Acoustic fall detection with Bootstrap Aggregating classifiers

Loukas Kominis
Department of ECE,
University of Patras,
Patras, Greece
loukaskom@gmail.com

Yanik Ngoko
Qarnot computing and University of Paris 13
Montrouge, France
yanik.ngoko@qarnot-computing.com

Christophe Cérin
University of Paris 13
Paris, France
christophe.cerin@lipn.univ-paris13.fr

***Abstract*—We consider the design of an acoustic fall detection system with the Qarnot model of computing. Qarnot introduced a computing model based on heaters that embed several processors and sensors. The network of Qarnot heaters in a home and building offers several advantages for the design of smart-environments systems when we consider the privacy, the real-life performance or the acceptance. The main goal of this work is to show the effectiveness of this platform for the resolution of smart-home problems and acoustic fall detection in particular. In addition to consider the Qarnot context, our work also innovates in applying bootstrap aggregation on acoustic fall detection. The results we obtained are very encouraging. We were able to build a classifier whose accuracy is higher than what we found in the state of the art on the problem. Finally, though the results were obtained in using Qarnot tools for the design of smart-environment systems, they can be generalized to other contexts.**

***Keywords*-acoustic fall detection; smart-environment; bootstrap aggregation**

I. Introduction

The demographic explosion of the eldery is offering and unprecendented opportunity for the development of a assisted living technologies. However, the question of the right architecture for such technologies is critical. By the past, several concerns were raised on existing solutions. The sensitive topics include the privacy, the acceptance or the real-life performance of such technologies. For the interested reader, we recommend the work of Igual et al. [1].

In this paper, we focus on the design of a fall detection system for elders. The conviction that supports our work is that with the Qarnot model of computing [1], we can design a fall detection system where several of the classical concerns regarding assisted living technologies are addressed. Qarnot introduced a utility computing model where the computing nodes are heaters that are embedded in homes, offices, buildings where they serve to heat. These special computing nodes (called Q.rads) embed several processors but also sensors related for instance to humidity, CO_2, noise, presence etc. The heaters are connected to the Internet and at the upper level, the network of geo-distributed Q.rads is exploited as an HPC cloud infrastructure (See Figure 1). At the scale of homes and buildings in which the Q.rads are deployed, this network serves as a smart-building platform with some services like the analysis of the air quality or the detection of the sounds of fire-alarms. For more details about the Qarnot model, we refer the interested reader to [2].

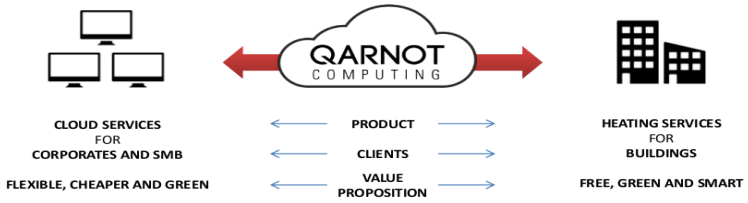
In the Qarnot vision, the process view of a smart-environment (home or building) consists of a forest of distributed worflows that are locally orchestrated in the Q.rads to which the environment belongs. Here, the data used by the processes are issued from the sensors embedded in Q.rads or remotely. A process could for instance analyze the level of CO_2 to make recommendations on the need to open windows. Another process could continuesly analyze the sound flow that are collected at the level of Q.rads in order to see whether or not there is a fall. Regarding data privacy, in this vision, we can guarantee that all data collected in a building will remain in the building. This changes from classical cloud-based smart-homes where the data are saved remotely in a datacenter. Regarding acceptance by elders, the Qarnot solution does not require any wearable device, nor a smartphone to collect user data. The user does not even need to care about the functioning of their heater. Finally for real-life performance, because of the sensing and computing power of the heaters [2], we have enough computing power to locally calibrate a machine learning model, for the specific environment in which we operate. In this paper, we will present the results we got on a distributed training system (machine learning system), we designed in order to detect falls with the Qarnot model of computing.

The automatic fall detection is an old smart-home topic. Several machine learning systems were already proposed for its resolution. A good survey on the diverse contributions can be found in [1], [3]. Despite the interest and quality of these solutions, our work has some specificities that were not addressed in past works. Indeed, past systems were often based on videos, accelerometers or smart-phones while we are mainly based on acoustic data. The choice to consider acoustic inputs comes from the fact that the Qarnot heaters do not embed cameras. In addition, we consider that the solution of an accelerometer or a smart-phone is not efficient in the viewpoint of the acceptance or usability by the elders.

[1] www.qarnot-computing.com

[2] A heater could embed 4 processors

(a) A Q.rad



(b) The Qarnot model for HPC-cloud

Figure 1: The Qarnot model

Acoustic fall detection systems were considered in the past. For instance, we can refer to the work of Xiaodan Zhuang [4] where GMM and SVM classifier were used and they managed to achieve a fall accuracy of 59% and 67% respectively with using only a far-field microphone. Other insteresting works on this field are the ones of Yun [5] and Muhammad [6] that use audio data to achieve classification of falls, each of them in a specific way.

Differently to these works, we innovate in two points. The first is that we designed a training system to run on the Qarnot cloud. This was achieved in using the Software Defined Kit of Qarnot to build a solution that is orchestrated by the Q.ware resoyrce manager. The second point is that differently from existing works, we used the bootstrap aggregation.

Bootstrap aggregating, also known as bagging, is a machine learning ensemble meta-algorithm which was designed to improve the stability and accuracy of machine learning algorithms by combining classifications of randomly generated training sets. Bagging can reduce variance and help to avoid the problem of overfitting. Usually it is applied to decision tree methods, but it can be applied to any method. The usage of this techniques leads to an important finding: we were able to build a meta-classifier, largely more accurate than all the other classifiers taken separately. We even obtained accuracy and True positive scores that we did not find in the literature.

The remainder of this paper is organized as follows. In Section II, we present the related works. Section III present Qarnot tools for smart environments. We detail here the architectural context in which we want to develop a fall detection system. Section IV discusses of the training of a fall detection system with three classifiers: SVM, KNN and decision tree. In Section V we reconsider the question with Bagging. In Section VI, we conclude.

II. Related Works

TODO

III. Qarnot tools for smart environments

To add info for QaIoT, Q.ware etc

IV. Building fall detection classifiers

A. Creating the Dataset

In order to be able to design an accurate system, it is really important to create a dataset of significant quality. As for a system like this, it is not easy to create a dataset from the very beginning, we used as main source an already existing Laboratory research dataset [7]. It contained video data which consisted of falls and other no-fall activities in different environments and especially Office, Home, Coffee room and Lecture room. The frame rate is 25 frames/s and the resolution 320x240 pixels. The dataset containes 191 videos of different time durations, from 6 sec to 1.20 minutes. The use of different environments makes the dataset more complete and representative in compare to others.

1) Audio Extraction: Using the Audacity software, we extracted the audio and isolated the fall and no-fall segments. The duration of all extracted segments is 5 seconds. Also a re-sampling was done to the audio files in order to have the same sampling rate (44100 Hz) in all of them and they were also converted to monophonic so that they can be used by our python code and libraries used.

2) Noise Mixing and Features extraction: Next step of the procedure was the audio files to get mixed with different types of noise (in all available combinations), so that our dataset is an accurate representation of real-life everyday sounds. At the end, the dataset consists of two major categories, Falls and No-falls, each of them containing a big variety of samples from all the different environments that were used to record the videos. So it is clear that our dataset in now complete. After finishing with the creation of the dataset, next step is to extract the features needed to run the algorithms. For this purpose, an open source library was used, the pyAudioAnalysis python library [8] which extracts specific features for each audio file. These features will be used to train the machine learning algorithms.

3) Subsets creation: The last step to the dataset creation, is to split it in Training and Testing subsets. The proportion remains constant for the entire duration of this experimental research project. Training subset includes 80% of the main dataset samples and Testing subset 20%. This split is re-

quired in order to produce learning curves for each classifier and also have cross-validation during training them.

B. SVM and kNN implementation

After our dataset was fully complete and ready to use in training, SVM and kNN classifiers were implemented on it. In part we present the work done in four experimentation Phases, where the algorithms run in different proportions of data and cover all possible scenarios. The proportion of falls and no falls will be increased and decreased respectively in each Phase in order to check how the algorithms respond on the amount of data given.

The Scikit-Learn python library is used for importing the code, needed for the classifiers, to our main python code. Scikit-Learn library was selected as it provides an easy, universal way for importing different classifiers.

In each classifier used, after training is finished, which is done using the Q.ware platform and actual Q.rads as computing nodes, the five most efficient classifiers are selected and a choice of the best one is made, among them, based on shortest training time. Because apart from the scores and high accuracy we try to accomplish, a time-efficient method is also important to be created. The most important part is the evaluation of the classifier, where the best classifier is used on both Testing and Training Dataset and see through the Learning Curves and the scores of the confusion matrix, which are presented, how the classifier responds as the amount of data is increasing.

1) Phases workflow and results: In total our experimental section consists of four Phases. In each of them, a different proportion of fall and no fall data was used in order to check the response of the algorithms. In Table 1, the proportions and total amount of data are presented for each Phase.

In Phase 1, only SVM classifier is trained and we chose to begin with this specific one as it is mostly used to other researches. The results though were definitely not what we expected as at first True-positive rate is really low and False-positive rate seemed unrealistic as a value. The duration of audio data in this phase (audio segment of fall or no fall mixed with noise) was 10 seconds and proportion of falls and no falls was 20-80%.

In Phase 2, it was decided to reduce duration of the audio data to 5 seconds in order to make it easier for the classifier to detect the fall part. It would also improve training time. Falls and no falls proportion was 40-60%. Also the KNN classifier was introduced and trained for the first try as an effort to compare results with SVM. Better results were produced to the SVM, as the True-positive rate increased by 20%. KNN had similar scores to the SVM as shown in the table, but in general these results are not yet acceptable in order for our system to be independent.

So moving on to Phase 3, SVM and KNN classifiers were trained again, duration of audio data is still 5 seconds and proportion of data is changed to 50-50% and we hoped to improve more the scores of the classifiers. But instead of our expectations for continuous improvement, the SVM results were dissapointing. Not only scores did not icrease but they deteriorated. KNN on the other hand, had better results. This difference in the results can be explained by the different definition of the two classifiers. As KNN detects similarities in a close nearby area it is profited by adding more fall data. In this phase, it was when we first thought of having a different approach on the way of classification, as the system seems to have a kind of sensitivity to the amount of data given each time, that it is not easy to regulate.

As it was needed to see if our thoughts had logical basis, we moved on to Phase 4. Here, the same settings are used, except the proportion of data used, which changed to 60-40% of falls and no falls. And the results, certified our theory of sensitivity. The SVM classifier, had a significant improvement and for sure these were the best scores achieved with this specific classifier. KNN still produced good results but not with such a huge improvement to its scores.

By checking the results shown in Table 2, which presents the scores of each classifier for all four experimentation Phases, the above described are more clear. Three are the categories presented. False-positive-rate (F.P.R), True-positive-rate (T.P.R) and Accuracy.

Table I: *Proportions of data, for each Phase*

PHASE	FALL SAMPLES	FALL(%)	NO FALL SAMPLES	NO FALL(%)
1	1600	20	4800	80
2	2400	40	4000	60
3	3200	50	3200	50
4	4000	60	2400	40

C. Figures

Here we present the learning curves for each of the four experimental Phases previously described in order to visualize our results.

D. Conclusion

After testing SVM and KNN classifiers in different amount of data and comparing their scores, it is clear that this kind of system behaves differently depending on the amount of data given each time. The SVM results leave no doubt to this statement. KNN has better scores, but as explained previously it is due to the way it works and classifies falls. Our main accomplishment is the one presented on next section.

V. FALL DETECTION WITH BAGGING

The results of the previous experimentations, made us considering a different approach for this type of system. The data sensitivity issue had to be solved. So in this Section, a

Table II: *Classifiers scores, in each Phase*

PHASE	CLASSIFIER	F.P.R	T.P.R	ACCURACY
PH.1	SVM	0.00	0.18	0.79
PH.2	SVM	0.12	0.45	0.75
	KNN	0.10	0.42	0.73
PH.3	SVM	0.40	0.20	0.40
	KNN	0.17	0.78	0.80
PH.4	SVM	0.20	0.92	0.90
	KNN	0.21	0.79	0.77

new classifier is introduced in order to help us achieve better scores and easier classification of falls and solve the data sensitivity issue. Bootstrap Aggregating, commonly known as Bagging, is an ensemble meta-estimator. It is important to give an explanation of Bagging in order to make clear for the reader, how we think it can help us. Bagging classifier establishes base estimators to randomly selected subsets of our main dataset. Then it aggregates their estimation to produce one final estimation for the entire dataset. This is its main advantage against other methods.

The randomly selected subsets assure that we can have an estimation for all possible combinations of data instead of choosing a fixed proportion. Furthermore, the idea of base estimators, aggregated to a final estimation, gives the opportunity to combine all estimations, in an effort to consider parts of the dataset were classification is done properly and other where it is not. So this section is also split in Phases. But with some differencies from the previous one. Here proportion of data is kept stable for every Phase and is 50-50% of falls and no falls, so that we can have enough samples of both categories. The duration of audio data is still 5 seconds. What is getting different at each Phase are the estimators, in an effort to see if we can achieve better results than the classifiers used in previous section.

A. Decision Tree base estimators

The first approach of classification with Bagging method was done using it's default parameters which include Decision Tree base estimators. The results produced were more than satisfying. They were much better than our initial expectations. Except from the fact that False-positive rate is under 20% combined this time with a True-positive rate which significantly increased to 90%, also Accuracy reached 90% and the Learning curves also represent a system with stable improvement as more data were given to it.

B. kNN base estimators

As the results of the previous Phase were really promising and much better than any previous try, it was decided make some simulations with other base estimators. In this phase, KNN base estimators were selected in an try to see if better results than the KNN classifier alone, can be produced. And as the results presented in Table 3, prove that this is a goal that can be achieved.

C. SVM base estimators

After running Bagging Classifier with Decision Tree and kNN estimators it was also really important to try it with SVM estimators in a try to have also in this category better results in prove our theory of data sensitivity which we hope to solve via the Bootstrap Aggregating method.

Table III: *Bagging Classifier scores with different estimators*

ESTIMATOR	F.P.R	T.P.R	ACCURACY
DECISION TREE	0.16	0.90	0.87
KNN	0.22	0.88	0.84
SVM	0.XX	0.XX	0.XX

D. Figures

After presenting the scores for each classifier trained, what is also an important tool for machine learning are Learning curves. This section is dedicated to the Learning curves of the classifiers trained during the different phases of this project. It is important to see who the system can benefit from adding more data.

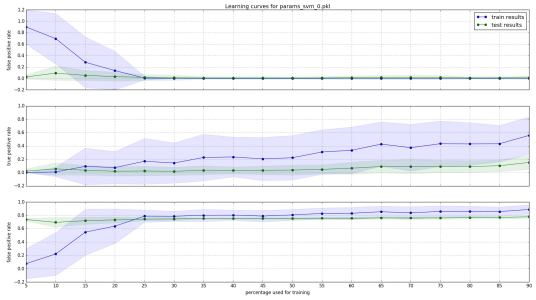


Figure 2: Learning curves, Phase 1, SVM

VI. CONCLUSION

REFERENCES

[1] R. Igual, C. Medrano, and I. Plaza, "Challenges, issues and trends in fall detection systems," *BioMedical Engineering OnLine*, vol. 12, no. 1, p. 66, 2013.

[2] Y. Ngoko, "Heating as a cloud-service, A position paper (industrial presentation)," in *Euro-Par 2016: Parallel Processing - 22nd International Conference on Parallel and Distributed Computing, Grenoble, France, August 24-26, 2016, Proceedings*, 2016, pp. 389–401.

[3] M. Muhammad, S. Ling, and S. Luke, "A survey on fall detection: Principles and approaches," *Neurocomput.*, vol. 100, pp. 144–152, jan 2013.

[4] X. Zhuang, J. Huang, G. Potamianos, and M. Hasegawa-Johnson, "Acoustic fall detection using gaussian mixture models and gmm supervectors," in *2009 IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2009.

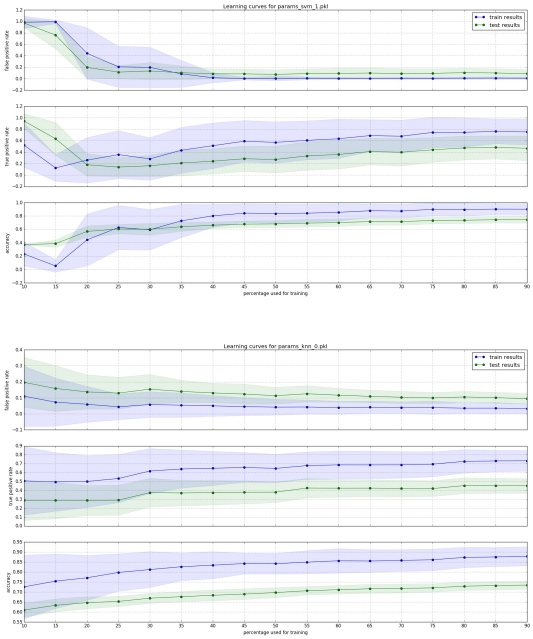


Figure 3: Learning curves, Phase 2, SVM & and KNN

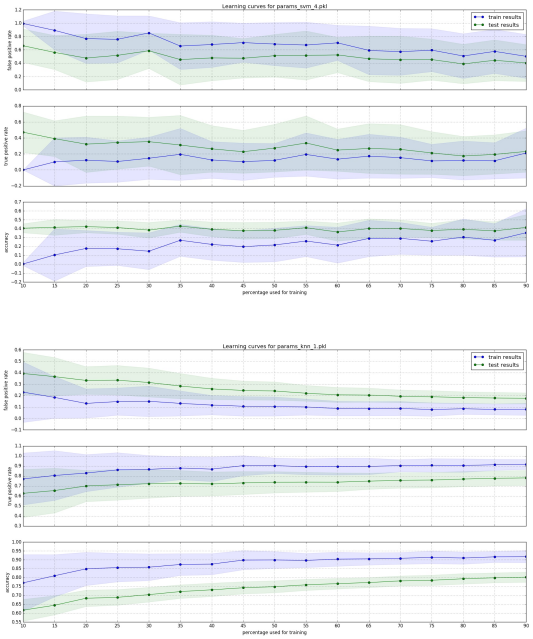


Figure 4: Learning curves, Phase 3, SVM and KNN

[5] Y. Li, K. C. Ho, and M. Popescu, "A microphone array system for automatic fall detection," *IEEE Transactions on biomedical engineering*, vol. 59, no. 2, may 2012.

[6] M. S. Khan, M. Yu, P. Feng, L. Wang, and J. Chambers, "An unsupervised acoustic fall detection system using source separation for sound interference suppression," *Signal Processing*, pp. 199–210, 2015.

[7] A. Trapet. (2013) Fall detection dataset. Le2i-Laboratoire Electronique, Informatique et Image. [Online]. Available: http://le2i.cnrs.fr/Fall-detection-Dataset?lang=en

[8] T. Giannakopoulos, "pyaudioanalysis: An open-source python library for audio signal analysis," *PloS one*, vol. 10, no. 12, 2015.

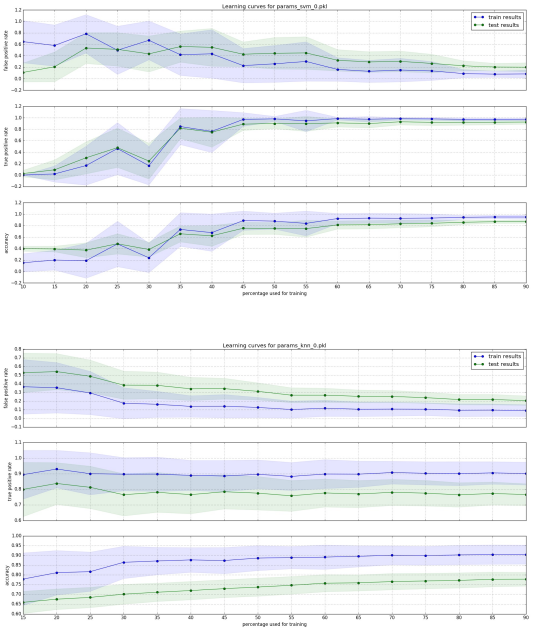


Figure 5: Learning curves, Phase 4, SVM and KNN

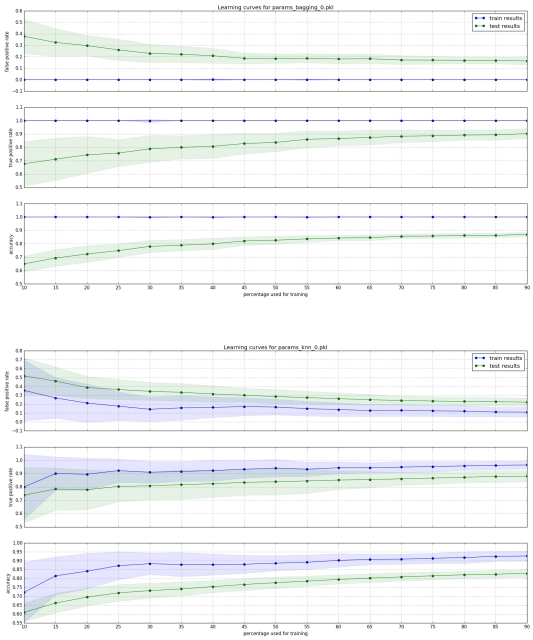


Figure 6: Learning curves, Bagging classifier, Decision Tree, KNN and SVM estimators