Fall Detection using Wearable Sensor Data
Programming for Big Data 2
CMPT 733

Gustavo Felhberg
Jorge Marcano
Muhammad R Myhaimin

16 / 04 / 2018

1 Introduction

Falls represent the second leading cause of accidental or unintentional injury deaths worldwide (behind only of road traffic injuries). Almost 50% of older adults who fall experience a minor injury and 25% of them experience a more serious injury such as a fracture. Each year, 2.8 million older people are treated in emergency units for injuries caused by falls [2].

The objective of this project is to analyze data regarding simulations of falls and non falls and train machine learning models to detect falls using data collected from wearable sensors.

2 Problem Definition

This project uses the results of a research at Simon Fraser University in 2015 which compares threshold-based algorithms with machine learning algorithms. The data were created by ten young participants wearing tri-axial sensors in their head, sternum, waist, both thighs and both ankles and simulating the common causes of falls observed in older adults, along with near-falls and activities of daily living.[3].

This project intends to use the data from the sensors, analyze theirs relationships, train different machine learning models and simulate the detection of falls using real time data.

3 Data Analysis

3.1 Features Engineering

Firstly we consolidated the data into a single Pandas DataFrame. The original data has a timestamp for each reading and 63 columns representing each one of the combinations of sensors, axes (X, Y, and Z) and measurements (acceleration, angular velocity, and magnetic fields). We included some implicit features extracted from the data files and folder names: type, subtype, number of the trial, number of the subject and the target value (0 as non-fall and 1 as fall). This feature engineering step was the base to initialize the Exploratory Data Analysis and the preprocessing steps to train the models. Additionally we included the vector sum XYZ for each one of the readings and measures. To compute this resultant measure using the formula $\sqrt{X^2 + Y^2 + Z^2}$.

3.2 Exploratory Data Analysis

The data was already clean and had no duplicated rows and no null values. Thus, the primary focus of the EDA was to reduce the number of features. We assumed that, not all the features are relevant to detect falls.

In order to reduce the dimensionality, we decided to use Pearson Correlation among the features and the target. Then, stipulating a threshold of 0.3, we selected six features. These features were from acceleration data of X axis of the sensors located in the head, sternum, waist, right and left thighs and right ankle. Next, we repeated the same analysis for the resultant calculations, but we reduced the threshold to 0.1 because no correlation using this data was higher than this value. Using this limit, we identified three most correlated features, which are magnetic fields readings of right and left ankles and left thigh.

For the above analysis, we used the whole data for each trial, mostly comprising a fifteen-seconds interval. However, as we can check the charts with those trials, the beginning and the end of the simulations are mostly with less abrupt movements, suggesting that they do not have a significant contribution to the identification of a fall. This way, we reduced the data of each trial using a four-second window around the absolute peak, i.e., the value of the reading with the highest magnitude. Repeating

the correlation analysis, we could identify an increase in the number of features with correlation above the thresholds from six to nine using the raw axes readings and from three to eleven using the resultant calculations.

3.3 Visualization

Our project has three major components of visualization. First, we developed the plots in Python and Jupyter Notebooks covering the EDA and the model training/test section, presenting the results we achieved. The second component was designed using a Jupyter dashboard extension [1] and includes two dynamic visualizations of the results of the trails, allowing the user to select a combination of types, subtypes, sensors, subjects and trial numbers and plot the graphics with the sensor readings. Lastly, the third component is a Flask application using DC.js, D3.js, and crossfilter.js that presents a dynamic dashboard to allow the user to compare the results of the tri-axial readings with the resultant calculation of the axes.

4 Fall Detection Models

In this section we'll explain the different preprocessing steps used to work with the data as well as the Machine Learning Models used.

4.1 Data preprocessing

The data provided to us for this project was not labeled. It was separated in different types of trials (ADLs, Near-Falls and Falls) but the rows of data themselves were not classified as fall or non-fall. In a fall trial the subject starts by walking, then trips and falls to the ground, then stays on the ground for a while, so only a fraction of each fall trial would have to actually be labeled as a fall. In our preprocessing we labeled the data, which was challenging due to conflicting definitions of "fall" used by different researchers. In this project we tried different labeling methods and compare the results of each one.

We presented 4 different labeling methods, but each one of them uses similar preprocessing steps asides from the labeling. Firstly we picked one or multiple body parts to use in the models (Head, Sternum,Waist, Thighs and Ankles). Then we picked one or multiple metrics to use in the models (Acceleration, Velocity and Magnetic Field). In the next step we grouped the data in 0.5 or 1.0 second windows depending on the labeling method and calculate the mean and variance of each column for each of these groups. This is to make the data more manageable and make the labeling and training process faster. Additionally, this step is skipped in the Convolutional Neural Network method since we want the CNNs to see the full data and try to find patterns in it. Afterwards we labeled each data row as "fall" (1) or "non-fall" (0) based on one of the four labeling methods explained in the following section. Finally we separated training and testing data. We used the last five subjects data as training data and first five subjects data as testing data, like the SFU researcher.

4.2 Labeling methods

In this section we'll explain all 4 of the labeling methods used. Each method requires a body part and metric to be picked to create the "window" of fall around. For the sake of simplicity we'll use the waist acceleration for every explanation, since those are the most relevant features and what was used in most of the testing we showed later.

4.2.1 Overlapping window method

For this method we don't use the resultant measures since this only uses the original data set provided without extra columns. At first we grouped the data in 0.5 seconds windows. The goal is to obtain the absolute peak acceleration of each axis (X,Y,Z) and create a window that starts from the minimum of these values minus 2 seconds to the maximum of these values plus 2 seconds. This creates a dynamic window size, where the minimum absolute peak is 4 seconds into the trial and the maximum is at 9 seconds into the trial, the window we consider a fall would be from 2 seconds to 11 seconds, but in can also be either bigger or smaller.

4.2.2 Resultant acceleration peak window method

First we grouped the data in 0.5 seconds small windows to make it more manageable. After this we find the Resultant acceleration peak of each trial and create a fall window of 4 seconds around it (2 seconds before and 2 seconds after the peak).

4.2.3 Resultant difference window method

This method requires a lot more computational power so we grouped the data in 1.0 second small windows to make it faster. The next step is to calculate the difference between the resultant acceleration of each two consecutive rows of data and find the peak of these differences. When this is found, we creates a window of 4 seconds around it (2 seconds before and 2 seconds after this peak).

4.2.4 Convolutional Neural Network method

This last method does not group the data in small windows before labeling to allow CNN access to full data to learn from any specific patterns in it. This uses the same labeling method as the Resultant acceleration peak window and then proceeds to reshape the data for it to be used in a 1D CNN, where we treat a data row as a "very thin" 1D image. We tried to use 2D CNNs as well, but the results weren't very promising so we omitted that in the final product.

4.3 Machine Learning Models

The first 3 labeling methods used SVMs to detect falls. We picked SVM because past research in the area of fall detection with sensor data had shown best performance with this model. For training our one vs one SVM model the best set of hyperparameters found via parameter tuning were: Radial basis function as kernel, kernel coef 0.01, degree 3 for the RBF

For the last method, the 1D Convolutional Neural Network used the following parameters : 50 filters, filter size 3, ReLU activation in the Conv layer, 1 Maxpooling, 0.01 LR in the Dense layer with a sigmoid activation function. The goal it to enable each filter to learn different patterns in the data. The filter size indicates the size of these patterns and the maxpooling ensures dimensionality reduction when picking the "patches" of data, best resemble these patterns.

5 Real Time Fall Detection

5.1 General structure

Our goal here was to use a pretrained model to give real time prediction using REST api (https). To achieve that we saved our trained data as a pickle file, read from it using a Python Flask server and give real time prediction.

5.2 AWS architecture

We deployed our server into a AWS ec2 instance to enable RESTful API. For ease of deployment of the trained model we uploaded our pickle file in S3 and made our flask server read from it before predicting. We can relay the prediction to our user as a http response and store it in a persistent storage such as AWS dynamodb.

5.3 Getting prediction from sensor

In a real life scenario the sensor readings can be transmitted to a local server such as a small raspberry pi device or a cell phone using bluetooth. The local server or the cell phone, can get fall predictions on the received data by calling the prediction server hosted in AWS using the REST api. If a fall is detected, the system can alert someone using email or text and they can take the necessary action to help the person who has fallen. Additionally they can report whether a fall actually happened or not which can be used to validate some of the user data and we can use this further train the model.

6 Results

There was a class imbalance(1:10 for fall vs non-falls) in our data. To address that we tried weighted prediction but it didn't give the best realistic result, so we omitted that in the report. We'll limit ourselves to showing only the best combinations of results obtained for each method and we'll show accuracy, recall, precision and specificity for each one of them. Additionally, all these results use the full set of data rows to train and test, there was no downsampling done to obtain them regardless of the imbalances in the data, this is due to the fact that it is not realistic to pick random rows from a trial that clearly follows a sequence of events. The only case in which the data was down-sampled was for the CNN results, since apparently these don't perform very well when using all ADL,Near-Fall and Fall trials, so we limited the test of this model to only detecting the fall in a Fall trials.

Overlapping Window (Waist Acceleration only)	Accuracy	0.782
	Recall	0.428
	Precision	0.854
	Specificity	0.962
Resultant Acc Peak Window (Waist Acceleration only)	Accuracy	0.934
	Recall	0.394
	Precision	0.743
	Specificity	0.986
Resultant diff Window (Head Acceleration only)	Accuracy	0.945
	Recall	0.477
	Precision	0.796
	Specificity	0.988
Convolutional Neural Nets (All body parts Acceleration)	Accuracy	0.787
	Recall	0.725
	Precision	0.571
	Specificity	0.809

7 Conclusions

From the results obtained it's easy to see that most of the methods produce a fairly good accuracy, precision and specificity, but low recall. This is most likely caused by the big imbalance in the data. The only one that doesn't follow this trend is the CNN method and this is because it only uses Fall trials to detect falls as a method to downsample the data. Notice that even if the recall in the other methods is low, the Precision hovers around the 75%-85% mark, which means that, even if our models don't do a great job at identifying the falls, when they do identify one, it's most likely true. Additionally, we need to consider that every trial has a fraction of it's rows that are considered part of the fall (since every row is a reading with 0.5-1.0 seconds in between each one), this means that in reality, as long as we're able to identify one of those rows correctly, an alert would be sent indicating that the person fell, and action can be taken, there is no need to identify the whole fall.

That said, the imbalance in the data was a big problem for this research and we have some suggestions to address this in the next section. Lastly, the models all seem to perform well when predicting non-falls, but, this is most likely due to the big number of rows that are classified as non-falls.

From this research we've learned how to create a real big data pipeline from analysis to preprocessing, model tuning, prediction and testing and deployment of a real application to use our product. Additionally, the setbacks that we had during the project, specifically related to the data imbalance and the conflicting definitions of fall have taught us that, at least in this area of research, there is a lot more work to be done outside of big data and machine learning to correctly solve this problem.

8 Suggestions and Future Work

Before moving onto what should or could be done as future work with this research, we wanted to go over two suggestions that could be applied to make this research easier to compare with past results as well as more understandable. The first is to settle for one definition of fall, different research shows that falls can be windows of different sizes (varying from 3 seconds to 7 seconds) and these windows are created around peaks of different metrics like acceleration on a certain axis to the resultant acceleration, it would be ideal if there was a clear definition of this so every research uses the same. Additionally, for future research, more fall data is required. The data is very imbalanced, and there is only so much that can be done with downsampling or upsampling, specially because up/downsampling in this specific case doesn't represent real scenarios, each trial is a time series of events, and picking random rows takes this away.

As for future work, it would be interesting to do some more studies on the use of convolutional neural networks to solve this problems. At the time, there is no proper way to tune the hyperparameters of a CNN, so it's entirely possible that with a better set of hyperparameters or maybe using more or different sensors, more interesting patterns can be found in the data, making it easier to train the model and detect falls with more accuracy. The use of more varied sensors in different body parts could also be incorporated in the research. For example, sensors on the wrists of the subject could be useful, as well as the use of a pressure metric. Lastly, while working on this project we found some research done on this subject that uses image and video data of a person when they fall (or don't) to predict the fall, combining the use of sensor data with image and video data could make the detection and prediction of falls easier and more accurate.

References

[1] Jupyter dashboards layout extension. `https://github.com/jupyter/dashboards`, 2018.

[2] Centers for Disease Control, National Center for Injury Prevention Prevention, and Control. Web–based injury statistics query and reporting system (wisqars). `https://www.cdc.gov/injury/wisqars/`, 2018.

[3] Edward J. Park Greg Mori Stephen N. Robinovitch Omar Aziz, Magnus Musngi. A comparison of accuracy of fall detection algorithms (threshold-based vs. machine learning) using waist-mounted tri-axial accelerometer signals from a comprehensive set of falls and non-fall trials. *Medical & Biological Engineering & Computing*, 2017.