Extrapolation of weight from smart scale data

Pablo Caballero\textsuperscript{a,}\textsuperscript{*}, Juan A. Ortega\textsuperscript{a}, Luis Gonzalez-Abril\textsuperscript{b}

\textsuperscript{a}Universidad de Sevilla, ETS Ingenieria Informatica, Avda. Reina Mercedes s/n, E-41012 Sevilla, Spain
\textsuperscript{b}Universidad de Sevilla, Facultad de Ciencias Economicas, Avda. Ramn y Cajal, 1, E-41018 Sevilla, Spain

Abstract

In the area of human digital twins, the designed model should be as close as possible to the reality. The weight variable is one of the interested parameters of these mathematical models, as well as its interaction with other vital signals. The aim of this paper is including the information gathered from weight sensors of a person in its digital twin. To do this it is described an algorithm that filters and forecasts the human weight to use it in indexes such as BMI. It has been applied to a real sample and the results obtained are good.

\textsuperscript{*} Corresponding author. Tel.: +34-605-458-552
\textit{E-mail address:} pabcabper@alum.us.es

1. Introduction

The creation of a human digital twin requires different vital signals, calculated indexes and some metrics. For complex models like these it is necessary to divide in simpler models which in turn can be related each other.

The monitorization [1] of the humans and the early disease prevention is the final aim of the human digital twins, trying to align the target of state machine of the model with the patterns of the human-machine [2]. The latest Internet of Things (IoT) developments [3] focused on the fitness provides to the market a sort of commercial ubiquitous devices [7], in particular smart bands and smart watches. Some examples such as the real time heart rate, the walking or resting heart rate, and the sleeping hours are stored in the smart watches like Apple HealthKit and cloud databases. This raw information can be queried by mobile applications as well as complex backend systems.

On the other hand, the increment of overweight and obesity [4, 5] has also opened the market to new smart devices, for instance smart scales. The weight or the body fat can be measured and recorded whenever the users want.
In a model, the body mass index [6] is widely used to measure the status in fitness models. Some actions can be suggested when the user is underweight, overweight or more.

Due to the information is gathered from several devices and every device can have one or more sensors; therefore, the collection of samples have different time frequencies. To relate the signals and therefore the collections of samples need to exist in a specific moment, so the model is split in steps.

The body mass [13] or commonly called weight has a tiny recurrency compared to heart rate so it is needed a forecast mechanism to match the values in one time step. This paper will be focused on the extrapolation of the weight received from a smart scale. After this standardization process the resulted samples can be used as dataset of machine learning models or cloud computing calculations.

The paper is organized as follows. Section 2 exposes the properties of the weight to get a picture of the problem. Secondly, in Section 3 the time intervals is defined. Then, in Section 4 the extrapolation algorithm is described. The acknowledgements are in Section 6. Finally the conclusions and further works are detailed in Section 5.

2. Properties of weight

Based on the anonymous survey “habits of weigh yourself” made for this paper on internet over 58 people on March 4, 2021; the following properties can be inferred from its results and they justify discretizing the signal of the weight and taking the values based on the closer samples.

2.1. Recurrence

The high variability of the hour of weigh yourself shown in Figure 1 does not allow to predict the moment when the sample of the weight is gathered.

Table 1. Common weigh yourself hour.

<table>
<thead>
<tr>
<th>Hour</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>8</td>
<td>13.8%</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>8.6%</td>
</tr>
<tr>
<td>8</td>
<td>19</td>
<td>32.8%</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>15.5%</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>3.4%</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>6.9%</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1.7%</td>
</tr>
<tr>
<td>19</td>
<td>2</td>
<td>3.4%</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>8.6%</td>
</tr>
<tr>
<td>21</td>
<td>3</td>
<td>5.5%</td>
</tr>
</tbody>
</table>

The recurrency per week is also highly variable, shown in Figure 2. Consequently, it is not possible to predict how often the samples are gathered.

Table 2. Number of weighing yourself per week.

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Weekly) 1</td>
<td>14</td>
<td>17.7%</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>7.6%</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>15.2%</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5.1%</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>19.0%</td>
</tr>
<tr>
<td>(Daily) 7</td>
<td>28</td>
<td>35.4%</td>
</tr>
</tbody>
</table>

Fig. 1. Common weigh yourself hour from Table 1.

Fig. 2. Number of weighing yourself per week from Table 2.
2.2. Variability

The variability throughout a day the weight could increase or decrease significantly based on the quantity of liquid or food, basal metabolic rate [8], activity [9], hormone levels [10], the frequency of bowel movement [11]... So, it is not possible forecasting more accurate for an specific day.

In the opposite case, it might be possible measure to the weight few times in a short period of time and it should represent the same moment. In some cases, mobile applications [16] linked to the smart scale ask for a confirmation when the difference is too big. For the current proposal, the value of the weight in a step will be the average of the values inside the same time step.

3. Time interval

As it was exposed before, during the creation of a model the samples of the vital signals are in different frequencies, so it is necessary to standardize in time intervals. The complex structure [14] has to be converted into a time interval defined by \([start, end]\).

The parameter \(\text{stepSize}\) is the number of minutes of one step. For this algorithm will be 1 min. So a time interval can be defined as the period of time \([t_i, t_{i+1}]\) where \(t_i\) is the beginning of the step and \(t_{i+1}\).

\[
t_{i+1} = t_i + \text{stepSize}
\]  \hspace{1cm} (1)

In the forecasting algorithm some time interval methods will be used, they are \text{intersect} and \text{expandMinutes}.

3.1. Method intersect

An interval \(x\) intersects with an interval \(y\) when they share part of their time. It is shown in Figure 3.

\[
\text{intersect}(x, y) = y.end \geq x.start \land y.start \leq x.end
\]  \hspace{1cm} (2)

| t | s------E--|
|  x | #-------|
| y1| s-------e--- y1 intersects with the right boundary of x |
| y2| s------e------ y2 intersects with the left boundary of x |
| y3| s--------e----- x contains y3 |
| y4| s----------e--- y4 contains x |

Fig. 3. Possible cases of intersect is true.

3.2. Method expandMinutes

Date types are specific of the programming languages, so it is recommended the usage of a common representation. Henceforth the dates will be represented in ticks\(^{1}\) [15] format. Due to some systems like Apple devices include the second precision and it is not needed, the smallest precision is minute and the seconds component can be ruled out.

The constant \(\text{ticksPerMinute}\) is the number of ticks in one minute, that is 600 000 000, and it is necessary to convert time difference in minutes.

An interval \(i\) is expanded in \(m\) minutes when the field \(\text{start}\) in moved \(m\) minutes before it and the field \(\text{end}\) is moved \(m\) minutes after it. This method is used when there are not any samples for a time interval and it is need to expand the search radius.

\[
\text{expand}(interval, m) = [i.start - (\text{ticksPerMinute} \times m), i.end + (\text{ticksPerMinute} \times m)]
\]  \hspace{1cm} (3)

\(^{1}\) One tick is 100 ns (nanoseconds) and date is the number of ticks since 12:00, January 1, year 1 in Gregorian calendar.
4. Procedure

Below, the involved methods and the resampling process are described. The extrapolation itself is part of the method searchSamples(samples, interval) and the general method to standardize all the samples is the method getAllStandardSamples(samples1, samples2, ... samplesn).

4.1. Filtering of bounding values

When information is extracted from a smart device, a potential problem could happen, users can add manually values and consequently that action could be a source of possible errors.

Based on a dataset [12] of 40 400 samples of height of 18 years old people, the bounding values depending on the gender have been obtained using BMI the formula can follow:

\[
bmi = \text{weight}[kg] ÷ \text{height}[m]^2
\]

\[
weight_i = bmi × \text{height}_i^2
\]

\[
weight = \text{average}(\text{weight}_i)
\]

Table 3. Extreme values for weights depending on the gender.

<table>
<thead>
<tr>
<th>Gender</th>
<th>Threshold</th>
<th>Weight(kg) Extreme Underweight BMI 16</th>
<th>Weight(kg) Moderate Underweight BMI 16</th>
<th>Weight(kg) Obesity III BMI 40</th>
<th>Weight(kg) Obesity IV BMI 50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Female</td>
<td>Minimum</td>
<td>39.50</td>
<td>41.97</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Female</td>
<td>Maximum</td>
<td>-</td>
<td>-</td>
<td>98.74</td>
<td>123.43</td>
</tr>
<tr>
<td>Male</td>
<td>Minimum</td>
<td>45.38</td>
<td>48.22</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Male</td>
<td>Maximum</td>
<td>-</td>
<td>-</td>
<td>113.46</td>
<td>141.83</td>
</tr>
</tbody>
</table>

The BMI values for extreme underweight and obesity IV have been taken as limits maximum and minimum respectively due to there are no more extreme categories. The categories moderate underweight and obesity III could be taken in a more restrictive model because more values are out of range. All the samples whose weight is out of range should be discarded. This filter can be applied in early steps in our process, so it is not needed to spend space and processing time.

4.2. Identify time forecast boundaries

All the samples extracted from the devices usually are within a time interval, so they have at least a tuple of three values: start date (samplei.start), end date (samplei.end) and the signal value (samplei.value).

In addition to the samples of weight there are also other samples like heart rate or height, and that boundaries belong to the model itself, so the entire set of samples is needed. One modification could be including the current time in this limit calculation so it will try to extrapolate until then.

The boundaries will be a tuple of the minimum date of all the samples of all the signals for the current model and the maximum date of all the samples.

\[
\text{getForecastBoundaries}(samples_1, ..., samples_n) = [\min(x.\text{start} | x \in \text{samples}_i), \max(x.\text{end} | x \in \text{samples}_i)]
\]
4.3. Definition of all intervals

It is needed to create a collection of time intervals, one per step. The method `getAllIntervalsPerStep` is defined in Algorithm 1.

The result is a collection of steps. Every step is followed by the immediate next step, and all the steps have the same size.

**Algorithm 1:** `getAllIntervalsPerStep(samples1, samples2, ... samplesn)`

- **samples:** Input. Set of samples.
- **Result:** The set of all intervals.

1. \((global\text{Start}, global\text{End}) \leftarrow\) getForecastBoundaries(\{x.start| x \in samples\_1\}, \{x.start| x \in samples\_2\}, ..., \{x.start| x \in samples\_n\})
2. \(interval\_start \leftarrow global\text{Start}\)
3. \(interval\_end \leftarrow global\text{Start} + (ticks\text{PerMinute} \times step\text{Size})\)
4. intervals \(\leftarrow \emptyset\)
5. while \(interval\_end < global\text{End}\) do
6.  \(intervals \leftarrow intervals \cup \{interval\}\)
7.  \(interval\_start \leftarrow interval\_end\)
8.  \(interval\_end \leftarrow interval\_end + (ticks\text{PerMinute} \times step\text{Size})\)
9. end
10. return intervals

4.4. Look for a possible value

Once the raw values are filtered, Algorithm 2 describes the process where the weight is assigned to every interval as well as the rest of the sets of samples.

It is possible to find more than one samples, however those values should be closer to each other so the average method is used.

**Algorithm 2:** `getAllStandardSamples(samples1, samples2, ... samplesn)`

- **samples:** Input. Set of samples. Every sample has a start time, an end time and a weight value.
- **Result:** The set of all weight samples.

1. \(intervals \leftarrow getAllIntervalsPerStep(samples)\)
2. \(result \leftarrow \emptyset\)
3. foreach \(interval\ in intervals\) do
4.  \(samplesInInterval \leftarrow searchSamples(samples, interval)\)
5.  if \(|samplesInInterval| > 0\) then
6.   \(currentSample\_start \leftarrow interval\_start\)
7.   \(currentSample\_end \leftarrow interval\_end\)
8.   \(currentSample\_value \leftarrow average(x.value| x \in samplesInInterval)\)
9.   \(result \leftarrow result \cup \{currentSample\}\)
10. end
11. end
12. return result

Below in Algorithm 3 the method `searchSamples` is defined. The result is a set of samples that are close to that interval.

The first part of the algorithm covers the corner cases of having a reduced number of samples. This case could happen when it is the first time that the standardization process is executed.

The second part tries to filter all the values that intersect with the current interval, so we could say that they belong to this interval.
In the third part it tries to increase the window time to catch a value. The increment is defined by the parameter \texttt{searchStep}, it is the number of minutes to be incremented (by default is 60). The limit of the search is defined by the parameter \texttt{searchRate} constant (by default its value is 4, so the maximum search will be a quarter of the total time). These parameter are visually explained in Figure 4.

![Fig. 4. Visual explanation of the parameters.](image)

A drawback of stopping the search using \texttt{maxSearchStep} (Algorithm 3, line 20) is when there are a small set of samples close in time then the difference between \texttt{lastInstance} and \texttt{firstInstance} is tiny, and it is also reduced by \texttt{searchRate} so the window time maybe is not enough. A possible solution less restrictive could be set a minimum of \texttt{maxSearchStep} or the least restrictive solution is removing that condition so it always will find at least one sample.

```plaintext
Algorithm 3: searchSamples(samples, interval)

\texttt{samples}: Input. Set of samples. Every sample has a start time, an end time and a weight value.
\texttt{interval}: Input. Current interval to evaluate.
\textbf{Result}: The set of samples for a given interval.

1 if $\mid$\texttt{samples}$\mid$ = 0 then
2 \hspace{1em} return $\emptyset$
3 else if $\mid$\texttt{samples}$\mid$ $\leq$ 1 then
4 \hspace{1em} return \texttt{samples}
5 end

6 $\texttt{samplesInCurrentWindowTime} \leftarrow \{x \mid x \in \texttt{samples} \land \text{intersect}($\texttt{interval}, $x$)$\}$
7 if $\mid$\texttt{samplesInCurrentWindowTime}$\mid$ $>$ 0 then
8 \hspace{1em} return \texttt{samplesInCurrentWindowTime}
9 end

10 \hspace{1em} \texttt{firstInstance} $\leftarrow$ \text{min}($\{$\texttt{x}.\text{start} $|$ $x \in \texttt{samples}$$\}$

11 \hspace{1em} \texttt{lastInstance} $\leftarrow$ \text{max}($\{$\texttt{x}.\text{end} $|$ $x \in \texttt{samples}$$\}$

12 \hspace{1em} $\texttt{maxSearchStep} \leftarrow ((\texttt{lastInstance} - \texttt{firstInstance}) \div \text{ticksPerMinute}) \div \texttt{searchRate}$

13 \hspace{1em} $\texttt{intervalSize} \leftarrow \texttt{stepSize}$

14 \hspace{1em} do
15 \hspace{2em} $\texttt{intervalSize} \leftarrow \texttt{intervalSize} + (2 \times \texttt{searchStep})$
16 \hspace{2em} $\texttt{expandedInterval} \leftarrow \text{expandMinutes}($\texttt{expandedInterval}, $\texttt{searchStep}$)$
17 \hspace{2em} $\texttt{samplesInExpandedInterval} \leftarrow \{x \mid x \in \texttt{samples} \land \text{intersect}($\texttt{expandedInterval}, $\texttt{castSampleToInterval}(x))$$\}$
18 \hspace{2em} while $\texttt{samplesInExpandedInterval} = \emptyset \land \texttt{intervalSize} \leq \texttt{maxSearchStep}$
19
20 return $\texttt{samplesInExpandedInterval}$
```
4.5. Algorithm example

An example of this algorithm is in Figure 5. The initial data is 5 intervals, from $I_0$ to $I_4$ defined in Table 4. The algorithm parameters are stepSize as 15 (min), searchRate as 4 and searchStep as 60 (min).

Table 4. Extrapolation example data.

<table>
<thead>
<tr>
<th>Interval</th>
<th>Start</th>
<th>Start (Date)</th>
<th>End</th>
<th>End (Date)</th>
<th>Weight (kg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_0$</td>
<td>6374678400 10^8</td>
<td>01/21/2021 00:00</td>
<td>6374678406 10^8</td>
<td>01/21/2021 00:01</td>
<td>99.0</td>
</tr>
<tr>
<td>$I_1$</td>
<td>6374687040 10^8</td>
<td>01/22/2021 00:00</td>
<td>6374687046 10^8</td>
<td>01/22/2021 00:01</td>
<td>98.0</td>
</tr>
<tr>
<td>$I_2$</td>
<td>6374695680 10^8</td>
<td>01/23/2021 00:00</td>
<td>6374695686 10^8</td>
<td>01/23/2021 00:01</td>
<td>98.0</td>
</tr>
<tr>
<td>$I_3$</td>
<td>6374704320 10^8</td>
<td>01/24/2021 00:00</td>
<td>6374704326 10^8</td>
<td>01/24/2021 00:01</td>
<td>99.0</td>
</tr>
<tr>
<td>$I_4$</td>
<td>6374712960 10^8</td>
<td>01/25/2021 00:00</td>
<td>6374712966 10^8</td>
<td>01/25/2021 00:01</td>
<td>96.0</td>
</tr>
</tbody>
</table>

After running the standardization code it will generate 384 intervals of 15 minutes per interval.

In $t_0$, $t_1$, $t_2$ and $t_3$ the values are original values. Between $t_0$ and $t_1$ there are 2 intervals whose values are 98.5 kg, this situation happens due to searching a value, the window time intersects with $I_0$ and $I_1$. Between $t_2$ and $t_3$ the same thing happens as well.

5. Conclusion and further work

The difficulty to forecast the weight inside a day due to its recurrency and variability has been exposed.

Some threshold values were proposed depending on the gender to filter possible wrong values. Cases of extreme underweight or extreme obesity would be non-allowed values.

The proposed algorithm is ready to be used and it extrapolates the weight value to moments close to the original instant. Every step will have a discrete value of the weight so it can be used to create a digital twin model in conjunction with other signals. If the value is out the search range then that time interval could be discarded if the weight is needed.

In a general perspective the forecasted values are valid, nevertheless for an specific moment the value will depend of how far in time is from a real value.

A future modification could be a second pass day by day and converting the discrete values in a curve. In every day, the variations at lunch or dinner could be also considered nevertheless a research on nutritional habits would be necessary.
6. Acknowledgements

This research has been partially supported by the Evolving towards Digital Twins in Healthcare (EDITH) Research Project (PGC2018-102145-B-C21, C22 (AEI/FEDER, UE)), funded by the Spanish Ministry of Science, Innovation and Universities.

References


