Community-Guided Learning: Exploiting Mobile Sensor Users to Model Human Behavior

Daniel Peebles and **Hong Lu** and **Nicholas D. Lane** and **Tanzeem Choudhury** and **Andrew T. Campbell**
Department of Computer Science
Dartmouth College
6211 Sudikoff Lab
Hanover, NH 03755

Abstract

Modeling human behavior requires vast quantities of accurately labeled training data, but for ubiquitous *people-aware* applications such data is rarely attainable. Even researchers make mistakes when labeling data, and consistent, reliable labels from low-commitment users are rare. In particular, users may give identical labels to activities with characteristically different signatures (e.g., labeling eating at home or at a restaurant as "dinner") or may give different labels to the same context (e.g., "work" vs. "office"). In this scenario, labels are unreliable but nonetheless contain valuable information for classification. To facilitate learning in such unconstrained labeling scenarios, we propose *Community-Guided Learning (CGL)*, a framework that allows existing classifiers to learn robustly from unreliably-labeled user-submitted data. CGL exploits the underlying structure in the data and the unconstrained labels to intelligently group crowd-sourced data. We demonstrate how to use similarity measures to determine when and how to split and merge contributions from different labeled categories and present experimental results that demonstrate the effectiveness of our framework.

Introduction

The recent explosion of the sensor-equipped smartphone market has created an irresistible opportunity for machine learning researchers. Millions of people now voluntarily carry more sensors and computational power than was available on specialized sensing devices just two years ago (Bao and Intille 2004; Choudhury et al. 2008). Constrained learning applications for these devices already abound: dozens of iPhone applications have been released that incorporate basic human sensing and learning. Citysense (Networks 2008) for example traces user movement patterns to find nightlife in San Francisco. But while such applications are encouraging, it is not hard to envision even larger, more general systems for learning more complex trends from user data.

The availability of this data presents an opportunity to radically change the way we build computational models of human behavior. No longer will it be necessary to prepare carefully controlled experiments using specially engineered sensing devices and fixed label sets; the sensors are already deployed, and many users already document their lives on services like Twitter. We believe large-scale learning systems will benefit greatly from the effective use of such unconstrained human data. Current approaches, however, are ill-equipped to deal with such data: they typically rely on a static well-defined set of labels and cannot deal with semantic discrepancies. Furthermore, to improve generality and reduce individual user requirements, labels are often pooled from multiple users—a technique that only works if the labels are consistent across people. In reality it might be beneficial for classification to split a class with a specific textual label into several sub-classes or merge classes with different labels. Supervised and semi-supervised learning techniques assume a rigid closed set of labels, and unsupervised algorithms do not incorporate labels at all.

In this paper, we propose *Community-Guided Learning (CGL)*, a novel framework for learning in a dynamic human environment. Community-based models are already widely used in other domains (e.g., Wikipedia, SETI@home, Freebase), where they successfully amortize individual users' shortcomings by incorporating contributions from other users. Analogously, our approach uses notions of similarity to incorporate data from multiple users in a flexible manner that neither places excessive weight on labels nor discards them entirely. More specifically, CGL uses existing unsupervised and supervised classifiers to find groupings of the input data that maximize robustness and classifier performance.

The novel contributions of our work are as follows:

- We propose and develop the CGL framework for learning models of human behavior from crowd-sourced sensor data.
- We demonstrate the effectiveness of similarity measures to regroup classes specified by imperfect labels from users.
- We present experimental results showing CGL's advantages over conventional learning techniques.

Related Work

There is much prior work that carefully collects and labels high-quality training data (Bao and Intille 2004; Lester, Choudhury, and Borriello 2006; Choudhury et al. 2008; Huynh, Blanke, and Schiele 2007) for learning models of

human behavior. Unsupervised activity discovery is another active area of research where no labeled data is used during training. Both approaches need labels to be specified at some stage to perform useful classification. To achieve reasonable performance, the domain is often restricted to simple behaviors (Thad et al. 2006) or relies on output from low-level supervised models (Huynh, Fritz, and Schiele 2008) during unsupervised learning. Semi-supervised (Stikic, Van Laerhoven, and Schiele 2008; Mahdaviani and Choudhury 2007) and multi-instance learning approaches (Stikic and Schiele 2009) have been proposed to deal with limited and inconsistent labels. In multi-instance learning, labels are associated with a set of data points that includes at least one positive data point but may also include points that are not from the labeled class. In all of these cases, class labels are considered to be fixed.

We are not the first to identify sensor-enabled consumer devices as an opportunity to radically alter an existing paradigm. In recent years researchers have been considering ways that mass deployments of mobile sensors (e.g., embedded in cell phones) can change people-centric data collection (Abdelzaher et al. 2007; Campbell et al. 2006; Burke et al. 2006; Krause et al. 2008). This work has focused primarily on privacy, mobile device resource limitations, and data fidelity. Lane et al. (Lane et al. 2008) propose a community-based technique that groups people based on their social clique and partitions their data accordingly to improve learning. However, the grouping is social-network-based and the label domain is still fixed. Hence, the challenges we identified above remain unsolved.

Community-guided Learning

Inevitably, any large learning system will have to deal with mislabeled data for reasons ranging from simple misunderstandings to malicious behavior. Naïvely accepting free-form user-provided labels will almost certainly undermine learning, as these may be overly broad or narrow and either way, will rarely be textually equal. For example, distinguishable (from the point of view of the sensor data) classes may be given a single label like "work". In such cases, modeling the distinct subclasses separately may lead to better performance. On the other hand, similar or indistinguishable classes may be given labels like "driving" and "commuting" and a joint model will perform better.

We present Community-Guided Learning as a framework to build classifiers using inconsistently labeled sensor traces. To achieve this, CGL trains data in groups not only defined by users' labels but also by properties of the data. In essence, we account for "soft ground truth" by first performing a clustering step to refine the classes suggested by labels and then training a supervised classifier on those clusters. We list specific steps below and provide figure 1 for a schematic overview:

- Take as input inconsistently labeled mobile sensor data from multiple users.
- Measure *intra-class similarity* between segments that have the same label to decide whether to keep the original grouping of the data or to split dissimilar segments into sub-groups and model them as different classes.
- Train classifiers using groupings from previous step.
- Measure the *inter-class similarity* between segments that have different labels and decide whether to keep the original groupings based on the labels or to merge similar segments that have different labels and model them as one class.
- Retrain classifiers using groupings from previous steps.

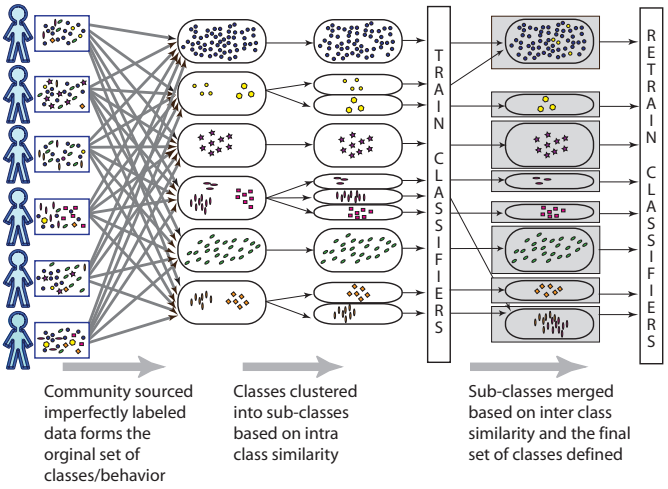

Figure 1: The main steps of CGL. Segments are first grouped according to user-provided labels. The class groupings are redefined based on inter- and intra-class similarity measures. Classifiers are built based on the resulting groupings.

In a deployed system, the above process can be applied iteratively using new contributions from the community to update classifiers.

Splitting and Merging User-Defined Classes

A primary goal in CGL is to preserve the user-provided labelings, as these are what they expect. But users cannot be expected to understand *how* to label their data such that the classifiers perform reliably. Instead, the reasoning system, guided by the labels and the input data, should make the final decision on how to re-partition that data to achieve good performance.

To that end, CGL first uses clustering to determine whether distinguishable classes have been assigned the same label. If so, the data is split into logical subclasses. Next, if indistinguishable classes have been assigned different labels, CGL unifies them. For example, data labeled by a user as 'driving' may include both urban and stationary segments – these different types of driving data will likely have different sensor signatures. This example should be clustered and split into two classes. However, in the merge phase, CGL may find the stationary subclass to be indistinguishable from other similar data, such as waiting at a drive-through, and will thus merge and model it as a single class. In the following section, we describe the two types of similarity metrics we use to split and merge the contributed segments and how

we utilize these similarity measurements in the learning process.

Defining Similarity. CGL uses *intra-class* and *inter-class* similarity measures, as described below:

Intra-class similarity. This is used during the splitting stage. The goal is to find the underlying groupings in data that share a label. This is purely data-driven and is effectively a clustering operation. Although many clustering algorithms exist and our technique can use any of them, in our experiments we choose Euclidean k-means (Bishop 2007) for its simplicity. Unfortunately, there is no agreed-upon solution for choosing the number of clusters k. We estimate k by evaluating the objective function for varying values of k—typically the objective function decreases rapidly when k is smaller than the correct number of clusters and flattens when k is larger. Thus, the location of an *elbow* in the objective function is a good choice for k (Milligan and Cooper 1985). The intuition behind this procedure is that there is some optimal clustering of the data, and if k is lower than the optimal number of clusters, the objective function will be high because distant points are being clustered together. On the other hand, if k is greater than the optimal value, clusters will get divided, causing a comparatively small decrease in objective.

If a class is split into two or more sub-classes, CGL treats them as independent classes for training, but associates them with the same user-visible label. This is purely for the user's benefit, and a different front-end could instead ask the user to refine the subcluster labels.

Inter-class similarity. Unlike the purely data driven intra-class similarity, we measure inter-class similarity by comparing the output of two different classifiers on the same input. The aim is to identify classes that may be labeled differently but belong to or is better modeled as a single category. For example, 'talking' and 'conversation' classes may both represent scenarios where a person is speaking. If the data belonging to two or more labels represent the same underlying class then these classifiers are likely to "agree" and are potential candidates for merging. Let us assume two classifiers, C_A and C_B. On a given test segment S, we compute C_A's performance on B and C_B's performance of A—let us assume they are based on confusion matrices conf_{AB} and conf_{BA} respectively. The agreement between the classifiers is defined to be the f-score computed from the combined confusion matrix $\mathrm{conf}_{AB+BA} = \mathrm{conf}_{AB} + \mathrm{conf}_{BA}$.

The intuition behind the similarity score is as follows: if two classes A and B are the same or very similar, C_A will perform well on B's data and C_B will perform well on A's data and the combined f-score will also be high. If the classes are dissimilar, C_A will perform poorly on B's data and vice-versa, resulting in a low combined f-score.

If the inter-class similarity score is greater than a experimentally determined threshold, the classes C_A and C_B are merged and used to train a unified classifier C_{AB}. We set this threshold such that merging operates conservatively, causing it only to be applied at levels of similarity that caused high subsequent f-scores during exploratory experiments. If classes are merged, the system associates the union of the labels to the newly defined class. But as with the split case described above, users could be prompted by a different front-end to confirm whether the classes should be merged and whether they want to re-label.

Training Classifiers

After classes are redefined based on the similarity measures, any classification algorithm can be trained and used for recognition. Since our primary goal is to test the effectiveness of CGL, we use simple boosted decision-stump classifiers (Friedman, Hastie, and Tibshirani 1998) in our current experiments to train binary activity classifiers. As with the clustering algorithms used during the splitting phase, we are not tied to a specific type of classifier and can use more sophisticated models if needed. However, boosted decision stumps have been used successfully in a variety of classification (Torralba and Murphy 2007) tasks including human activity recognition (Lester et al. 2005; Blanke and Schiele 2009).

For each activity A_i , we iteratively learn an ensemble of weak binary classifiers $C^i = {c_1}^i, {c_2}^i, {c_3}^i, ..., {c_M}^i$ and their associated weights ${\alpha_m}^i$ using the variation of the AdaBoost algorithm proposed by (Viola and Jones 2001). The final output is a weighted combination of the weak classifiers. The prediction of classifier C_i is:

$$C^i = \mathrm{sgn}(\sum_m {\alpha_m}^i {c_m}^i)$$

Supervised training is done after *both* the split and merge steps based on the new class groupings produced by those stages.

Evaluation

In this section, we describe our dataset and the experiments we conducted to evaluate the performance of CGL, and test CGL's ability to cope with two common forms of human labeling error: (i) inconsistencies in user-provided labels and (ii) incorrect user-provided label boundaries.

Dataset

We collected a 33-hour audio dataset of high-level activities and their associated contexts, as shown in table 1. We recruited five researchers to run a custom audio logger on jailbroken iPhones and instructed them to loosely but truthfully provide freeform labels of anything they chose. In the resulting dataset, we found that the labels consisted of activities performed (e.g., driving, eating, working) or the type/name of places the users visited (e.g., supermarket, office, library, restaurant).

It is worth noting that while the labels themselves were freeform, they were semantically meaningful to the user (i.e., the user did not lie or attempt to mislead the labeling process in any way). Also, our subjects made sure to provide good label boundaries for their data, so that the segmentation of activities is correct in our dataset. For the experiments involving incorrect boundaries, we intentionally disrupt boundaries between classes in our data to simulate real scenarios and then evaluate against our original data.

Category	Labels
Working	working, office, name of the building/room
Driving	driving, car, vehicle
Transport	bus, vehicle, cab, airplane
Eating	eating, lunch, dinner, kitchen, Dirt Cowboy, Quiznos, Boloco, Ramuntos, Five Olde, Novack
Shopping	shopping, supermarket, grocery store
Gym	gym

Table 1: Audio dataset

Domain	Features
Time	ZCR, RMS, low energy frame rates
Frequency	Spectral entropy, flux, centroid, bandwidth, normalized phase deviation, band energy
Cepstral	13 MFCC with DC component removed

Table 2: Features extracted from audio dataset

Data processing and feature extraction

To calculate similarity scores and perform classification, we extract a set of features from the recorded audio. We choose acoustic features that emphasize important characteristics of the data and have been used successfully in other mobile audio sensing applications. Our feature set contains 39 features, computed from the time, frequency, and cepstral domains. Table 2 gives an overview of these features.

Of the time-domain features, the root mean squared (RMS) value is a measure of the average amplitude of a frame, the zero-crossing rate (ZCR) is the number of time-domain zero-crossings within a frame and correlates with the frequency content of a signal. Low energy frame rate is the number of frames within a window that have an RMS value less than 50% of the mean RMS for the entire window (Scheirer and Slaney 1997).

In the frequency domain, the spectral entropy gives cues about the frequency pattern of the sound; spectral flux is the L_2-norm of the spectral amplitude difference vector of two adjacent frames and measures the change in the shape of the spectrum; the spectral centroid is the balancing point of the spectral power distribution; spectral bandwidth (Li et al. 2001) is the width of the range of the frequencies that the signal occupies; normalized phase deviation, introduced in (Dixon 2006), shows the phase deviations of the frequency bins in the spectrum, weighted by their magnitude.

Finally, we use coefficients of the Mel-frequency cepstral domain, a very commonly used feature in speech recognition systems that mimics human perception (Peltonen et al. 2002) (Mckinney and Breebaart 2003).

As we deal with large datasets and seek to summarize the sound of high-level activities, we chose a fairly long frame length (window of samples used to calculate features) compared to that common in speech applications: a half-second, with non-overlapping frames.

CGL Stages

Here we explain the experimental procedure for each stage of CGL. According to the framework, user-contributed data is initially grouped based on user-provided labels, the data in these groups is then clustered during the intra-class similarity step. Classifiers are trained based on the clusters found with similar classes then merged together before a final training step occurs to complete the process.

Intra-class similarity measurements. We start by clustering the data using Euclidean k-means, determining k as described previously. Note that this procedure can pick $k = 1$, and often does.

Figure 2 shows the result of principal component analysis (PCA) on the driving data. The figure clearly demonstrates three distinct clusters and in fact k-means also splits the driving class into three clusters but operates on the full 39-dimensional data. Table 3 lists the classes that had more than one cluster.

Classifier training. The clusters produced by the previous step are then used to train boosted decision stump classifiers. Assuming that a class C gets broken into subclasses by the previous step, we construct a classifier for C in the following manner:

- For each subclass, consider points within its cluster as positive examples and points outside (including other subclasses of the same class) as negative examples.
- Run 10 iterations of Adaboost on the above to get a classifier for each subclass, making sure to give the positive and negative classes equal weight.
- The output of our classifier for class C is the disjunction of the outputs of each of its subclasses' classifiers.

For example, if a class is found to have three subclasses, we train one boosted decision stump classifier for each of these. The compound classifier returns true for a given sample if (and only if) any of the three classifiers returns true on that sample.

Inter-class similarity measurements. After obtaining the set of classes and classifiers from the split stage, we determine the inter-class similarity between each pair of classes. Given two classes A and B and their respective classifiers C_A and C_B, we compute the similarity as follows:

- Run classifier C_A on B's data and classifier C_B on A's data
- Compute the confusion matrix conf_{AB} for C_A's when applied to B's data, and vice versa
- Compute the f-score of the combined confusion matrix $\mathrm{conf}_{AB+BA} = \mathrm{conf}_{AB} + \mathrm{conf}_{BA}$
- The combined f-score is used as the similarity measure between A and B

The similarity matrix thus generated can be seen as a complete graph with edge weights corresponding to pairwise similarity. Raising this matrix to a power simulates transitivity of similarity. For example, raising it to the second power "walks" one step over all the edges, accentuating similar elements and de-emphasizing dissimilar ones.

Based on the similarity matrix, we chose a conservative threshold experimentally chosen to favour merges only for highly similar classes.

Classifier re-training. After classes are merged, classifiers for the new categories are trained using the same

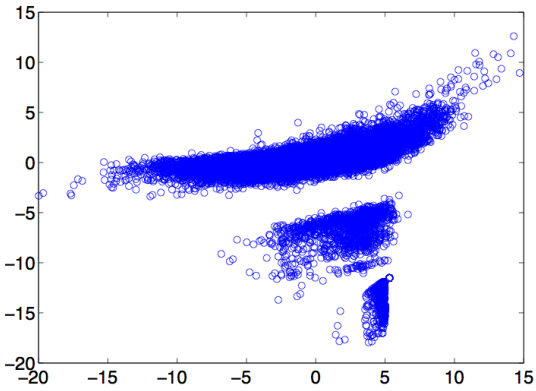

Figure 2: Two-dimensional PCA on driving data, showing three clear clusters.

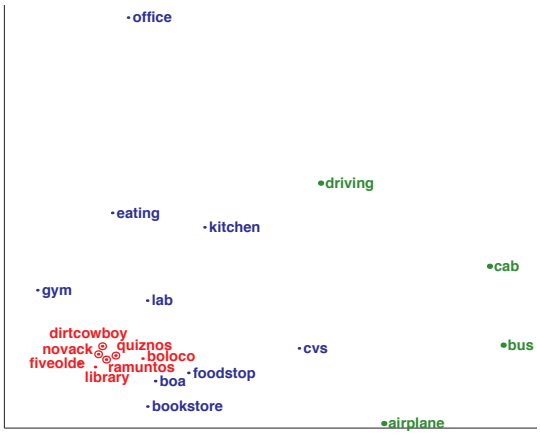

Figure 3: Multidimensional Scaling results for similarity between classes. MDS axes have no meaningful units.

boosted decision stump classifier. If classes A and B get merged in the previous step, we construct a classifier C_{AB} in the following manner:

- Consider points in A *or* B as positive examples and points outside as negative examples.
- Run 10 iterations of Adaboost on the above to get the parameters for classifier C_{AB}.
- The output of the classifier on A, B, or AB is simply the classification output of C_{AB}.

Class	k	f**-score**		**Precision**		**Recall**	
		pre	*post*	*pre*	*post*	*pre*	*post*
Driving	3	80.4	82.3	89.4	89.2	73.0	76.5
Eating	2	64.4	60.6	70.2	68.1	59.5	54.6
Gym	2	83.7	85.6	83.4	82.0	84.0	89.4
Lab	3	81.5	81.1	78.5	71.3	84.6	93.9
Office	2	85.4	85.5	77.8	76.9	94.8	96.3

Table 3: Performance pre and post splitting on classes that contained multiple clusters.

Class	f**-score**		**Precision**		**Recall**		**Accuracy**	
	pre	*post*	*pre*	*post*	*pre*	*post*	*pre*	*post*
eating	87.5	90.2	84.6	88.5	90.6	92.0	87.1	90.0
communal places	88.6	91.7	85.6	89.2	91.8	94.4	88.2	91.5
transportation	93.6	83.5	95.4	93.4	91.9	75.6	93.7	85.1

Table 4: Performance before and after for two examples where CGL will merge two subsets of a tight cluster (visible in figure 3). In the third row we see the result of merging related sub-clusters that CGL would not select for merging, although all subclusters are associated with transportation merging the results in worse performance.

Scenario	**Precision**		**Recall**		**Accuracy**		f**-score**	
	naive	*CGL*	*naive*	*CGL*	*naive*	*CGL*	*naive*	*CGL*
gym/driving	76.4	97.2	69.5	93.6	69.5	95.5	67.4	95.4
eating/lab	81.7	77.2	81.7	88.2	81.7	81.1	81.7	82.3
airplane/bus	91.4	99.0	91.9	99.8	91.6	99.4	91.7	99.4

Table 5: Performance under the class boundary errors experiment for a model using CGL relative to one that does not.

Experimental Results

Two common types of human error from low-commitment users during labeling are: (i) inconsistent class definitions between people since they are free to use their own definitions and (ii) unreliable boundaries that mark the start and end of classes that often occur due to user distraction. To study the ability of CGL to cope with these deficiencies we perform individual experiments, each of which focuses on one of these sources of error. In all experiments we evaluate CGL using standard five-fold cross-validation and tables 3 and 5 show the mean of the folds. As performance will not change for unsplit classes, we provide the average performance numbers: 88.7% accuracy and 89.3% f-score.

Intra-class similarity and splitting performance. In our experimental data we find frequent disagreement of class definition occurs within our users (e.g., users providing different labels for the same activity). Table 3 and 4 demonstrate the ability of CGL to handle this type of human error. Table 3 shows classification performance for both the user-provided groupings (*pre*) and CGL's split groups (*post*). For the majority of these classes, we see an improvement in the overall performance as measured by the f-score—usually the best number to use for evaluating whether the classifier is correctly recognizing the true examples and rejecting the false ones. In particular, the recall numbers go up significantly for all but one class (eating). But as we will see later, eating may be a candidate for merging with other classes that are capturing eating related events (i.e., the classes corresponding to various eateries).

Inter-class similarity and merging performance. To evaluate the empirical result of inter-class similarity, we apply non-classical multidimensional scaling (MDS) to the similarity matrix and plot the result in figure 3. Distances between points in this figure are proportional to the differences in the similarity. The figure shows a clump of highly similar eateries in the bottom-left region as well as some fairly dis-

similar transportation classes. The strange position of the 'library' class in the plot may be explained by the fact that our library includes a cafeteria area ('novack', also present in the plot). Additionally, we computed the similarity between the two eating subclasses and the various eateries. One of the eating subclasses is quite similar to the eateries (f-score = 0.70) and is a potential candidate for merging while other subclass is completely dissimilar (f-score = 0.0).

Performance of regrouping. To calculate the performance of the post-split (*post*) classifiers, we consider the disjunction of each subclass classifier's output. To evaluate the benefits of merging, we evaluate individual classes' performance before merging and see if retraining merged classes improves the original performance. Table 4 shows that performance increases consistently for merging classes CGL determines to be similar. In this table each class refers to more than one user-provided grouping with the following associations in use, obtained by picking all classes that exceed a threshold of similarity: eating $\rightarrow$ { dirtcowboy, novack, quiznos, ramuntos }, communal places $\rightarrow$ { dirtcowboy, novack, quiznos, ramuntos, boloco, fiveolde, library }. In contrast, if semantically similar classes like transportation $\rightarrow$ { cab, airplane, driving, bus }, that our technique considers dissimilar are merged blindly, the performance decreases significantly, as shown in the third row of table 4.

Handling boundary errors. To measure the impact of pronounced class boundary errors we perform an experiment which uses the original data set but synthetically increases class boundary errors. We deliberately splice additional audio from a different activity into a class segment identified by the user. We base each of these synthetic examples of class pollution off naturally occurring scenarios we observe during the experiment. We present three scenarios in table 5 such as: data from the gym is mistakenly mixed with data sampled during the drive home or data acquired while eating is accidently combined with data from in the lab. We compare the performance of CGL to boosted decision stump classifiers that treat the labels from users at face value. In the first scenario in the table, "gym/driving", CGL outperforms the naïve benchmark's accuracy by 41.5% and its other metrics by a similar margin. In contrast, CGL's accuracy is marginally outperformed in the second scenario "eating/lab". The reason for this is that the gym and driving classes are very easily split due to their large similarity distance (see figure 3). Due to the CGL is cleanly able to separate this class pollution and so it has little impact on the classifiers that are trained. However, in the case of "eating/lab" the classes themselves are have similar signatures (some of the eating data was even collected in the lab) so the improvement can only ever be minor.

Conclusion

In this paper, we introduced CGL as a novel framework for building robust context classifiers. The experimental results showed that CGL can overcome the limitations of existing techniques in coping with inconsistent labels, which are inevitable in real-world scenarios. By dynamically regrouping the classes that are modeled, CGL can recognize a wide range of classes more robustly than the conventional "train in a controlled environment then deploy" approach. Furthermore, user contributions can be intelligently shared to minimize unnecessary duplication of effort. Our work on CGL raises a number of open questions that directs our future work. The results presented, although promising, were validated using a community of five people, not hundreds, and the experiments were conducted offline. There are both implementation and algorithmic challenges to deploying CGL at a large scale. In particular, repeated retraining will not be scalable for a mobile inference systems. We plan to investigate online learning algorithms (Blum 1998; Oza and Russell 2001) to reduce computational demands. The CGL paradigm assumes that the users provide some labeled training data. Sensor data is straightforward to collect but labels are considerably harder because user interaction is required (Horvitz and Apacible 2003; Horvitz, Koch, and Apacible 2004; Fogarty et al. 2005). A key challenge in context modeling, which we plan to address, is to automatically detect opportune moments when users would be willing to label small chunks of their data. In summary, we plan to develop online and active learning techniques to address scalability and labeling challenges with the eventual goal of creating a large-scale CGL deployment.

References

Abdelzaher, T.; Anokwa, Y.; Boda, P.; Burke, J.; Estrin, D.; Guibas, L.; Kansal, A.; Madden, S.; and Reich, J. 2007. Mobiscopes for human spaces. *IEEE Pervasive Computing* 6:20–29.

Bao, L., and Intille, S. S. 2004. Activity recognition from user-annotated acceleration data. In Ferscha, A., and Mattern, F., eds., *Pervasive*, volume 3001 of *Lecture Notes in Computer Science*, 1–17. Springer.

Bishop, C. M. 2007. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition.

Blanke, U., and Schiele, B. 2009. Daily routine recognition through activity spotting. In *LoCA '09: Proceedings of the 4th International Symposium on Location and Context Awareness*, 192–206. Berlin, Heidelberg: Springer-Verlag.

Blum, A. 1998. On-line algorithms in machine learning. In *Developments from a June 1996 seminar on Online algorithms*, 306–325. London, UK: Springer-Verlag.

Burke, J.; Estrin, D.; Hansen, M.; Parker, A.; Ramanathan, N.; Reddy, S.; and Srivastava. 2006. Participatory sensing. In *In: Workshop on World-Sensor-Web (WSW): Mobile Device Centric Sensor Networks and Applications*.

Campbell, A. T.; Eisenman, S. B.; Lane, N. D.; Miluzzo, E.; and Peterson, R. A. 2006. People-centric urban sensing. In *WICON '06: Proceedings of the 2nd annual international workshop on Wireless internet*, 18. New York, NY, USA: ACM.

Choudhury, T.; Borriello, G.; Consolvo, S.; Haehnel, D.; Harrison, B.; Hemingway, B.; Hightower, J.; Klasnja, P.; Koscher, K.; LaMarca, A.; Landay, J. A.; LeGrand, L.; Lester, J.; Rahimi, A.; Rea, A.; and Wyatt, D. 2008. The mobile sensing platform: An embedded system for activity recognition. *Appears in IEEE Pervasive Magazine - Special Issue on Activity-Based Computing* 7(2):32–41.

Dixon, S. 2006. Onset detection revisited. In *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-06)*, 133–137.

Fogarty, J.; Hudson, S. E.; Atkeson, C. G.; Avrahami, D.; Forlizzi, J.; Kiesler, S.; Lee, J. C.; and Yang, J. 2005. Predicting human interruptibility with sensors. *ACM Trans. Comput.-Hum. Interact.* 12(1):119–146.

Friedman, J.; Hastie, T.; and Tibshirani, R. 1998. Additive logistic regression: a statistical view of boosting. *Annals of Statistics* 28:2000.

Horvitz, E., and Apacible, J. 2003. Learning and reasoning about interruption. In *ICMI '03: Proceedings of the 5th international conference on Multimodal interfaces*, 20–27. New York, NY, USA: ACM.

Horvitz, E.; Koch, P.; and Apacible, J. 2004. Busybody: creating and fielding personalized models of the cost of interruption. In *Proceedings of CSCW '04*, 507–510. New York, NY, USA: ACM.

Huynh, T.; Blanke, U.; and Schiele, B. 2007. Scalable recognition of daily activities with wearable sensors. In *3rd International Symposium on Location- and Context-Awareness (LoCA)*.

Huynh, T.; Fritz, M.; and Schiele, B. 2008. Discovery of activity patterns using topic models. In *UbiComp 2008*.

Krause, A.; Horvitz, E.; Kansal, A.; and Zhao, F. 2008. Toward community sensing. In *Proceedings of IPSN '08*. Washington, DC, USA: IEEE Computer Society.

Lane, N. D.; Lu, H.; Eisenman, S. B.; and Campbell, A. T. 2008. Cooperative techniques supporting sensor-based people-centric inferencing. In Indulska, J.; Patterson, D. J.; Rodden, T.; and Ott, M., eds., *Pervasive*, volume 5013 of *Lecture Notes in Computer Science*, 75–92. Springer.

Lester, J.; Choudhury, T.; Kern, N.; Borriello, G.; and Hannaford, B. 2005. A hybrid discriminative/generative approach for modeling human activities. In *In Proc. of the International Joint Conference on Artificial Intelligence (IJCAI*, 766–772.

Lester, J.; Choudhury, T.; and Borriello, G. 2006. A practical approach to recognizing physical activities. In Fishkin, K. P.; Schiele, B.; Nixon, P.; and Quigley, A. J., eds., *Pervasive*, volume 3968 of *Lecture Notes in Computer Science*, 1–16. Springer.

Li, D.; Sethi, I. K.; Dimitrova, N.; and McGee, T. 2001. Classification of general audio data for content-based retrieval. *Pattern Recogn. Lett.* 22(5):533–544.

Mahdaviani, M., and Choudhury, T. 2007. Fast and scalable training of semi-supervised crfs with application to activity recognition. In *In Proc. of the Advances of Neural Information Processing Systems 20 (NIPS 2007)*.

Mckinney, M., and Breebaart, J. 2003. Features for audio and music classification. In *Proceedings of the International Symposium on Music Information Retrieval*, 151–158.

Milligan, G. W., and Cooper, M. C. 1985. An examination of procedures for determining the number of clusters in a data set. *Psychometrika* 50:159–179.

Networks, S. 2008. Website. `http://www.sensenetworks.com/`.

Oza, N. C., and Russell, S. 2001. Online bagging and boosting. In Jaakkola, T., and Richardson, T., eds., *Eighth International Workshop on Artificial Intelligence and Statistics*, 105–112. Key West, Florida. USA: Morgan Kaufmann.

Peltonen, V.; Tuomi, J.; Klapuri, A.; and Jyri. 2002. Computational auditory scene recognition. In *In IEEE Intl Conf. on Acoustics, Speech, and Signal Processing*, 1941–1944.

Scheirer, E., and Slaney, M. 1997. Construction and evaluation of a robust multifeature speech/music discriminator. In *ICASSP '97: Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97)-Volume 2*, 1331. Washington, DC, USA: IEEE Computer Society.

Stikic, M., and Schiele, B. 2009. Activity recognition from sparsely labeled data using multi-instance learning. In *Proceedings of LoCA '09*, 156–173. Berlin, Heidelberg: Springer-Verlag.

Stikic, M.; Van Laerhoven, K.; and Schiele, B. 2008. Exploring semi-supervised and active learning for activity recognition. In *In Proc. of IEEE International Symposium on Wearable Computing*.

Thad, D. M.; Minnen, D.; Starner, T.; Essa, I.; and Isbell, C. 2006. Discovering characteristic actions from on-body sensor data. In *In Proc. of IEEE International Symposium on Wearable Computing*, 11–18.

Torralba, A., and Murphy, K. P. 2007. Sharing visual features for multiclass and multiview object detection. *IEEE Trans. Pattern Anal. Mach. Intell.* 29(5):854–869. Senior Member-Freeman, William T.

Viola, P., and Jones, M. 2001. Robust real-time object detection. In *International Journal of Computer Vision*.