Common Sense Based Joint Training of Human Activity Recognizers

Shiaokai Wang   **William Pentney**
Ana-Maria Popescu
University of Washington

Tanzeem Choudhury
Matthai Philipose
Intel Research

Abstract

Given sensors to detect object use, common-sense priors of object usage in activities can reduce the need for labeled data in learning activity models. It is often useful, however, to understand *how* an object is being used, i.e., the *action* performed on it. We show how to add personal sensor data (e.g., accelerometers) to obtain this detail, with little labeling and feature selection overhead. By synchronizing the personal sensor data with object-use data, it is possible to use easily specified common-sense models to minimize labeling overhead. Further, combining a generative common sense model of activity with a discriminative model of actions can automate feature selection. On observed activity data, automatically trained action classifiers give 40/85% precision/recall on 10 actions. Adding actions to pure object-use improves precision/recall from 76/85% to 81/90% over 12 activities.

1 Introduction

Systems capable of recognizing a range of human activity in useful detail have a variety of applications. Key problems in building such systems include identifying a small set of sensors that capture sufficient data for reasoning about many activities, identifying the features of the sensor data relevant to the classification task and minimizing the human overhead in building models relating these feature values to activities. The traditional approach, using vision as the generic sensor [Moore *et al.*, 1999; Duong *et al.*, 2005], has proved challenging because of the difficulty of identifying robust, informative video features sufficient for activity recognition under diverse real-world conditions and the considerable overhead of providing enough labeled footage to learn models.

A promising recent alternative [Philipose *et al.*, 2004; Tapia *et al.*, 2004] is the use of *dense sensing*, where individual objects used in activities are tagged with tiny wireless sensors which report when each object is in use. Activities are represented as probabilistic distributions over sequences of object use, e.g., as Hidden Markov Models. These simple models perform surprisingly well across a variety of activities and contexts, primarily because the systems are able to detect consistently the objects in use. Further, because objects used in day-to-day activities are common across deployment conditions (e.g., most people use kettles, cups and teabags in making tea), it is possible to specify broadly applicable prior models for the activities simply as lists of objects used in performing them. These priors may either be specified by hand or mined automatically off the web [Perkowitz *et al.*, 2004]. Given these simple models as priors and unlabeled traces of objects used in end-user activities, a technique for learning with sparse labels (e.g., EM) may be used to produce customized models with no per-activity labeling [Wyatt *et al.*, 2005].

Object use is a promising ingredient for general, low-overhead indoor activity recognition. However, the approach has limitations. For instance, it may be important to discern aspects of activities (e.g., whether a door is being opened or closed) indistinguishable by object identity, it is not always practical to tag objects (e.g., microwaveable objects) with sensors, and multiple activities may use similar objects (e.g., clearing a table and eating dinner). One way to overcome these limitations is to augment object-use with complementary sensors.

Wearable personal sensors (especially accelerometers and audio), which provide data on body motion and surroundings, are a particularly interesting complement. These sensors are unobtrusive, relatively insensitive to environmental conditions (especially accelerometers), and previous work [Lester *et al.*, 2005; Bao and Intille, 2004; Ravi *et al.*, 2005] has shown that they can detect accurately a variety of simple activities, such as walking, running and climbing stairs. We focus on how to use these sensors to detect fine-grained arm actions *using* objects (such as "scoop with a spoon", "chop with a knife", "shave with a razor" and "drink with a glass"; note that the physical motion depends on the kind of action *and* the object used), and also how to combine these actions with object-use data from dense sensors to get more accurate activity recognition.

We present a joint probabilistic model of object-use, physical actions and activities that improves activity detection relative to models that reason separately about
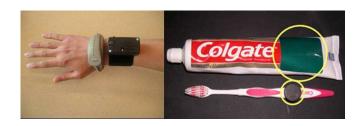
Figure 1: Sensors: iBracelet and MSP (left), RFID tagged toothbrush & paste (right), tags circled.
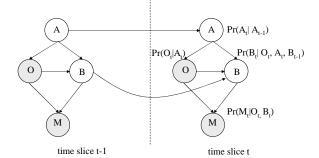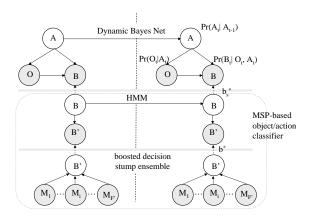

Figure 2: Joint models for activity and action recognition. (a) Full joint DBN (above), and (b) a derived layered model used in this paper (below). Dotted arrows represent the assignment of the MAP estimate of a variable in one layer as its observed value in the next.

these quantities, and learns action models with much less labeling overhead than conventional approaches. We show how to use the joint prior model, which can be specified declaratively with a little "common sense" knowledge per activity (and could in principle be mined from the web), to automatically infer distributions over the labels based on object-use data alone: e.g., given that object "knife" is in use as part of activity "making salad", action "chop with a knife" is quite likely. We therefore call our technique *common sense based joint training*. Given class labels, Viola and Jones (2001) have shown how to automatically and effectively select relevant features using boosting. We adapt this scheme to work over the distributions over labels, so that our joint model is able to perform both parameter estimation and feature selection with little human involvement.

We evaluate our techniques on data generated by two people performing 12 activities involving 10 actions on 30 objects. We defined simple commonsense models for the activities using an average of 4 English words per activity. Combining action data with object-use in this manner increases precision/recall of activity recognition from 76/85% to 81/90%; the automated action-learning techniques yielded 40/85% precision/recall over 10 possible object/action classes. To our knowledge this work is the first to demonstrate that simple commonsense knowledge can be used to learn classifiers for activity recognizers based on continuous sensors, with no human labeling of sensor data.

2 Sensors

Figure 1 shows the sensors we use. We wear two bracelets on the dominant wrist. One is a Radio Frequency Identification (RFID) reader, called the iBracelet [Fishkin *et al.*, 2005], for detecting object use, and the other is a personal sensing reader, called the Mobile Sensing Platform (MSP) [Lester *et al.*, 2005], for detecting arm movement and ambient conditions.

The iBracelet works as follows. RFID *tags* are attached to objects (e.g., the toothbrush and tube of toothpaste in Figure 1) whose use is to be detected. These tags are small, 40-cent battery-free stickers with an embedded processor and antenna. The bracelet issues queries for tags at 1Hz or faster. When queried by a reader up to 30cm away, each tag responds (using energy scavenged from the reader signal) with a unique identifier;

the identifier can be matched in a separate database to determine the kind of object. The bracelet either stores timestamped tag IDs on-board or transmits read tags wirelessly to an ambient base station, and lasts 12 to 150 hours between charges. If a bracelet detects a tagged object, the object is deemed to be in use, i.e., hand proximity to objects implies object use.

The MSP includes eight sensors: a six-degree-of-freedom accelerometer, microphones sampling 8-bit audio at 16kHz, IR/visible light, high-frequency light, barometric pressure, humidity, temperature and compass. The data (roughly 18,000 samples per second) is shipped in real time to an off-board device, such as a cellphone which currently stores it for later processing. In the near future, we hope to perform the processing (inference, in particular), in real-time on the cellphone. To compact and focus the data, we extract $F = 651$ features from it, including mean, variance, energy, spectral entropy, FFT coefficients, cepstral coefficients and band-pass filter coefficients; the end-result is a stream of 651-dimensional feature vectors, generated at 4Hz. We will write $S_N = s_1, \ldots, s_N$ for the stream of sensor readings, where each s_i is a paired object name and a vector of MSP features.

3 Techniques

3.1 Problem Statement

Let $A = \{a_i\}$ be a set of activity names (e.g., "making tea"), $B = \{b_i\}$ be a set of action names (e.g., "pour"), $O = \{o_i\}$ be a set of object names (e.g., "teabag") and $M = \{m_i\}$ be the set of all possible MSP feature vectors.

We assume coarse "commonsense" information linking objects, activities and actions. Let $OA = \{(a, O_a)|a \in A, O_a = \{o_1, \ldots, o_{n_a}\}$ s.t. o_i is often used in $a\}$ (e.g., $(a, O_a) = $ (make tea, {kettle, milk, sugar, teabag})). Let $BOA = \{(a, o, B_{a,o} = \{b_1, \ldots, b_{m_{a,o}}\})|a \in A, o \in O_a$ s.t. b_i is performed when using object o as part of $a\}$ (e.g., (make tea, milk, {pour, pick up})).

In monitoring peoples' day-to-day activities, it is relatively easy to get large quantities of unlabeled sensor data S_N. Getting labeled data is much harder. In what follows, we assume no labeling at all of the data. Finally, although training data will consist of synchronized object-use and personal sensor data, the test data may contain just the latter: we expect end-users to transition between areas with and without RFID instrumentation.

Given the above domain information and observed data, we wish to build a classifier over MSP data that:

- Infers the current action being performed and the object on which it is being performed.

- Combines with object-use data O (when available) to produce better estimates of current activity A.

- For efficiency reasons, uses $F' \ll F$ features of M.

3.2 A Joint Model

The Dynamic Bayesian Network (DBN) of Figure 2(a) captures the relationship between the state variables of interest . The dashed line in the figure separates the variables representing the state of the system in two adjacent time slices. Each node in a time slice of the graph represents a random variable capturing part of the state in that time slice: the activity (A) and action (B) currently in progress, and the MSP data (M) and the object (O) currently observed. The directed edges are inserted such that each random variable node in the induced graph is independent of its non-descendants given its parents. Each node X_i is parameterized by a *conditional probability distribution* (CPD) $P(X_i|\text{Parents}(X_i))$; the joint distribution $P(X_1, \ldots, X_n) = \Pi_{i=1\ldots n}P(X_i|\text{Parents}(X_i))$.

Our DBN encodes the following independence assumptions within each time slice:

1. The MSP signal M is independent of the ongoing activity A given the current object O and action B. For instance, hammering a nail (action "hammer", object "nail") will yield the same accelerometer and audio signatures regardless of whether you're fixing a window or hanging a picture. On the other hand, M is directly dependent both on O and B: the particular hand motion varies for action "pull" depending on whether object "door" or object "laundry line" are being pulled. Similarly, pulling a door entails a different motion from pushing it.

2. The action depends unconditionally both on current activity and the object currently in use. For instance, if the activity is "making pasta" the probability of action "shake" will depend on whether you are using object "salt shaker" or "pot". Similarly, if the object is "pot" the probability of action "scrub" will depend on whether the activity is "making pasta" or "washing dishes".

3. The object used is directly dependent on the activity. In particular, even if the action is a generic one such as "lift", the use of the object "iron" significantly increases the probability of activity "ironing".

Given initial guesses (derivable from OA and BOA) for the conditional probabilities, the problem of jointly re-estimating the parameters, estimating $P(M|O, B)$ and selecting a small discriminatory subset of features of M based on partially labeled data can be viewed as a semi-supervised structure learning problem, traditionally solved by structural variants of EM [Friedman, 1998]. However, structural EM is known to be difficult to apply tractably and effectively. We therefore trade off the potential gain of learning a joint model for the simplicity of a layered model.

3.3 A Layered Model

Figure 2(b) shows the layered model we use. The new structure is motivated by the fact that recent work [Viola and Jones, 2001; Lester *et al.*, 2005] has shown that a tractable and effective way to automatically select features from a "sea of possible features" is to boost an ensemble of simple classifiers sequentially over possible features. Our new model therefore replaces the node M of the original DBN, with a separate discriminative classifier smoothed by an HMM. Classification in the layered structure proceeds as follows:

1. At each time slice, the F' features are fed into the boosted classifier (described further below) to obtain the most likely current action-object pair b^* . The classification so obtained is prone to transient glitches and is passed up a level for smoothing.

2. The HMM is a simple temporal smoother over actions: its hidden states are smoothed action estimates, and its observations are the unsmoothed output b^* of the ensemble classifier. The MAP sequence of smoothed action/object pairs, b_s^*, is passed on to the next layer as observations. The HMM is set to have uniform observation probability and uniform high self-transition probability t_s. If we just require action classification on MSP data, we threshold the smoothed class by margin; if the margin exceeds threshold t_{smooth} (set once by hand for all classes), we report the class, else we report "unknown".

3. If we wish to combine the inferred action with object data to predict activity, we pass the smoothed action class up to the DBN. The DBN at the next level (a truncated version of the one described previously)

treats o and (the action part of) b_s^* as observations and finds the most likely activity A.

Inference requires parameters of the DBN to be known. The commonsense information OA and BOA can be converted into a crude estimate of conditional probabilities for the DBN as follows:

- $P(o|a) = \frac{p}{n_a}$ if $o \in O_a$, $\frac{1-p}{|O|-n_a}$ otherwise. Essentially, we divide up probability mass p (typically $\gg 0.5$) among likely objects, and divide up the remainder among the unlikely ones.

- $P(b_t|a,o) = \frac{p'}{m_{a,o}}$ if $b \in B_{a,o}$, $\frac{1-p'}{|B|-m_{a,o}}$ otherwise, as above (note $m_{a,o}$ and n_a are defined in section 3.1).

- $P(a_t|a_{t-1}) = \frac{\epsilon}{|A|-1}$ if $a_{t-1} \neq a_t$, and $1-\epsilon$ otherwise. This temporal smoothing encourages activities to continue into adjacent time slices.

Hyperparameters p, p' and ϵ are currently selected once by hand and do not change with activities to be classified or observed data. If these crude estimates are treated as priors, the parameters of the DBN may be re-estimated with unlabeled data S_N using EM or other semi-supervised parameter learning schemes.

It remains to learn the boosted classifier. We take the approach of Viola and Jones (2001). Their scheme (as any conventional boosting-based scheme) requires labeled examples. Given that we only have unlabeled MSP data S_N, we need to augment the usual scheme. A simple way (which we call MAP-Label) to do so would be to use the object-use data $o_1, \ldots, o_N$ from S_N and the DBN to derive the most likely sequence $b_1, \ldots, b_N$ of actions, and to treat the pair (b_i, o_i) as the label in each time step. This approach has the potential problem that even if many action/object pairs have comparable likelihood in a step, it only considers the top-rated pair as the label and ignores the others. We therefore explore an alternate form of the feature selection algorithm, called VirtualBoost, that works on data labeled with *label distributions*, rather than individual labels: we use the DBN and object-use data to generate the marginal distribution of B at each time slice (we refer to this technique as Marginal-Label). When a boosted classifier is trained using this "virtual evidence" [Pearl, 1988], VirtualBoost allows the trainer to consider all possible labels in proportion to their weight.

Table 1 specifies VirtualBoost. For simplicity we assume two classes, although the algorithm generalizes to multiple classes. The standard algorithm can be recovered from this one by setting P_{ic} to 1 and removing all sums over classes c. The goal of both algorithms is to identify $\leq M$ features, and corresponding single-feature multiple-class classifiers T_{f^*m} and weights α_m, the weighted combination of the classifiers is maximally discriminative.

The standard algorithm works by, for each iteration m, fitting classifiers T_{fm} for each feature f to the labeled data weighted to focus on previously misclassified examples, identifying as T_{f^*m} the classifier that

Initialize weights $w_i = 1/N, i = 1, \ldots, N$
Iterate for $m = 1, \ldots, M$:
1. $T_{fm}(s) \leftarrow \text{fit}(W, f)$ for all features f
2. $E_{fm} \leftarrow \sum_{i,c} w_i P_{ic} \mathcal{I}(c \neq T_{fm}(s_i))$
3. $f^* \leftarrow \arg\min_f E_{fm}$
4. $\alpha_m \leftarrow \log\left(1 - E_{f^*m}/E_{f^*m}\right)$
5. $w_i \leftarrow \sum_c P_{ic} w_i \exp(\alpha_m \mathcal{I}(c \neq T_{f^*m}(s_i)))$
6. Re-normalize w_i to 1
Output
$T(s) = \arg\max_c \sum_{m=1}^{M} \alpha_m \mathcal{I}(c = T_{f^*m}(s))$

where $\text{fit}(W, f) =$
$t_+ \leftarrow \sum_i w_i P_{i+} s_i[f] / \sum_i w_i P_{i+}$
$t_- \leftarrow \sum_i w_i P_{i-} s_i[f] / \sum_i w_i P_{i-}$
$t \leftarrow (t_+ + t_-)/2$
return $\lambda s. + \textbf{if } \text{sgn}(s-t) = \text{sgn}(t_+ - t), - \textbf{ else}$

Table 1: The VirtualBoost Algorithm

1	make tea (6,1)		a	lift to mouth
2	eat cereal (2,1)		b	scoop
3	make sandwich (3,1)		c	chop
4	make salad (2,1)		d	spread
5	dust (1,1)		e	dust
6	brush teeth (2,1)		f	brush
7	tend plants (2,1)		g	wipe horizontally
8	set table (9,1)		h	wipe vertically
9	clean windows (2,1)		i	drink
10	take medication (2,1)		j	shave
11	shave (3,1)			
12	shower (2,1)			

Table 2: Activities (l) and actions (r) performed

minimizes the weighted error, attributing a weight α_m to this classifier such that high errors result in low weights, and re-weighting the examples again to focus on newly problematic examples. VirtualBoost simply replaces all computations in the standard algorithm that expect individual class labels c_i with the expectation over the distribution P_{ic} of the label. It is straightforward to show that this new variant is sound, in the sense that if $D = (s_1, P_{1c}), \ldots, (s_i, P_{ic}), \ldots, (s_N, P_{Nc})$ is a sequence that when input to VirtualBoost yields classifier T, then the "sample sequence" $D' = \ldots, (s_i, c_{i1}), (s_i, c_{i2}), \ldots, (s_i, c_{iK}), \ldots$ where the c_{ij} are samples from P_{ic}, when fed to the conventional algorithm, will produce the classifier T for large K.

4 Evaluation Methodology

Our evaluation focuses on the following questions:

1. How good are the learned models? In particular, how much better are the models of activity with action information combined, and how good are the action models themselves? Why?

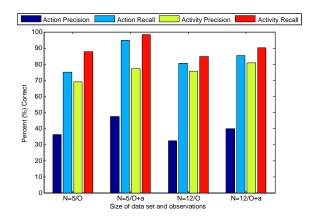2. How does classifier performance vary with our design choices? With hyperparameters?

Figure 3: Overall precision/recall

Figure 4: Precision/recall breakdown

3. How necessary were the more sophisticated aspects of our design? Do simple techniques do as well?

To answer these questions, we collected iBracelet and MSP data from two researchers performing 12 activities containing 10 distinct actions of interest using 30 objects in an instrumented apartment. The activities are derived from a state-mandated Activities of Daily Living (ADL) list; monitoring these activities is of interest to elder care professionals. Four to ten executions of each activity were recorded over a period of two weeks. Ground truth was recorded using video, and each sensor data frame was annotated in a post-pass with the actual activity and action (if any) during that frame, and designated "other" if not.

Table 2 lists the activities and actions performed. Each activity is followed by the number of tagged objects, and the number of actions labeled, in that activity; e.g., for making tea, we tagged 6 objects (spoon, kettle, tea, milk, sugar and saucer) and tracked one action (scoop with a spoon). We restricted ourselves to one action of interest per activity, although our system should work with multiple actions. We tag multiple objects per activity; 6 of the activities (make tea, eat cereal, make salad, make sandwich, set table, clean window) share at least one object with another activity. Note that for our system to work, we do not have to list or track exhaustively either the objects or the actions in an activity, an important consideration for a practical system.

In our experiments, we trained and classified the unsegmented sensor data using leave-one-out cross-validation over activities: we trained over all but one of the instances for each activity and tested on the last one, rotated the one that was left out, and report aggregate statistics. Since many time slices were labeled "other" for actions, activities or both, we use both precision (fraction of slices when our claim was correct) and recall (fraction of slices when we detected a non-"other" class correctly). In some cases, we report the F-measure $= 2PR/(P+R)$ a standard aggregate goodness measure, where P and R are precision and recall.
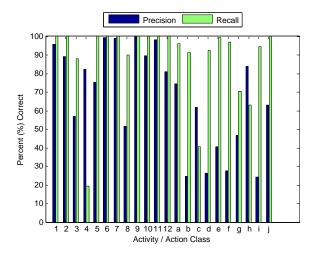
5 Results

Figure 3 displays overall activity/action detection precision/recall (P/R) for four configurations of interest. Each configuration has the form $(N = 5|12/O[+a])$. $N = 5$ attempts to detect activities 1-5 of Table 2, whereas $N = 12$ detects them all. In the $N = i|O$ configuration, we assume just RFID observations, whereas $N = i|O + a$ assumes MSP observations for the action classifier in addition. For each configuration, we report P/R of activity and action detection. The $N = 5$ configuration yields similar results to $N = 12$, so we focus on $N = 12$ below.

Three points are key. First, comparing corresponding O and A bars for activity, we see that adding MSP data does improve overall P/R from 76/85% to 81/90%. Second, our action classifier based on MSP data has 41/85% P/R; given that we have 10 underlying action/object classes all fairly evenly represented in the data, this is far better than any naive guesser. The low precision reflects the automatically-trained classifier's inability to identify when no action of interest is happening: 56% of our false positives are mis-labeled "other" slots. Third, the relatively high action P/R (32/80%) with just objects $(N = 12/O)$ reveals a limitation of our dataset: if object-use data is available, it is possible to predict the current action almost as well as with additional MSP data. In our dataset, each activity/object combination has one action associated with it, so guessing the action given activity/object is fairly easy.

Figure 4 breaks down P/R in the $N = 12$ case over activities 1-12 and actions 1-10. The main point of interest here (unfortunately not apparent in the figure) is the interaction between object and MSP-based sensing: combining the two sensors yields a jump in recall of 8, 15, 39 and 41% respectively in detecting activities 4, 8, 9 and 2 respectively, with no activity deteriorating. In all these cases the activities involved shared objects with others, so that actions were helpful. Activity 4, making
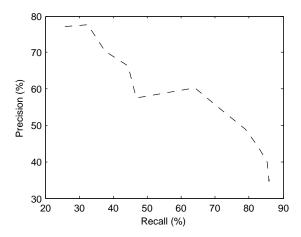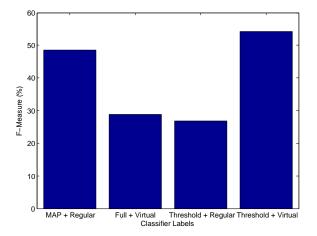
Figure 5: Precision/recall vs. margin threshold

Figure 6: Impact of virtual evidence handling techniques

are associated with a single object, it is not surprising that using the most likely action performs quite well. In some cases, because of overlap in object-use, Marginal-Label sprinkles in incorrect labels with correct ones. In these cases, the third approach ascribes too much weight (i.e., 1.0) to the (possibly incorrect) answer, whereas the fourth approach mitigates the label choice via its weight. Complete VirtualBoost seems to get distracted by needing to analyze all 10 actions every time.

6   Conclusions

Feature selection and labeling are known bottlenecks in learning sensor-based models of human activity. We have demonstrated that it is possible to use data from dense object-use sensors and very simple commonsense models of object use, actions and activities to automatically interpret and learn models for other sensors, a technique we call *common sense based joint training*. Validation of this technique on much larger datasets and on other sensors (vision in particular) is in progress.

References

[Bao and Intille, 2004] Ling Bao and Stephen S. Intille. Activity recognition from user-annotated acceleration data. In *Pervasive 2004*, pages 1–17, 2004.

[Duong *et al.*, 2005] Thi V. Duong, Hung Hai Bui, Dinh Q. Phung, and Svetha Venkatesh. Activity recognition and abnormality detection with the switching hidden semi-markov model. In *CVPR*, pages 838–845, 2005.

[Fishkin *et al.*, 2005] Kenneth P. Fishkin, Matthai Philipose, and Adam Rea. Hands-on RFID: Wireless wearables for detecting use of objects. In *ISWC 2005*, pages 38–43, 2005.

[Friedman, 1998] Nir Friedman. The Bayesian structural EM algorithm. In *UAI*, pages 129–138, 1998.

[Lester *et al.*, 2005] Jonathan Lester, Tanzeem Choudhury, Nicky Kern, Gaetano Borriello, and Blake Hannaford. A hybrid discriminative/generative approach for modeling human activities. In *IJCAI 2005*, pages 766–772, 2005.

[Moore *et al.*, 1999] Darnell J. Moore, Irfan A. Essa, and Monson H. Hayes. Exploiting human actions and object context for recognition tasks. In *ICCV*, pages 80–86, 1999.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[Perkowitz *et al.*, 2004] Mike Perkowitz, Matthai Philipose, Kenneth P. Fishkin, and Donald J. Patterson. Mining models of human activities from the web. In *WWW*, pages 573–582, 2004.

[Philipose *et al.*, 2004] M. Philipose, K.P. Fishkin, M. Perkowitz, D.J. Patterson, H. Kautz, and D. Hahnel. Inferring activities from interactions with objects. *IEEE Pervasive Computing Magazine*, 3(4):50–57, 2004.

[Ravi *et al.*, 2005] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman. Activity recognition from accelerometer data. In *AAAI*, pages 1541–1546, 2005.

[Tapia *et al.*, 2004] Emmanuel Munguia Tapia, Stephen S. Intille, and Kent Larson. Activity recognition in the home using simple and ubiquitous sensors. In *Pervasive*, pages 158–175, 2004.

[Viola and Jones, 2001] Paul A. Viola and Michael J. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, pages 511–518, 2001.

[Wyatt *et al.*, 2005] Danny Wyatt, Matthai Philipose, and Tanzeem Choudhury. Unsupervised activity recognition using automatically mined common sense. In *AAAI*, pages 21–27, 2005.

salad, is particularly hard hit because both objects used (knife and bowl) are used in other activities. Resolving this ambiguity improves P/R for the sole action associated with making salad by 62/41%.

Figure 5 shows how P/R of action detection changes when margin threshold t_{smooth} of section 3.3 varies from -0.8 to 0.2. We use $t_{\text{smooth}} = -0.25$ in this section. We also varied boosting iterations M to 5, 10, 20, 40 and 60; the best matching P/R rates were 36/61, 41/77, 40/85, 36/88 and 35/86%. We use $N = 20$.

Figure 6 compares VirtualBoost to simpler approaches. The first bar shows the effect on action detection (on F-measure) of MAP-Label combined with regular boosting-based feature selection; the second shows Marginal-Label with VirtualBoost. The third shows Marginal-Label with the single class with highest probability chosen as label (if probability is higher than a fixed threshold), combined with regular boosting. The fourth uses Marginal-Label, picks the most-likely class as label if above threshold, and feeds it *along with its probability* to a variant of VirtualBoost. Given that all our actions