Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 192 (2021) 2761–2768

Procedia Computer Science

www.elsevier.com/locate/procedia

25th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Extrapolation of weight from smart scale data

Pablo Caballero[a,*], Juan A. Ortega[a], Luis Gonzlez-Abril[b]

[a]Universidad de Sevilla, ETS Ingeniera Informtica, Avda. Reina Mercedes s/n, E-41012 Sevilla, Spain
[b]Universidad de Sevilla, Facultad de Ciencias Econmicas, 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.

Keywords: WEIGHT; EXTRAPOLATION; ALGORITHM; DIGITAL-TWIN; SIGNAL PROCESSING

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.

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

1877-0509
10.1016/j.procs.2021.09.046

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.

Hour	Count	Percentage
None	8	13.8 %
7	5	8.6 %
8	19	32.8 %
9	9	15.5 %
10	2	3.4 %
11	4	6.9 %
12	1	1.7 %
19	2	3.4 %
20	5	8.6 %
21	3	5.5 %

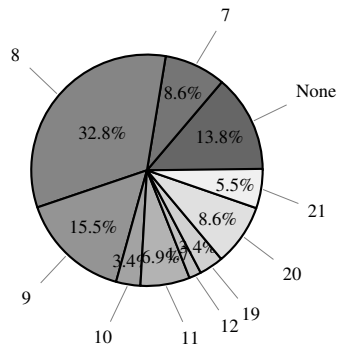

Fig. 1. Common weigh yourself hour from Table 1.

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.

Frequency	Count	Percentage
(Weekly) 1	14	17.7 %
2	6	7.6 %
3	12	15.2 %
4	4	5.1 %
5	15	19.0 %
(Daily) 7	28	35.4%

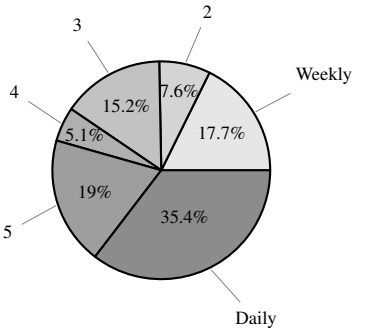

Fig. 2. Number of weighing yourself per week from Table 2.

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 `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}.

$$t_{i+1} = t_i + stepSize \tag{1}$$

In the forecasting algorithm some time interval methods will be used, they are `intersect` and `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.

$$intersect(x, y) = y.end \geq x.start \wedge y.start \leq x.end \tag{2}$$

```
t  ---------------->
x  -----S-----E-----
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 `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 *start* in moved *m* minutes before it and the field *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.

$$expand(interval, m) = [i.start - (ticksPerMinute \times m), i.end + (ticksPerMinute \times m)] \tag{3}$$

[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(`$samples_1$`, `$samples_2$`, ... `$samples_n$`)`.

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:

$$bmi = weight[\text{kg}] \div height[\text{m}]^2 \tag{4}$$

$$weight_i = bmi \times height_i^2 \tag{5}$$

$$weight = average(weight_i) \tag{6}$$

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

Gender	Threshold	Weight(kg) Extreme Underweight BMI 16	Weight(kg) Moderate Underweight BMI 16	Weight(kg) Obesity III BMI 40	Weight(kg) Obesity IV BMI 50
Female	Minimum	39.50	41.97	-	-
Female	Maximum	-	-	98.74	123.43
Male	Minimum	45.38	48.22	-	-
Male	Maximum	-	-	113.46	141.83

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 ($sample_i.start$), end date ($sample_i.end$) and the signal value ($sample_i.value$).

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.

$$getForecastBoundaries(samples_1, ..., samples_n) = [min(x.start|x \in samples_i), max(x.end|x \in samples_i)] \tag{7}$$

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

```
1  (globalStart, globalEnd) ←
     getForecastBoundaries({x.start|x ∈ samples_1}, {x.start|x ∈ samples_2}, ..., {x.start|x ∈ samples_n})
2  interval.start ← globalStart
3  interval.end ← globalStart + (ticksPerMinute × stepSize)
4  intevals ← ∅
5  while interval.end < globalEnd do
6  |   intevals ← intevals ∪ {interval}
7  |   interval.start ← interval.end
8  |   interval.end ← interval.end + (ticksPerMinute × stepSize)
9  end
10 return intevals
```

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 ← getAllIntervalsPerStep(samples)
2  result ← ∅
3  foreach interval in intervals do
4  |   samplesInInterval ← searchSamples(samples, interval)
5  |   if |samplesInInterval| > 0 then
6  |   |   currentSample.start ← interval.start
7  |   |   currentSample.end ← interval.end
8  |   |   currentSample.value ← average(x.value|x ∈ samplesInInterval)
9  |   |   result ← result ∪ {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 `searchStep`, it is the number of minutes to be incremented (by default is 60). The limit of the search is defined by the parameter `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.

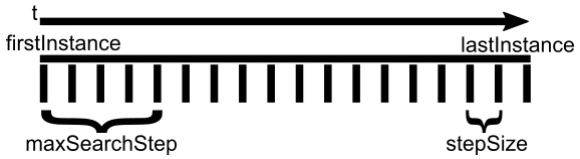

Fig. 4. Visual explanation of the parameters.

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

Algorithm 3: searchSamples(samples, interval)

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

```
1  if |samples| = 0 then
2  |   return ∅
3  else if |samples| ≤ 1 then
4  |   return samples
5  end
6
7  samplesInCurrentWindowTime ← {x|x ∈ samples ∧ intersect(inteval, x)}
8  if |samplesInCurrentWindowTime| > 0 then
9  |   return samplesInCurrentWindowTime
10 end
11
12 firstInstance ← min({x.start|x ∈ samples})
13 lastInstance ← max({x.end|x ∈ samples})
14 maxSearchStep ← ((lastInstance − firstInstance) ÷ ticksPerMinute) ÷ searchRate
15 intervalSize ← stepSize
16 do
17 |   intervalSize ← intervalSize + (2 × searchStep)
18 |   expandedInterval ← expandMinutes(expandedInterval, searchStep)
19 |   samplesInExpandedInterval ← {x|x ∈
   |     samples ∧ intersect(expandedInterval, castSampleToInterval(x))}
20 while samplesInExpandedInterval = ∅ ∧ intervalSize ≤ maxSearchStep
21
22 return 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.

Interval	Start	Start (Date)	End	End (Date)	Weight (kg)
I_0	6 374 678 400 10^8	01/21/2021 00:00	6 374 678 406 10^8	01/21/2021 00:01	99.0
I_1	6 374 687 040 10^8	01/22/2021 00:00	6 374 687 046 10^8	01/22/2021 00:01	98.0
I_2	6 374 695 680 10^8	01/23/2021 00:00	6 374 695 686 10^8	01/23/2021 00:01	98.0
I_3	6 374 704 320 10^8	01/24/2021 00:00	6 374 704 326 10^8	01/24/2021 00:01	99.0
I_4	6 374 712 960 10^8	01/25/2021 00:00	6 374 712 966 10^8	01/25/2021 00:01	96.0

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.

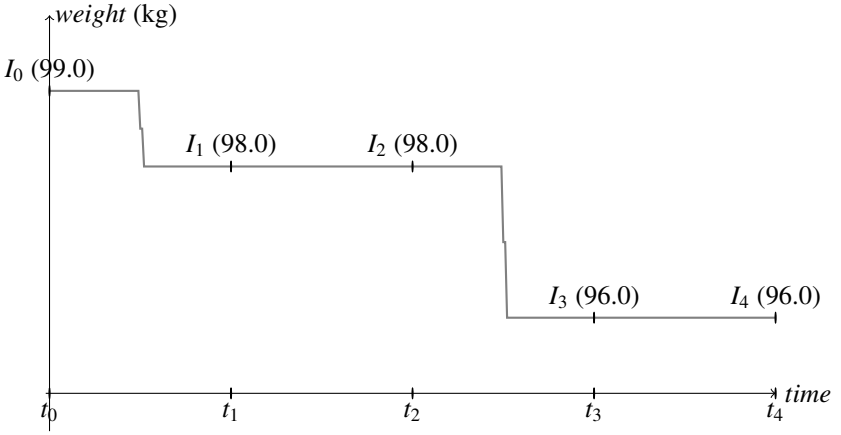

Fig. 5. Extrapolation example.

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

[1] Kang, JS. & Chung, K. & Hong, E.J. 2021 Multimedia knowledgebased bridge health monitoring using digital twin. DOI: https://doi.org/10.1007/s11042-021-10649-x

[2] Hafez, Wael. 2020. Human Digital Twin: Enabling Human-Multi Smart Machines Collaboration. DOI: http://dx.doi.org/10.1007/978-3-030-29513-4_72

[3] Barricelli, Barbara & Casiraghi, Elena & Gliozzo, Jessica & Petrini, Alessandro & Valtolina, Stefano. 2020. Human Digital Twin for Fitness Management. IEEE Access. PP. 1-1. DOI: http://dx.doi.org/10.1109/ACCESS.2020.2971576

[4] Health Effects of Overweight and Obesity in 195 Countries over 25 Years. New England Journal of Medicine, Volume 377, Number 1, 2017, Pages 13-27. DOI: https://doi.org/10.1056/NEJMoa1614362

[5] Roberto, Christina A & Swinburn, Boyd & Hawkes, Corinna & Huang, Terry T-K & A Costa, Sergio & Ashe, Marice & Zwicker, Lindsey & Cawley, John H & Brownell, Kelly D Patchy progress on obesity prevention: emerging examples, entrenched barriers, and new thinking. The Lancet, Volume 385, Issue 9985, 2015, Pages 2400-2409, ISSN 0140-6736. DOI: https://doi.org/10.1016/S0140-6736(14)61744-X

[6] Jan, Arif & Weir, Connor. 2019. BMI Classification Percentile And Cut Off Points. Available in: https://pubmed.ncbi.nlm.nih.gov/31082114/

[7] Mark Weiser. 1999. The computer for the 21st century. SIGMOBILE Mob. Comput. Commun. Rev. 3, 3 (July 1999), 311. DOI: https://doi.org/10.1145/329124.329126

[8] Schofield, WN. 1985. Predicting basal metabolic rate, new standards and review of previous work. Human nutrition. Clinical nutrition, Volume 39 Suppl 1, Pages 5-41. PMID: 4044297. https://www.researchgate.net/publication/285910339_Predicting_basal_metabolic_rate_new_standards_and_review_of_previous_work

[9] Matthews, JJ. & Stanhope, EN. & Godwin, MS. & Holmes, MEJ. & Artioli, GG. The Magnitude of Rapid Weight Loss and Rapid Weight Gain in Combat Sport Athletes Preparing for Competition: A Systematic Review. Int J Sport Nutr Exerc Metab. 2019 Jul 1;29(4):441452. PMID: 30299200. DOI: https://doi.org/10.1123/ijsnem.2018-0165

[10] Bray, G. & Fisher, D. & Chopra, I. 1976. Relation of thyroid hormones to body-weight. The Lancet, Volume 307, Issue 7971, Pages 1206-1208. DOI: https://doi.org/10.1016/S0140-6736(76)92158-9

[11] Sanjoaquin, M. & Appleby, P. & Spencer, E. & Key, T. 2004. Nutrition and lifestyle in relation to bowel movement frequency: A cross-sectional study of 20 630 men and women in EPICOxford. Public Health Nutrition, 7(1), 77-83. DOI: https://doi.org/10.1079/PHN2003522

[12] NCD Risk Factor Collaboration. A century of trends in adult human height. eLife 2016;5:e13410. DOI: https://doi.org/10.7554/eLife.13410

[13] Apple Developer Documentation. BodyMass. Retrieved April 7, 2021 from https://developer.apple.com/documentation/healthkit/hkquantitytypeidentifier/1615693-bodymass

[14] Apple Developer Documentation. HKQuantitySample. Retrieved April 7, 2021 from https://developer.apple.com/documentation/healthkit/hkquantitysample

[15] Microsoft Corporation. DateTime.Ticks Property (System). Retrieved April 7, 2021 from https://docs.microsoft.com/en-us/dotnet/api/system.datetime.ticks?view=net-5.0

[16] Huami Inc. Mi Fit. App Store Preview. Retrieved April 13, 2021 from https://apps.apple.com/us/app/mi-fit/id938688461