Detecting Harmful Hand Behaviors with Machine Learning from Wearable Motion Sensor Data

Lingfeng Zhang and **Philip K. Chan**
Florida Institute of Technology, Melbourne, FL 32901
lingfeng2013@my.fit.edu , pkc@cs.fit.edu

Abstract

We propose a machine learning approach for detecting harmful hand behaviors in autistic children. Our approach extracts features from a wearable motion sensor, applies the Random Forest algorithm, and uses sequence post-processing to correct potentially incorrect classifications. Our experimental results indicate that our proposed system can get a 96% accuracy rate in detecting harmful behavior, 95% accuracy in distinguishing harmful behaviors from normal but similar behaviors.

Introduction

Human activity recognition becomes increasingly popular because it has a strong potential for services such as sports tracking, health care, special needs, and security. A faculty member at our university helps treat autistic children. One of his goals is to design therapies that help autistic children, who sometimes perform harmful actions to themselves. To evaluate the effectiveness of his therapies, he would like to count the number of harmful actions in an automated manner, instead of having someone count manually. This project tries to detect harmful hand behaviors from motion sensor data, which can be used for counting harmful behaviors. Some other studies focus on hand behavior recognition, but our target is detecting some abnormal behaviors such as aggressive attacks toward the head or knees. Since abnormal behaviors occur much less frequently than normal behaviors, we need to handle the imbalance data issue.

In this paper, the raw sensor data was acquired from a single wearable sensor located on the user's wrist. We first use a windowing process to segment the raw data into fixed-size windows. Then, we extract features from each window. A large set of features can lead to large computational overhead and may potentially decrease the accuracy of classification. Therefore, selecting a small and efficient set of features plays an important role. We use Random Forest to combine multiple decision trees to improve accuracy. We also design a Sequence Post-Processing method that tries to correct the potentially incorrect classifications from the Random Forest.

Our empirical results indicate over 96% in the true positive rate of the slapping class. We optimize the results by selecting the best combination of forest size and feature subset size for Random Forest. Furthermore, our results indicate that Sequence Post-Processing helps decrease the false positive rate.

Related Work

Some of the studies focus on the body movement recognition. Krause et al. (2003) and Mannini et al. (2010) use the Hidden Markov Model (HMM) as their classification algorithm, and they did experiments to prove that a good sampling rate can improve the accuracy. Li et al. (2014) use Singular Value Decomposition (SVD) to capture the underling relationship among the features and reduce the number of features. Features obtained from SVD are helpful for discriminating between the classes. They use the Back Propagation Neural Network (BPNN) as their classification algorithm. Yang et al. (2008) use Principal Component Analysis (PCA) to reduce the feature dimensionality, and design a divide-and-conquer method to separate the dynamic and static states. Finally, they use BPNN to perform classification. Yang et al. (2010) use Naive Bayes as the classification algorithm, which assumes that the features are conditionally independent given the class.

Some studies focus on the hand behavior recognition. Hong et al. (2000) compute features from the input video images which are 2D positions of a user and use Finite State Machine (FSM) to be the classification algorithm. Pylvanaimen (2005) uses HMM. The recognition system has a pre-processing step which removes the effect of device orientation from the data. Wu et al. (2009) use Support Vector Machine (SVM), which is a margin classifier, and compares the performance in two ways: user-dependent and user-independent.

The data for human activity recognition can also come from multiple sensors resources, or other resources instead of accelerometer. Nam et al. (2013) acquire data from an accelerometer sensor and a single grid-based image sensor. The data in different channels are processed by SVM, and the results show a great improvement by fusing data from the two sensors. Ward et al. (2006) use two sensors: accelerometer and microphone. The authors use linear discriminant analysis to analyze the sound data and HMM to analyze the acceleration data. Anjo et al. (2012) use a real time continuous video stream as data and BPNN for classification.

Approach

The main problem addressed by this paper is how to detect harmful actions from motion sensor data. In this section, we introduce the main components of the overall system. First, for data pre-processing, we extract features and generate instances. Then we use Random Forest to perform classification. Finally, we use Sequence Post-Processing to improve the classification from Random Forest by trying to correct potentially incorrect classifications.

Data Pre-processing and Feature Extraction

To obtain the raw sensor data, an associate of the behavior analysis professor simulated slapping and drinking behaviors. The reason for choose slapping and drinking is that the two behaviors are very similar and we would like to study if we can distinguish them. Besides the sensor data, we have an action table that contains the starting/ending time for each action.

The raw data are measurements from a wearable sensor, which includes an accelerometer and a gyroscope. From the accelerometer, we use Linear Acceleration, which includes the object's acceleration and excludes Earth's gravitation. For both the accelerator and gyroscope, the senor provides measurements in the x, y, z directions every 10 milliseconds. Which generates 100 samples per second.

Since the starting/ending time labels for user actions have a precision of 1 second while the sensor has a precision of 10 ms, we need to refine the precision of the time labels for user actions. We first plot the data and manually determine more precise starting/ending time labels up to two decimal places for the user actions. Finally, we label each record (sample) as one of three classes: slapping, drinking, and "no action."

One single data record does not contain enough information for our machine learning system. Therefore, we need to window the data, so that we can extract some useful information/features from a group of data records. To determine the window size, we set it to be the length of the shortest action in our data set. That is, the window size is large enough to represent an action, but it is not too large to contain irrelevant data. In our data set, the window size is 25 (i.e., a duration of 250 ms). For each window, we then label it with a class that is associated with more than 50% of the data records.

Since the current behavior might be influenced by the previous behavior, we concatenate three previous windows to the current window to construct an instance for machine learning. The class label for the instance is the class label for the current window.

In this paper, we use seven features, which are: (1) mean value; (2) the absolute value between min and max: the difference between min and max is useful in discriminating whether there is an action happening now or not; (3) root mean square value: this feature is to describe the distribution of the data in one period; (4) standard deviation; (5) linear regression: linear regression is to represent the trend of a group data; (6) Pearson correlation between axes: Pearson correlation can represent the relation in two dimensions, either positive related or negative related; (7) Pearson correlation between accelerometer and gyroscope in the same axis. Overall, each window has 39 features.

Random Forest and Imbalance Data Set

Our data set is an imbalance data set, because the number of negative class instances ("no action") is much larger than the number of positive class instances ("slapping" or "drinking"). Random Forest (Breiman 2001) generally achieves higher accuracy and handles a large number of features. We generate multiple trees to solve the imbalance data set problem. The data set for each tree contains all instances from the positive (minority) class and the same number of instances from the negative (majority) class. That is, we down sample the negative class to create multiple balanced data sets for the trees.

As mentioned above, there are two parameters in Random Forest: the size of a feature subset for each node and the number of trees. According to Breiman (2001), he uses 100 trees, and the feature subset size is $\log_2 M + 1$, where M is the size of the entire feature set. But we found that, using these two values do not obtain a significant improvement in our data set. Therefore, we want to find the best combination of the feature subset size and the forest size. We find these two parameters by using a validation set.

After a random forest is built, we get the initial classifications of each instance in the test set. However, according to initial experiments, we find that using the majority voting strategy is not highly accurate. Therefore, we want to find a more appropriate vote threshold method to determine the class for each instance, not just use the class with the most votes as in the majority voting method.

To find the voting threshold, we use Random Forest to classify instances in the validation set. For each instance, we gather three values: the number of trees that predict the instance as the slapping class (S), drinking class (D), or no-action class (N). Figure 1 illustrates how we choose the threshold for the drinking class. The X axis represents the number D – N. The Y axis represents the number of instances. If an instance shows strong characteristics of being a drinking class, D will be larger than N and D – N is positive; otherwise, D – N is negative. That is, we expect the blue curve to decrease, while the red curve increases. The two curves cross each other with the same number of instances for the drinking class and the no-action class. For example, in Figure 1 the crossing point is indicated by the green dotted vertical line, the crossing point located at D – N is 12. To minimize the error for the validation set, we choose the crossing point to be the threshold for D – N for classifying an instance to be in the drinking class. That is, if D – N of an instance is larger than the threshold, the instance is labeled as the drinking class; otherwise the instance is labeled as the no-action class. The threshold for the slapping class is determined in a similar way with S - N in the X axis.

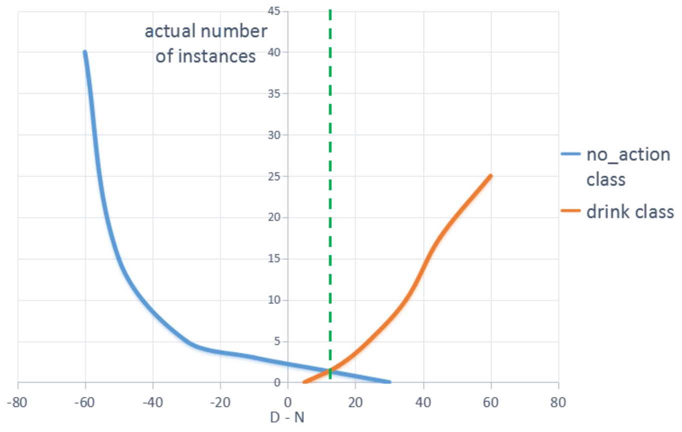

Figure 1. Selecting the voting threshold for the drinking class.

Sequence Post-processing

The Sequence Post-Processing step is used to potentially correct some incorrect classifications by a tree or Random Forest. We analyze sequences of classifications and determine if each sequence is consistent. For example, we assume that a drinking action lasts two seconds, if only one drinking instance exists while the previous and next instances belong to the no action class, this drinking instance is likely a wrong classification.

Based on the training set, we determine several parameters to correct probably incorrect predictions that come from Random Forest. First, we determine the minimum duration and maximum duration of drinking (or slapping) action. Second, we identify the minimum interval between any two drinking (or slapping) actions.

Due to space limitations, we outline our algorithm--further details are in (Zhang, 2015). We first determine drink_gap, drink_min, and drink_max from the training set. drink_gap represents the minimum number of instances in No_action class which are between one drink action and another action; drink_min and drink_max correspond to the minimum and maximum number of instances among all the drink actions. As long as a drinking class instance exists, the algorithm groups the drinking instances to form a drinking block. If the previous drink action is too close to the current drink action, then the program will merge these two drink actions into one action as long as it does not exceed the maximum length for a drink action (drink_max). That is, when appropriate, we merge two drinking actions into one, which might be incorrectly separated. Otherwise, the previous drink action has an enough interval with the current drink action and the program sets the current drink action as a legal drink action.

Experimental Evaluation

In this paper, we use three data sets to evaluate our algorithm performance. All the data come from a wearable sensor on the user's wrist. The first data set includes 9 knee-slapping actions. This data set has 47 seconds of data and each second has 33 records. The second data set includes 10 head-slapping and 5 tea-drinking actions. These actions happen randomly. This data set has 1773 seconds of data and each second has 100 records. The third data set includes 29 head-slapping and 29 tea-drinking actions. This data set has 310 seconds of data and each second has 100 records. The head-slapping and knee-slapping motions last approximately 0.5 second, and the tea-drinking motion lasts around 2 seconds for all data sets mentioned above.

Since the data sets are not balanced, using accuracy to measure performance is not sufficient. We consider the slapping and drinking motions as different positive classes, and the remaining instances ("no actions") to be of the negative class. We calculate the true positive rate and the false positive rate for the two positive classes. Also, since we select a desirable threshold, we do not report results in terms of ROC.

Table 1 Confusion matrix of classification (each cell has number of instances)

		Predicted class		
		slapping	drinking	no action
Actual class	slapping	a	b	c
	drinking	d	e	f
	no action	g	h	i

Using the confusion matrix in Table 1, we define the true positive rate of slapping as a/(a+b+c), the true positive rate of drinking as e/(d+e+f), and the false positive rate of no action as (g+h)/(g+h+i). In behavior analysis, we would also like to estimate whether the system can correctly identify each slapping action and drinking action. Therefore, to measure the performance of actions, we use precision and recall. Using the confusion matrix in Table 1, the recall of slapping is a/(a+b+c), the precision of slapping is a/(a+d+g); the recall of drinking is e/(d+e+f), the precision of drinking is e/(b+e+h).

We extracted the features from the raw data set. We set 25 records in one window, and we used the continuous 4 windows to be one instance. 39 features were extracted in one window; thus, 39 * 4= 156 features in one instance. For evaluation purposes, we divide the data set into a training set and a test set. We use 1/4 of the whole data set to be the test set, and 3/4 of the whole data set to be the training set. In the training set, we chose 1/3 data to be the validation set; the validation set is not involved in the training process. The training, testing, and validation sets kept in similar class distributions.

For each of the three data sets, we used six methods to evaluate the performance. Details are listed in Table 2.

Table 2 The different methods with different components

	Decision Tree	Random Forest	Sequence Post-Processing	Validation Threshold	Best Combination of Random Forest
method 1	√				
method 2		√			
method 3		√	√		
method 4		√	√	√	
method 5		√			√
method 6		√	√	√	√

In method 1, we evaluate the performance of the Decision Tree algorithm. In method 2, we want to compare the difference between Decision Tree and Random Forest. In method 3, we expect that there is some improvement with the addition of Sequence Post-Processing to Random Forest. In method 4, we expect that adding validation threshold could help the system improve the accuracy. In method 5, we want to test whether using the best combination of forest size and feature subset size could improve the performance of Random Forest. In method 6, we add all the components, and expect it to have the best performance among the six methods.

For each method, we conduct ten experiments with different pairs of training and testing sets. In each experiment, we measure two true positive rates (for slapping and drinking) and one false negative rate. We report the average value of the ten experiments.

Results from the first data set

The first data set contains 9 knee-slapping motions. The results are shown in Figure 2 and Figure 3.

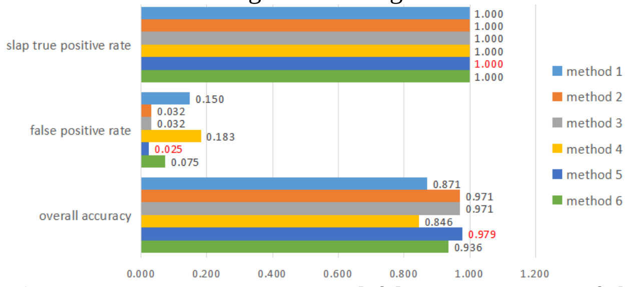

Figure 2. True positive rate and false positive rate of the first data set

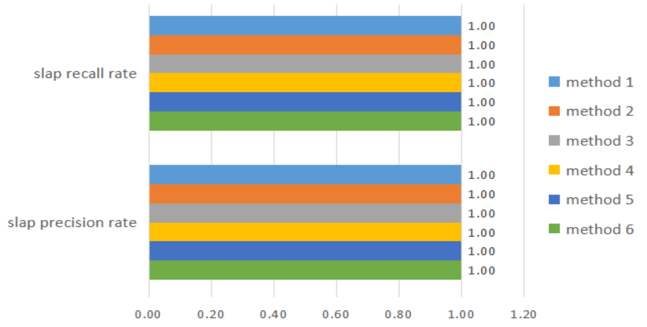

Figure 3. Actions recall rate and precision rate of the first data set

We observe the following points from Figure 2: (1) The true positive rate of the slapping class is 100% in all methods. (2) Method 2 and 3 have the same result because Sequence Post-Processing does not affect the slapping class. (3) In method 4, we include the validation threshold, but the false positive rate increases, and the overall accuracy decreases. Because there are only 9 slapping actions in this data set, part of the 9 actions will be split to be the validation set, which makes the threshold less accurate. (4) In method 5, we use the combination of 100 trees and 15% feature subset rate in Random Forest. Comparing with other methods, method 5 yields the highest true positive rate in the drinking class, and the lowest false positive rate in No_action class, which show that the combination of the two parameters improves the accuracy for Random Forest. (5) As we observe from method 4, adding validation threshold affects the false positive rate. Method 6 contains all the components in method 4, thus the false positive rate in method 6 must be influenced by the validation threshold. However, the false positive rate of drinking in method 6 is lower than it is in method 4, which shows that the best combination of two parameters for Random Forest can improve the accuracy of using validation threshold. For the recall rate and precision rate, all the methods have 100% from Figure 3. One possible reason is that the data set is less complicated and each slapping action can be easily identified.

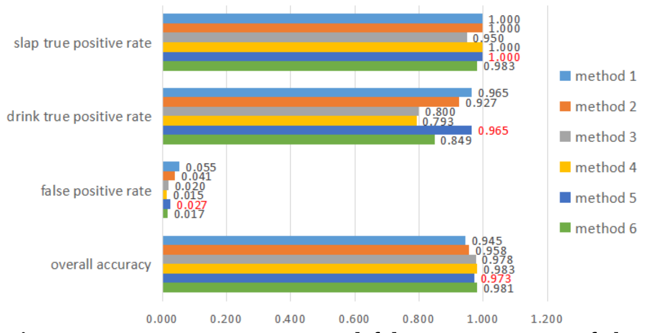

Figure 4. True positive rate and false positive rate of the second data set

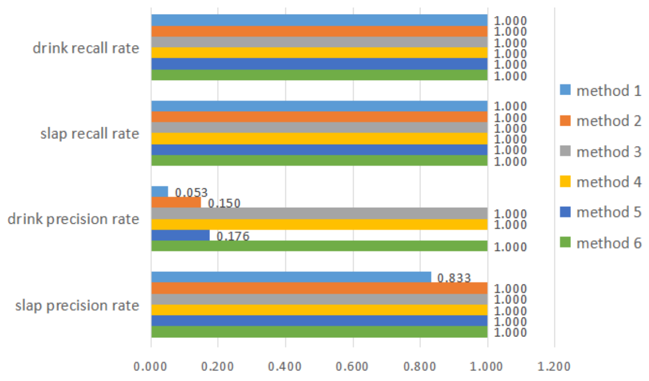

Figure 5. Actions recall rate and precision rate of the second data set

Results from the second data set

The second data set contains 10 head-slapping and 5 tea-drinking motions. The results are shown in Figure 4 and Figure 5.

According to the results in Figure 4, we yield a good true positive rate for the slapping class. In method 1, using the normal standard Decision Tree could already enable us to obtain high accuracy in both slapping and drinking classes. In method 2, Random Forest decreases the false positive rate of no action, but also decreases the true positive rate of drinking. In method 3 and method 4, we obtain the low false positive rates: 2% and 1.5%; however, the true positive rates of drinking class significantly fell to 80% and 79.3%. The trade-off of drinking or no action exists in methods 3 and 4. As the results indicate in method 5, finding the best combination of forest size and feature subset size for Random Forest is a reasonable method. We use 100 trees and 20% feature subset size for Random Forest in this data set. In method 6, since we use the best combination of the two parameters for Random Forest, the true positive rate of drinking becomes better than it is in methods 3 and 4, but is still lower than it was in methods 1, 2 and 5. However, the false positive rate in method 6 is much lower than it is in methods 1 and 2. Therefore, we observe that Sequence Post-Processing and validation threshold have a tradeoff between the true positive rate in drinking and the false positive rate in No_action in this data set. Considering the overall performance, the method 5 is the best choice.

In the second data set, there are only five drink actions, some of them are located in the test set. Therefore, the number of actual drink actions in the test set will be just one or two. As long as there are some incorrectly classified drinking actions, it will rapidly decrease the precision of drinking--the results are shown in Figure 5. That is one possible reason why in all methods the precision of drinking is low. However, in method 3 after Sequence Post-processing is added, it prevents predicting the drink actions which are too short or the interval with slapping action is too short. That is one possible reason the precision rate of drink action to increase in method 3.

Results from the third data set

The third data set contains 29 head-slapping and 29 tea-drinking motions. The results are shown in Figure 6 and Figure 7.

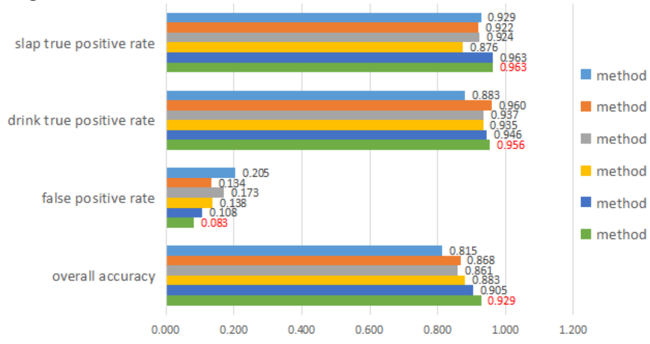

Figure 6. True positive rate and false positive rate of the third data set

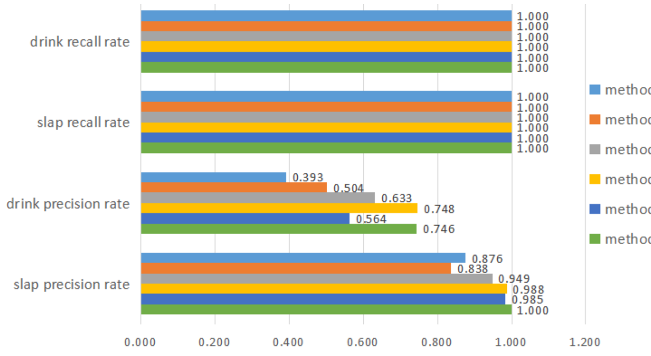

Figure 7. Actions recall rate and precision rate of the third data set

From Figure 6, Decision Tree does not achieve 100% accuracy. Moreover, the accuracy of the drinking class falls to 88.3%, and the false positive rate increases to 20.5%. According to the results from Decision Tree, the data set might contain noise. After implementing Random Forest in method 2, the accuracy of drinking increases, and the false positive rate falls to 11.7%. These results indicate that Random Forest has a strong ability to handle the noisy data. From method 3 to method 4, there is no improvement by implementing the validation threshold or Sequence Post-Processing. On the contrary, using validation threshold in method 4 has a bad effect on the accuracy of slapping. One possible reason is that Random Forest is weak in finding the threshold of the slapping class. In method 5, we use 100 trees and 30% feature subset rate for Random Forest. As we expect, using the best combination of forest size and feature subsets size of Random Forest significantly improves the performance. The accuracy of slapping and drinking reaches 96.3% and 94.6% respectively, and the false positive rate decreases from 13.8% to 10.8%. In method 6, we observe that the false positive rate falls to 8.3%, the true positive rate of drinking increases from 94.6% to 95.6%, and the true positive rate of slapping remains the same. It shows that using validation threshold and Sequence Post-Processing has a good effect on decreasing the false positive rate of No_action, and increasing the true positive rate of drinking in this data set.

According to Figure 7, Decision Tree has a drink precision of just 39.3%. However, Random Forest increases the drink precision rate to 50.4%. After adding validation threshold and Sequence Post-processing, the drink precision reaches at 74.8%. As we expect, method 6 obtains the best performance in both recall and precision. It shows that Sequence Post-processing could correct some of the mistakes. Results from the third data set shows that finding the best combination of forest size and size of feature subset for Random Forest is important.

Contributions and Limitations

Our research has a few contributions. First, we propose an algorithm for detecting harmful behaviors among autistic children using a wearable motion sensor. For evaluating the effectiveness of therapies, the algorithm can be used to automate the process of counting harmful behaviors.

Second, according to the experimental results, we obtain an over 96% true positive rate of the slapping class in the three data sets. At the same time, we also obtain an over 95% true positive rate of the drinking class. These achievements give us confidence in detecting the harmful hand actions and distinguishing harmful hand actions from normal hand actions.

Third, our algorithm could find a more effective combination of forest size and feature subset size for Random Forest, which yields a better performance than the parameter values used by Brieman (2001). For different data sets, the combination of these two parameters may change, so automating the selection of parameter values is necessary.

Fourth, we propose the Sequence Post-Processing method to correct some of the mistakes. The method finds parameters from the training data to decide if an action is too short, too long, or too close to another action and tries to correct possible mistakes in a sequence of classifications. Our experimental results indicate that Sequence Post-Processing helps decrease the false positive rate.

Our study is limited to data sets with two harmful behaviors from one person. Further studies would include more harmful behaviors from multiple people.

References

Anjo, M. D. S., Pizzolato, E. B., & Feuerstack, S., (2012) A real-time system to recognize static behaviors of Brazilian sign language (libras) alphabet using Kinect. In *Proceedings of the 11th Brazilian Symposium on Human Factors in Computing Systems*, pp. 259-268.

Zhang, L. (2015). Detecting Harmful Hand Behaviors with Machine Learning from Wearable Motion Sensor Data. MS Thesis, Florida Institute of Technology, Melbourne, FL.

Breiman, L. (2001) Random forests. *Machine Learning*, 45(1):5-32.

Hong, P., Turk, M., & Huang, T. S., (2000) Behavior modeling and recognition using finite state machines. In *Automatic face and behavior recognition, fourth IEEE international conference on* pp. 410-415.

Krause, A., Siewiorek, D. P., Smailagic, A., & Farringdon, J., (2003) Unsupervised, dynamic identification of physiological and activity context in wearable computing. *Proceedings. Seventh IEEE International Symposium on Wearable Computers*. p. 88.

Li, C., Lin, M., Yang, L. T., & Ding, C., (2014) Integrating the enriched feature with machine learning algorithms for human movement and fall detection. The *Journal of Supercomputing*, 67(3), pp.854-865.

Mannini, A., & Sabatini, A. M., (2010) Machine learning methods for classifying human physical activity from on-body accelerometers. *Sensors*, 10(2), pp.1154-1175.

Nam, Y., Rho, S., & Lee, C., (2013) Physical activity recognition using multiple sensors embedded in a wearable device. *ACM Transactions on Embedded Computing Systems (TECS)*, 12(2), p.26.

Pylvänäinen, T. (2005) Accelerometer based behavior recognition using continuous HMMs. In *Pattern Recognition and Image Analysis*, pp. 639-646.

Wu, J., Pan, G., Zhang, D., Qi, G., & Li, S., (2009) Behavior recognition with a 3-d accelerometer. In *Ubiquitous intelligence and computing*, pp. 25-38.

Ward, J. A., Lukowicz, P., Troster, G., & Starner, T. E., (2006) Activity recognition of assembly tasks using body-worn microphones and accelerometers. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10), pp. 1553-1567.

Yang, J. Y., Wang, J. S., & Chen, Y. P., (2008) Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers. *Pattern recognition letters*, 29(16), pp. 2213-2220.

Yang, X., Dinh, A., & Chen, L., (2010) Implementation of a wearerable real-time system for physical activity recognition based on naive Bayes classifier. In *Bioinformatics and Biomedical Technology (ICBBT)*, pp. 101-105.