An investigation into smartphone based weakly supervised activity recognition systems

William Duffy, Kevin Curran, Daniel Kelly, Tom Lunney

School of Computing, Engineering & Intelligent Systems, Ulster University,
MS Building, University of Ulster, Magee Campus, Derry BT48 7JL
Email: duffy-w@ulster.ac.uk

Abstract

With smart-devices becoming increasingly more commonplace, methods of capturing an individual's activities are becoming feasible. This is more generally performed through questionnaires or within unnatural environments bringing drawbacks in accuracy or requiring impractical conditions. This paper presents a simpler method of data collection which reduces the complications of typical activity data collection by collecting labels directly from a user. Instead of capturing activity beginning and end times, user requests are made at time intervals and labels are populated to feature vectors. These methods can provide a simpler method of data collection and could provide a solution to the annotation problem within activity recognition.

Keywords – Gaussian Means; Activity Recognition; Multiple-Instance Learning; Random Forest; Support Vector Machines

1. Introduction

Globally, an increasing number of people are using smartphones daily. Many of these devices contain embedded sensors which accurately measure acceleration and orientation, providing solutions to challenging issues associated with activity recognition.

Healthcare advancements have resulted in an aging population, where the elderly have become a larger percentage of the overall population [1], bringing with it associated rises in operating costs [2]. Therefore, methods of reducing these costs are advantageous, for instance, activity recognition for patient rehabilitation [3] and monitoring [4]. Additionally, activity recognition uses are not limited to the medical sector with systems being developed for exercise tracking [5] and biometrics [6]. While smartphone sensors allow for the simple capture of raw data, traditional supervised machine learning requires detailed labels or annotations in conjunction with the raw data. Raw sensor data is readily available however acquiring annotations is a laborious, costly and time consuming task [7]. This is both due to the complexities regarding setup of data collection experiments [8] and ensuring that annotations collected are sufficient and accurate. To ensure the accuracy of annotations provided, most of these experiments are performed within a laboratory where participant's movements can be closely monitored.

A potential solution to this annotation problem is to bring activity recognition into naturalistic settings and utilize embedded smart device sensors to provide data for discrimination of activities. Annotations can be collected from users via a diary study technique known as experience sampling [7]. This method of data collection is similar to the physician-administered questionnaires which the medical field currently uses for activity recognition [9]. However, an in the moment request from a user's device will be less likely to be affected by recall bias then questionnaires and questionnaires performance is relatively low compared with laboratory-based experiments [10]. However, as this solution provides much fewer and less accurate annotations than a standard supervised classifier, it could potentially affect classifier performance.

This paper sets out to show that even when weak and noisy annotations are collected via a heavily restricted collection process in comparison to a standard supervised training set, methods can be utilized to reduce the associated negative impact on classifier performance. While these methods are unlikely to fully equal that of a comprehensively annotated training set, they provide foundations for a practical approach to activity recognition. Weak supervision methodologies, rather than fully supervised learning, will be utilized to extract supervision signals, or training annotations, from these noisy sources with the overall goal of reducing the burden and cost associated with detailed annotation of activity training sets. The key contributions of this paper are evaluation of bespoke methods for activity recognition, using concepts from weak supervision and experience sampling.

Experiments will be performed using ideas from Multiple Instance Learning, a weak supervision methodology which places feature vectors into collections known as bags [11]. Simulated collection of bag level labels will be performed, resulting in each label being assigned too many feature vectors. As it cannot be known what activities are performed in the bag, it will be difficult to tell which features the bag level label applies to. A Gaussian Means [12] hierarchical clustering algorithm can be used to assign bag labels to feature vectors. Following this, we can investigate several methods to further increase bag classification accuracy. Validation methods will then be performed to test how well the weakly labelled training set performs in comparison to the best-case scenario of a fully annotated training set. The methodology is provided in section 3, section 4 outlines the results, section 5 discusses the results and finally, section 6 provides conclusions and some future work.

2. Related Work

Annotations have previously been gathered using experience sampling for activity recognition. Some of these have annotated a single feature vector [13], [14] whereas others have captured annotations for multiple feature vectors [7], [15]. Different structures of user requests have also been applied, ranging from asking what activity a user is currently performing [13] to the activity they have performed the most over a certain time frame [7]. Several methods have also been tested to populate these experience sampling captured annotations to a larger pool of unlabeled data. Multiple Instance Learning, which places multiple feature vectors in bags and uses modified classifiers which can handle these bags, has shown efficacy [7]. Other methods such as Graph Label Propagation [7] and Active Learning [15] have also been tested. Weakly supervised techniques running at a substantial experience sampling interval of 10 minutes allowed an increase in classifier accuracy from 70.2% to 84.8% by populating labels based on Euclidean distance [13]. Another consideration is that even with experience sampling, users may not accept a system if it is overly intrusive, for instance, making excessive user requests for input. While not tested specifically for activity recognition, methods of reducing user requests by employing a cost vs benefit ratio have been investigated [16]. Other work has tried to reduce the number of user requests by making predictions on future requests. This was performed by calculating the measure of confidence a classifier had in each prediction. This method was shown to reduce the number of user requests by up to 35% with insignificant effects on classifier accuracy [14]. Transition-aware systems have been tested previously, however, they have only been used in fully supervised systems [17]. This work approaches the issue differently by gathering user-provided weak labels in combination with zero-shot learning to find these transitions. Although this means that we will never have the contextual labels e.g. sitting-to-standing, it may be possible to infer them.

3. Methodology

3.1. Dataset

The dataset used was the Human Activities and Postural Transitions dataset [17], this dataset was deemed appropriate as it was captured on smart devices, which has an emphasis on naturalistic data collection. The dataset contains static and dynamic activities and the postural transitions between the static activities. The data is captured from 30 participants which range in age from 19-48 years. Data is collected using a smartphone which is attached to the participant's waist, both the accelerometer and gyroscope signals are captured at a rate of 50Hz. The distribution of labels in both the training and test set are evenly distributed except for the transitions, however, we expect this to have lower occurrences as they only happen for a few seconds at a time.

3.2. Feature Extraction

As certain activities can be difficult to differentiate on acceleration values alone [18], features are computed from both accelerometer and gyroscope data. Both embedded sensors capture tri-axial data providing 6 axes from which features can be computed. Raw values are applied to a median filter, and then de-noised using a low pass Butterworth filter. The filter had a corner frequency specified at 20Hz. A second Butterworth filter with a corner frequency of 0.3Hz is then applied to the accelerometer data; this splits the accelerometer data into body acceleration and acceleration due to gravity. This body acceleration, along with angular velocity is then used to calculate signal Jerk. Signal magnitude is also calculated from the body acceleration, gravity acceleration, body Jerk, body gyroscopic data, and the gyroscopic Jerk. This magnitude is calculated using Euclidean Norms. A fast Fourier transform is applied to body acceleration, body Jerk, body gyroscopic data, bodily acceleration magnitude, gyroscopic magnitude, and gyroscopic Jerk magnitude. From these signals we then calculate mean, standard deviation, median absolute deviation, maximum value, minimum value, signal magnitude area, energy measure, interquartile range, signal entropy, correlation coefficients between two signals, auto-regression coefficient, frequency component with the largest magnitude, mean frequency with a weight average, skewness, kurtosis, frequency interval energy of the Fourier transform window and the angle between vectors. These calculations result in a 561-width feature vector, each of which is captured from a sliding window of approximately 2.56 seconds [17].

3.3. Experience Sampling (ES)

Experience sampling is a type of diary study method which captures data from users, within an activity recognition system it provides a feasible method of capturing labels [7]. Previous work has used different methods of applying the labels gathered by experience sampling to feature vectors. In some cases, the label gathered has been directly applied to a single feature vector which was generated from the signal data at the time of the request [13]. While in another method the labels are applied to a "bag" of feature vectors [7]. One advantage of gathering a label for a single feature vector is that it ensures that the feature vector has the correct label. However, in reality, this approach could be problematic, the user may not actually be willing at that time to provide a label, and the physical movement of a device, while a label is being provided, could also cause issues in selecting the correct feature vector. However, gathering a label for a bag of feature vectors is much more feasible. For example, if the system is being used by someone monitoring a patient, it allows them to provide a label at any time within a time interval. However, this also leaves the difficult problem of estimating which feature vectors the label applies to within the bag. Obviously, requesting data from a user could cause problems with the user's acceptance of the system [16], so of these two methods of applying labels to data the “bagging” method will be used as it does not require the user to immediately respond.

Algorithm 1 - Find clusters with Gaussian Means

```
{Input: b = feature vector of bag to be clustered, s = g-means strictness value}
b_GM = GMEANS(b, s)
c = findBiggestCluster(b_GM)
return c
```

As we are collecting a single label from the user and have no way of knowing when one activity begins and another ends. Therefore, experience sampling will be simulated by contacting the original ground truth at certain intervals. These intervals are going to be pre-defined as 0.5, 1, 1.5, 2, 2.5, 10, 30 and 60 minutes. Ideally, to minimize the user burden associated with data collection, longer bags would be an advantage, but they could reduce the amount of data available to the classifier. With all likelihood, the activities will either be longer or shorter than the specified time interval, requiring some method to decipher which feature vectors within the bag correctly apply to the activity label. In a real-world scenario, the user request would ask for which activity was performed the most during the time interval. In order to simulate this the label which occurs the most within the bag will be captured from the ground truth. If in any cases there is no clear majority label within the bag, the bag will be skipped.

3.4. Multiple-Instance Learning

Where normal fully supervised models learn from individual feature vectors, multiple instance learning places data into collections. These collections are known as “bags”, meaning that a single label can have multiple sets of feature vectors associated with it [11]. These bags are then placed into a modified supervised classifier, an example is a bag level classifier known as MI-SVM [19]. Multiple Instance Learning operates on the principles that; a bag will be labelled negative if all instances within it are negative and a bag will be labelled positive if at least one instance within it is positive [20].

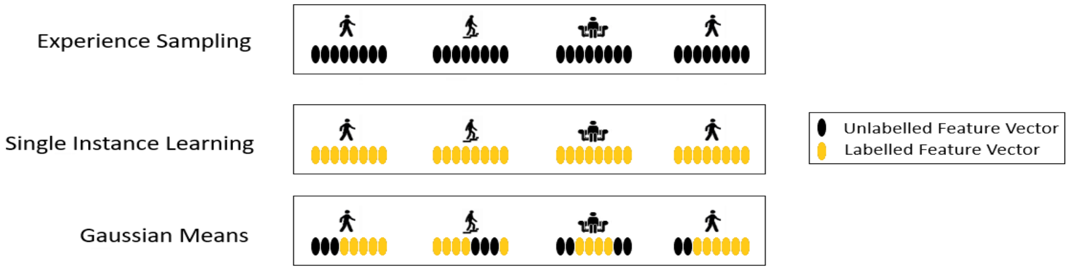

Figure 1 – Population of bag-level labels to individual feature vectors

A key problem in the case of this work is we have no way of identifying “negative” bags. Since we are only capturing a single label per bag, it is likely that a bag will contain activities other than the activity identified as the majority activity. It is therefore not possible to accurately define any bag as negative due to the unknown activities which may occur in the bag. This combined with an aim to predict individual feature vectors rather than bag labels means this work will only use Multiple-Instance classifiers for comparison. An experience sampling interval refers to the time between requests given to the user to provide a label. While the frequency in requests at this stage can be quite high, they are much easier to provide. For instance, the 1-minute window will have 4.3% of the original training sets labels and require that 337 user requests be simulated. However, instead of requiring start times, end times, and ensuring the accuracy of these times the user only needs to provide the name of the activity which they performed the most. MI-SVM classification has previously been used for activity recognition [7] so it provides a good baseline for comparison with other methods provided within this paper.

3.5. Finding the activity within bags

Since each user request only collects a single label which encompasses multiple feature vectors, we investigate methods to eliminate feature vectors which the label applies to. Based on the assumption that the collected label is the one performed the most over the specified time frame, the investigated methods will identify similar feature vectors within the bag. Two main methods to perform this similarity measure will be investigated:

1. Single Instance Learning: Assign the bag level label to each feature vector within the bag. It is a ‘naïve’ learner in that it will incorrectly label a large number of feature vectors, but has been shown to be effective in certain cases [21].
2. Bag clustering: Use of a clustering algorithm to search for the largest single cluster within the bag.

These methods including the experience sampling are shown within Figure 1. Experience sampling shows to only have bag level labels, single instance learning has given each feature vector within the bags the same label as the bag label. The clustering-based algorithm has discovered the largest cluster within the bag and only applied labels to these feature vectors. Figure 1 demonstrates that for a certain time interval, experience sampling will gather one label for a bag of multiple feature vectors but does not label any specific feature vectors. Single Instance Learning then populates each

Algorithm 2 - Detect and reduce outliers

```
{Input: c = feature vectors of biggest cluster, t = percentage of data points to keep}
centre = findCentrePosition(c)
For each feature vector j within c:
    d{j} = EuclideanDistance(c{j}, centre)
s_d = sortByMinimumFirst(d)
n = size(s_d) * t
return s_d{0:n}
```

of these bag labels to the entire bag, resulting in a high per bag recall rate, but potentially poor precision rate. The cluster analysis approach attempts to only provide the appropriate feature vectors in the bag with the bag label, hopefully maintaining a similar recall rate to Single Instance Learning while improving the precision rate.

3.6. Bag Clustering

Since many clustering algorithms require the number of clusters to be specified, this presents a problem. The core aim of the proposed bag clustering is to cluster the bag into a set of K clusters, where each cluster represents a different activity. Each user request provides only a single label. There is therefore no way of knowing the number of possible clusters in a bag.

In order to estimate the number of clusters and therefore the largest of these clusters, we propose the use of a method known as Gaussian means clustering (g-means) which attempts to find the k value in k-means. The g-means algorithm incrementally grows the number of k-means centers and checks to ensure the data comes from a Gaussian distribution. For each of the centers found, if the data appears to be from a Gaussian distribution then this center will be accepted. However, if it isn’t from a Gaussian distribution then a larger number for k may be required.

Algorithm 3 - Detect activity transitions

```
{Input: RF = random forest trained on all labelled feature vectors,
trainingData = full training data without labels}
Transitions = vector of zeros the same length as training set
Predictions = Get individual tree output from RF on training set
For each output in Predictions:
    classesOutput = number of classes predicted in output
    if classesOutput equal to number of classes within training set:
        transitions[output] = 1
```

Once g-means has detected and found the clusters within the data, the largest of these clusters will be extracted and the user's label applied to the corresponding feature vectors. Feature vectors which are outside of the largest cluster are removed from the bag. This idea is illustrated in Algorithm 1 where the bag of feature vectors b and the strictness value s for g-means are input. It then clusters the bag, resulting in a number of clusters b_GM. The biggest cluster within b_GM is then found and returned as c.

3.7. Reducing outliers

Even with the g-Means algorithm discovering the largest cluster within the bag, it is still possible for incorrect feature vectors to exist within the cluster. To reduce this chance, the centroid position within the cluster is found and the Euclidean distance from each feature vector to the centroid calculated. Any feature vectors which lie outside of a threshold from the cluster centroid position can be assumed to be outliers and removed from the cluster.

Algorithm 4 - Reduce manual user input

```
{Input: RF = random forest trained on all labelled feature vectors, data = feature vector of bag, Threshold = prediction threshold}
Confidence = 2-width matrix of zeros the same length as data
TreePredictions = GetIndividualTreeOutput(RF, data)
For each output o in TreePredictions:
    ModePrediction = mode(TreePredictions{output})
    Confidence{output, 0} = RF.predict{data}
    Confidence{output, 1} = Percentage of Trees in TreePredictions which voted for in ModePrediction
BagMode = mode(Confidence{:, 0})
BagModePercent = Percentage of predictions in Confidence which contain the BagMode **
If bagModePercent >= Threshold:
    return bagMode
else
    return -1
```

Although this method could reduce recall, it should improve precision. If a system of this type is to be used, label population performance could incrementally degrade due to previous poor label population performance, making a higher precision favorable. Pseudocode for this idea is provided in Algorithm 2. It takes the cluster c which contains the biggest cluster returned by Algorithm 1 and a percentage t which is the percentage of the cluster to keep as input. The centre position of c is found by calculating the mean. The Euclidean distance between each of the feature vectors in c and the centre is calculated and stored in d. This distances d are then sorted by smallest first and stored in s_d. The n smallest distances in s_d are found by multiplying the threshold t by the size of s_d. These n distances within s_d are then outputted.

3.8. Transition detection

We propose training a model to provide further insight into the feature vector labels and specifically detect feature vectors which correspond to transition activities. Feature vectors will either remain as the label determined during g-means and outlier removal stages or will be given a new transition label. Due to the weakly supervised nature of this work, it is not feasible to classify transitions into specific types of transitions such as sitting-to-standing. We therefore classify all types of transitions the same. Previous work has used an ensemble learner to provide insight into the confidence in a prediction [14]. We leverage the outputs of each individual tree in a Random Forest classifier and analyze these votes to determine how much confidence the classifier has in its prediction in order to detect transition activities. Experience sampling will only collect labels of actual activities rather than the transitions between them. We can therefore look for transitions in bags by looking for the feature vectors which the classifier has extremely low confidence. Algorithm 3 shows how the transition detection is implemented. A supervised random forest classifier is trained on feature vectors and labels identified through g-means and outlier reduction steps described in Sections 3.6 and 3.7. Each feature vector is then tested on the trained random forest classifier and feature vectors which the classifier has a low classification confidence of for all classes is given a transition label. Confidence is determined by counting how many classes the individual trees voted for against each feature vector. If they have voted for a number of classes equal to the number of classes in the weakly labelled training set then its confidence is low. If the random forest is not confident that the feature vector relates to a specific activity, the feature vector is therefore classified as a transition.

As Random Forest can output the votes from each tree individually and for each feature vector, we can look at the variance in these votes and determine how much confidence the classifier has in its prediction. This same idea could be used for the detection of transitions. As the simulated experience sampling will only be collecting the labels of actual activities rather than the transitions between them, we can look for transitions in bags by simply looking for feature vectors which the classifier has extremely low confidence.

3.9. Reducing manual user inputs

As an extension to the random forest confidence methods of finding the transitions within bags, the same principle will be applied to reduce the number of user requests and more importantly any future requests. This is performed in two stages. The first stage is an initial training stage using an experience sampling process where users will provide labels via experience sampling. The second stage utilizes what is learnt from stage 1 to make attempt to make automatic predictions for bag labels as opposed to asking users to provide bag labels. Bag predictions will be made in instances where the classifier has a high confidence, otherwise it will revert to asking the user to provide the label. By having each individual tree in a Random Forest make a prediction on each point within the bag. The votes for each point from each tree are then aggregated into the mode vote label for each feature vector and the percentage of the votes for the most common label.

For a bag of 10 feature vectors long, it will output a matrix holding the mode vote for each of these feature vectors and their percentage of the total votes. The mode of all the mode votes is found and its percentage of the aggregated votes is found. If this percentage is above a threshold then the prediction is used, if not, then the experience sampling request will be performed. Once this process has finished for the current bag the Random Forest will be retrained with the new data and the same process will be repeated for the next bag.

Algorithm 5 - Activity Recognition System

```
{Input: trainingData = matrix of all training feature vectors, testingData = matrix of all testing feature vectors, trainingLabels = vector of all training labels, testingLabels = vector of all testing labels, outlierThreshold = Percentage of data points to keep in outlier removal}
WeaklySupervisedLabels = Matrix of zeros the same length as data
Bags = ConvertToBags(trainingData, bagSize)
For each Bag b in Bags:
    //Capture label or make prediction
    If index of Bag b > Total number of Bags:
        bagLabel = Run Algorithm 4 on Bags[b]
    Else:
        bagLabel = Capture label by experience sampling
    //Find the positions in the bag the label applies
    LabelPositions = Run Algorithm 1 on Bags[b]
    //Reduce outliers
    LabelPositions = Run Algorithm 2 on LabelPositions with outlierThreshold
    //Apply these labels to WeaklySupervisedLabels
     WeaklySupervisedLabels[LabelPositions * b] = bagLabel
End For
//Find transitions
Train a Random Forest on WeaklySupervisedLabels and Training Data
Transitions = Run Algorithm 3 with this Random Forest on TrainingData
WeaklySupervisedLabels[Transitions] = TransitionLabel
//Validate on classifiers
```

Algorithm 4 provides implementation details on this idea. It takes a Random Forest RF which has been pre-trained on all known labelled feature vectors, data with is the feature vectors of the bag to be predicted upon and Threshold which will be our prediction threshold all as input. First an empty matrix Confidence, equal in length to data and 2-width, is created to store our confidence values. For each output o from TreePredictions, the mode tree vote will be calculated. RF's prediction for the data placed and the percentage of individual trees which voted for the mode prediction will be stored in Confidence. After the loop, the mode prediction for data is calculated and stored in BagMode. The percentage of predictions for this BagMode is calculated and if this value is over the Threshold the bagMode will be taken as a prediction for the bag. Otherwise, the algorithm will return -1 in which case the system will ignore the prediction made and make a user request instead.

3.10. Full Algorithm Implementation

Algorithm 5 provides details on the full implementation of this system. It begins by taking the training and testing data as input parameters. A matrix of zeros the same length as the training data is created to store the labels obtained through weak supervision. The training data is then converted into bag form. For each of these bags, it first checks if more than half of the bags have been processed. If we have passed the half-way point, then it will begin trying to make predictions on the bag labels by running Algorithm 4. Otherwise it will capture the bag through simulated experience sampling. Algorithm 1 run now be run on Bag b, this cluster returned by Algorithm 1 will be run through Algorithm 2 to try to reduce any bag outliers with outlierThreshold. The resultant cluster will now be applied to WeaklySupervisedLabels with the label gathered either through the experience sampling or bag prediction performed earlier. Once all bags have been processed, a random forest RF is trained on WeaklySupervisedLabels and the feature vectors in trainingData by running Algorithm 3. These transitions are then added the WeaklySupervisedLabels and the data can now be validated on a classifier.

3.11. Classification

Several classifiers are being used as part of this work, one of which will be used for confidence measurements and the other which will be used for final testing of the system.

3.11.1. Random Forest

Random Forest is an ensemble learner which is made up of a collection of decision trees. This Random Forest [21], implementation is being used as it provides an output of the individual votes from each tree. Using these votes, we can infer the classifiers confidence in its prediction, and therefore look for feature vectors which the system has very low confidence in. These positions may then be marked as potential transition points. This Random Forest is also used for the prediction of the future bag level labels.

3.11.2. Support Vector Machines (SVM)

Support vector machines have been previously proven to perform well in the classification of activity data [6]. They are also capable of handling a large feature space without significant performance impact [22], useful in this situation as activity recognition systems are destined to be applied to lower power devices where efficiency is important.

3.11.3. K-Nearest Neighbours

A lazy classifier which uses Euclidean distance to find the nearest feature vectors within the feature space. This classifier has been shown to be capable in classifying activity data and it generally outperforms Naïve Bayes [23].

3.11.4. Multi-layer Perceptron

A Multilayer Perceptron is a type of feedforward neural network. Previous work has shown that it can classify activity data relatively well. It has been shown to outperform logistic regression and decision trees [24].

4. Results

4.1. Evaluation protocol

Evaluation will be performed using a training/test split. To reduce bias, the participants in the training set will be completely independent of those in the test set. The training set will comprise of 70% of the data and the test set will contain the remaining 30%. Cross validation is not being used as the training/test split allows comparison with other methods. Since this paper sets out to test how usable a training set captured using experience sampling is, we will remove all ground truth labels from the training set. A small number of labels will then be captured with simulated user requests and then populated to other unlabelled feature vectors.

4.2. Full ground truth

Before conducting experiments to evaluate the performance of the weakly supervised techniques we first perform baseline performance evaluations on a fully supervised system to ascertain the "gold standard" performance. In order to find this the fully labelled training set will be used in multiple supervised classifiers. They are SVM, Random Forest, k-Nearest Neighbours [22] and Multi-layer Perceptron [23]. A grid search has been employed to optimize each of the four classifiers. This was initially performed in previous experimentation and resulted in the following parameters.

- **Random Forest**: 50 estimators with no maximum depth
- **Multi-layer perceptron**: Uses the Adam stochastic gradient descent optimization method with a Relu activation layer, a hidden layer size of 100 is used with an alpha value of 0.001
- **SVM**: An RBF kernel is used with a C value of one with an automated gamma value
- **k-Nearest Neighbours**: Uses a number of nearest neighbours set to 5.

As can be seen from table 1, of the classifiers given Multi-Layer Perceptron and SVM have shown the best performance in both accuracy and F1 score.

Table 1 – Accuracy and F1 Score of fully labelled ground truth

Classifier	Accuracy	F1 Score
SVM	94.10%	94.29%
K-Nearest Neighbours	89.80%	90.14%
Random Forest	91.00%	91.14%
ML Perceptron	94.84%	94.43%

4.3. Populating bag-level labels to instance level

Two methods of automatically assigning bag labels to the individual feature vectors are going to be tested. As a baseline comparison, we first evaluate a naïve labelling method where no attempt to restrict the labels is performed and bag level labels are assigned to all the feature vectors within each bag. The second method will use the proposed Gaussian Means based method to cluster the data before applying labels, while they method may reduce the overall number of true positives found, we postulate that the removal of feature vectors using g-means will decrease the false discover rate. The Gaussian means algorithm uses a "strictness" value. Preliminary experiments identified that a strictness value of 4 provided the desired balance between precision and recall.

Table 2 – Precision and Recall of Single Instance Learning

ES Time Interval	0.5 mins	1 mins	1.5 mins	2 mins	2.5 mins	10 mins	30 mins	60 mins
Precision	0.78	0.64	0.54	0.48	0.44	0.19	0.10	0.10
Recall	0.81	0.67	0.55	0.51	0.45	0.23	0.19	0.19

4.3.1. Single Instance Learning

Single instance learning will indiscriminately assign the bag label to all feature vectors in the given bag. Table 2 shows the precision and recall scores for propagating a single label to each bag. As expected, performance is significantly reduced each time the time interval is increased. However, even with the smaller bags, the performance does not compare with the performance of fully supervised training.

4.3.2. Proposed Method

For this method to assign labels to the feature vectors within the bag, it will first cluster the data using g-means clustering. Once the largest cluster has been found, the bag label will be applied to these feature vectors and the unused labels will be left unlabeled. The thresholding step to reduce bag outliers will be applied with a threshold value of 95%, preliminary experiments found that this value provided the lowest reductions in recall. Table 3 shows the results of training using the proposed weakly supervised method. The precision performance is excellent for the smaller windows, likely due to the lower chances of having noisy labels when compared to single instance learning.

Table 3 – Precision and recall of Gaussian Means

ES Time Interval	**0.5 mins**	**1 mins**	**1.5 mins**	**2 mins**	**2.5 mins**	**10 mins**	**30 mins**	**60 mins**
Precision	0.94	0.87	0.74	0.72	0.65	0.31	0.18	0.31
Recall	0.86	0.73	0.66	0.62	0.54	0.30	0.13	0.13

4.3.3. Label Propagation Discussion

With the results of both methods collected, a decision can be made about which of the two label populating algorithms is most effective. Tables 2 and 3 show that of the two methods of assigning labels, the proposed algorithm performs significantly better in terms of precision. Recall is higher for single instance learning in the longer windows, however, performance has already degraded below usable by this point. There is a chance that single instance learning will get a higher number of true positives, however, it is more important to get a higher percentage of correct labels. This is due to a false negative only reducing the amount of data a classifier must learn from, whereas a false positive will actively confuse the decision boundary of a classifier. The proposed algorithm shows superior results in all lengths of experience sampling window lengths, and thus further experiments will be carried out on that method.

4.4. Transition Detection

This section will describe experiments to evaluate the performance of the transition detection method described in Section 3.9. We propose detecting transitions by using a Random Forest trained on the completed weakly labelled training set. Transitions are classified by identifying feature vectors which have a very low classifier confidence for all activity classes.

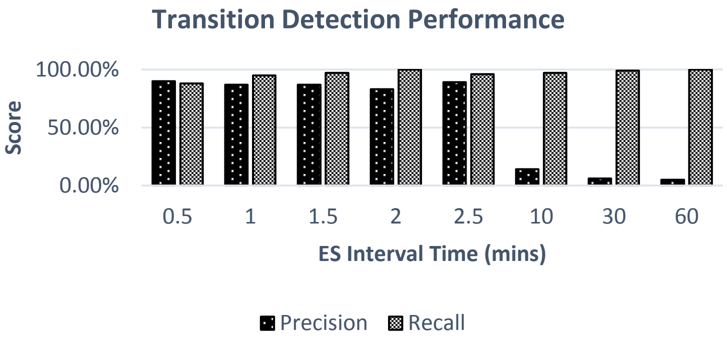

Figure 2 – Precision and recall of the detected transitions at multiple experience sampling window length

The experience sampling label collection process will only ever capture actual activities and not transitions between them. In this weakly supervised scenario, labels specifically indicating the presence of transitions will therefore never be available. Figure 2 shows the precision and recall results of detecting the transitions. The shorter experience sampling windows perform with good precision and recall. We attribute this good performance to the fact that these classifiers are being provided with a large amount of data, this results in the classifier being confident in making predictions. It also means that it will be more likely to select positions in which confidence is low. Conversely, larger experience sampling windows provide the classifier with small amounts of data due to the limits in the dataset size at this sampling frequency. The transition points are validated by comparison with the transitions within the ground truth. The algorithm performs very well in finding transition points and up to an experience sampling window of 2.5 minutes. The recall results are high at the longer experience sampling windows with extremely low precision. This is due to the algorithm not having enough data to find these transitions accurately and it, in fact, labels almost every feature vector as a transition. This means it almost never misses any transitions, but when you compare the ratio of how many of these feature vectors are labelled correctly versus the total number of transitions labelled, it results in a low precision value. This issue is being solely caused by

Table 4 – Classification results of multiple classifiers

Accuracy					F1 Score			
SVM	**KNN**	**RF**	**MLP**	**ES Time Interval (mins)**	**SVM**	**KNN**	**RF**	**MLP**
0.93	0.87	0.89	0.91	**0.5**	0.93	0.87	0.88	0.90
0.88	0.79	0.81	0.78	**1**	0.88	0.77	0.80	0.77
0.66	0.65	0.63	0.61	**1.5**	0.56	0.58	0.54	0.53
0.49	0.47	0.46	0.44	**2**	0.38	0.40	0.39	0.38
0.48	0.48	0.43	0.45	**2.5**	0.38	0.41	0.36	0.38
0.05	0.12	0.08	0.09	**10**	0.01	0.11	0.06	0.08
0.05	0.07	0.60	0.05	**30**	0.01	0.04	0.03	0.02
0.05	0.09	0.05	0.06	**60**	0.01	0.06	0.02	0.02

a lack of data and can be solved in real world scenarios by only allowing transitions to be discovered once a classifier has gathered a certain level of performance.

4.5. Classification Results

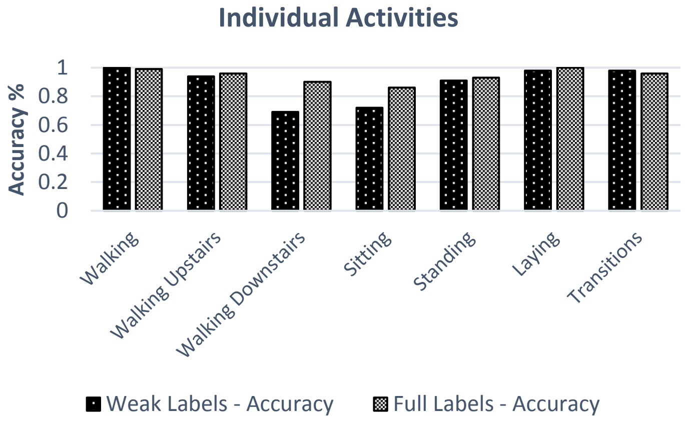

Figure 3 – Classification of individual activities with weak labels vs full ground truth

The following section will test the overall performance of the proposed algorithm combined with the transition detection algorithm and make direct comparisons with a fully supervised system.

From Table 4, it is clear that the SVM performs best in all cases. However, after the one-minute window length the performance drops significantly, and at 2 minutes it has dropped below what could be considered reasonable performance. The SVM performs best of the four classifiers tested and as one minute shows the longest window where the system is still effective. We therefore further investigate the individual activity classification efficiency compared to a fully labelled training set for the SVM. Figure 3 shows the comparison of accuracy for both the fully labelled training set and the weakly labelled training set. It can be seen that most of the performance is lost in the walking downstairs and sitting activities. However, in some cases, the weakly supervised system outperforms the fully labelled training set, this is apparent in the transitions and walking.

Table 5 – Confusion matrix showing SVM performance at one-minute interval

	Walking	**Walking Upstairs**	**Walking Downstairs**	**Sitting**	**Standing**	**Laying**	**Transitions**
Walking	494	0	2	0	0	0	0
Walking Upstairs	28	441	2	0	0	0	0
Walking Downstairs	83	46	289	0	0	0	2
Sitting	0	0	0	368	132	0	8
Standing	0	0	0	38	507	0	11
Laying	0	0	0	0	0	534	11
Transitions	2	0	0	0	1	0	163

Table 5 shows the confusion matrix for the classification results achieved using the weakly labelled training set at a one-minute interval with the SVM classifier. The reason for its reduced performance in classifying walking downstairs is due to it being confused with walking and walking upstairs. Similarly, sitting is being confused with standing. The reason for this is due to walking downstairs, walking and walking upstairs all being dynamic movements with a similar cyclic pattern. Whereas, sitting and standing are both static activities with no movement and hence provides similar acceleration profiles. These similarities in the activities mean they can be difficult for a classifier to discriminate, especially if the classifier has less information than normal. We perform an additional experiment to compare the proposed weakly supervised methods with Multiple Instance SVM (MI-SVM), another weakly supervised algorithm which has been previously used for activity recognition [7].

Figure 4 shows the F1 score comparison between the proposed method and MI-SVM. These scores are taken from the 0.5-minute window, providing MI-SVM with the best opportunity to classify the data. The experiments found that although MI-SVM had good precision scores, the associated recall scores are very low, ultimately leading to the low F1 scores seen in Figure 4. As MI-SVM performance has degraded below what could be considered acceptable at the 0.5 minute intervals, the longer windows are not tested.

4.6. Reducing manual user input

In a real-world setting, we envisage the system collecting more and more bag labels as time progresses. There is an opportunity to pre-train a classifier which can be used automatically assign labels to bags. These automatically assigned labels can be used to further reduce the number of user requests performed and therefore reduce the overall burden on the user. A Random Forest will be retrained each time a user input request is performed, this classifier will then make predictions on the next bag to be labelled. Experiments to evaluate this technique will only be performed on the smaller experience sampling intervals due to the lack of information available in the larger intervals. This lack of information will lead to a system which may be able to make predictions after enough

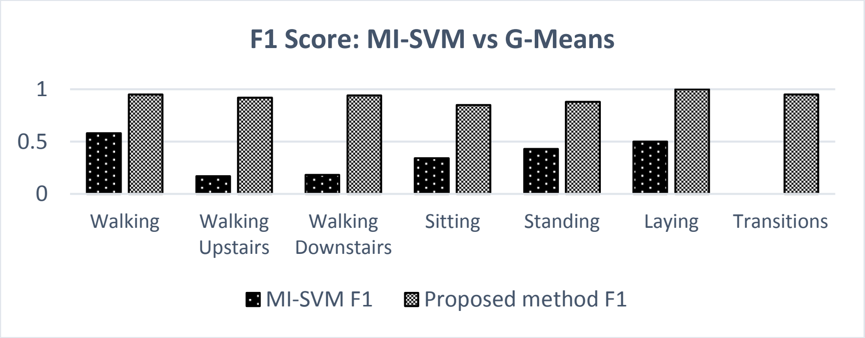

Figure 4 – Comparison of F1 Scores of proposed method and MI-SVM

time has passed, but within the constraints of the dataset will not have gained enough confidence to make accurate predictions. However, as this method is deciding on an entire bag of feature vectors, a mistake here could result in many incorrectly labelled feature vectors resulting in a very significant impact on classifier accuracy. A threshold value of 0.95 was found to provide the best balance between incorrect predictions and number of predictions made. Due to the random nature of the random forest classifier, the entire experiment is repeated 10 times to smooth out any outlier performances. The train/test split does not change during these repeated experiments. The average reduction in predictions and prediction accuracy from each of the 10 iterations will then be taken. This experiment is not being run in tandem with the previous experiments as the dataset is not large enough to support the longer windows, rather it is being shown as a proof of concept. We found that at the 0.5-minute interval, 108 predictions are made on average. This reduces the overall number of predictions by approximately 16%. Of these 108 predictions, 98.3% are correct, this results in an insignificant degradation of overall classifier F1 score by 1%. However, at the 1-minute window, the number of user requests is only reduced by 2% and while all of these predictions where correct it shows that the shorter the window, the more predictions can be made. This is simply because more data is available for prediction and there is more chance that the system will have a higher confidence in its prediction. The percentage reduction in predictions is the number of successful predictions the system made, and the accuracy figures show the number of extra predictions the system incorrectly made.

5. Discussion

Collecting labels via user request does have some limitations, mostly in that entire activities can be missed. Short activities can also be missed by the wording of the request or a very long-time interval.

A possible solution could be through attempting to detect sequences of data which are more valuable to a classifier. For example, a weakly supervised system could look for sequences which a classifier has less confidence in. The fully supervised performance results in this work match fully supervised experiments in the literature in that SVM is a competent classifier for activity data. While the Multi-Layer Perceptron performed the best in the initial experiment with fully supervised data, this performance suffered once labels where restricted. In terms of labelling instances within the bags, the g-means based method had far superior performance in ensuring that the labels are applied to the correct instances. The g-means based method in comparison to the naïve method of applying a single label to the entire bag aims to restrict the labels to feature vectors which the label most likely applies. Standard data collection experiments need accurate recording of activity start-times, end-times and the activity label itself. These experiments are invasive, requiring laboratory conditions and continual user monitoring to maximize classification performance. With the training set contains approximately 769 points when one activity changes to another activity or transition, it would result in 1538 instances when a time would need to be logged. Our experiment set out to provide a simpler method of collecting data which could be performed within the user's natural environment and even the very short 0.5-minute window would only require 674 instances when an activity label would need to be provided. In addition, the effort and time requires to provide a single weak activity label is significantly less than the effort required to specific the specific start and end time of a label. Experiments showed that both the 0.5 and 1 minute experience sampling time intervals provide a significantly more feasible and less burdensome data collection process with accuracy losses of only 1.4% and 6.1% respectively. The proposed algorithm also outperforms the MI-SVM classifier where results show that MI-SVM had significantly lower rates of recall. One issue not addressed is the compute performance impact of the experience sampling request prediction. The Random Forest is re-trained in a semi-online fashion, as each prediction or simulated user request provides more information for the next prediction. The retraining of this model may have negative effects on compute and battery performance. One possible workaround is to reduce the rate at which the Random Forest is retrained and instead provide several predictions before retraining. This, however, will likely reduce the effectiveness of the system.

6. Conclusion

The methods presented within this paper test the efficacy of several methods in reducing the requirement for infeasible data collection experiments. Instead of accurately collecting the beginning and end times of activities and having users placed within unnatural environments, users or individuals monitoring a user can provide a single label at a certain frequency. However, a fully supervised classifier is never going to be provided with noisy labels. This means that a weak classifier is always going to be somewhat less efficient and will either need extra correctly labelled feature vectors to compensate or employ a methodology which can handle these noisy labels. In this work we have attempted to reduce noisy labels through the correct tuning of the Gaussian Means algorithm. From this single label, g-means clustering based method is applied to the data to discover the appropriate labels within that time interval. Algorithms are then used to discover transitions within these activities and possibly reduce the number of future data requests. Results show that a 0.5-minute experience sampling time interval only reduced classifier performance by 1.4% but also required substantially fewer labels than typical data collection experiments. The proposed method also outperforms MI-SVM. Multiple time intervals were tested but result show that the longer the intervals, the less data is available, and the dataset currently being used quickly runs out of data. This results in incrementally worse classifier accuracies, however, it does set a proof of concept that which could possibly allow a system to begin with small intervals and incrementally grow these intervals over time. Future work will consider removing the time intervals and instead focus on finding the start time of new activities. An example could be making a request at a time when a classifier has low confidence in the current data. This could have two effects; it could allow the discovery of new activities and it could also increase the classifier performance in activities. Another possible direction of this work is to first identify how many noisy labels are being created and measure the effect this has on classification performance relative to normal fully supervised learning. In the case that the noisy labels are causing a substantial performance impact then methods such as importance reweighting [24] can be tested.

7. References

[1] T. Rault, A. Bouabdallah, Y. Challal, and F. Marin, "A survey of energy-efficient context recognition systems using wearable sensors for healthcare applications," *Pervasive and Mobile Computing*, vol. 37. pp. 23–44, 2017.

[2] C. Wong, Z. Q. Zhang, B. Lo, and G. Z. Yang, "Wearable Sensing for Solid Biomechanics: A Review," *IEEE Sensors Journal*, vol. 15, no. 5. pp. 2747–2760, 2015.

[3] E. J. W. Van Someren *et al.*, "A new actigraph for long-term registration of the duration and intensity of tremor and movement," *IEEE Trans. Biomed. Eng.*, vol. 45(3), pp. 386–395, 1998.

[4] S. Jiang *et al.*, "CareNet: An Integrated Wireless Sensor Networking Environment for Remote Healthcare," in *Proceedings of the 3rd Int. ICST Conference on Body Area Networks*, 2008.

[5] N. Alshurafa *et al.*, "Designing a Robust Activity Recognition Framework for Health and Exergaming Using Wearable Sensors," *IEEE J. Biomed. Heal. Informatics*, vol. 18, no. 5, pp. 1636–1646, 2014.

[6] Y. L. Zheng *et al.*, "Unobtrusive sensing and wearable devices for health informatics," *IEEE Trans. Biomed. Eng.*, vol. 61, no. 5, pp. 1538–1554, 2014.

[7] M. Stikic, D. Larlus, S. Ebert, and B. Schiele, “Weakly supervised recognition of daily life activities with wearable sensors,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 12, pp. 2521–2537, 2011.

[8] W. Y. Wong, M. S. Wong, and K. H. Lo, “Clinical applications of sensors for human posture and movement analysis: a review.,” *Prosthet. Orthot. Int.*, vol. 31, no. 1, pp. 62–75, 2007.

[9] D. Kelly and B. Caulfield, “An investigation into non-invasive physical activity recognition using smartphones.,” *Conf. Proc. IEEE Eng. Med. Biol. Soc.*, vol. 2012, pp. 3340–3, 2012.

[10] R. J. Shephard, “Limits to the measurement of habitual physical activity by questionnaires,” *Br J Sport. Med*, vol. 37, pp. 197–206, 2003.

[11] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez, “Solving the multiple instance problem with axis-parallel rectangles,” *Artif. Intell.*, vol. 89, no. 1–2, pp. 31–71, 1997.

[12] G. Hamerly and C. Elkan, “Learning the k in k means,” *Adv. neural Inf. Process.*, vol. 17, pp. 1–8, 2004.

[13] W. Duffy, K. Curran, D. Kelly, and T. Lunney, “Addressing the problem of Activity Recognition with Experience Sampling and Weak Learning,” in *Intelligent Systems Conference (IntelliSys)*, 2018, pp. 1–13. doi.org/10.1007/978-3-030-01054-6_86, Springer

[14] W. Duffy, K. Curran, D. Kelly, and T. Lunney, “Reducing the intrusion of user-trained activity recognition systems,” in *The 29th Irish Signals and Systems Conference (ISSC 2018)*, 2018.

[15] M. Stikic, K. Van Laerhoven, and B. Schiele, “Exploring semi-supervised and active learning for activity recognition,” *2008 12th IEEE Int. Symp. Wearable Comput.*, pp. 81–88, 2008.

[16] A. Kapoor and E. Horvitz, “Experience sampling for building predictive user models,” *twenty-sixth Annu. CHI Conf. Hum. factors Comput. Syst. - CHI ’08*, pp. 657–66, 2008.

[17] J.-L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, and D. Anguita, “Transition-Aware Human Activity Recognition Using Smartphones,” *Neurocomputing*, vol. 171, pp. 754–767, 2016.

[18] S. Liu, R. Gao, D. John, J. Staudenmayer, and P. Freedson, “Multi-Sensor Data Fusion for Physical Activity Assessment,” *IEEE Trans. Biomed. Eng.*, vol. PP, no. 99, pp. 1–1, 2011.

[19] S. Andrews, T. Hofmann, and I. Tsochantaridis, “Multiple Instance Learning with Generalized Support Vector Machines,” *AAAI*, pp. 943–944, 2002.

[20] R. C. Bunescu and R. J. Mooney, “Multiple instance learning for sparse positive bags,” *Proc. 24th Int. Conf. Mach. Learn.*, vol. 79, no. June, pp. 105–112, 2007.

[21] S. Ray and M. Craven, “Supervised versus multiple instance learning: An empirical comparison,” in *… 22nd international conference on Machine learning*, 2005, pp. 697–704.

[22] P. Gupta and T. Dallas, “Feature selection and activity recognition system using a single triaxial accelerometer,” *IEEE Trans. Biomed. Eng.*, vol. 61, no. 6, pp. 1780–1786, 2014.

[23] J. R. Kwapisz, G. M. Weiss, and S. a. Moore, “Activity recognition using cell phone accelerometers,” *ACM SIGKDD Explor. Newsl.*, vol. 12, no. 2, p. 74, 2011.

[24] Liu, T. & Tao, D., 2016. Classification with Noisy Labels by Importance Reweighting. IEEE Transactions on Pattern Analysis and Machine Intelligence.