marginal. When debugging system, we found that only one weak learner is trained to classify the activity **Sitting**, that means the weak learner overfits the retraining dataset and is unable to classify **Sitting** during prediction stage if the activity presents variance. However, when we lower the *score* threshold and collect more examples for retraining, the f-score achieves 0.94.

In order to confirm the usefulness of extra features, we look deep into our system and count the proportion of weak learners that are trained on the new features during the retraining process. Since AdaBoost is able to automatically select the weak learner that has the minimum weight error rate in each iteration, the more that the weak learners are trained on the new features, the more discriminative the new features are. As is presented in Figure 6, for most of the dataset the proportions of weak learners trained on new features are more than 50%. From the figure we can see that dataset SD-BELT and SD-POCKET have the proportions of 62% and 38% respectively. The underlying reason is that, for the dataset SD-BELT the accelerometer features can better distinguish standing and sitting, and then during the retraining process, more weak learners are trained on the accelerometer data. While in SD-POCKET dataset, the retraining process terminates unexpectedly early for activity **Sitting**, and fewer weak learners are trained on the retraining dataset and hence the new features cannot be sufficiently leveraged for performance improvement.

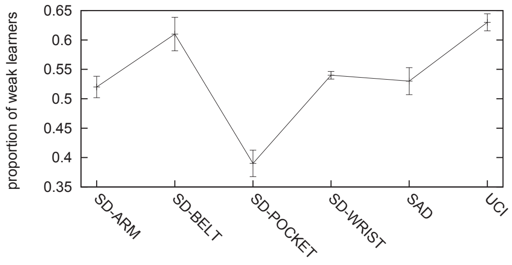

Figure 6: Proportion of weak learners trained on new features during the retraining process across the datasets.

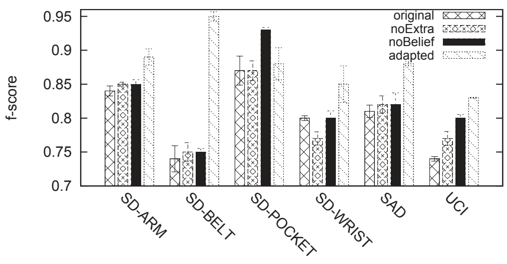

Figure 7: Comparison with *noBelief* and *noExtra* in terms of f-score.

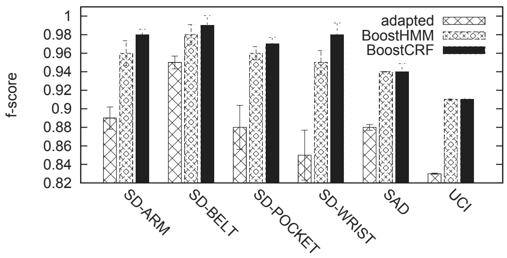

Figure 8: Combining adapted AdaBoost with HMM and CRF.

5.4 Role of belief propagation

In this subsection, we will examine the role that belief propagation plays in our system. For comparison, we do not perform belief propagation on the intermediate predictions of AdaBoost and choose the most confident examples for retraining, referred to as *noBelief*. We also compare with the setting without belief propagation and not considering the extra features (acceleration features), referred to as *noExtra*. Therefore, *noExtra* is exactly the traditional semi-supervised learning that selects the most confident examples to adapt the model, while *noBelief* still considers the incorporation of extra features.

The configurations for these two methods are the same as ours except that the confidence threshold is set to be 0.7 to select examples for retraining. The result is presented in Figure 7, from which we can see that for most of the datasets, *noBelief* and *noExtra* provide marginal f-score improvement. In some case, *noExtra* even experiences performance loss. The reasons are two-fold. On the one hand, high-confidence examples are usually less informative and make less contribution to the f-score improvement. On the other hand, it is difficult to set a universal confidence threshold for all datasets. For example, in the dataset SAD, the activity **Sitting** is frequently classified with a confidence lower than 0.7 (the confidence threshold). Due to the enforcement of retraining data balance, insufficient data of sitting results in a small amount of retraining dataset and hence, less contribution in f-score improvement. While in the dataset SD-WRIST, a confidence threshold of 0.7 introduces the noisy examples and has a negative impact on the recognition performance.

An exception is found in the dataset SD-POCKET, in which the *noBelief* achieves the f-score as high as 0.93, as gyroscope performs better than accelerometer in pocket position, confirmed by [19]. Therefore, initial model with gyroscope is able to correctly recognize most of the activities with high confidence, and provides true labels for the retraining with the combination of accelerometer and gyroscope data, hence the resulting model can then significantly improve the recognition performance. As discussed in the previous subsection, our method is able to obtain 0.94 in f-score when we lower the *score* threshold.

It should be noted that for most of the datasets (except UCI), traditional semi-supervised method (*noExtra*) does not provide performance improvement. However, it does not necessarily mean the contradiction between our experiments and previous work [21]. In our cases, the recognition performance is limited by the discriminative power of the features rather than the amount of training data, as we build the initial model with sufficient training data, especially for the later two datasets which include activity data from multiple users. The dataset SD-POCKET supports our conclusion. Both *noExtra* and *noBelief* take the exactly the same data for retraining, but only *noBelief* results in model refinement, due to the fact that it incorporates acceleration features.

To conclude, by incorporating newly discovered features, our method outperforms traditional methods that simply consider the most confident example, and belief propagation followed by examples selection scheme achieves significant improvement in terms of the recognition performance.

5.5 Role of graphical model

In this subsection, we evaluate the recognition performance by combining AdaBoost with CRF, which is to smooth the accidental predictions given by AdaBoost.

The results are shown in Figure 8, from which we can see that by temporarily smoothing the outliers, the f-score can be improved by 7.9% and 8.2% with BoostHMM and BoostCRF respectively. The figure also shows that BoostCRF performs slightly better than BoostHMM, which has been confirmed by previous work [24]. The reason is that, BoostHMM makes strong assumptions among the variables while BoostCRF have more flexible structures and relationships between connected nodes. Actually, when we look at the results provided by BoostHMM, examples of some continuous activity are still sporadically classified as other classes.

For the datasets SAD and UCI, BoostCRF seems to present no advantage over BoostHMM. This is because we only perform one iteration during the inference process for BoostCRF. It seems that one iteration is not enough to converge the model, because more iterations can still improve the f-score, as shown in Figure 9. It should be noted that, the authors in [2] use Support Vector Machines (SVMs) to clas-

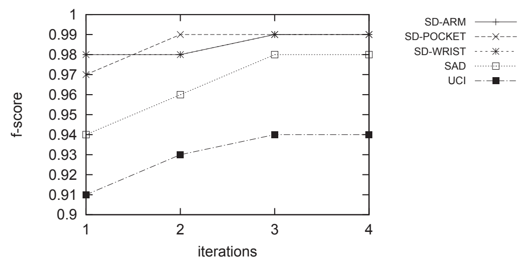

Figure 9: F-score corresponding to the number of iterations during inference process for BoostCRF.

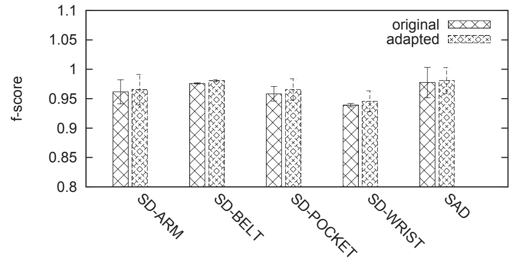

Figure 10: Performance(f-score) improvement by incorporating magnetometer features, we do not experiment on dataset UCI as it does not provide magnetometer data.

sify the activity with the same dataset, UCI, and obtain the average accuracy of 89.0%. By comparison, we are able to achieve the f-score of 94.0% with BoostCRF. However, we only use the 80 gyroscope features while they build their model on the all of the 561 features.

5.6 Investigation of the usefulness of extra context

In this subsection, we investigate the conditions under which the extra context cannot help with the accuracy improvement. To this end, we make the following assumptions and perform experiment with the datasets to validate those hypothesises.

1. When the extra context provides less discriminative information compared with existing features.

2. When the initial model is not accurate enough to perform adaptation.

The basic idea is that extra context, which cannot better characterize the activities classes or are less discriminative than the features upon which the initial model is built, are automatically ignored during the retraining process. Secondly, if the initial model is not accurate enough, mis-classified examples would be selected for retraining and jeopardise the model. To validate the first assumption, we build the initial model with accelerometer and gyroscope data. During the learning and adaptation stage, the examples contain accelerometer, gyroscope and magnetometer data. As magnetometer data is demonstrated to be overfitting [19], magnetic features are less likely to be incorporated during the adaptation stage. The results are illustrated in Figure 10, from which we can see the f-score improvement is insignificant, less than 1% on average. Figure 11 provides a more insightful reason, which shows that only a small portion of weak learners are trained on magnetic features, since they are less discriminative than acceleration features and angular velocity features.

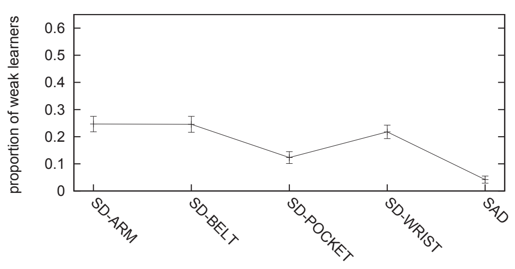

Figure 11: Percentage of weak learners that are trained on magnetometer features during the adaptation process.

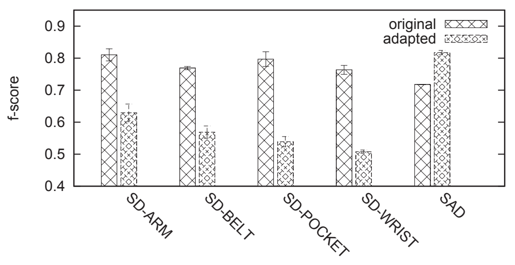

Figure 12: Performance(f-score) decrement with an inaccurate initial model.

In order to validate the second assumption, we limit the size of initial training dataset, so that the initial model would overfit the dataset and result in an inaccurate classifier. We use 5% of the training data to build the initial model, and present the results in Figure 12. From the figure one can see that, the adapted model would be negatively affected if the initial model is not accurate enough. The underlying reason is that wrongly predicted examples are added to retrain the model. One potential solution to this problem is to be more conservative and increase the weight α_3 in Eq.(8). However, it is out of the scope of this paper and is left for future work.

6. CONCLUSION

In this paper, we propose methods to automatically incorporate dynamically available contexts for activity recognition in dynamic environments. We build the initial activity recognition model with training data, and choose the profitable examples to adapt and refine the model. AdaBoost can automatically select the most discriminative features during the adaptation process. We also leverage the temporal information of human behaviour to boost the performance, both in the off-line data analysis and online predictions.

Experimental results show that the recognition performance can be significantly improved with dynamically discovered data sources. The proposed method is able to select the valuable examples to adapt and refine the model without human intervention, and the combination with graphical models is able to further improve the recognition accuracy.

From the experiments with the later two datasets, our methods can also be used to perform activity personalization, where general model built with multiple users is then adapted to the specific user at run time. In this light, building the general model is the first step to performing personalization. In the future, we will learn the general model from the data of multiple users, without the constraints that the data has to be labelled or requires the exactly the same data sources.

7. ACKNOWLEDGMENTS

This work is partially supported by NICTA (National ICT Australia).

8. REFERENCES

[1] F. Albinali, S. Intille, W. Haskell, and M. Rosenberger. Using wearable activity type detection to improve physical activity energy expenditure estimation. In *In Ubicomp'2010*.

[2] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *Ambient assisted living and home care*. 2012.

[3] A. Brajdic and R. Harle. Walk detection and step counting on unconstrained smartphones. In *Ubicomp'2013*.

[4] L. Chen, C. D. Nugent, and H. Wang. A knowledge-driven approach to activity recognition in smart homes. *TKDE'2012*.

[5] Y. GRANDVALET. Semi-supervised learning by entropy minimization. *NIPS'2005*.

[6] P. Hevesi, S. Wille, G. Pirkl, N. Wehn, and P. Lukowicz. Monitoring household activities and user location with a cheap, unobtrusive thermal sensor array. In *Ubicomp'2014*.

[7] P. Hu, J. Indulska, and R. Robinson. An autonomic context management system for pervasive computing. In *PerCom'2008*.

[8] S. Kang, Y. Lee, and C. Min. Orchestrator: An active resource orchestration framework for mobile context monitoring in sensor-rich mobile environments. In *PerCom'2012*.

[9] M. Keally, G. Zhou, G. Xing, J. Wu, and A. Pyles. Pbn: towards practical activity recognition using smartphone-based body sensor networks. In *Sensys'2011*.

[10] J. R. Kwapisz, G. M. Weiss, and S. A. Moore. Activity recognition using cell phone accelerometers. *sensorKDD'2011*.

[11] O. D. Lara, A. J. Pérez, M. A. Labrador, and J. D. Posada. Centinela: A human activity recognition system based on acceleration and vital sign data. *Pervasive and mobile computing*, 2012.

[12] J. Lester, T. Choudhury, N. Kern, G. Borriello, and B. Hannaford. A hybrid discriminative/generative approach for modeling human activities. In *IJCAI'2005*.

[13] L. Liao, T. Choudhury, D. Fox, and H. Kautz. Training conditional random fields using virtual evidence boosting. In *IJCAI'2007*.

[14] G. Luis and O. Amft. mining relations and physical grouping building-embedded sensors and actuator. In *PerCom'2015*.

[15] T. Maekawa, Y. Yanagisawa, Y. Kishino, K. Ishiguro, K. Kamei, Y. Sakurai, and T. Okadome. Object-based activity recognition with heterogeneous sensors on wrist. In *Pervasive'2010*.

[16] A. Reiss and D. Stricker. Personalized mobile physical activity recognition. In *ISWC'2013*.

[17] D. Riboni and C. Bettini. Cosar: hybrid reasoning for context-aware activity recognition. *Personal and Ubiquitous Computing*, 2011.

[18] K. Sankaran, M. Zhu, X. F. Guo, A. L. Ananda, M. C. Chan, and L.-S. Peh. Using mobile phone barometer for low-power transportation context detection. In *Sensys'2014*, 2014.

[19] M. Shoaib, H. Scholten, and P. J. Havinga. Towards physical activity recognition using smartphone sensors. In *UIC/ATC'2013*.

[20] M. Stikic, D. Larlus, and B. Schiele. Multi-graph based semi-supervised learning for activity recognition. In *ISWC'2009*.

[21] M. Stikic, K. Van Laerhoven, and B. Schiele. Exploring semi-supervised and active learning for activity recognition. In *ISWC'2008*.

[22] F.-T. Sun, Y.-T. Yeh, H.-T. Cheng, C. Kuo, and M. Griss. Nonparametric discovery of human routines from sensor data. In *PerCom'2014*.

[23] E. M. Tapia, T. Choudhury, and M. Philipose. Building reliable activity models using hierarchical shrinkage and mined ontology. In *Pervasive'2006*.

[24] T. Van Kasteren, A. Noulas, G. Englebienne, and B. Kröse. Accurate activity recognition in a home setting. In *Ubicomp'2008*.

[25] S. Wang, W. Pentney, A.-M. Popescu, T. Choudhury, and M. Philipose. Common sense based joint training of human activity recognizers. In *IJCAI'2007*.

[26] J. Wen and J. Indulska. Discovering latent structures for activity recognition in smart environments. In *2014 IEEE UIC/ATC/ScalCom Multi-Conference*, 2015.

[27] J. Wen and M. Zhong. Activity discovering and modelling with labelled and unlabelled data in smart environments. *Expert Systems with Applications*, 42(14):5800–5810, 2015.

[28] J. Ye, S. Dobson, and S. McKeever. Situation identification techniques in pervasive computing: A review. *Pervasive and mobile computing*, 2012.

[29] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *Information Theory, IEEE Transactions on*, 2005.

[30] P. Zappi, C. Lombriser, T. Stiefmeier, E. Farella, D. Roggen, L. Benini, and G. Tröster. Activity recognition from on-body sensors: accuracy-power trade-off by dynamic sensor selection. In *EWSN'2008*.

[31] K. Zhan, S. Faux, and F. Ramos. Multi-scale conditional random fields for first-person activity recognition. In *PerCom'2014*.

[32] Z. Zhao, Y. Chen, J. Liu, Z. Shen, and M. Liu. Cross-people mobile-phone based activity recognition. In *IJCAI'2011*.