

A Smartphone-Based System for Detecting Falls Using Anomaly Detection

Vincenzo Carletti^(✉), Antonio Greco, Alessia Saggese, and Mario Vento

Department of Information Engineering,
Electrical Engineering and Applied Mathematics, University of Salerno,
Fisciano, Italy

{vcarletti, agreco, asaggese, mvento}@unisa.it
<http://mivia.unisa.it>

Abstract. As reported by the World Health Organization, falls are a severe medical and financial issue; they represent the second leading cause of unintentional injury death, after road traffic injuries. Therefore, in recent years, the interest in realizing fall detection systems is considerably increased. Although the overall architecture of such systems in terms of its basic components is consolidated, the definition of an effective method to detect falls is a challenging problem due to several difficulties arising when the system has to work in the real environment. A very recent research trend is focused on the realization of fall detection systems running directly on a smartphone, so as to avoid the inconvenience of buying and carrying additional devices. In this paper we propose a novel smartphone-based fall detection system that considers falls as anomalies with respect to a model of normal activities. Our method is compared with other very recent approaches in the state of the art and it is proved to be suitable to work on a smartphone placed in the trousers pocket. This result is confirmed both from the achieved accuracy and the required hardware resources.

Keywords: Smartphone-base falls detection · Embedded pattern recognition · Anomaly detection · One-class classification

1 Introduction

The World Health Organization (WHO) defines a fall as an *event which results in a person coming to rest inadvertently on the ground or floor or other low level*. Globally, falls are a major public health problem [1–3]. An estimated 424.000 fatal falls occur each year, making them the second leading cause of unintentional injury death, after road traffic injuries. Though not fatal, every year approximately 37.3 million falls are severe enough to require medical attention. Such falls are responsible for over 17 million DALYs lost (the Disability-Adjusted Life Year is a measure of overall disease burden, expressed as the number of years lost due to ill-health). The largest morbidity occurs in people aged 65 years or older, young adults aged 15–29 years and children aged 15 years or younger. But, more

generally, elderly, children, disabled individuals, workers, athletes and patients with visual, balance, gait, orthopaedic, neurological and psychological disorders are all at risk from falls. Furthermore, fall-related injuries have relevant financial costs and as highlighted by a recent analysis conducted in Canada, the implementation of effective prevention strategies could create a net savings of over USD 120 million each year. For this reason, there is a growing interest of the scientific community in realizing effective fall detection systems [4–6].

A fall detection system can be defined as a device whose main objective is to alert when a fall event has occurred. Its main purpose is to distinguish whether a given event is a fall or a normal Activity of Daily Living (ADL). The general structure of a fall detection system is shown in Fig. 1; it can be considered a typical pattern recognition system, composed of a batch of sensors, a feature extraction stage and a detection (or classification) stage. An additional component is generally used to notify the detection of fall to other people that will promptly act to help the fallen one.

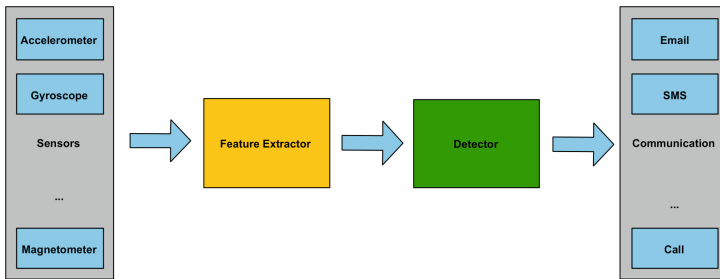


Fig. 1. Common structure of a fall detection system.

In a real-life scenario, a fall detection system potentially reduces some of the adverse consequences of a fall and, more specifically, it can have a direct impact on the rapid provision of assistance after a fall. But, to be effective, this system has to provide the best trade-off between false negatives and false positives. Indeed, if the system does not detect a fall, the safety of the monitored person could be jeopardized. On the other hand, if the system reports an excessive number of false positives, the users could perceive it as ineffective and useless.

Reviewing the current scientific literature [4–8] it is possible to identify two classes of fall detection systems: *context-aware systems* and *wearable systems*. In Context-aware systems the sensors are deployed in the environment to detect falls, such as cameras, acoustic sensors, pressure sensors, infrared sensors, lasers and Radio Frequency Identification (RFID) tags. The advantage is that the targeted person does not need to wear or carry any special device. But they are suitable for those situations where the environment to monitor is well defined and circumscribed such as hospitals, nursing house and other indoor environment. When the activities performed by the user take place both indoors and outdoors and involve going from one place to another, the approach of this type of systems becomes unsuitable due to the restrictions imposed on the mobility of the

person. Wearable fall detectors have been developed in response to the problems affecting context-aware systems. They are specialized sensor-based devices that detect falls by analyzing users motion. Most of these devices are composed by a tri-axis accelerometer, and optionally additional sensors (such as gyroscopes and magnetometers), a dedicated elaboration units and wireless connectivity. The availability on recent smartphones of all the required elements makes them a suitable solution to realize a self-sufficient fall detection system [5]. Despite that, the use of a smartphone have some challenging issues that can affect the accuracy and the precision of the system. For instance, the *quality of the sensors* vary among different models of the same manufacturer and can not be sufficient to detect falls in some real situations; the *placement of the smartphone* (waist, pocket, hand, arm and so on) influences the measures collected from the sensors (indeed, moving the smartphone from the shirt pocket to the arm the accelerations can be very different); the *power consumption* sets some limits to the number of sensors, the data sampling rate and the complexity of the algorithm used to perform the task. As discussed in [5], to deal with these challenges, the majority of the systems running on smartphone use very simple fall detection algorithms that are based on fixed or adaptive thresholds applied on the data collected from the accelerometer and require a fixed placement of the phone. Only recently, thanks to the improved hardware resources and battery life of the new smartphones, more complex and accurate approaches based on machine learning techniques have been proposed [9–13]. Nevertheless, as discussed in [14], all the traditional approaches suffer from a high false positive rate when they are used in a real environment. The reason lies in the fact that these methods suppose to be able to collect enough data concerning real falls. This is a weak assumption because it is not easy to collect real falls, especially when the system has to monitor complex activities such as running, biking or skating. An alternative, proposed in [14, 15], is to consider fall detection as an anomaly detection problem, where the detector is trained only on the ADLs and considers all the anomalies as falls. In this paper we propose a smartphone-based fall detector using Self-Organizing Maps (SOMs) to deal with the constraints depending on the limited hardware resources of a smartphone. In order to proof the effectiveness and the efficiency of the proposed method, we have compared SOMs with other well-known classifiers at the state of the art: k -Nearest Neighbor and Support Vector Machine. In particular, the performance comparison considered both the accuracy and the resource requirements of each method and has been conducted directly on a smartphone.

2 The Proposed Method

Anomaly detection refers to the problem of finding patterns in data that do not conform to the expected behaviour [16]. The aim of the anomaly detection is to define a methodology through which establish a region representing the normal behaviour and classify any observation in the data which does not belong to this normal region as an anomaly. In the case of fall detection, the system

has to consider as normal the data representing the ADLs for a given activity and as anomalies (or falls) any unknown situation. Defining such a region that includes every possible ADL is not a trivial problem, indeed, the boundary between normal and abnormal instances is often not precise. In particular, abnormal observations very close to the region boundary can be considered as normal, and vice-versa. This problem is further enhanced by the noise: if the data representing ADLs are affected by noise they could be very similar to abnormal data and so not easy to distinguish. Furthermore, in many domains the ADLs can evolve and a current notion of normal instances might not be sufficiently representative in the future. For instance, in the case of sport activities, the experience of a user can increase, so he would be able to perform more difficult actions that were not considered in the model of the ADLs initially provided to the system.

It is important to point out that the system has to run in real-time using very limited hardware resources and battery in order to allow the normal operation of the smartphone. Therefore, on the one hand, the system has to be robust with respect to the noise and the small variations that can affect the normal instances. But, on the other hand, the model of ADLs used by the detector has to be as much compact as possible in order to require a low computational cost and to provide a decision with a low amount of memory. In addition, the hardware constraints could not allow to update the model directly on the smartphone, and, if the classifier requires an expensive training procedure, it could not be performed on the smartphone.

The previous considerations have a high impact on the features and the classifier that we can use to realize the system. As for the features, they have to be easy to compute in real-time and provide a compact representation. Similarly to Medrano [14] and Micucci [15], we used only the accelerations obtained from the three-axis accelerometer of the smartphone. Even though it could not be the best choice, we have decided to avoid the use of other sensors available on the smartphone, such as the gyroscope and the magnetometer, so as to reduce at minimum the energy required by the system. Indeed, the more sensors we use, the higher the battery consumption is. Moreover, differently from the other sensors, all the smartphones are equipped with good quality accelerometers because they are very cheap.

In more details, the three-axes accelerometer provides three raw measures (x, y and z axes) for each sample, as shown in Fig. 2. According to most of the current scientific literature, we have used a sampling rate of 50 Hz that is adequate to our purpose and compatible with the highest sampling rate allowed by the majority of the smartphone (85 Hz). So that, the signal produced by the accelerometer is arranged in non-overlapped time windows of 50 samples. Each window is characterized by a feature vector of 150 features (50 features for each axis) representing the raw measures collected from the sensor. As discussed in [15] raw features can provide a good accuracy without requiring further elaborations to extract the feature vector; thus, we reduce the time and the battery required by the feature extraction stage.

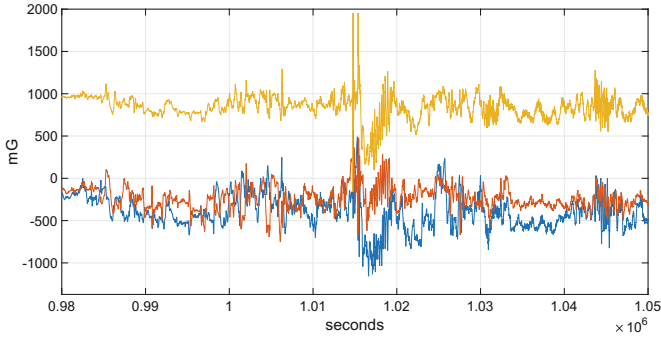


Fig. 2. Common fall pattern measured by the accelerometer of a smartphone. The accelerations on the three axes x, y and z are shown in red, blue and yellow, respectively. (Color figure online)

Concerning the selection of a suitable classifier, the anomaly detection is nicely addressed using *One-Class Classifiers* (OCCs) [16–19]. Indeed, the problem is to determine, in the data space, the boundary to distinguish between normal and abnormal patterns, i.e. ADLs and falls. In Micucci et al. [15], the authors compared the performance of a Support Vector Machine (SVM) and a k -NN, used both as one-class and binary classifiers. Differently from them, we propose to use Self-Organizing Maps since they are able to generate a very compact model and the decision stage has a very low computational cost.

SOMs are feed-forward neural networks composed of only two levels of neurons. Their main purpose is to produce a discrete representation of the input space, where the instances lies. This is performed by adopting the *competitive learning*, an unsupervised method that makes the network able to divide the input space in different partitions, each of them represented by one or more prototypes (i.e. output neurons). In a multi-class problem each partition is composed of instances belonging to the same class, but a class can be represented by one or more partitions. A new instance is classified on the basis of the class the closest prototype belongs to. Alternatively, the class of the new instance can be decided similarly to the k -NN, by considering the class of the k nearest prototypes. In the case of anomaly detection, all the partitions shaped by the SOM are related to the normal class; thus, when a new instance comes, the classifier checks whether it belongs or not to the normal region by considering the average distance between the new instance and the k nearest prototypes. If such a distance is higher than a given *rejection threshold*, the instance is considered abnormal (i.e. a fall).

Therefore, recalling the architecture shown in Fig. 1, we propose a fall detection system where the features are obtained by collecting raw data from the accelerometer, the detection is performed by a SOM that is trained out of the smartphone using only ADLs and the communication is carried out by using the basic function of a phone, such as calls and messages.

3 Experiments

As previously discussed, the limited hardware resources of a smartphone (or other wearable systems) introduces additional constraints; thus, the most accurate method could not be the most suitable to run in real-time on a smartphone. In our experiments, we have firstly compared the accuracy of SOM, k -NN and ν -SVM on a test environment in Matlab by considering two performance measures: the *sensitivity* (SE) (a.k.a the recall) which represents the true positive rate, i.e. the number of falls that are correctly recognized, and the *specificity* (SP) that is the true negative rate or, in our case, the number of properly classified ADLs. It is important to note that if we have a high sensitivity and a low specificity the system will generate many false alarms. Differently, if the specificity is high and the sensitivity is low the system will have many miss alarms. So that, it is clear that the aim is to tune the classifier in order to have the best trade-off between the two indices.

Successively, once the best parameters for each classifier have been assessed, we have verified the hardware and battery requirements on a Huawei P8 lite equipped with Android 6.

Table 1. Area under the curve (AUC) of k -NN considering different values of k .

	$k = 1$	$k = 3$	$k = 5$	$k = 7$
Medrano	0.976	0.976	0.975	0.974
Sisfall	0.706	0.705	0.704	0.705

Table 2. Area under the curve (AUC), sensitivity (SE) and specificity (SP) of ν -SVM considering different values of ν .

		$\nu = 0.1$	$\nu = 0.2$	$\nu = 0.3$	$\nu = 0.4$	$\nu = 0.5$	$\nu = 0.6$	$\nu = 0.7$	$\nu = 0.8$	$\nu = 0.9$	$\nu = 1.0$
Medrano	<i>AUC</i>	0.978	0.998	0.978	0.977	0.977	0.975	0.975	0.974	0.974	0.974
	<i>SE</i>	0.132	0.198	0.398	0.589	0.641	0.812	0.839	0.901	0.924	0.948
	<i>SP</i>	1.000	1.000	1.000	1.000	0.992	0.969	0.961	0.939	0.973	0.922
Sisfall	<i>AUC</i>	0.975	0.984	0.985	0.985	0.985	0.985	0.984	0.984	0.981	0.956
	<i>SE</i>	0.421	0.511	0.549	0.601	0.619	0.580	0.581	0.563	0.638	0.827
	<i>SP</i>	1.000	1.000	1.000	1.000	1.000	1.000	0.998	0.997	0.995	0.919

3.1 Datasets

The experiments have been conducted on two very recent publicly available datasets, Medrano et al. [14] and Sucerquia et al. [20],

The former has been specifically realized to investigate the use of anomaly detection techniques for fall detection in smartphone-based solution. It is

composed of ADLs and simulated falls recorded with the built-in three-axes accelerometer of a smartphone (a Samsung Galaxy Mini running Android). and considers eight different typologies of falls: forward falls, backward falls, left and right-lateral falls, syncope, falls sitting on empty chair, falls using compensation strategies to prevent the impact, and falls with contact to an obstacle before hitting the ground. Each fall has been repeated three times (for a total of 24 fall simulations per subject) and has been completed on a soft mattress in a laboratory environment. During the falls, participants wore a smartphone in both their two pockets. On the other hand, the ADLs have not been collected in a laboratory environment, but under real-life conditions. In particular, participants carried a smartphone in their pocket for at least one week to record everyday behaviour. On average, about 800 ADLs records have been collected from each subject. The dataset is divided in entries, each of them representing a time window of 6 seconds around a peak.

The SisFall dataset is composed of simulated activities of daily living and falls. This dataset has been generated in a laboratory environment with the collaboration of 38 volunteers divided in two groups: elderly people and young adults. Both the groups contains men and women. Differently from Medrano, SisFall has not been used by Micucci et al. [15] and is composed also of more interesting situations such as running and jogging people. The duration of each entry varies from 12 seconds, for simple ADLs, to 100 seconds, for more complex activities.

Both the datasets have been divided in ADLs and simulated falls because only the former have been used to train the systems, while the latter have been used to validate and test the classifiers. In more details, the *training set* consists of the 40% of the ADLs while the *validation set* is composed of the 30% of the ADLs and the 50% of the falls. The remaining part has been used as *test set*.

3.2 Results

The accuracy and the computational requirements of each classifier can vary on the base of how we set its parameters. We have assessed the best setup of each classifier by analyzing the *Area Under the Curve* (AUC) for different combination of their parameters. Additionally, only for the ν -SVM, we have considered the combined evolution of the specificity and sensitivity curves so as to avoid the overfitting.

Before discussing the validation of the ν -SVM, it is important to recall the meaning of the parameter ν in order to properly understand the results in Table 2. According to the technical report of Schölkopf [21], ν allows to control the model complexity; it defines how much the process of definition of the separating hyperplane will lie upon the training instances. In particular, low values of ν state that the hyperplane will be completely defined on the training set, while high values of ν assert that the hyperplane will take the training data just as an indication. Then in the former case, we risk to overfit the training set, while in the latter we risk to have an excessive generalization with a more complex model. In our results (see Table 2) it is possible to understand that even

if the AUC is high for most of the values of ν , the best value for ν on both the datasets is 1.0, because it provides the best trade-off between the miss and false alarms. Moreover, the high values of specificity and sensitivity guarantee that the system is not too specialized on the training set.

Then, Table 1 shows the validation results of k -NN. The accuracy of the k -NN is not improved when k grows; we have selected the lowest value of k so has the best trade-off between the accuracy and the computational cost of the prediction.

As for the SOMs, the aim is to have as less neurons as possible so as to reduce the complexity of the model used on the smartphone. Analysing the validation results in Table 3, it is possible to note that increasing the number of neurons or the value of k the accuracy of the classifier is not substantially affected. Thus, we have selected 9 neurons and $k = 3$.

Table 3. Area under the curve (AUC) for different values of the number of prototype vectors and the parameter k on both the datasets. The best values for each combinations of the parameters are highlighted in bold.

	k	<i>Neurons</i>							
		9	16	25	36	49	64	81	100
Medrano	$k = 1$	0.947	0.960	0.964	0.969	0.972	0.976	0.976	0.974
	$k = 3$	0.961	0.971	0.971	0.970	0.972	0.974	0.973	0.974
	$k = 5$	0.936	0.962	0.972	0.970	0.973	0.974	0.973	0.973
	$k = 7$	0.879	0.951	0.973	0.971	0.973	0.975	0.974	0.973
SisFall	$k = 1$	0.943	0.931	0.944	0.950	0.943	0.941	0.949	0.949
	$k = 3$	0.964	0.959	0.959	0.958	0.960	0.957	0.964	0.955
	$k = 5$	0.978	0.971	0.968	0.966	0.960	0.960	0.966	0.960
	$k = 7$	0.987	0.977	0.971	0.969	0.959	0.962	0.966	0.962

The results achieved by using the best setup for each dataset are shown in Table 4. As regards the Medrano Dataset, even if the ν -SVM has achieved the best accuracy, there is a small difference among the three classifiers. The reason lies in the fact that the ADLs composing the dataset are very simple so the falls are easy to distinguish. The results are completely different when more complex ADLs are considered, as in the case of the SisFall dataset. Indeed, in activities like jogging and running the accelerations measured by the system can be very similar to those representing falls. In these situations, that are very frequent in a real environment, the k -NN achieved a very low specificity and an acceptable sensitivity. This is a symptom of the fact the classifier often decides for a fall and confuses it with ADLs. The ν -SVM, instead, is more precise, but the recall is still low with a high value of false negatives. So it is clear that the best accuracy is achieved by the SOMs. Furthermore, if we join the results in Table 4 with those in Table 3, it is evident that the best accuracy is obtained with only 9 prototypes;

Table 4. Comparison of the performance on Medrano Dataset and SisFall.

	Medrano			SisFall		
	<i>k</i> -NN	SVM	SOM	<i>k</i> -NN	SVM	SOM
Sensitivity	0.952	0.952	0.944	0.842	0.824	0.912
Specificity	0.944	0.954	0.918	0.410	0.929	0.981

so we can suppose that SOMs will have very low resources requirement during the detection stage.

The comparison performed on the smartphone confirms the feasibility of the proposed method. In fact, the results in Table 5 show the efficiency of SOMs with respect to the other two classifiers. *K*-NN and ν -SVM need complex models to achieve a good accuracy, so they require also more time and battery to detect falls. Differently from them, SOMs are suitable to our purpose, since they are faster and cheaper due to a simpler model. They require just 0.04 seconds to classify 1 second of measures with a good battery consumption.

Table 5. Performance comparison in terms of hardware resources on a Huawei P8 lite.

	<i>k</i> -NN	SVM	SOM
Time	9 s	9 s	0.04 s
Memory	40 Mb	40 Mb	28 Mb
Battery (10 min)	$\geq 2\%$	$\geq 2\%$	1%

4 Conclusion

In this paper we have proposed and realized a smartphone-based fall detection system that can be comfortably placed in the trousers pocket. Due to the constraints imposed by the limited hardware resources and the battery of a smartphone we have adopted Self-Organizing Maps to distinguish normal activities and falls. They have the advantage to produce a very compact model and, consequently, to limit the computational cost required during the detection. The suitability of SOMs have been confirmed by the experimental evaluation, which demonstrated the effectiveness of the proposed method both in terms of accuracy and hardware requirements. To the best of our knowledge, the relevance of these results is enhanced by the fact that this is the first performance comparison of fall detection machine learning approaches performed on a smartphone.

References

1. Sadigh, S., Reimers, A., Andersson, R., Laflamme, L.: Falls and fall-related injuries among the elderly: a survey of residential-care facilities in a swedish municipality. *J. Commun. Health* **29**(2), 129–140 (2004)

2. Kalace, A., Fu, D., Yoshida, S.: Who Global Report on Falls Prevention in Older Age. World Health Organization Press, Ginevra (2008)
3. Igual, R., Medrano, C., Plaza, I.: Challenges, issues and trends in fall detection systems. *BioMed. Eng. Online* **12**(1), 1–24 (2013)
4. Delahoz, Y.S., Labrador, M.A.: Survey on fall detection and fall prevention using wearable and external sensors. *Sensors* **14**(10), 19806–19842 (2014)
5. Habib, M.A., Mohktar, M.S., Kamaruzzaman, S.B., Lim, K.S., Pin, T.M., Ibrahim, F.: Smartphone-based solutions for fall detection and prevention: challenges and open issues. *Sensors* **14**(4), 7181–7208 (2014)
6. Koshmak, G., Loutfi, A., Linden, M.: Challenges and issues in multisensor fusion approach for fall detection: review paper. *J. Sens.* (2016)
7. Brun, L., Saggese, A., Vento, M.: Dynamic scene understanding for behavior analysis based on string kernels. *IEEE Trans. Circ. Syst. Video Technol.* **24**(10), 1669–1681 (2014)
8. Leo, M., Medioni, G., Trivedi, M., Kanade, T., Farinella, G.M.: Computer vision for assistive technologies. *Comput. Vis. Image Underst.* **154**, 1–15 (2017)
9. Choi, Y., Ralhan, A.S., Ko, S.: A study on machine learning algorithms for fall detection and movement classification. In: 2011 International Conference on Information Science and Applications, pp. 1–8, April 2011
10. Abbate, S., Avvenuti, M., Bonatesta, F., Cola, G., Corsini, P., Vecchio, A.: A smartphone-based fall detection system. *Pervasive Mob. Comput.* **8**(6), 883–899 (2012). Special Issue on Pervasive Healthcare
11. Albert, M.V., Kording, K., Herrmann, M., Jayaraman, A.: Fall classification by machine learning using mobile phones. *PLoS ONE* **7**(5), 1–6 (2012)
12. Shi, Y., Shi, Y., Wang, X.: Fall detection on mobile phones using features from a five-phase model. In: 2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing, pp. 951–956, September 2012
13. Ali Fahmi, P.N., Viet, V., Deok-Jai, C.: Semi-supervised fall detection algorithm using fall indicators in smartphone. In: Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication, pp. 122:1–122:9 (2012)
14. Medrano, C., Igual, R., Plaza, I., Castro, M.: Detecting falls as novelties in acceleration patterns acquired with smartphones. *PLoS ONE* **9**(4), 1–9 (2014)
15. Micucci, D., Mobilio, M., Napoletano, P., Tisato, F.: Falls as anomalies? An experimental evaluation using smartphone accelerometer data. *J. Ambient Intell. Humanized Comput.* **8**(1), 87–99 (2017)
16. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. *ACM Comput. Surv.* **41**(3), 15:1–15:58 (2009)
17. Bartkowiak, A.M.: Anomaly, novelty, one-class classification: a comprehensive introduction. *Int. J. Comput. Inf. Syst. Ind. Manag. Appl.* **3**(1), 61–71 (2011)
18. Pimentel, M.A.F., Clifton, D.A., Clifton, L., Tarassenko, L.: Review: a review of novelty detection. *Sig. Process.* **99**, 215–249 (2014)
19. Khan, S.S., Madden, M.G.: One-class classification: taxonomy of study and review of techniques. *Knowl. Eng. Rev.* **29**(3), 345–374 (2014)
20. Sucerquia, A., Lpez, J.D., Vargas-Bonilla, J.F.: SisFall: a fall and movement dataset. *Sensors* **17**(1), 198 (2017)
21. Platt, J., Schlkopf, B., Shawe-Taylor, J., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Technical report, November 1999