Irregular Movements from Senior Citizens

1. What is the problem?

Currently, there exists a system that records data such as time, acceleration and rotation of an object that is in the possession of seniors most of the time.

The data transmitted could be segmented as follows: significant movements and insignificant movements (the former refers to movements like: standing up, falling, walking, whereas the latter refers to very tiny 'noise-like' data[1]).

The first problem (henceforth referred to as Problem I) is that there currently exists no model that can differentiate between such movements.

Once a an accurate enough model is created that could differentiate between such movements, it is to be deployed in conjunction with the device (either on the device or otherwise) such that there is no transfer of personal data over the internet.

After this point, the 'data' transferred for analysis would only be data that is labelled by the model as 'significant movements'[2].

The first part of this data could be used to alert caregivers in case the senior is in an accident (such as falling), thereby acting as a PERS.

The second part of this data would be non-emergency movement data (labelled as such; this includes activities like walking).

The second problem (henceforth referred to as Problem II) is that, once the second part of the data is obtained, there would be no model that could detect any anomalies in said data within a given period of time (i.e. a couple of hours or days).

An example use case would be: if the senior walks significantly slower, or with irregularities that aren't major enough to be deemed emergencies, but enough to raise an alarm. For example: if the senior walks extremely slow (or not at all) due to fever.

2. How can you use data to solve the problem?

The API created by Rohan is capable of providing movement data from the device[3].

My understanding is that this data will need to be fairly personal and thus adapt to each individual. So there would be a 'training' time where the user uses the device and feeds information to the model before it can start making any perditions.

---

[1] Mr Jimenez mentioned 'breathing', but this type of data should be more defined.

[2] See Question / Concern 1 for the specification of this data

[3] See Question / Concern 2 for concerns about the data

Is there a way to prevent this?

3. What data do you need?

Types of motion that I would expect and the data needed:

Type of motion	Data Category	Data needed
	Day to day movements	
Standing up from a seated position	Significant	Accelerometer data, time
Bending down to pick up something	Significant	Gyro data
Walking	Significant	Accelerometer data
Turning	Noise	Gyro data
Turning *(device turning inside the person's pocket)*	Noise	Gyro data
	Emergency movements	
Falling	Significant	Accelerometer, Gyro data, time (for examining whether a repeated action instigates a fall?)
	Sleep related movements	
Turning in bed	Noise	Time, Gyro
Changing from a lying position to a seated position	Significant (can signal waking up)	Time, Gyro data, Accelerometer data

As a result of this, for **Problem I** we'd need all the data. To reduce dimensions, it might be worth considering the parameters as scalars, and not vectors. I.e. a large movement in Acceleration_X is equivalent to a large movement in Acceleration_Y (this is a consideration because the orientation of the device itself might change while in the senior's pocket). Based on the table above, it seems to me that magnitude is enough. Would have to experiment with this, as it may result in a loss of data.

For **Problem II**, I'd have to do a bit more exploration to understand the relationships between the 7 variables (3 angular directions, 3 accelerations, 1 time). My worry is that reducing dimensions on this type of analysis might result in underfitting. For more information on the analytical method of choice, see Section 8.

In general, it's worth recording the time of the day, so it might be worth adding that data input to all of the 'movements'.

4. Where will you get the data from (all sources), and how will you get it?

All of the data will come from the personal device.

5. Is the data representative of the problem that you are trying to solve?

Yes.

6. What extra work is required to manipulate the data?

For Problem II, integrating the acceleration values to find things like speed, distance might be worth doing (although note that this process might be fairly inaccurate).

Acceleration and Gyro values may have to be combined to give a magnitude, though this may be inappropriate for Problem II, which requires more data.

7. In what way can you visualise the data to get a solution?

For **Problem I**, assuming that we've reduced the dimensions of the data, you should examine scatter plots of data against each other, as well as a 3D plot to check the clustering process.

For **Problem II**, plotting the time series data is a good start.

8. Analytical method

For **Problem I,** steps are:

1) Clustering the data, to find two (or three) groups which are: significant movements (emergency), significant movements (normal), noise movements

    1) Based on the data that we have, it seems that we can get 2 clusters nicely based on the following reasons: noise will be captured well because Acceleration and Gyro data will have very small magnitudes. Significant movements may be difficult to split into "emergency" and "normal", unless the magnitudes for emergency are really high. Should try with k = 2 and 3 clusters to see outcomes
    2) A second algorithm to consider would be agglomerative clustering as can create nice diagrams that visually show how the data 'splits'. This might be helpful to determine the "emergency' vs. "normal" data types
    3) Both can be done using Python's sklearn library

4) For evaluating the data, we can artificially create a labelled dataset since we control the data going into the system

2) Classifying new data[4], see **Section 10 for deployment.**

For **Problem II**, a more detailed analysis is given below. This assumes that there exists a method capable of recognising significant movements vs. noise movements, and recording said 'significant movements'.

8.1. Reiteration of the type of data that you will get

The data will be [Acc_X, Acc_Y, Acc_Z, Omega_X, Omega_Y, Omega_Z, time]. All the data would be 'significant' movement of type 'normal', not emergency.

The data would be sent at different times (this can be configured using the arduino / API). Let's say that data is being sent every **10 minutes.** So you would have 6 * 24 data points per day.

Another 'dimension' to the data becomes the day on which the data is sent.

This way, you can have a time series data for each day, for all of the motions captured by the arduino.

8.2. Algorithms that might come in handy

1) Isolation Forest:

Recursively partitions data, such that the partition with less data points is the next area to be partitioned.

2) STL Decomposition:

This is not a machine learning algorithm, but rather an algorithm that tries to show underlying trends, such as seasonal trends. It's a preprocessing step when examining the actual data.

3) DBSCAN:

Not necessarily for outlier detection, though it can capture outliers. This might be a useful clustering algorithm for trying to understand the time series data that you get from the Arduino.

4) LSTM Neural Networks:

Unlike the isolation forest which tries to locate the anomaly, LTSM neural networks try to 'recreate' the trend using a simplified or compressed dataset, and then find anomalies based on how much they deviate from said model.

5) ARIMA:

[4] See Question / Concern 4

Typically, neural networks work for when you have large datasets, and thus may be inappropriate for the purposes of this project. This is where ARIMA can come in handy, as it relies on a moving average. It does not require large datasets.

6) Prophet:

Works best for datasets that:

- contain extended time periods of historical observations
- have multiple strong seasonalities
- have irregular events
- have large outliers
- have non-linear growth trends that approach a limit

8.3. Comparison of algorithms for anomaly detection

Algorithm	Pros	Cons	Suitability
Isolation forest	• Works with highly dimensional data • Robust and easy to optimise	• Visualising results is complicated	Useful for finding anomalies in time series data, so for example for a single day
LSTM	• Can learn hidden interdependencies given large amounts of data	• Data requirement • Setting hyperparameters might be difficult	Useful for finding anomalies in time series data for a long period of time
ARIMA	• Does not require large datasets	• Does not work well with non-stationary time series (i.e. moving average) (however there is a variation of this known as seasonal ARIMA)	Useful for finding anomalies in time series data, so for example for a single day
Prophet	• Quick and flexible experience	• Does not partition data into hours, only has functionality for days / month /years • Based on an additive model	Useful for finding anomalies in time series data for a long period of time

8.4. Relevant Python libraries

Prophet: The syntax is based on the sklearn library[5]. Has options for 'daily' seasonality (but not for hourly or parts of the data)

sklearn: has many algorithms and is useful for evaluating models

Tensorflow / Keras: has many deep learning algorithms, for example LSTM.

[5] https://www.youtube.com/watch?v=95-HMzxsghY

PyTorch: a more low level AI library

9. Does your model answer the problem?

Need to build the model first before we get to this stage.

10. How will you deploy the model?

In general, to deploy your model you will need a “Deployment” Account.

Based on my understanding, you can export a model to an offline device (https://www.ibm.com/support/knowledgecenter/SSHGWL_1.2.3/local/remotedeploy.html).

There is also Amazon Snowball[6] which can be used to deploy offline models.

One might be able to use Arduino ML libraries to have everything on the Arduino itself[7] [8]. See Question / Concern 5 for more.

However, there are alternative ways for deploying models:

- https://towardsdatascience.com/how-to-easily-deploy-machine-learning-models-using-flask-b95af8fe34d4
- https://mlinproduction.com/deploying-machine-learning-models/
- https://tvm.apache.org/docs/tutorials/frontend/deploy_model_on_rasp.html

11. What feedback can you get from stakeholders regarding the model?

Need to test the model with out of sample data.

Questions / Concerns:

1. Walking data is fairly useful, though it is distinct to falling data or to ‘sudden’ movement data, as such you would ideally want 3 segments:
   - emergency type data, such as falling (alert being sent with high urgency)
   - behavioural data, such as walking trends over a period of time (alert being sent when walking trends fall much lower than expected)

---

[6] https://aws.amazon.com/snowball/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc

[7] https://blog.arduino.cc/2019/10/15/get-started-with-machine-learning-on-arduino/

[8] https://eloquentarduino.github.io/2019/11/you-can-run-machine-learning-on-arduino/

- noise data: not entirely sure on what this would encompass, but it could include things such as rotation during bedtime, movements while sat on a couch, breathing, etc

2. I'd like to clarify what type of data we have available.
   - Volume: how much data is there?
   - Veracity: will the data be accurate? My presumption is that this would be Rohan's own personal data, his own movements. Would this be appropriate for a senior?

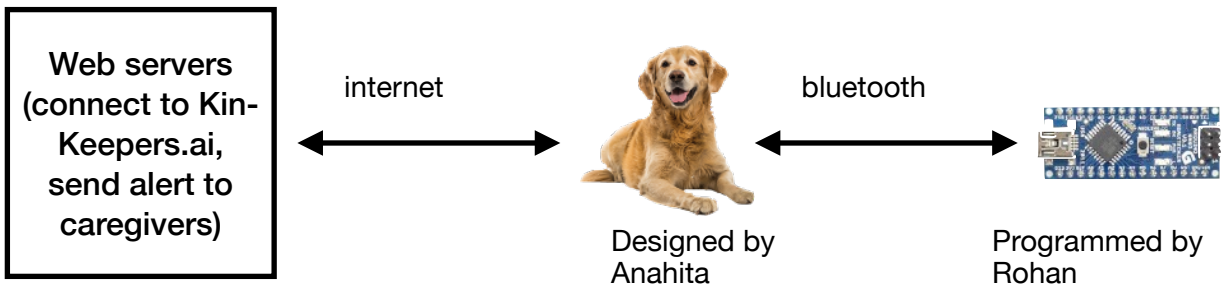
   Unsure about the device itself. Based on the description, I understood that it is not 'fixed' in position, but can be put inside a pocket. As such, it's orientation when with the person might change considerably, so it adds unnecessary dimensions to the data. It might be worth combining the 'vectors' into scalars to reduce this problem. Will this impact data analysis though?

3. Would it be worth also be worth segmenting the data into 'sleep' data as well? As in, what time the senior sleeps, what time they get up from bed, what time they go to the toilet?

4. I'm not sure how I would classify the data based on the clusters. In general, it doesn't seem like a good idea to use supervised learning after unsupervised learning, as that might add a lot of noise to the data. It may be that you'd have to update your model every time your data points increase by a significant percentage (i.e. say 10%) to ensure that your cluster model is 'up to date' all the time. Any thoughts on this?

   Either way, possible algorithms include using SVM, KNN, Random Forest.

5. I need a re-brief of how the system is gonna work. Where are the bluetooth signals being sent to? Is this a good schematic of what's happening?



So the steps of this AI project COULD be:

I. Using CSV file generated by Rohan, cluster data into "significant motion" and "noise"
II. Verify this using purposely labelled data (compare with ground truth)
III. Create a classifier based on this that is deployed on the Arduino itself (using TinyAI modules), enabling only datasets that are 'significant' to be sent to to the PET
IV. Create 'significant' movement time series data (measured at regular intervals and for different days or times of the day, i.e. time series data for the morning)
V. Create a model that is able to perform anomaly detection on this type of data, deployed on the PET (for example, on a Raspberry Pi)
VI. The next steps would be to enable the PET to communicate with web servers if the an anomaly is detected (via a connection from the Raspberry Pi to the servers)

There could be individual steps that separate the significant data into ‘emergency’ and ‘normal’ data, where the former would signal a different alert.

Where does the Fluid detection team come in?

My contribution:

I’ve confirmed with Abbas and our meetings are @20:30 CET on Mondays and Fridays, so I can attend the daily standup meetings.

As for my other priorities (Website etc…), I can manage my day such that I prioritise this project over them.

Links:

- https://towardsdatascience.com/what-does-it-mean-to-deploy-a-machine-learning-model-dddb983ac416
- https://www.saedsayad.com/model_deployment.htm
- https://en.wikipedia.org/wiki/Isolation_forest
- https://towardsdatascience.com/anomaly-detection-with-time-series-forecasting-c34c6d04b24a
- https://www.unicon.net/insights/blogs/building-and-deploying-machine-learning-ml-model
- https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a
- https://developer.squareup.com/blog/so-you-have-some-clusters-now-what/
- https://www.newgenapps.com/blog/random-forest-analysis-in-ml-and-when-to-use-it/
- https://www.reddit.com/r/statistics/comments/2082vx/what_to_do_after_clustering/
- https://www.quora.com/Is-supervised-learning-commonly-carried-out-after-clustering
- https://stackoverflow.com/questions/22300830/can-k-means-clustering-do-classification
- https://www.kdnuggets.com/2017/02/yhat-support-vector-machine.html
- https://medium.com/datadriveninvestor/choosing-the-best-algorithm-for-your-classification-model-7c632c78f38f
- https://towardsdatascience.com/there-are-two-very-different-ways-to-deploy-ml-models-heres-both-ce2e97c7b9b1
- https://blog.statsbot.co/time-series-anomaly-detection-algorithms-1cef5519aef2
- https://www.kaggle.com/vinayjaju/anomaly-detection-using-facebook-s-prophet
- https://medium.com/seismic-data-science/anomaly-detection-using-prophet-a5dcea2c5473
- https://towardsdatascience.com/anomaly-detection-time-series-4c661f6f165f
- https://towardsdatascience.com/5-ways-to-detect-outliers-that-every-data-scientist-should-know-python-code-70a54335a623
- https://towardsdatascience.com/anomaly-detection-def662294a4e
- https://towardsdatascience.com/anomaly-detection-with-lstm-in-keras-8d8d7e50ab1b
- https://keras.io/examples/timeseries/timeseries_anomaly_detection/

- https://machinelearningmastery.com/lstm-autoencoders/
- https://towardsdatascience.com/a-note-about-finding-anomalies-f9cedee38f0b
- https://towardsdatascience.com/anomaly-detection-with-time-series-forecasting-c34c6d04b24a
- https://www.curiousily.com/posts/anomaly-detection-in-time-series-with-lstms-using-keras-in-python/
- https://github.com/sahandha/eif
- https://facebook.github.io/prophet/
- https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality/
- https://towardsdatascience.com/lstm-autoencoder-for-anomaly-detection-e1f4f2ee7ccf
- http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- https://towardsdatascience.com/prophet-vs-deepar-forecasting-food-demand-2fdebfb8d282
- https://www.empiwifo.uni-freiburg.de/teaching/summer-term-13/Material%20Time%20Series%20Analysis/classicalts