Using Graphs to Improve Activity Prediction in Smart Environments based on Motion Sensor Data

S. Seth Long and Lawrence B. Holder

Washington State University

Abstract. Activity Recognition in Smart Environments presents a difficult learning problem. The focus of this paper is a 10-class activity recognition problem using motion sensor events over time involving multiple residents and non-scripted activities. This paper presents the results of using three different graph-based approaches to this problem, and compares them to a non-graph SVM approach. The graph-based approaches are generating feature vectors using frequent subgraphs for classification by an SVM, an SVM using a graph kernel and nearest neighbor approach using a graph comparison measure. None demonstrate significantly superior accuracy compared to the non-graph SVM, but all demonstrate strongly uncorrelated error both against the base SVM and each other. An ensemble is created using the non-graph SVM, Frequent Subgraph SVM, Graph Kernel SVM, and Nearest Neighbor. Error is shown to be highly uncorrelated between these four. This ensemble substantially outperforms all of the approaches alone. Results are shown for a 10-class problem arising from smart environments, and a 2-class one-vs-all version of the same problem.

1 Introduction

Activity Recognition in Smart Environments is made difficult by the dynamics of everyday life, particularly in settings involving multiple residents without a particular set of activities to perform. Activities are seldom performed in exactly the same way from one time to the next, particularly if the resident is multitasking.

The Washington State University Smarthome project uses motion sensors to detect activity within a dwelling. These sensors are placed in strategic locations in order to report relevant information on the inhabitant's location (Figure 1).

The focus of this paper is a 10-class activity recognition problem arising from this project. Two residents and a pet lived in the smart environment in Figure 1. Data from the motion sensors during this period is annotated with 10 different activity labels, for non-overlapping activities. Not all the collected data is associated with one of these 10 activities, and data annotated with a particular activity label may contain sensor events not related to the activity in question. A simple SVM-based classifier (described in Section 3) gives a baseline accuracy of 68% in determining the correct activity label (using cross-10 validation). In order to improve upon this baseline accuracy, this paper presents a method of considering an activity to be a graph, and three graph-based approaches to improve accuracy. Frequent Subgraph SVM uses presence or absence of a set of frequent subgraphs to generate a feature vector, which is then classified by a standard SVM.Graph Kernel SVM uses a kernel which compares graphs directly based on the number of similar walks. Nearest Neighbor uses a graph comparison function as a distance measure for a Nearest Neighbor algorithm. Each of these three graph-based methods, along with the non-graph SVM, are shown to have uncorrelated error. An ensemble of these three plus the baseline method yields higher accuracy than any of the methods alone.

The Cairo test environment (Figure 1) consists of a 3-bedroom apartment with two residents and two types of sensors relevant to this work. Limited area motion sensors report any motion within an area of about a square meter. Area motion sensors report any motion within a room. The 10 labeled activities are: Work in office (46), Take medicine (44), Sleep (102), Night Wandering (67), Lunch (37), Leave Home (69), Laundry (10), Dinner (42), Breakfast (48), Bed to Bathroom (30). The number in parenthesis is the number of times the indicated activity occurs in the dataset. The learning algorithms are provided with motion sensor data only. Certain other attributes such as time [5] have proven to be useful as well, but are not considered in this work.

2 Previous Work

Deshpande et al. use feature vectors created from frequent subgraphs to analyze chemical structures in [7]. Feature vector generation is separated from classification, so that any classification algorithm can be used. Deshpande et al. use a Support Vector Machine. This is similar to the Frequent Subgraph SVM used in this work.

Graph-based kernels for Support Vector Machines, including the direct product kernel used in this paper, are described by Gärtner et al. in chapter 11 of Mining Graph Data [4] and in [8]. It is compared with a cycle-based kernel, which is quicker to compute but results in reduced accuracy.

Nearest Neighbor algorithms using graph edit distance were used in [11]. This approach was tested here, but accuracy was not as good as the system of points defined in Section 3.

Much research has been done on the Smarthome project. Singla et al. [1] use a Naive Bayes Classifier and Hidden Markov Model to classify activities similar to the ones in this paper, although generated in a more scripted manner and in a different living space. They present results on an 8-class problem of a similar nature. Kim et al. [9] provide a theoretical overview of the task, using Hidden Markov Models, Conditional Random Field, and Emerging Patterns. Crandall and Cook use a Naive Bayes Classifier and Hidden Markov Models to distinguish multiple inhabitants in a smart environment [6]

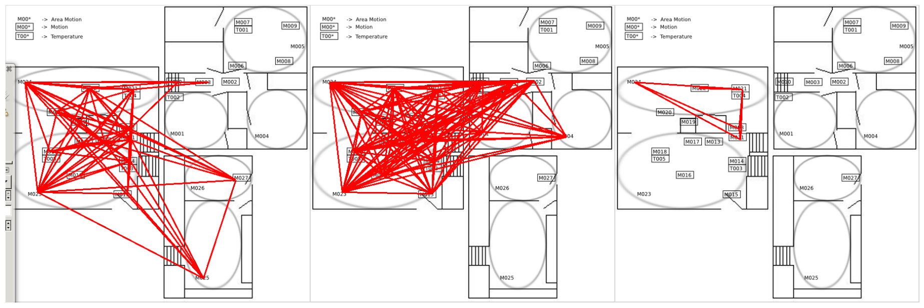

Fig. 1. The left two graphs are both of breakfast. Both include many sensor events outside the usual area of the activity. The rightmost picture is a frequent subgraph among all breakfast graphs.

The frequent substructures are discovered by Gaston [12], an established frequent subgraph miner for which the source code is available. The Support Vector Machines use libsvm [2], a freely available Support Vector Machine library.

3 Methods Used

A graph is constructed out of a sequence by considering each sensor to be a node, and connecting two sensors if they are visited in sequence. Figure 1 shows examples of graphs generated by such a process. No attempt is made to avoid connecting sensors which cannot physically be visited in sequence. This becomes significant when there are two inhabitants performing possibly different activities at the same time and in different locations within the environment.

Non-Graph SVM. A graph-unaware Support Vector Machine using a Radial Bias Function kernel was used as a representative of the non-graph-based approach. Support Vector Machines are a leading classification method [10] [2] [13], and the Radial Bias Function is a good general-purpose kernel given appropriate parameters [2] [15]. Input was given as feature vectors containing only the sensors triggered in a particular activity. This is represented as a vector of booleans, and is equivalent to the node labels in a graph representing the event.

Frequent Subgraph SVM. As indicated in figure 1, a graph of an activity may contain many extraneous edges. However, some edges will be common in all graphs from that activity. Figure 1 shows a graph which is common to all activities classified as "breakfast". A graph which contains this subgraph is more likely to represent the activity "breakfast" than one that does not.

A set of frequent subgraphs was found for each of the 10 classes. The presence or absence of each of these subgraphs in a graph to be classified was used to form a feature vector. A subgraph was only considered present if all of the edges in it are represented in the graph to be classified, with the requirement that the node labels match exactly. This allows the classifier to exploit the fact that particular activities tend to take place in particular locations.

Once the data is represented as a set of feature vectors, it can be classified using standard learning algorithms. An SVM with RBF kernel obtained higher accuracy than a decision tree algorithm, a naive Bayes classifier, and an artificial neural net.

SVM with Graph Kernel. Support Vector Machines do not interact directly with the examples they are classifying, but rather use a kernel function which returns a similarity measure. Training and test examples are given directly to this kernel function. Therefore, if a kernel function is used which takes as input a pair of graphs rather than a pair of feature vectors, a support vector machine can operate on graphs.

A direct product kernel [4] was used to compare graphs. This kernel compares walks in two graphs to determine similarity. It depends heavily on cross product between graphs. Consider a graph without edge labels to be represented as a set of vertexes V, a set of edges E, and a set of node labels α such that $G = (V, E, \alpha)$. The cross product of two graphs $g_1 = (V_1, E_1, \alpha_1)$ and $g_2 = (V_2, E_2, \alpha_2)$, is defined as $g_1 \times g_2 = (V_{g_1 \times g_2}, E_{g_1 \times g_2})$, where

$$V_{g_1 \times g_2} = \{(v_1, v_2) \in V_1 \times V_2 : \alpha_1(v_1) = \alpha_2(v_2)\}$$
$$E_{g_1 \times g_2} = \{((u_1, u_2), (v_1, v_2)) \in V_{g_1 \times g_2} \times V_{g_1 \times g_2} : (u_1, v_1) \in E_1, (u_2, v_2) \in E_2\}$$

Node are labeled for the sensor they represent. Labels do not need to directly appear in $g_{g_1 \times g_2}$, given that the following definition of the direct product kernel depends only on the adjacency matrix of $g_{g_1 \times g_2}$. The direct product kernel is defined by:

$$k(g_1, g_2) = \sum_{i,j=1}^{|V_{g_1 \times g_2}|} \left[\sum_{l=0}^{\infty} M^l_{g_1 \times g_2} \right]_{ij}$$

Entries in the adjacency matrix $M_{g_1 \times g_2}$ correspond to the number of walks between nodes i and j of length l. 10 is used as a computationally-feasible substitute for ∞. As noted in [8], graph kernels are computationally expensive. An $O(n^3)$ algorithm is proposed in [14], but was not used for this work. All values from this kernel were precomputed in parallel using a 296-node cluster.

Nearest Neighbor. Nearest Neighbor algorithms can be adapted to classify graphs given a suitable distance measure between graphs. This can be provided in a number of ways. The method used here is to assign "similarity points". 1 point is given for a matching node or edge in both graphs, and 4 points is lost for a node which cannot be matched in the other graph. This is intended to emphasize the location of the event, while still giving some consideration to the patterns of movement during it. This was converted into a distance by $1/points$ if more than three points were obtained, and a maximal distance of 1 otherwise. Accuracy is not greatly affected by changing the point values.

Some work has been done on using graph edit distance with a nearest neighbor algorithm [11]. This was tried using the "graph match" feature from Subdue [3]. This feature gives a "matching cost" which can be adapted to serve as a distance. This resulted in lower accuracy and required more time to run, and was not pursued further.

Ensemble. An ensemble of all 4 techniques mentioned above was used to increase performance beyond any of the methods alone. Weights were assigned such that a ranking of Frequent Subgraph SVM, the non-graph SVM, Nearest Neighbor, Graph Kernel SVM was established and no algorithm has more than twice the weight of any other. This is a ranking from most accurate algorithm to least.

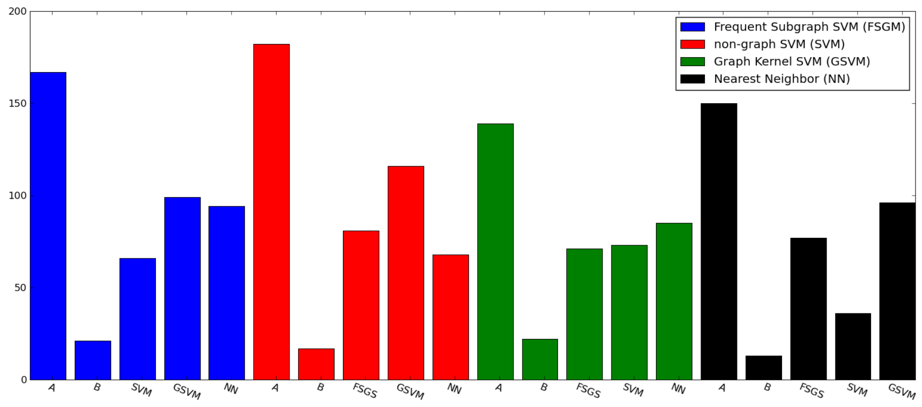

Fig. 2. Error correlation of all 4 methods. A indicates the number of times the indicated (by color) method was right and at least one other method was wrong. B indicates the number of times the indicated method was right, and all other methods were wrong. Each of the remaining three bars of each color represent the number of times the method indicated by color was right, when the method indicated by label was wrong.

4 Results

Accuracy obtained for each method, using 10-cross validation on the 10-class problem, was 66.66 for Subgraph SVM, 66.0 for Non-graph SVM, 58.5 for Graph SVM, 60.5 for Nearest Neighbor, and 72.5 for an ensemble of the previous four. The conditions for a successful ensemble are reasonably accurate classifiers and uncorrelated error. Error correlations are given in Figure 2. There are 53 examples that no algorithm classifies correctly, 73 correctly classified by exactly one algorithm, 89 by two, 129 by three, and 151 by all four. On 11 examples, two algorithms are correct but are outvoted by a higher-weighted pair of algorithms. The Nearest Neighbor algorithm appears to be the least useful in this comparison, however, when removed from the ensemble the overall accuracy drops to 70.1%.

There are 73 test examples which are only classified correctly by one algorithm. The voting weights used ensure that in all these 73 cases, the answer from the frequent subgraph SVM is used (this resulted in greater overall performance than using the graph kernel SVM in this role). This results in 52 misclassifications, out of 136 misclassifications total. In addition to these 52, there are 11 cases where two algorithms agree on the correct solution, but are outvoted by the other two algorithms, which have greater weight. Only 53 examples are not classified correctly by any algorithm. Weighting is irrelevant for 280 examples correctly classified by at least three algorithms.

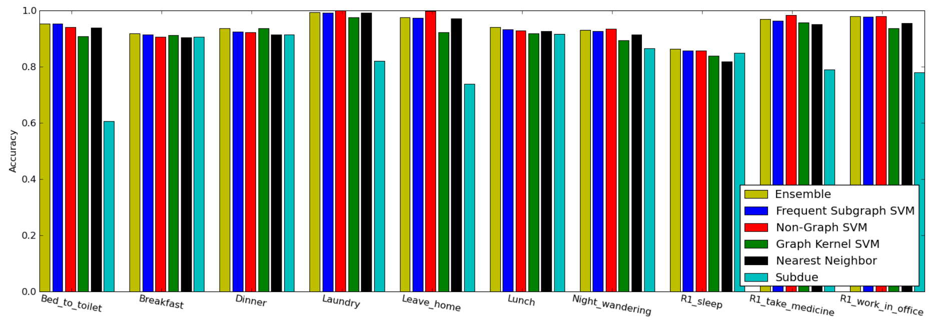

Fig. 3. Comparison on 2-class one-vs-all problems. 3-cross validation used for accuracy.

Class	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Bed to toilet (1)		1.0	0.98	0.87	0.92	0.98	0.87	0.75	0.97	0.88
Breakfast (2)	1.0		0.72	1.0	1.0	0.75	0.99	1.0	0.97	0.98
Dinner (3)	0.98	0.72		0.98	0.99	0.73	0.99	0.99	0.96	0.97
Laundry (4)	0.87	1.0	0.98		0.96	0.97	0.97	0.97	0.96	1.0
Leave home (5)	0.92	1.0	0.99	0.97		0.99	0.99	1.0	0.99	1.0
Lunch (6)	0.98	0.75	0.73	0.97	0.99		0.99	0.99	0.96	0.97
Night wandering (7)	0.87	0.99	0.99	0.97	0.99	0.99		0.76	0.96	0.88
Sleep (8)	0.75	1.0	0.99	0.97	0.98	0.99	0.76		0.97	0.92
Take medicine (9)	0.97	0.97	0.96	0.96	0.98	0.96	0.96	0.97		0.95
Work in office (10)	0.88	0.98	0.97	1.0	0.99	0.97	0.88	0.92	0.94	

Fig. 4. Table showing the accuracy in distinguishing one class from another using the ensemble. The three meals are the most difficult to distinguish from each other, followed by sleep, night wandering, and bed to toilet. Entries in blue indicate greater than 95% accuracy. Entries in red indicate accuracy below 80%.

2-Class Comparison. In order to determine which categories are particularly easy and hard to distinguish from all the others, a one-vs-all 2-class comparison is used. Accuracy in differentiating each class from all the others is calculated, as in Figure 3. Note that given the 10-class problem, an algorithm which simply guesses the majority class in every case (that is, that the example presented to it is not of the class under consideration) would score 90% accuracy on average. As such, it is unsurprising that nearly every accuracy calculated was above 90%.

The classes showing good performance do not correspond with those with high or low prevalence. Laundry is one of the most successful classes, with the non-graph algorithm scoring a perfect 100%, and the ensemble missing only a single example, and is the least frequent class. Leave Home is the second most frequent class, and is also perfectly distinguished by the non-graph approach (100% accuracy), as well as the third highest accuracy that the ensemble scored on any category. Sleep is the hardest category to distinguish, followed by Night Wandering (which produces particularly dissimilar graphs) and the three meals.

Figure 3 was generated using optimal parameters for each algorithm, for each category. Frequent Subgraph SVM and non-graph SVM vary in performance based on the RBF kernel parameter γ and the SVM parameter C. Using optimal parameters for each category, rather than a single set of parameters for all categories, gives a fair assessment of the difficulty of each 2-class problem.

One vs One Comparison. The 2-class comparison of Figure 3 provides a good indication of which categories are particularly easy or hard, but not which categories they are easily mixed up with. Figure 4 addresses this problem. The majority of pairs of activities can be distinguished with greater than 95% accuracy (33 out of 45, or 73%). As expected, classes which resulted in high accuracy in Figure 3 are not easily confused with any other class. For example, Laundry can be distinguished from 9 out of 10 other classes with greater than 96% accuracy, the remaining category (bed to toilet) with 87% accuracy. Take medicine, can be distinguished from every other category with 95% or greater accuracy.

In the case of the three meals, the meals are well distinguished from non-meal activities but not each other. Breakfast is traditionally defined as the first meal of the day, and given this definition no machine learning is required at all to differentiate it from other meals. Replacing all three meals with one generic "meal" activity label results in 81.8% overall accuracy.

5 Conclusion

Representing time-based sensor data as a graph provides additional information which is useful for classification. Although none of the graph-based algorithms significantly outperformed the non-graph SVM, they provided uncorrelated error which demonstrates that the graph-based approach is capable of correctly classifying graphs which cannot otherwise be classified correctly. Additionally, the different graph approaches used in the ensemble (Frequent Subgraph SVM, Graph Kernel SVM, and Nearest Neighbor) had uncorrelated error with each other.

This is a useful result in terms of overall classification accuracy. The ensemble gains 6.5% over the best classifier alone on the 10-class problem. It is also a promising result. If three graph-based classifiers with uncorrelated error on this problem exist, there may be others as well.

More work is needed to characterize when each classifier succeeds and fails. A complete success in this area would increase accuracy by over 12%, simply by correctly classifying the examples for which at least one classifier had the correct solution.

References

1. Tracking activities in complex settings using smart environment technologies. *International Journal of BioSciences, Psychiatry and Technology 1* (2009), 25–36.
2. Chang, C.-C., and Lin, C.-J. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
3. Cook, D. J., and Holder, L. B. Graph-based data mining. *IEEE Intelligent Systems 15*, 2 (2000), 32–41.
4. Cook, D. J., and Holder, L. B. *Mining Graph Data*. John Wiley & Sons, 2006.
5. Crandall, A., and Cook, D. Attributing events to individuals in multi-inhabitant environments. In *Proceedings of the International Conference on Intelligent Environments* (2008).
6. Crandall, A., and Cook, D. Coping with multiple residents in a smart environment. *Journal of Ambient Intelligence and Smart Environments 1*, 4 (2009), 323–334.
7. Deshpande, M., Kuramochi, M., Wale, N., and Karypis, G. Frequent substructure-based approaches for classifying chemical compounds. *IEEE Transactions on Knowledge and Data Engineering 17*, 8 (2005), 1036–1050.
8. Gaertner, T., Flach, P., and Wrobel, S. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of the 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop* (August 2003), Springer-Verlag, pp. 129–143.
9. Kim, E., Helal, S., and Cook, D. Human activity recognition and pattern discovery. *IEEE Pervasive Computing 9* (2010), 48–53.
10. Muller, K., Mika, S., Ratsch, G., Tsuda, K., and Scholkopf, B. An introduction to kernel-based learning algorithms. *IEEE transactions on neural networks 12*, 2 (2001), 181–201.
11. Neuhaus, M., and Bunke, H. Edit distance-based kernel functions for structural pattern classification. *Pattern Recognition 39*, 10 (2006), 1852–1863.
12. Nijssen, S., and Kok, J. N. A quickstart in frequent structure mining can make a difference. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2004), ACM, pp. 647–652.
13. Scholkopf, B., Sung, K., Burges, C., Girosi, F., Niyogi, P., Poggio, T., and Vapnik, V. Comparing support vector machines with Gaussian kernels to radial basis function classifiers. *IEEE Transactions on Signal Processing 45*, 11 (1997), 2758–2765.
14. Vishwanathan, S., Borgwardt, K., and Schraudolph, N. Fast computation of graph kernels. *Advances in Neural Information Processing Systems 19* (2007), 1449.
15. Wang, W., Xu, Z., Lu, W., and Zhang, X. Determination of the spread parameter in the Gaussian kernel for classification and regression. *Neurocomputing 55*, 3 (2003), 643–664.