Improving Activity Classification for Health Applications on Mobile Devices using Active and Semi-Supervised Learning

Brent Longstaff, Sasank Reddy, Deborah Estrin
Center for Embedded Networked Sensing
University of California Los Angeles
blongstaff, sasank@ucla.edu, destrin@cs.ucla.edu

***Abstract*—Mobile phones' increasing ubiquity has created many opportunities for personal context sensing. Personal activity is an important part of a user's context, and automatically recognizing it is vital for health and fitness monitoring applications. Recording a stream of activity data enables monitoring patients with chronic conditions affecting ambulation and motion, as well as those undergoing rehabilitation treatments. Modern mobile phones are powerful enough to perform activity classification in real time, but they typically use a static classifier that is trained in advance or require the user to manually add training data after the application is on his/her device. This paper investigates ways of automatically augmenting activity classifiers after they are deployed in an application. It compares active learning and three different semi-supervised learning methods, self-learning, En-Co-Training, and democratic co-learning, to determine which show promise for this purpose. The results show that active learning, En-Co-Training, and democratic co-learning perform well when the initial classifier's accuracy is low (75-80%). When the initial accuracy is already high (90%), these methods are no longer effective, but they do not hurt the accuracy either. Overall, active learning gave the highest improvement, but democratic co-learning was almost as good and does not require user interaction. Thus, democratic co-learning would be the best choice for most applications, since it would significantly increase the accuracy for initial classifiers that performed poorly.**

I. Introduction

As mobile phones become increasingly pervasive and sophisticated, they are being used for a greater variety of applications. With their GPS speed and accelerometer motion sensors, they can be used to monitor activity levels for health applications. For example, Ambulation [11] is an application that uses activity classification to monitor mobility patterns over time to help doctors accurately determine the progress of ambulatory patients. This type of system automatically determines the user's mobility mode (still, walking, running, etc). The user's activity data stream can be primary data, context/metadata, and user interface input.

As primary data, activity levels can indicate disease progression and clinical care plan efficacy in the context of heart disease, neuromuscular disease, and mental illness. Activity level tracking also provides quantitative feedback for health behavior changes, as pedometers are currently used [4]. Activity is also a good metric by which to compare the efficacy of rehabilitation treatments for stroke, hip replacement, and other mobility-impacting treatments.

Activity data are also useful as contextual data or metadata. They provide context for other health measures such as physiological self-tests (blood pressure, blood glucose, weight) as well as reporting of symptoms and side effects. Finally, activity traces can improve the user interface mechanisms across a range of applications by increasing the relevance and adherence of its users. For example, they could be reminded or triggered for action or input at a convenient moment, or a moment of interest to the study. Because of the importance of this activity data stream, we investigate how to improve the performance of activity classification using smartphones.

Smartphones classify activities using models created by machine learning algorithms. The machine learning process uses training data to create a model for the different activities. These data serve as examples to the machine learning algorithm, so it can associate certain attributes of the data with each activity. Each training data point includes the label of its associated activity as well as the attributes that are chosen as indicators of the activity. In this case, activity classification uses speed and acceleration data, which are indicative of the user's motion, which is linked to his/her activity. After a machine learning algorithm uses the training data to create a model, it can be used to classify unlabeled data to determine what activity was being performed when the data were sampled.

In the past, activity classifiers for mobile devices have been statically created ahead of time and then used as-is for classification. Labeled data are collected and used to build the model, which is then used in applications. To work well for various users, classifiers must be robust to variations in how each user does the activities. This is typically achieved by training on data from multiple users, so that the classifier is not overtrained towards any one user's particular data. Applications like Apple's Nike+iPod [1], supplement this by allowing the user to calibrate the classifier, but this requires the user to indicate which activity they are calibrating and then doing it, which is the same process as the original training of a classifier.

This paper investigates ways of improving the classification model even after the user has begun using the activity classifier. The reason for this is twofold: first, it is time-consuming and difficult to collect all the training data necessary to create

PERVASIVEHEALTH 2010, March 22-25, Munchen, Germany
Copyright © 2010 ICST 978-963-9799-89-9
DOI 10.4108/ICST.PERVASIVEHEALTH2010.8851

a robust classifier that is accurate out-of-the-box, and second, even if a classifier is trained on multiple users, it can still improve by adapting itself to the particular user. This work investigates methods of further training classifiers after a user begins to use them, using active and semi-supervised learning. We compare versions of self-learning [14], democratic co-learning [15], En-Co-Training [5], and active learning [6] to determine which has the most potential for improving the accuracy of mobile activity classifiers.

II. Related Work

Activity classification has been an interesting field of research in machine learning. Most of the work has focused on creating a static classifier, which would then be used for activity classification as-is. For example, Bao and Intille [2] developed algorithms to classify physical activities using data from accelerometers worn on different parts of the body. They collected user-annotated data by asking participants to perform a series of everyday tasks. They then trained classifiers on these user-annotated data, using mean, energy, frequency-domain entropy, and correlation of acceleration data. The features were calculated on an overlapping sample window of several seconds (each sample appears in two windows). Decision table, IBL, C4.5, and naïve Bayes classifiers were tested, and the C4.5 decision tree classifiers were found to be the most accurate. The classifier was tested both by including and excluding data from the test user in the training data. While some activities were recognized well with subject-independent training data, others required user-specific training data to be accurate, and suggested the need for further study of the power of user-specific training sets.

Lester et al.[7] developed a personal health activity recognition system using multimodal sensor devices at 3 locations on the users' bodies: the waist, shoulder, and wrist. The devices used a microphone, visible and IR light sensors, an accelerometer, a compass, a barometer, and sensors for detecting temperature and humidity. The data from these were used to compute 651 features, of which the top 50 were selected for classification. Static classifiers were used to provide inputs for hidden Markov models, which recognize activities in continuous time chunks. They compared the accuracy of the activity classifier when it was trained on a varying number (one to twelve) of users and found that their classifier works well out of the box if trained on a larger set of people. It performed well even when the users whose data were used for testing was not in the training set. However, when training with data from users in the test set, accuracy was higher, showing that personalized training data can improve accuracy.

Several different methods of improving an existing classifier have been investigated [17]. One type, self learning, has been shown to work for text analysis. Yarowsky [14] developed an algorithm for classifying ambiguous word meanings, such as "plant," which can be a life form or a manufacturing facility. The classifier uses a small amount of labeled "seed" data to train a classifier, which then is used to classify the unlabeled data. Confidently classified samples are then added to the seed set and the classifier is augmented. This process is repeated iteratively until the algorithm converges on a stable set of training samples. If a previously added example drops below the expected confidence with a later version of the classifier, it is removed from the training set to remove initial misclassifications. This method was able to achieve 97% accuracy, an improvement to the 92% given by the existing Schütze Algorithm.

Self-learning is also applied to image processing by Li et al. [8], who introduced OPTIMOL, an algorithm which uses Bayesian incremental learning to simultaneously build datasets and learn the model describing them. It runs the classifier on new images and accepts some that are selected by the classifier. It then augments the dataset with the accepted new data, and trains a new classifier on only these new data. It then repeats the process. During each iteration, OPTIMOL only accepts some of the images that the classifier selects in order to avoid overly favoring images with a large resemblance to the current ones, which would result in overspecialization. Also, the classifier is only trained on the new data chosen in the current iteration so that OPTIMOL can work with very large datasets without having memory issues. In tests, OPTIMOL achieved near-human accuracy in accurate dataset collection.

Another type of algorithm, co-learning, was first developed by Blum et Mitchell [3]. Unlike self-learning, co-learning is a multi-view semi-supervised algorithm. Co-learning uses two classifiers, each trained on a different view, with the strong assumption than each view is independently sufficient for classification. The views must also not be perfectly correlated. For example, web pages might be classified both by words on the page and by the text of hyperlinks to that page on other webpages. First, co-learning trains the two classifiers with the labeled data. Both were naïve Bayes classifiers. These data include all the features for both views. The classifiers then iteratively label unlabeled data, and each classifier adds its most confident predictions to the training set. Both classifiers are retrained with the augmented data. The co-training method was tested with the aforementioned web page example, and had fewer than half the errors of a simple supervised training method.

Guan et al. introduced En-Co-Training [5], which modifies Blum's algorithm to work without requiring multiple views. It uses three classifiers that are trained on the same view of the data, and relies on different machine learning methods to create the diversity required for co-learning to perform well. It adds unlabeled data to the training set when all three classifiers agree on the prediction. By using three classifiers instead of the two used by Blum, they can use majority voting for predictions and also ensure a higher degree of confidence in the samples that are added to the training set. They implemented this with activity classification using 40 accelerometers strapped to the users' legs and found that it resulted in a lower error rate than using each classifier separately. Just using the voting method to decide using three classifiers helped, but the semi-supervised learning also contributed to decreasing the error rate as well.

Zhou and Goldman[15] have a method called democratic

Digital Object Identifier: 10.4108/ICST.PERVASIVEHEALTH2010.8851
http://dx.doi.org/10.4108/ICST.PERVASIVEHEALTH2010.8851

co-learning, a single-view semi-supervised technique that uses multiple classifiers with different inductive bias. These classifiers are trained on the same data to vote for predictions for unlabeled data. Their predictions are used to label unlabeled data which are then added to the training sets of the classifiers that voted differently than the majority. This approach is different from co-learning because it is single view instead of multi-view. It relies on the difference in inductive bias instead of different feature sets. This relaxes the restrictive constraint imposed by [3], so it can be used when there are not multiple sufficient and redundant feature sets. They also presented another use for democratic classifiers in active learning. This method uses multiple classifiers to make a prediction on an unlabeled sample and takes their confidence-weighted vote entropy to determine the priority for active sampling. The greater the disagreement among the classifiers is, the less confident the combined prediction is, so the priority of prompting the user for a label is higher.

Zhou and Li [16] describe tri-training, which overcomes [3]'s requirement for multiple sufficient and redundant features sets as well. Tri-training first generates three different classifiers with the same labeled training data, and then uses them to classify unlabeled data. When two of them agree on a prediction, they label the example with their prediction and augment the third classifier with the newly labeled example. They tested the tri-training algorithm using J4.8 decision trees, BP neural networks, and naïve Bayes classifiers as the three classifiers. When tested on UCI data sets, it had a lower average classification error rate than self-training and co-training, although the co-training was not done under ideal circumstances, since there were not two redundant, sufficient views. Instead, the features were randomly partitioned.

Unlike self-learning and co-learning, active learning requires user input. Thus, the challenge changes from choosing the most accurately classified samples to deciding which samples to ask the user to label. Kapoor and Horvitz [6] compare several methods of determining when to prompt the user for data labels, the goal being to prompt the user to label a data sample when the value of the label justifies the cost of interrupting the user. The methods used were random probe, uncertainty probe (uncertain classification), decision-theoretic probe, which weighs the costs as well as the benefits of the probe, and decision-theoretic dynamic probe, which has different models for different contexts. The random probe issues probes at random times. The uncertainty one uses a predictive model on the data collected up to that point and issues probes when the classifier's prediction in the current situation has low confidence. The decision-theoretic probe takes into account the user's state (busy or not) as well as the benefit of the probe. When the benefit outweighs the predicted cost, a probe is sent. Finally, the decision-theoretic dynamic probe extends the decision-theoretic one by adding flexibility across different contexts that the model may not necessarily recognize. To test the different methods, a program named BusyBody was installed on subjects' PCs. It would issue probes asking the subjects if they were busy or not to build a model to predict when the user is highly uninterruptable. Versions of BusyBody with the four different probing policies were given to different users. The annoyance the interruptions caused and the accuracy of the generated model were measured. The best methods built a better model and were less annoying to users. The decision-theoretic ones performed better because they considered the cost of issuing a probe, rather than just when it was beneficial to get a label. Uncertainty probing was slightly better than random probing because it prompted for the most valuable labels even though it did not take into account the cost.

Stikic et al. [12] examined semi-supervised (both self-training and co-training) and active learning for improving activity recognition. For self-training, a classifier was built with labeled data, and then iteratively augmented with the samples having the fifty most confidently predictions from each iteration. The co-training algorithm was multi-view, like in [3], using two types of sensors, each of which is sufficient for classification. They used accelerometers and infra-red motion sensors. Co-training performed better than self-learning, but the latter usually increased performance as well over the starting classifier. The active learning algorithm had two ways of triggering a user prompt, both based on classification uncertainty. The first was to prompt for the classifications which had the lowest confidence, and the second was two prompt when the two classifiers (from co-learning) disagreed. Performance was better with the former method, but both provided accuracy improvements. Unlike the approach analysed in this paper, these active and semi-supervised learning algorithms were only evaluated as a way to facilitate the creation of the initial classifier by requiring less training, rather than as a way to improve a classifier in use.

Lu et al. also worked on a machine learning system for mobile devices. They created SoundSense [9], an application which uses sound recorded by the microphone on iPhones to determine the context and sound type. It starts with some general classes of sound contexts, such as music or talking. It uses an unsupervised learning algorithm to discern new classes of context and prompts the user to identify them. For example, if the user is driving and the phone detects from the noises associated with driving are a different from or a subclass of the existing classes of sound, it will prompt the user, who will input driving as the name. From then on, the phone will be able to classify driving by the sound. SoundSense, like this work, uses the phone for machine learning, but for a different purpose. They used unsupervised learning to distinguish new classes for classification, while here it is used to improve the classification accuracy within predetermined classes.

In this paper, we compare the performance of semi-supervised and active learning methods for personalization of activity classifiers. Activity classification can work well out of the box, but training it for each user can improve accuracy, so an automatic or convenient way of making the classifier more personalized would be useful. We implement a self-learning technique similar to Stikic et al.'s method, where the most confident predictions are used to label unlabeled data and

Digital Object Identifier: 10.4108/ICST.PERVASIVEHEALTH2010.8851
http://dx.doi.org/10.4108/ICST.PERVASIVEHEALTH2010.8851

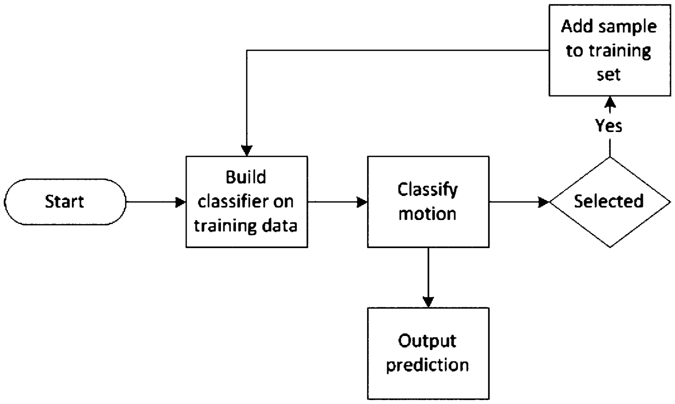

Fig. 1. Classifier improvement

augment the existing classifier. We implement two co-learning algorithms. The first is Guan's En-Co-Training. The other one is a version of Zhou and Goldman's democratic co-learning that is slightly modified to work on a mobile phone. Since we do not have two sufficient and redundant views, we could not use Blum's original co-learning method. Finally, we simulate the active learning method of prompting the user when the confidence of classification was low, similar to Stikic's method and Kapoor's uncertainty probe. Kapoor's decision-theoretic probing methods would not work in this case however, since the activity classifier needs to learn uncertain data from all of the mobility classes, and there is no additional busyness classifier. Thus, the decision to request a label is based solely on the confidence of the prediction.

III. Methods

A typical way of implementing activity classification in mobile systems is to first train a classification and then implement that classifier as static logic in the program. It does not change unless the software is updated. However, this does not allow the classification to adapt to the user or improve over time. Work in semi-supervised learning suggests that if learning continues after deployment of the software, it can improve the accuracy of the classification over time. These improvement algorithms collect new samples to add to the training data and the classifier is periodically retrained on the augmented training dataset. Figure 1 shows the basic logic of this process. In this paper the performance of several semi-supervised learning methods, as well as the potential of active learning, are compared in the context of activity classification.

A. Self-learning

The first of the semi-supervised learning methods investigated is self-learning. Self-learning employs a single classifier, which is used to classify unlabeled data. When the classifier's confidence in its prediction for a sample is high, it labels that sample with its prediction and adds it to the training set. If only the most confident predictions are used as labels, it should increase the accuracy of the classifier. In general, adding more data tends to increase the accuracy of a classifier. However, adding a data point with the wrong label can decrease it. To make the ratio of new training data with the correct label to those labeled incorrectly high enough to raise the overall accuracy, only the samples with the highest confidence are used.

B. Co-learning

Another type of semi-supervised learning method is co-learning, which uses multiple classifiers which can learn from each other. There are two different approaches considered here: En-Co-Training [5] and a method based on democratic co-learning [15]. Both of these are single-view adaptations of Blum's [3] co-learning algorithm; all the classifiers are initially trained on the same data, but differ in the machine learning method (for example, one may be a decision tree, while another could be a naïve Bayes classifier).

1) En-Co-Training: En-Co-Training is like self-learning but with two important differences. First, it uses the consensus of three different classifiers, rather than the confidence of one, to determine that it is confident enough with a prediction. It labels data for the training set when there is consensus and all three classifiers are retrained on the common training set. The second difference is that it takes advantage of having three classifiers and takes the majority vote to determine the prediction for each sample.

2) Democratic co-learning: Democratic co-learning also uses multiple classifiers, but has a different way of selecting and using new samples as training data than En-Co-Training does. Each classifier has its own separate data set, although all are initialized with the same starting data. Then the classifiers are run on new, unlabeled data. The majority classification of each sample is used to label it, and then the labeled sample is added to the training set of the classifiers whose predictions disagreed with the majority. Zhou and Goldman's version of democratic co-learning ran tests on these values to predict whether the potential noise of mislabeled data would be offset by the larger training set. The version implemented in this comparison instead uses the confidence of the predictions to indicate the priority, so the new training data will have the largest difference between the sum of the confidences of the majority predictions and the sum of the confidences of the dissenting predictions. This modification would allow a mobile phone to identify the samples to use for retraining without having the entire original training set stored on the phone. That way the phone could identify the samples for retraining, but offload the classification process to a server, which would receive the new samples from the phone, update the model, and send the updated classifier to the phone.

C. Active learning

Active learning, as opposed to semi-supervised learning methods like co-learning and self-learning, does not use predictions as labels. Rather, it chooses samples of interest and asks the user to label them manually. The data labeled by the user are then added into the training data to recreate the

Digital Object Identifier: 10.4108/ICST.PERVASIVEHEALTH2010.8851
http://dx.doi.org/10.4108/ICST.PERVASIVEHEALTH2010.8851

classifier. Samples are chosen based on the confidence of prediction, but instead of using those with a high confidence like self-learning, those with the lowest confidence are selected. Since the prediction is not going to be used as the label, there is no reason to want an highly confident prediction. Rather, active learning seeks to find the most informative samples so as to get the most benefit out of inconveniencing the user. Predictions that have a low confidence indicate that the model could benefit from that kind of example.

IV. Approach

This work used a simple activity classification scenario to compare the different semi-supervised learning methods. The possible activities were staying in one place, walking, and running. These activities were accessible to all participants, since they required no specific equipment (bicycle, car, etc.). The features used for classification were GPS speed and statistics on the magnitude of acceleration calculated once per second. These statistics were the mean, variance, and the FFT coefficients between 1 and 10 Hz, similar to those used by Reddy et al. [10] in their activity classifier. Reddy achieved a high level of accuracy using GPS and accelerometer features, so they were a good choice for the classifiers in this research.

The subjects who collected the data for the classifier used an HTC Android Dev Phone 1, which had an application that allowed them to keep track of the amount time they had recorded for each activity. First, 17 participants collected labeled training data for the base classifiers. This relatively large group size provided a variety of training data so the initial classifier can be more robust. Then, 15 other subjects collected data to use as the unlabeled data in the tests, although the data were collected with labels to determine the accuracy of the results. They collected 30 minutes of each activity, for a total of 90 minutes of data per participant. The walking and running paces were left to individual preference, as was the position (standing or sitting) of the still activity, since a real application should be able to detect activities as each user is accustomed to doing them. The phones were held in the hand or worn on the hip (belt or pocket) or armband of the participants. Participants were requested to consistently wear the phone in the same place on their bodies for all the activities, but users could individually choose where they preferred to keep it.

The machine learning algorithms used Weka's [13] implementation of machine learning algorithms. The performance of the semi-supervised learning methods were tested with several different sizes of initial training datasets. To compare the quality of the samples chosen by each method fairly, the same number of samples was added to the training data for each method. Each method ranks the samples it chooses according to its selection criteria. For instance, self-learning ranks by confidence (with high confidence coming first). Multiple iterations were tested, so that the training set could be updated after each part of the new data is classified to allow the algorithm to augment the classifier as it went along. For example, algorithm 1 shows the algorithm for testing self-learning. The co-learning algorithms classify with a vote since they use three classifiers. To discern the effect of this from the effect of the actual co-learning process, the performance of the classifier is measured both democratically and with only a decision tree. The democratic one would be used in practice, but the decision tree shows the effect of the augmented classifier without the added advantage of voting.

For self-learning and active learning, the machine learning method was a C4.5 decision tree. The co-learning algorithms use three types of classifier: a C4.5 decision tree, naïve Bayes classifier, and a support vector machine using the Sequential Minimal Optimization algorithm. All decision trees had a minimum leaf size of 10.

Algorithm 1 Self-learning test

```
for $i \in iterations$ do
  $newData \leftarrow unlabeled.subset(i)$
  for $sample \in newData$ do
    $prediction \leftarrow classifier.classify(sample)$
    $priority \leftarrow 1 - prediction.getConfidence()$
    $priorityQueue.add(sample, prediction, confidence)$
  end for
  for $j \leq newSamplesPerIteration$ do
    $trainingData.add(priorityQueue.pop())$
  end for
  $classifier.rebuild(trainingData)$
end for
```

V. Results

The comparison of the active and semi-supervised learning algorithms revealed several results. First, all but self-learning significantly increase the accuracy if the base classifier is around 80%. However, they did not improve on classifiers that already had an accuracy closer to 90%. Adding new data over multiple iterations instead of all at once increases their effect on the accuracy of the classifier. Table I shows the performance of the different methods. The "Unlabeled" column specifies what percentage of the total data used was unlabeled. The table gives the mean and 95% confidence interval of the change in accuracy between the original classifier and the one augmented with new data. These values reflect the increase or decrease of the percentage of the correctly classified instances, rather than the percentage increase of the original number of correct instances. For example, a value of 5% would mean that if the original classifier had 85% accuracy, the new one would have 90% correct. The top 480 points (about 10%) of new data chosen by the learning algorithms were added to the classifier. Evaluation was done using 10-fold cross-validation over the new data.

For six of the ten starting classifiers, the active learning and two of the three semi-supervised learning methods significantly boosted the classification accuracy. For the remaining four initial classifiers, most or all of the results do not deviate significantly from zero. The methods that improve the accuracy (active learning, En-Co-Training, and democratic co-learning) perform consistently for a given starting classifier;

Digital Object Identifier: 10.4108/ICST.PERVASIVEHEALTH2010.8851
http://dx.doi.org/10.4108/ICST.PERVASIVEHEALTH2010.8851

TABLE I
PERCENTAGE CHANGE FROM BASE CLASSIFIER WITH 480 NEW DATAPOINTS OVER EIGHT ITERATIONS AND A CONFIDENCE INTERVAL OF 95%

	Self-Learning	Active Learning	En-Co-Training		Democratic Co-learning	
Unlabeled	DT only	DT only	DT only	Democratic	DT only	Democratic
50%	$-1.27\% \pm 2.07\%$	$2.15\% \pm 2.85\%$	$-0.91\% \pm 2.15\%$	$-0.34\% \pm 2.67\%$	$-2.06\% \pm 3.08\%$	$-0.63\% \pm 2.85\%$
55%	$-5.35\% \pm 5.66\%$	$3.17\% \pm 4.87\%$	$-6.64\% \pm 6.46\%$	$0.67\% \pm 0.66\%$	$-1.46\% \pm 3.14\%$	$0.38\% \pm 0.87\%$
60%	$3.31\% \pm 4.41\%$	$17.13\% \pm 7.95\%$	$5.53\% \pm 5.29\%$	$13.05\% \pm 7.20\%$	$14.38\% \pm 8.31\%$	$15.07\% \pm 8.00\%$
65%	$0.05\% \pm 0.28\%$	$12.38\% \pm 7.28\%$	$0.88\% \pm 1.66\%$	$6.34\% \pm 3.43\%$	$8.59\% \pm 8.08\%$	$10.48\% \pm 6.34\%$
70%	$0.17\% \pm 0.54\%$	$9.35\% \pm 6.41\%$	$0.04\% \pm 0.58\%$	$5.04\% \pm 3.14\%$	$7.99\% \pm 5.76\%$	$8.41\% \pm 5.82\%$
75%	$3.31\% \pm 4.41\%$	$9.79\% \pm 6.44\%$	$1.65\% \pm 6.51\%$	$6.69\% \pm 4.61\%$	$9.03\% \pm 6.31\%$	$9.12\% \pm 6.31\%$
80%	$-0.02\% \pm 0.03\%$	$1.48\% \pm 2.31\%$	$-0.01\% \pm 0.03\%$	$1.14\% \pm 0.80\%$	$0.54\% \pm 1.40\%$	$1.03\% \pm 1.11\%$
85%	$1.38\% \pm 1.87\%$	$8.77\% \pm 6.57\%$	$0.23\% \pm 0.55\%$	$5.45\% \pm 3.51\%$	$7.80\% \pm 6.40\%$	$8.84\% \pm 6.12\%$
90%	$-0.63\% \pm 0.89\%$	$3.13\% \pm 4.50\%$	$0.10\% \pm 1.54\%$	$1.41\% \pm 1.56\%$	$0.51\% \pm 2.15\%$	$1.02\% \pm 1.95\%$
95%	$-1.74\% \pm 1.33\%$	$8.90\% \pm 5.03\%$	$1.82\% \pm 2.79\%$	$6.27\% \pm 4.08\%$	$8.72\% \pm 6.48\%$	$8.97\% \pm 6.56\%$

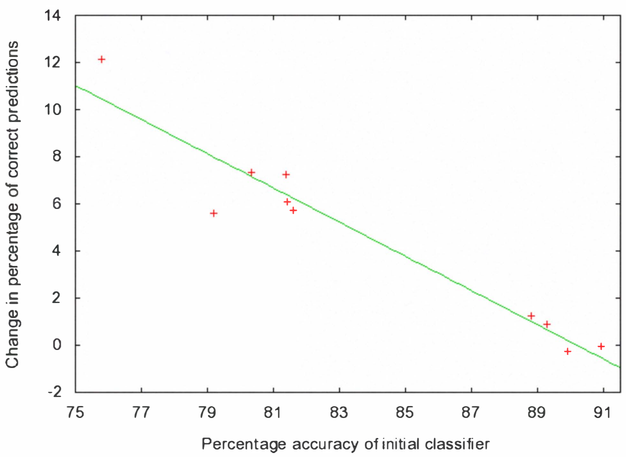

Fig. 2. Correlation between initial classifier accuracy and average increase in accuracy from the four methods

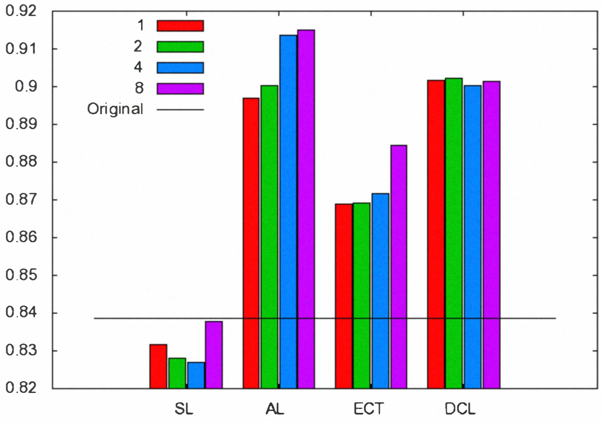

Fig. 3. Classification accuracy by method and number of iterations

either they all perform well or none of them do. This suggests that the potential effectiveness of the semi-supervised and active learning methods are highly dependent on the starting classifier. If the starting classifier is conducive to improvement, then they are more successful. As Figure 2 shows, the overall performance is correlated to the accuracy of the original classifier. When the initial accuracy is low (around 75-80%) to begin with, active and semi-supervised learning increase accuracy, whereas when it starts at around 90%, the improvement methods have little effect. Since all the methods (except self-learning, which has little effect) exhibit this trend, it is not specific to a particular algorithm. Rather, it shows that active and semi-supervised learning algorithms increase accuracy when the initial classifier has a lower accuracy, but if the accuracy is already high, then the algorithms have little effect. While this keeps these learning methods from producing a classifier that approaches perfection, they are still able to significantly increase performance when the existing classifier is in need of improvement.

Another detail to note in Table I is that the En-Co-Training algorithm only performs well when using the vote of the three classifiers instead of just the decision tree, indicating that the democratic classification, not just the co-learning, is responsible for the success of the method. On the other hand, democratic co-learning increases accuracy even when only classifying with the decision tree. The democratic classification process offers an added boost to performance, but democratic co-learning's semi-supervised learning algorithm that is responsible for most of the improvement.

A final point to consider in Table I is that democratic co-learning is quite competitive with active learning, which means that application developers can achieve comparable accuracy without the drawbacks of active learning. As a supervised learning method, active learning does not have to rely on the classifier's guesses, but it has the disadvantages of disturbing the user or missing data when the user is too busy to provide a label. When possible, an automatic process is preferable, so it is very fortunate that one of the semi-supervised learning algorithms provides similar performance.

In an application using one of these methods, the user would periodically run the algorithm to update the classifier with new data collected since the previous time. To evaluate the effect of multiple iterations, the algorithms were run with different numbers of iterations. Figure 3 shows the results with 1, 2, 4, and 8 iterations. The amount of new data added is the

Digital Object Identifier: 10.4108/ICST.PERVASIVEHEALTH2010.8851
http://dx.doi.org/10.4108/ICST.PERVASIVEHEALTH2010.8851

same regardless of how many iterations there are. For more iterations, a corresponding fraction of the new data is added. The original classifier's performance is shown by a horizontal line. For most of the methods, the more iterations, the higher the accuracy. This is because, even though the same amount of new data is being added, multiple iterations allow the classifier to improve while it is still selecting and labeling new points.

VI. Future Work

Another interesting area of investigation is the effect of semi-supervised and active learning on a classifier's accuracy when the user's way of performing activities differs from how it was done when the training data were recorded. For example, someone who has a limp or uses a walker would have a different gait from those of the people in the training set, so the classifier would be less accurate for them. In applications like Ambulation [11], this case would not be uncommon, since it is designed for ambulatory patients. Data for cases such as these would show if the personalization methods described in this work could help the classifier adjust to these variations.

VII. Conclusions

This paper tested the feasibility of using various semi-supervised and active learning methods to improve activity classification on mobile phones after application deployment. This would allow health and fitness monitoring applications to record the user's activity data stream with an increasing degree of accuracy as it adapts to each user. In cases where the original classifier's performance was around the 75-80% accuracy range, most of them had significant improvement over the original classifier, but when the starting accuracy was already high (about 90%) they did not. Self-learning never demonstrated any improvement, while active learning and both varieties of co-learning performed well, depending on the initial classifier. For any given starting classifier, either all three classifiers succeeded, or none did. On average, none of the methods (except self-learning in one case) showed a statistically significant decrease in accuracy, so an application could implement one of them for the possible improvement without a corresponding risk of losing accuracy. Although the drop could be significant for some individual users, the gain could be as well. This is nothing new to classification however, as the initial classifier will vary in accuracy between users even without applying semi-supervised or active learning methods. Finally, one of the most encouraging results is the fact that this version of democratic co-learning performs almost as well as active learning. Active learning is much more difficult to implement effectively and requires user interaction, which is a considerable drawback. Many patients would not want to burden themselves with the task, so it is very advantageous to have a semi-supervised learning method that can perform as well or better than supervised ones. Taking this into consideration, the algorithm that shows the most promise is democratic co-learning. However, it does have the downside of running three classifiers at once on the mobile device. If this becomes too energy-intensive or difficult, active learning could be used instead, but it would force the user to provide input. There is no reason to prefer En-Co-Training because it has the same requirements as democratic co-learning and does not increase performance as much. Overall, democratic co-learning is the best choice for medical applications, since it significantly increases accuracy without burdening the patient with additional interaction with the device.

VIII. Acknowledgments

This work is supported in part by NSF Cooperative Agreement #CCR-0120778 and NSF Grant #CNS-0627084. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding entities. We thank Google for their mobile phone hardware donation.

References

[1] Apple. Nike+ipod. http://www.apple.com/ipod/nike/.

[2] L. Bao and S. Intille. Activity recognition from user-annotated acceleration data. *Lecture Notes in Computer Science*, pages 1–17, 2004.

[3] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, page 100. ACM, 1998.

[4] S. Consolvo, D. McDonald, T. Toscos, M. Chen, J. Froehlich, B. Harrison, P. Klasnja, A. LaMarca, L. LeGrand, R. Libby, et al. Activity sensing in the wild: a field trial of ubifit garden. 2008.

[5] D. Guan, W. Yuan, Y. Lee, A. Gavrilov, and S. Lee. Activity recognition based on semi-supervised learning. In *Proceedings of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 469–475. IEEE Computer Society, 2007.

[6] A. Kapoor and E. Horvitz. Experience sampling for building predictive user models: a comparative study. 2008.

[7] J. Lester, T. Choudhury, and G. Borriello. A practical approach to recognizing physical activities. *Lecture Notes in Computer Science*, 3968:1–16, 2006.

[8] L. Li and L. Fei-Fei. Optimol: automatic online picture collection via incremental model learning. *International Journal of Computer Vision*, pages 1–22.

[9] H. Lu, W. Pan, N. Lane, T. Choudhury, and A. Campbell. SoundSense: scalable sound sensing for people-centric applications on mobile phones. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 165–178. ACM New York, NY, USA, 2009.

[10] S. Reddy, J. Burke, D. Estrin, M. Hansen, and M. Srivastava. Determining transportation mode on mobile phones. In *Proceedings of The 12th IEEE Int. Symposium on Wearable Computers*, 2008.

[11] J. Ryder, B. Longstaff, S. Reddy, and D. Estrin. Ambulation: a tool for monitoring mobility patterns over time using mobile phones. In *2009 International Conference on Computational Science and Engineering*, pages 927–931, 2009.

[12] M. Stikic, K. Laerhoven, and B. Schiele. Exploring Semi-Supervised and Active Learning for Activity Recognition. In *Proceedings of the 12th IEEE International Symposium on Wearable Computers (ISWC)*, 2008.

[13] I. Witten. *Weka: Practical Machine Learning Tools and Techniques with Java Implementations.* Dept. of Computer Science, University of Waikato, 1999.

[14] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics Morristown, NJ, USA, 1995.

[15] Y. Zhou and S. Goldman. Democratic co-learning. In *Proceedings of the 16th IEEE international conference on tools with artificial intelligence*, pages 594–202. Washington, DC: IEEE Computer Society Press, 2004.

[16] Z. Zhou and M. Li. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, pages 1529–1541, 2005.

[17] X. Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2007.

Digital Object Identifier: 10.4108/ICST.PERVASIVEHEALTH2010.8851
http://dx.doi.org/10.4108/ICST.PERVASIVEHEALTH2010.8851