Improving Wearable-based Fall Detection with unsupervised learning

Mirko Fáñez
Instituto Tecnológico de Castilla y León, Burgos, 09001, Spain
mirko@mirkoo.es

José R. Villar
University of Oviedo, Computer Science Department, Oviedo, 33005, Spain
villarjose@uniovi.es

Enrique de la Cal
University of Oviedo, Computer Science Department, Oviedo, Asturias, 33005, Spain
delacal@uniovi.es

Víctor M. González
University of Oviedo, Electrical Engineering Department, Gijón, 33204, Spain
vmsuarez@uniovi.es

Javier Sedano
Instituto Tecnológico de Castilla y León, Burgos, 09001, Spain
javier.sedano@itcl.es

February 10, 2020

Abstract

Fall detection (FD) is a challenging task that has received the attention of the research community in the recent years. This study focuses on FD using data gathered from wearable devices with tri-axial accelerometers (3DACC), developing a solution centered in elderly people living autonomously. This research includes three different ways to improve a FD method: i) an analysis of the event detection stage, comparing several alternatives, ii) an evaluation of features to extract for each detected event and, iii) an appraisal of up to 6 different clustering scenarios to split the samples in subsets that might enhance the classification. For each clustering scenario, a specific classification stage is defined. The experimentation includes publicly available simulated fall data sets. Results show the guidelines for defining a more robust and efficient FD method for on-wrist 3DACC wearable devices.

Keywords: Fall Detection; Unsupervised Learning; Clustering; One-Class Classifier

1 Introduction

Fall Detection (FD) is a major challenge with many applications to healthcare, work safety, etc. The best rated commercial products only reach an 80% of success [19]. The publicly available solutions include context-aware systems (like video systems [24]) or wearable systems [13]. These latter allow care institutions to optimize their services at lower cost and be available to a wider part of the population. They are crucial because the high percentage of non-completely self-sufficient people (e.g. elderly people), and their desire to live autonomously at their own house. By far, tri-axial accelerometers (3DACC) represents the preferred wearable-based solution [3, 11, 12, 23, 25].

For instance, a feature extraction stage and Support Vector Machines have been applied directly in [23, 25], using some transformations and thresholds with very simple rules [3, 12, 14]. A comparison of different classification algorithms has been presented in [11], and several threshold-based fall detection algorithms using 3DACC data were presented in [3, 9, 10]. There are also studies concerned with the dynamics in a fall event [2, 7]. Additionally, Abbate et al. proposed the use of these dynamics as the basis of the FD algorithm [1], with moderate computational constraints but a high number of thresholds to tune. The common characteristic in all these solutions is that the wearable devices were placed on the waist or in the chest. In [15], the sensors were placed on the wrist, using Abbate et al. solution plus an over-sampling data balancing stage (using the Synthetic Minority Over-sampling Technique - SMOTE) and a feed-forward Neural Network.

This study makes use of the solution in [1, 16] as the base line, experimenting with several options for the event detection and the feature extraction. Furthermore, a clustering stage is introduced to split the domain in smaller sub-problems; each obtained

cluster induces a different classification model specialized in the assigned instances. Up to 6 different clustering scenarios are evaluated, each one with a different classification strategy. The solution is evaluated using publicly available simulated FD data sets; the experimentation evaluates each part of this study, leading to the selection of the best configuration for FD.

The structure of this study is as follows. The next section deals with the description of the base line solution. Afterwards, the different issues to study in this research are introduced in 3. The complete experimentation is described in Sect. 4, while the obtained results and discussion are included in Sect. 5. The study ends with the conclusions draw from this research.

2 A three stage FD solution

The base-line solution is the FD system proposed in [1, 16], where a peak detection is followed by a feature extraction; these features are then classified using either Neural Networks (NN) and Support Vector Machines (SVM) among other well-known machine learning techniques. The dynamics of a fall used to detect peaks and the very simple Finite State Machine (FSM) that is used to detect the falls are shown in the left and right part of Figure 1.

The data gathered from a 3DACC located on the wrist is processed using a sliding window. A peak candidate is detected if the magnitude of the acceleration a_t is higher than a specified value $th_1 \times g$ (th_1 is a predefined threshold and $g = 9.8$ m/s is the gravity) followed by a period of calm greater than 2500 milliseconds (the sum of the two timers used in the FSM). The time of the peak occurrence is call *peak time* (*pt*, **point 1** in the left part of Figure 1).

The **impact end**, *ie* (**point 2**), denotes the end of the fall event; it is the last time for which the a_t value is higher than $th_2 \times g$ ($th_2 = 1.5$). Finally, the **impact start**, *is* (**point 3**), denotes the starting time of the fall event, computed as the time of the first sequence of an $a_t <= th_3 \times g$ ($th_3 = 0.8$) followed by a value of $a_t >= th_2 \times g$. The *impact start* must belong to the interval $[ie - 1200\ ms, peak\ time]$. If no *impact end* is found, then it is fixed to *peak time* plus 1000 ms. If no *impact start* is found, it is fixed to *peak time*. The interval [*impact start*, *impact end*] is called *peak window*.

Whenever a fall-like peak is found, the following transformations are then calculated using the data in the *peak window*:

- Average Absolute Acceleration Magnitude Variation, $AAMV = \sum_{t=is}^{ie-1} \frac{|a_{t+1} - a_t|}{N}$, with N the number of samples in the interval.
- Impact Duration Index, $IDI = impact\ end - impact\ start$.
- Maximum Peak Index, $MPI = max(a_t)$, $\forall t \in [is, ie]$.

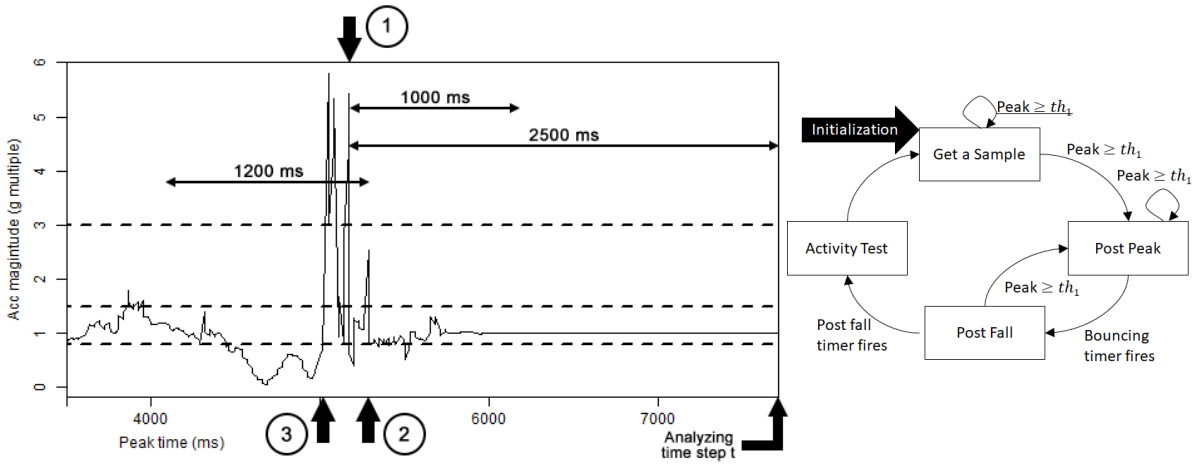

Figure 1: The base-line solution. To the left, the dynamics of a fall and the times and thresholds used for defining the peak window. To the right, the finite state machine.

- Minimum Valley Index, $MVI = min(a_t)$, $\forall t \in [is - 500, ie]$.
- Peak Duration Index, $PDI = peak\ end - peak\ start$, with *peak start* defined as the time of the last magnitude sample below $th_{PDI} = 1.8 \times g$ occurred before pt, and *peak end* defined as the time of the first magnitude sample below $th_{PDI} = 1.8 \times g$ occurred after pt.
- Activity Ratio Index, ARI, calculated as the ratio between the number of samples not in $[th_{ARIlow} = 0.85 \times g, th_{ARIIhigh} = 1.3 \times g]$ and the total number of samples in the 700 ms interval centered in $(is + ie)/2$.
- Free Fall Index, FFI, the average acceleration magnitude in the interval $[t_{FFI}, pt]$. The value of t_{FFI} is the time between the first acceleration magnitude below $th_{FFI} = 0.8 \times g$ occurring up to 200 ms before pt; if not found, it is set to $pt - 200ms$.
- Step Count Index, SCI, measured as the number of peaks in the interval $[pt - 2200, pt]$.

The real-time fall detection is proposed as follows (see Fig. 2). The acceleration magnitude value for each instant of time t is analyzed looking for a peak that marks where a fall event candidate appears. Whenever a 2500 ms of low activity occurs after a peak, the entire time series (TS) for the last 7500 ms is passed to the feature extractor, which determines the *impact end*, *impact start* and all the remaining features. These features are passed to the trained clustering and classifier models, which determine whether it was a FALL or NOT_FALL.

Figure 2: Block diagram of the base line method. Samples are continuously monitored to detect peaks. When a peak is detected, a feature extraction stage takes place. Finally, the extracted features are classified.

3 Improvement issues to analyze

The base line method described before has several drawbacks. For instance, some of the features extracted might have small variation between samples belonging to different labels. Moreover, the peak window usually approximates to $[pt, ie]$, which also reduces the significance of some features. Finally, some upper classification limits might be found due to considering a single classifier; if a suitable division in clusters of the domain is found then several classifiers can be obtained and the overall performance can be enhanced.

All these issues have been addressed in this study. On the first hand, the event detection method described before (using a peak threshold and an FSM) is compared with the alternative proposed in a previous research [8]. On the second hand, unsupervised learning is used to group the instances that arise from each detected peak; up to 6 different clustering configurations (from now on, they are called scenarios) are presented and compared. In between, an alternative for the feature subset presented in [8] is compared with the feature subset proposed in [1, 16].

3.1 An event detection alternative

As mentioned before, the event detection method produces a peak window closer to $[pt, ie]$, penalizing several extracted features. In past research [8], we have used this event detection together with the same FSM and threshold th_1 proposed in [1]. In this research we perform differently: we fix the peak window to $[pt-200, pt+1000]$; this change makes some features to become irrelevant, so they need to be substituted by new independent ones. This modification was suggested by the fact that a relevant number of instances included nearly constant features, such as IDI.

Besides, the threshold proposed in [1] is fixed for all the users. We have found that this might not be the case [16], proposing an optimized value. However, we do believe this threshold must be specific for each user; in this study we propose a method to determine it. To do so, walking was defined as the standard activity for determining this value. The user should walk during a relatively short period; meanwhile, the 1000 maximum values of the acceleration magnitude are recorded ($a_{max}[k]$, with $k \in \{1, \ldots, 1000\}$). The value of the threshold is determined as $th_1 = 0.9 \times \sum_{k=1}^{1000} a_{max}[k]/1000$, that is, a value a bit smaller than the mean maximum value of the acceleration when walking.

Furthermore, the mean and standard deviation during this walking period will be stored and used later to standardize the TS.

3.2 A second set of features

If the event detection from previous subsection is used, the features proposed in [1, 16] might become meaningless. Therefore, in [8] a new set of features was evaluated. This second set of features eliminates those transformations that were found meaningless after fixing the peak window size and includes three more variables that focus on specific intervals of the peak window. The list of the new set of features is:

- Average Absolute Acceleration Magnitude Variation ($AAMV$).
- Maximum Peak Index (MPI).
- Minimum Valley Index (MVI).
- Free Fall Index (FFI).
- Step Count Index (SCI).
- Pre-impact Activity Index ($PrAI$), measuring the AAMV in the part of the fall window before the peak, Eq. 1.

$$PrAI = \sum_{t=is}^{pt} \frac{|a_{t+1} - a_t|}{N} \tag{1}$$

- Post-impact Activity Index ($PoAI$), measuring the AAMV in the part of the fall window after the peak, Eq. 2.

$$PoAI = \sum_{t=pt}^{pt+500} \frac{|a_{t+1} - a_t|}{N} \tag{2}$$

- Long-term Activity Index (LAI), measuring the AAMV in the interval where the user must be more or less still after a fall, Eq. 3.

$$LAI = \sum_{t=pt+500}^{ie} \frac{|a_{t+1} - a_t|}{N} \tag{3}$$

3.3 Unsupervised learning scenarios

As explained before, event detection is performed on a TS; for each detected peak the defined feature subset is computed. Therefore, a new data set is extracted containing

an instance for each detected peak. In previous research [1, 15, 8, 22], all the instances were considered to obtain a single model which would be used to classify incoming new instances. However, the performance of the models showed an upper bound which was not possible to overcome. In this research we consider the divide and conquer approach to split the data into clusters, and to classify each cluster with a suitable technique.

This study includes the evaluation of several scenarios of clustering; the clustering is based on the K-Means Clustering Algorithm using R *stats package* [20] version v3.5.0. Each scenario defines a specific data pre-processing before clustering. Furthermore, each scenario also defines a technique to classify the instances belonging to each cluster. To classify, two different methods were proposed: i) Support Vector Machine (SVM) classification using R *e1071 package* [17] version v1.7.0, and ii) k-Nearest Neighbour (k-NN) classification using R *class package* [21] version v7.3.15.

All these options are described next; in all of them, we use the ratio of 0.4 to determine whether there is class balance or not:

Scenario 1 *Clustering all the data together.* The clustering algorithm analyzes all the instances; the optimal number of clusters is fetched from 2 to 20 using the elbow's rule. In any resulting group where all the instances belong to the same label A (FALL or NOT_FALL), a new instance belonging to the group is labelled as A. On the other hand, when for a group there exists instances belonging to different classes, a new instance assigned to this group is labelled using a specific SVM. That is, for each of these groups a SVM model is obtained with the instances belonging to the group; whenever the balance of the instances is in compromise, SMOTE balancing is applied [6].

Scenario 2 *Clustering the NOT_FALL data only.* In this scenario, the training data set is split in NOT_FALL and FALL instances; the mean and standard deviation of the NOT_FALL instances are used later to standardize all them before determining what cluster an instance belongs to. The clusters are determined using the normalized NOT_FALL training data. Afterwards, the normalized FALL training instances are assigned to the closest cluster. However, the centroids are not updated. As before, when all the instances in a cluster belong to the same class, this latter is used to classify new instances assigned to this cluster. When a cluster includes instances from more than one class, a SVM trained with the instances of the cluster (if needed, balanced with SMOTE as well) is used to classify new instances assigned to the group.

Scenario 3 *Clustering the NOT_FALL data only adding the FALL data and recomputing the centroids.* This scenario works as in the previous case for the clustering but just recomputing the clusters's centroids.

Scenario 4 *Clustering data from each label independently and using the computed centroids.* This is a totally different concept. The training NOT_FALL instances and the training FALL instances are clustered independently. The optimum number of clusters K varies from 4 to 22 for NOT_FALL instances, and from 2 to 12 for FALL instances.

The centroids from both sets of clusters (FALL and NOT_FALL) are used for a k-NN classifier for which better value k is also found. A new instance is labeled as FALL if the probability of FALL (p_F) returned from the k-NN model is higher that the probability of NOT_FALL (p_{NF}) plus a certain threshold ($p_F > p_{NF} + th$). Vice versa, when $p_{NF} > p_F + th$, the instance is classified as NOT_FALL. For those cases in which there is no clearly greater probability, the Equation 4 and Equation 5 are used. The highest value decides what label will be assigned. The value of th has been set experimentally to 0.10.

$$pp_F(instance) = (1 - \frac{\sum_{i=0}^{Q_F} dist(instance, {centroid_i}^F)}{\sum_{\substack{i=0 \\ CL \in \{F,\ NF\}}}^{Q_{CL}} dist(instance, {centroid_i}^{CL})}) * p_F \quad (4)$$

$$pp_{NF}(instance) = (1 - \frac{\sum_{i=0}^{Q_{NF}} dist(instance, {centroid_i}^{NF})}{\sum_{\substack{i=0 \\ CL \in \{F,\ NF\}}}^{Q} dist(instance, {centroid_i}^{CL})}) * p_{NF} \quad (5)$$

We call pp_F the predicted probability of being a FALL and pp_{NF} the predicted probability of being a NOT_FALL for an instance. These predicted probabilities are computed for the instance. The denominator makes use of the same number of clusters belonging to each class; more specifically, the minimum between the number of FALL clusters (Q_F) and the number of NOT_FALL clusters (Q_{NF}). For the label with higher number of clusters, the closest clusters are used.

Scenario 5 *One-class SVM for each cluster.* Clustering takes place using the training NOT_FALL instances only. For each cluster, the assigned NOT_FALL instances are used in training a one-class SVM (OCSVM). To classify a new instance, the OCSVM from the closest cluster is used to determine whether it belongs to the normal NOT_FALL class or it should be labeled as FALL.

Scenario 6 *95% PCA + Scenario 4.* In this scenario we apply PCA to the normalized subset of features, trying to reduce the dimension of the input. The level of representation is set to 95%. Then, the same scheme as in Scenario 4 is used.

4 Experimentation set up

This section deals with up to four different topics: the data sets used in this research and the three stages in the experimentation. These three stages are i) the comparison of the two event detection methods, ii) the comparison of the two feature extraction subsets and iii) the evaluation of the clustering scenarios together with the classification results.

For the last part of the evaluation, the best event detection and the best feature subset will be chosen. Each of these topics is cover in the following 4 different subsections.

4.1 Experimental data sets

This experimentation includes two publicly available simulated falls data sets; these data sets were gathered with participants wearing, at least, one 3DACC sensor placed on a wrist. Each participant performed several repetitions from a certain catalogue of Activities of Daily Living (ADL) and fall types. Each of the repetitions is stored as a multivariate TS with the three acceleration axis, along with the corresponding label of the ADL or fall type. Due to the lack of homogeneity in the ADL set and in the experimentation design [4], in this study we will manage each data set independently. Furthermore, we are going to focus on the fall types that are common to both data sets: we consider only forward, backward and lateral falls when the participant is standing still upright.

The data sets that are going to be considered are:

UMA Fall data set [5] : 17 participants for a total of 531 TS (208 of them are labelled with one of the possible fall types). The sampling frequency is 20 Hz. Includes forward, backward and lateral simulated falls, as well as several ADL (running, hopping, walking and sitting). The number of repetitions of each ADL or fall varies from one user to another. We have only considered those participants with more than 20 TS, provided that at least 9 of them are simulated falls. Although participant 17 did not performed any backward fall, we kept it due to the number of falls of the other types.

Ozdemir&Barshan [18] : 17 participants for a total of 4406 fall TSs and 8891 ADL TSs. The sampling frequency is 25 Hz. Includes 20 different FALL classes: front-lying, front-protecting-lying, front-knees, back-lying, syncope, etc.; as well as 16 types of ADL. Each participant performs, on average, 5 runs of every ADL and FALL simulation.

4.2 Evaluation of event detection methods

Two different aspects are evaluated: on the one hand, what event detection method identifies more fall events. On the second hand, the number of fall alarms proposed by

each method is analyzed. It is worth noticing that this evaluation is performed for each data set independently.

Therefore, we split the data sets in two: the FALL TS and the NOT_FALL TS. With the first group (FALL TS), the number of peak candidates should be ideally equal to the number of TS. For the second group (NOT_FALL TS), we manually split again in two: those TS gathered from high level ADL (HADL, such as Running, etc.) and those TS gathered from low level ADL (LADL, such as standing still, etc.). For the LADL there should not be any peak candidate; for the HADL, the smaller the number of peak candidates the better.

The activities selected as LADL are WALKING, CLIMBING UPSTAIRS, CLIMBING DOWNSTAIRS, LIMPING, WASHING DISHES, VACUUMING, SWEEPING, LYING AND STANDING, SITTING AND STANDING, RESTING, SLEEPING, STANDING, SITTING, LYING, RISING, SQUATTING, and COUGHING. The activities selected as HADL are: RUNNING, JOGGING, JUMPING, HOPPING, STUMBLE, SAWING, and BENDING.

For each case, we record the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). With these values, the performance of the two event detection methods will be evaluated.

4.3 Evaluation of the feature subsets

It is worth noticing that the feature subset in [1, 16] is not compatible with the event detection proposed in [8]: due to the fact that the peak window is of fixed size, several features become meaningless. Therefore, for this evaluation, the event detection and feature subset described in Sect.2 will be compared with the event detection and feature subset detailed in Sect. 3.1 and Sect. 3.2.

To select the best feature subset, we use the corresponding event detection method to identify the peaks and then, for each peak window, we compute the feature subset. A final classification stage using SVM follows afterwards, so a label is assigned to each 8 features instance. The best parameter subset will be found using 10-fold cross validation. The overall performance obtained with the best parameter subset and the complete data set will be used as comparison. As before, this evaluation is performed once for each data set.

4.4 Evaluation of the different scenarios

The aim of this experimentation is to determine what scenario performs better together with the previously selected best event detection method and feature subset. Because each scenario includes a classification method, the results of this experimentation stage is measured in terms of classification error, with the TP, TN, FP and FN, and the associated

measurements Sensitivity and Specificity.

The experimentation runs as follows. For each data set, perform a 10-fold shuffling of the participants, splitting the participants in 10 different subsets to be used later. At each cross-validation stage, we perform as follows. For each participant belonging to a train fold, standardize the TS as proposed in [16] in case that the Abbate et al pair of <event detection, feature subset> is the best one; otherwise, standardize the TS using the mean and standard deviation of the tuning Walking ADL (see Sect. 3.1). Afterwards, perform the event detection and the feature extraction for each detected peak.

The instances of feature extraction computed for the TS of all the participants from training constitute the *Scenario Training Data Set* (STDS). Performing similarly with the participants that belong to the validation data set we obtain the *Scenario Validation Data Set* (SVDS). Both STDS and SVDS are z-scored with the STDS mean and standard deviation.

Finally, with the STDS we perform the scenario training (for all the scenarios), while we use the SVDS for the validation.

5 Results and discussion

The results follow the explicit order shown in the previous section. Therefore, Table 1 to Table 3 include the results obtained in the comparison of the event detection methods. The results concerning the comparison of the two features subsets are depicted in Table 4. Finally, the comparison among the different scenarios is shown in Table 5.

In Table 1, the results when the event detection method analyzes each of the TS labelled as FALL are shown. The ideal case would be that a peak should be detected for all the FALL TS belonging to each data set. The real case is that there are peaks that have raised due to a fall that has not been detected. However, the improvements with the proposal in [8] are clear: up to a 16% in the decrease of the number of undetected falls.

On the other hand, Table 2 and Table 3 show the performance of the event detection methods when analyzing low activity ADL TS and high activity ADL TS. As it can be seen, the number of peaks detected when having high level activities (HADL) is also higher with the proposal in [8]. Although these are false alarms, it is interesting noticing that the method detects these peaks that are to be classified later, relying in the quality of the classifier.

However, when analyzing the LADL, it can be seen that the proposal from [1, 16] fires at a higher rate than the proposal in [8]. In this case, the smaller the number of detected peaks the better as long as these are low level activities.

Table 1: Comparison of the event detection methods when analyzing the FALL TS from each data set. Sens represents the Sensitivity.

Dataset	Abbate algorithm					Neurocomputing'19 algorithm				
	TN	TP	FP	FN	Sens	TN	TP	FP	FN	Sens
UMA Fall	0	153	0	57	0.7285	0	162	0	48	0.7714
OzmedirBarshan	0	4112	0	294	0.9332	0	4231	0	175	0.9602

Table 2: Comparison of the event detection methods when analyzing the NOT_FALL TS from each data set. The figures are the obtained counters.

Dataset		Abbate algorithm				Neurocomputing'19 algorithm			
		TN	TP	FP	FN	TN	TP	FP	FN
UMA Fall	High-Act	0	132	0	98	0	141	0	89
	Low-Act	68	0	23	0	70	0	21	0
OzmedirBarshan	High-Act	0	1023	0	1191	0	952	0	1262
	Low-Act	5960	0	717	0	5982	0	695	0

Table 3: Comparison of the event detection methods when analyzing the FALL TS from each data set. The statistical results.

Dataset		Abbate algorithm		Neurocomputing'19 algorithm	
		Sens	Spec	Sens	Spec
UMA Fall	High-Act	0.5739	0	0.6130	0
	Low-Act	0	0.7472	0	0.7692
OzmedirBarshan	High-Act	0.4620	0	0.4299	0
	Low-Act	0	0.8926	0	0.8959

Table 4 shows the results obtained for the comparison of the two feature subsets. In this case, the classification Sensitivity and Specificity are included; these values were obtained with the best parameter subset using 10-fold cross validation. The set of features detailed in Sect. 3.2 clearly outperforms the proposed in [1, 16]; therefore, this feature subset will be used in the last stage of this experimentation.

Finally, the comparison of the different scenarios using the event detection and feature subset detailed in Sect. 3.1 and Sect. 3.2 is shown in Table 5. The results in this table are the counters, the Sensitivity and the Specificity for the validation data set; these values were obtained for the best parameter subset found with 10-fold cross validation in each scenario. All these figures were obtained training the SVM with the caret package in R. The number of clusters for each scenario were {8, 6, 6, 6, 22 for NOT_FALL and 8 for FALL, 6, 8} for Scenario 1 to Scenario 6, respectively. In case of Scenario 5, the best number of neighbors was 5, with a th set to 0.10.

Table 4: Comparison of the statistical results of the two feature subsets evaluated for each data set.

Dataset	Abbate algorithm		Neurocomputing'19 algorithm	
	Sens	Spec	Sens	Spec
UMA Fall	0.8038	0.8170	0.8393	0.9350
OzmedirBarshan	0.8177	0.9737	0.9202	0.9940

A quick glance at these results shows the Scenario 4 as the best learning configuration. Reporting a high remarkable performance with the OzmedirBarshan data set and a more discrete one with the UMA Fall data set. These results suggest that perhaps introducing probabilistic models or methods based in possibility (such Fuzzy Rule Based Systems) might improve the FD performance. Surprisingly, Scenario 6 showed very poor results; we need to investigate why these unexpected figures.

Interestingly, if you compare Table 4 with Table 5, it might seem that just using SVM is enough; no clustering might be needed. However, it is important to remember that the results depicted in Table 4 were obtained with the training data set, that is, after the cross validation to find the best configuration; this is the output of the model with the best parameter subset when trained with the complete training data set, as stated in the package caret. In case of the results in Table 5, these results were obtained with the validation data set, which explains the worse performance.

Table 5: Comparison of the different Scenarios' performance. Sens and Spec stand for Sensitivity and Specificity, correspondingly.

Scenario	Dataset	TN	TP	FP	FN	Sens	Spec
Scenario 1	UMA Fall	255	129	66	81	0.6142	0.7943
	OzmedirBarshan	6312	3260	2579	1146	0.7399	0.7099
Scenario 2	UMA Fall	223	144	98	66	0.6857	0.6947
	OzmedirBarshan	7023	3659	1868	747	0.8304	0.7898
Scenario 3	UMA Fall	272	135	49	75	0.6428	0.8473
	OzmedirBarshan	6857	3521	2034	885	0.7991	0.7712
Scenario 4	UMA Fall	263	148	58	62	0.7047	0.8193
	OzmedirBarshan	8090	4094	801	312	0.9291	0.9099
Scenario 5	UMA Fall	194	151	127	59	0.7190	0.6043
	OzmedirBarshan	5866	3985	3025	421	0.9044	0.6597
Scenario 6	UMA Fall	214	137	107	73	0.6523	0.6667
	OzmedirBarshan	6143	3327	2748	1079	0.7551	0.6909

6 Conclusions

The detection of falls is very important mainly for the elderly population. In this study, the use of an unsupervised learning stage prior to the classification of an event is analyzed.

Up to 6 different scenarios are proposed, including i) Clustering all the data together, ii) Clustering the NOT_FALL data only, iii) Clustering the NOT_FALL data only, adding the FALL data, and re-calculating the centroids, iv) Clustering data from each label independently and using the computed centroids as the training data of a k-NN algorithm, v) Clustering the NOT_FALL instances plus learning One-class SVM for each cluster and vi) 95% PCA + Scenario 4. Furthermore, the comparison of two event detection methods and of two feature extraction subsets is performed as well. In the experimentation, two publicly available simulated fall data sets including ADLs have been used.

Results show that the best performance has been found for i) the event detection based on selecting an activity as the base line, standardizing the TS with the mean and standard deviation of the user when performing this ADL, ii) the new feature subset proposed in [8] and detailed in this research and, iii) the fourth scenario that independently cluster the NOT_FALL TS and the FALL TS, using the obtained centroids as the training data of a k-NN classifier. Further research includes introducing probabilistic and possibility models and also studying the poor performance obtained with the PCA method.

Acknowledgement

This research has been funded by the Spanish Ministry of Science and Innovation under project MINECO-TIN2017-84804-R and by the Grant FCGRUPIN-IDI/2018/000226 project from the Asturias Regional Government.

References

[1] Abbate, S., Avvenuti, M., Bonatesta, F., Cola, G., Corsini, P., AlessioVecchio: A smartphone-based fall detection system. Pervasive and Mobile Computing 8(6), 883–899 (Dec 2012)

[2] Abbate, S., Avvenuti, M., Corsini, P., Light, J., Vecchio, A.: Wireless Sensor Networks: Application - Centric Design, chap. Monitoring of human movements for fall detection and activities recognition in elderly care using wireless sensor network: a survey, p. 22. Intech (2010)

[3] Bourke, A., O'Brien, J., Lyons, G.: Evaluation of a threshold-based triaxial accelerometer fall detection algorithm. Gait and Posture 26, 194–199 (2007)

[4] Casilari, E., Santoyo-Ramón, J.A., Cano-García, J.M.: Analysis of public datasets for wearable fall detection systems. Sensors 17(1513), 4324 – 4338 (2017), http://www.mdpi.com/1424-8220/17/7/1513

[5] Casilari, E., Santoyo-Ramn, J.A., Cano-Garca, J.M.: Umafall: A multisensor dataset for the research on automatic fall detection. Procedia Computer Science 110(Supplement C), 32 – 39 (2017), http://www.sciencedirect.com/science/article/pii/S1877050917312899

[6] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research pp. 321–357 (2002)

[7] Delahoz, Y.S., Labrador, M.A.: Survey on fall detection and fall prevention using wearable and external sensors. Sensors 14(10), 19806–19842 (2014), http://www.mdpi.com/1424-8220/14/10/19806/htm

[8] Fañez, M., Villar, J.R., de la Cal, E., González, V.M., Sedano, J., Khojasteh, S.B.: Anomaly intelligent fall detection: mixing user-centered and generalized models. in evaluation for Neurocomputing (2019)

[9] Fang, Y.C., Dzeng, R.J.: A smartphone-based detection of fall portents for construction workers. Procedia Eng. 85, 147–156 (2014)

[10] Fang, Y.C., Dzeng, R.J.: Accelerometer-based fall-portent detection algorithm for construction tiling operation. Autom. Constr. 84, 214–230 (2017)

[11] Hakim, A., Huq, M.S., Shanta, S., Ibrahim, B.: Smartphone based data mining for fall detection: Analysis and design. Procedia Computer Science 105, 46–51 (2017), http://www.sciencedirect.com/science/article/pii/S1877050917302065

[12] Huynh, Q.T., Nguyen, U.D., Irazabal, L.B., Ghassemian, N., Tran, B.Q.: Optimization of an acc. and gyro.-based fall det. algorithm. Journal of Sensors (2015)

[13] Igual, R., Medrano, C., Plaza, I.: Challenges, issues and trends in fall detection systems. BioMedical Engineering OnLine 12(66) (2013), http://www.biomedical-engineering-online.com/content/12/1/66

[14] Kangas, M., Konttila, A., Lindgren, P., Winblad, I., Jämsaä, T.: Comparison of low-complexity fall detection algorithms for body attached accelerometers. Gait and Posture 28, 285–291 (2008)

[15] Khojasteh, S.B., Villar, J.R., de la Cal, E., González, V.M., Sedano, J., YAZG̈AN, H.R.: Evaluation of a wrist-based wearable fall detection method. In: 13th International Conference on Soft Computing Models in Industrial and Environmental Applications. pp. 377–386 (2018)

[16] Khojasteh, S.B., Villar, J.R., Chira, C., González, V.M., de la Cal, E.: Improving fall detection using an on-wrist wearable accelerometer. Sensors 18(5) (2018)

[17] Meyer, D., Dimitriadou E, Hornik K., Weingessel A., Leisch F., Chang C.C., Lin C.C.: Probability Theory Group (Formerly: E1071), TU Wien - Package 'e1071' (2019), https://cran.r-project.org/web/packages/e1071/e1071.pdf

[18] Ozdemir, A.T., Barshan, B.: Detecting falls with wearable sensors using machine learning techniques. Sensors 14, 10691–10708 (2014)

[19] Purch.com: Top ten reviews for fall detection of seniors. www.toptenreviews.com/health/senior-care/best-fall-detection-sensors/ (2018)

[20] R Core Team and contributors: K-Means Clustering in R Stats Package (2019), https://stat.ethz.ch/R-manual/R-devel/library/stats/html/kmeans.html

[21] Ripley B. and Venables W.: Functions for Classification - Package 'class' (2019), https://cran.r-project.org/web/packages/class/class.pdf

[22] Villar, J.R., de la Cal, E., Fañez, M., González, V.M., Sedano, J.: User-centered fall detection using supervised, on-line learning and transfer learning. Progress in Artificial Intelligence 8(4), 453–474 (Dec 2019), https://doi.org/10.1007/s13748-019-00190-2

[23] Wu, F., Zhao, H., Zhao, Y., Zhong, H.: Development of a wearable-sensor-based fall detection system. International Journal of Telemedicine and Applications 2015, 11 (2015), https://www.hindawi.com/journals/ijta/2015/576364/

[24] Zhang, S., Wei, Z., Nie, J., Huang, L., Wang, S., Li, Z.: A review on human activity recognition using vision-based method. Journal of Healthcare Engineering 2017 (2017)

[25] Zhang, T., Wang, J., Xu, L., Liu, P.: Fall detection by wearable sensor and one-class svm algorithm. In: Huang DS., Li K., I.G. (ed.) Intelligent Computing in Signal Processing and Pattern Recognition, Lecture Notes in Control and Information Systems, vol. 345, pp. 858–863. Springer Berlin Heidelberg (2006)