Handling noise and missing values in sensory data

Hoogendoorn, Mark; Funk, Burkhardt

published in
Machine Learning for the Quantified Self
2018

DOI (link to publisher)
10.1007/978-3-319-66308-1_3

document version
Publisher's PDF, also known as Version of record

document license
Article 25fa Dutch Copyright Act

Chapter 3
Handling Noise and Missing Values in Sensory Data

In the previous chapter we have aggregated the sensory data and put it neatly into a matrix **X**. By doing so, we are able to reduce some noise. However, it is likely that **X** still contains faulty or noisy measurements that pollute our data and hinder us from working on the machine learning tasks defined in Sect. 2.4. For instance, GPS sensors might be imprecise and the estimated position might jump between the northern and southern hemisphere. The same holds for accelerometers and nearly all types of sensors. Furthermore, some measurements could be missing, e.g. the heart rate monitor might temporarily fail. Although a variety of machine learning techniques exist that are reasonably robust against such noise, the importance of handling these issues is recognized in various research papers (see e.g. [128]). We have three types of approaches at our disposal that can assist us here:

1. We can use approaches that detect and remove *outliers* from the data.
2. We can *impute* missing values in our data (that could also have been outliers that were removed).
3. We can *transform* our data in order to identify the most important parts of it.

We will consider a number of approaches that fall within these categories. An overview of them is shown in Table 3.1 including some characteristics and a very brief summary. Note that nearly all these approaches are tailored towards numerical attributes, except for the distance based outlier detection algorithms and the mode and model-based imputation.

M. Hoogendoorn and B. Funk, *Machine Learning for the Quantified Self*,
Cognitive Systems Monographs 35, https://doi.org/10.1007/978-3-319-66308-1_3

Table 3.1 Methods discussed in this chapter

Approach	Purpose	Specific for $\mathbf{X}^{\mathcal{T}}$	Number of attibributes considered	Brief summary
Chauvenets criterion	Outlier detection	No	1	Identify values for an attribute that are unlikely given a single normal distribution to describe the data.
Mixture model-based outlier detection	Outlier detection	No	1	Identify values for an attribute that are unlikely given a combinations of distributions to describe the data.
Simple distance-based outlier detection	Outlier detection	No	$1, \ldots, p$	Identify instances with a great distance to other points.
Local outlier factor	Outlier detection	No	$1, \ldots, p$	Identify instances that have a lower local density than its neighboring points.
Mean imputation	Missing value imputation	No	1	Impute the mean value for an attribute for an unknown value or outlier.
Median imputation	Missing value imputation	No	1	Impute the median value for an attribute for an unknown value or outlier.
Mode imputation	Missing value imputation	No	1	Impute the mode value for an attribute for an unknown value or outlier.
Interpolation-based imputation	Missing value imputation	Yes	1	Impute the value for an attribute by extrapolating the previous and next measurement.
Model-based imputation	Missing value imputation	No	1	Impute the value for an attribute by creating a model to predict it.
Kalman filter	Outlier detection & Missing value imputation	Yes	$1, \ldots, p$	Estimate expected values based on historical observations and impute them when values are too deviant.
Lowpass Butterworth filter	Transformation	Yes	1	Remove periodic irrelevant data of a single attribute over time.
Principal Component Analysis	Transformation	No	p	Condense most of the variability of the data in a set of new features.

3.1 Detecting Outliers

When we get started with our dataset we are potentially confronted with some extreme values that are highly unlikely to occur. We will call these *outliers*. When working with data from physical sensors as in our case, outliers are very likely. We define an outlier as follows:

Definition 3.1 An outlier is an observation point that is distant from other observations (cf. [53]).

Observations in the sense of this definition can be two different things: we can consider single values of one attribute X_j as an observation (x_i^j), or we can consider complete instances as an observation (x_i). We will see that some approaches we discuss can only handle single attributes while others can cope with complete instances.

We can have two types of outliers: those caused by a *measurement error* and those simply caused by *variability* of the phenomenon that we observe or measure. Typically, we are interested in getting a full picture on what was caused by the phenomenon under study while we try to get rid of the measurement errors. When considering our example Arnold, a measurement of a heart rate of 300 would be considered a measurement error (unless our friend has some form of superpowers) whereas a heart rate of 195 might be very uncommon but could simply be a measurement of Arnold trying to push his limits. While we would clearly like to remove the measurement errors or replace them with more realistic values (which will hopefully yield a better performance of our machine learning approaches), we should be very careful not to remove the outliers caused by the variability in the measured quantity itself. Obviously, life will not always be as clear cut, and we are in need of approaches that can assist us in the process. One approach is to remove measurement errors based on domain knowledge rather than based on machine learning. For example, we know that a heart rate can never be higher than 220 beats per minute and cannot be below 27 beats per minute (the current world record). So, we remove all values outside of this range and interpret them as missing values. This will often be the right choice, but there are situations in which outliers by their existence carry information, e.g. a heart rate above 220 bpm is not possible but might reflect a situation of extreme physical stress causing the chest strap not to work properly. Hence, there is the possibility that we filter out important information.

An additional problem we might encounter is that domain knowledge is not widely accessible or to a large extent it is unknown how to define outlier for a domain. What can we do, if we do not possess this type of domain knowledge and have no up-front knowledge on what an outlier is? Below, we will treat various approaches that can help us to remove outliers. We will assume we do not have any knowledge on what outliers are. Hence, we consider it being an unsupervised problem. Be aware that this process is **dangerous** as there is a high risk of removing points that are not measurements errors and might in fact be the most interesting points in our dataset. One thing we can do is perform visual inspection to make sure we do not remove any

valuable information. Alternatively, we can also just try whether we improve on our machine learning tasks when we remove them. We roughly follow the categorization of [59] for outlier detection algorithms, discussing distribution-based models first, followed by distance-based approaches.

3.1.1 Distribution-Based Models

The first approach we will consider for outlier removal is based on the probability distribution of the data. Here, the data should follow a certain known distribution and we remove those that are outside of certain bounds of the distribution. These approaches are mainly targeted at single attributes X_j.

3.1.1.1 Chauvenets Criterion

When we consider Chauvenets criterion (cf. [35]), we assume the data to follow the normal distribution. Given that we have N measurements for attribute X_j, we compute the mean μ and standard deviation σ of our data:

$$\mu = \frac{\sum_{n=1}^{N} x_n^j}{N} \tag{3.1}$$

$$\sigma = \sqrt{\frac{\sum_{n=1}^{N} (x_n^j - \mu)^2}{N}} \tag{3.2}$$

Together, these values define a normal distribution $\mathcal{N}(\mu, \sigma^2)$. According to Chauvenet's criterion we reject a measurement from a dataset of size N when its probability of observation is less than $\frac{1}{2N}$. A generalization of this criterion is to replace the value 2 with a parameter c, we will follow this generalization in the remainder of this explanation. We can compute the probability of observing a value of at most x_i^j as follows:

$$P(X \leq x_i^j) = \int_{-\infty}^{x_i^j} \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{(u-\mu)^2}{2\sigma^2}} \delta u \tag{3.3}$$

A point is considered an outlier when one of the following two cases holds:

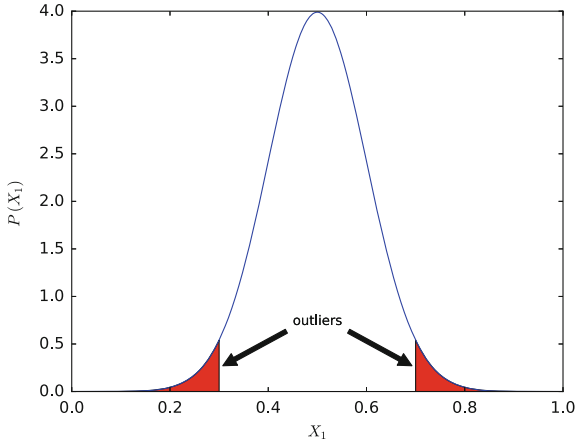


Fig. 3.1 Outlier detection based on Chauvenet's criterion

$$(1 - P(X \leq x_i^j)) < \frac{1}{cN} \tag{3.4}$$

$$P(X \leq x_i^j) < \frac{1}{cN} \tag{3.5}$$

where c is a positive constant number roughly between 1 and 10 that specifies the degree of certainty for the identification of outliers given the assumption of a normal distribution. A higher c corresponds to higher chance that identified outliers are truly outliers. Graphically, the outliers are visualized in Fig. 3.1. The red areas reflect a low probability (less than $\frac{1}{cN}$) of observing measurements that are not outliers—in other words we assume measurements in this area to be outliers (remember, we do not have ground truth here, although we do assume the normal distribution). Alternatives to Chauvenet's criterion exist that are based on the same assumptions yet a bit more sophisticated, see e.g. [52].

3.1.1.2 Mixture Models

While the previous approach is straightforward, it does assume that a single distribution can be fitted to our measurements. This might not always be realistic. If we think of accelerometer data, our data could for example follow a combination of two normal distributions one bump for measurements in case of a user being inactive and one for the same user being active. We can solve this problem with *mixture models*. Assuming we have K distributions to describe our data, e.g. K normal distributions $\{\mathcal{N}(\mu_1, \sigma_1), \ldots, \mathcal{N}(\mu_K, \sigma_K)\}$, we would then like to find the values of the parameters for those individual distributions (i.e. $\mu_1 \ldots \mu_k, \sigma_1, \ldots, \sigma_K$) that when combined best describe the data. We formulate the probability of observing a value x_n^j for a specific measurement:

$$p(x_n^j) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n^j | \mu_k, \sigma_k) \tag{3.6}$$

With

$$\sum_{k=1}^{K} \pi_k = 1 \tag{3.7}$$

$$\forall k : 0 < \pi_k \leq 1 \tag{3.8}$$

Here, π_k expresses the weight of each distribution. The sum of the weights is scaled to 1 to make sure the total area under the curve remains 1. We need to find the parameters that maximize the probability of observing the data we have measured, specified by means of the likelihood:

$$L = \prod_{n=1}^{N} p(x_n^j) \tag{3.9}$$

In other words, we maximize the product of the probabilities of observing our attribute values; the higher the probabilities of the individual attribute values the higher the product. One way to do so is the Expectation-Maximization algorithm (cf. [18]) which you can explore in the exercises. Once we have found the best parameters, we can consider identifying outliers again by considering the probability of each observation. Points with the lowest probabilities are candidates for removal. The exact criterion to apply here depends on the data at hand. In order to find the best number of distributions to fit the data, multiple approaches have been developed, see [61] for an overview.

3.1.2 Distance-Based Models

A second type of algorithm to detect outliers is to consider the distance between a point and the other points in the dataset. We will treat distance metrics between points in more detail in Chap. 5. For now assume that we have a metric to compute the distance between two instances x_i and x_j called $d(x_i, x_j)$. This is different from the distribution-based approach which only focused on individual attributes. Of course, one can also consider individual attributes for the distance-based approaches (just consider $p = 1$).

3.1.2.1 Simple Distance-Based Approach

The first approach takes a global view towards the data: we consider the distance of a point to all other points. We define a certain minimum distance d_{min} within which we consider a point to be close to another point. We say that a point is an outlier when more than a fraction f_{min} of the points in the dataset is at a distance of more than d_{min} (cf. [72]) from that point. Formally:

$$outlier(x_i) = \begin{cases} 1 & \frac{\sum_{n=1}^{N} d\_over(x_i, x_n, d_{min})}{N} > f_{min} \\ 0 & otherwise \end{cases} \tag{3.10}$$

where

$$d\_over(x, y, d_{min}) = \begin{cases} 1 & d(x, y) > d_{min} \\ 0 & otherwise \end{cases} \tag{3.11}$$

Again, the parameter settings for f_{min} and d_{min} are crucial for this approach in order to work well. An example is shown in Fig. 3.2, where we have two relevant attributes, X_1 and X_2 representing the x- and y-axis of the accelerometer data of Arnold. The red dots represent data points where the x-axis and y-axis stand for their values of X_1 and X_2 respectively. Assuming we want to determine whether the black dot is an outlier, we compute for each point whether it occurs within distance d_{min}. This distance from our point is indicated by the circle, assuming a certain distance metric.

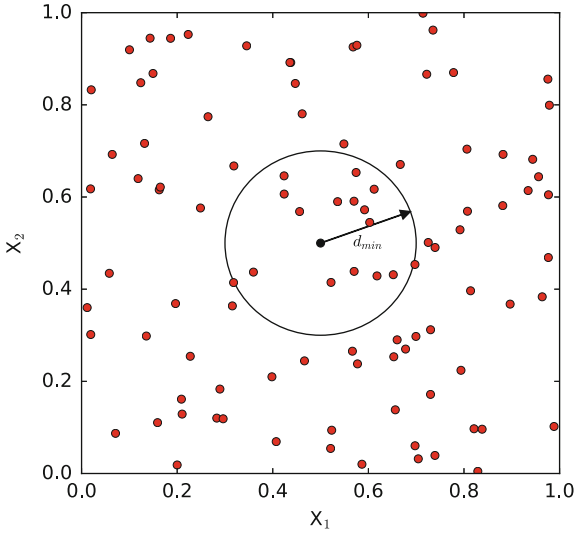


Fig. 3.2 Outliers based on distance

We see that fifteen other points lie within this distance while all other are outside. Depending on our parameter settings we could consider this an outlier or not.

3.1.2.2 Local Outlier Factor

Instead of taking a global look at points the local outlier factor approach (cf. [30]) only takes points into account that surround it. Some areas in our data space might be quite dense while others are not. Taking this into account might improve our detection of outliers. In addition, the approach specifies a degree of outlierness (i.e. the likelihood of an instance being an outlier). This was missing in the previous distance based approach, although it would not be difficult to change the previous approach slightly to account for this. Imagine the scenario shown in Fig. 3.3. We see data points on the lower left that form a sort of cluster, let us call this cluster 1. On the upper right we see a similar, yet less dense cluster, which we call cluster 2. Consider the two points visualized by black dots, point 1 being the lower left point, point 2 the one in the upper right. Given the shape of cluster 1, point 1 is likely to be considered an outlier while point 2 is probably not, given the shape of cluster 2. Hence, even though the distances to the points in the cluster are similar we treat them differently.

Let us dive into the approach in a bit more detail. The first step taken is to define the k-distance k_{dist} of a point x_i. This is defined as the largest distance among the distances of the k closest points. To describe this, we define the following constraints for $k_{dist}(x_i)$ of a point x_i:

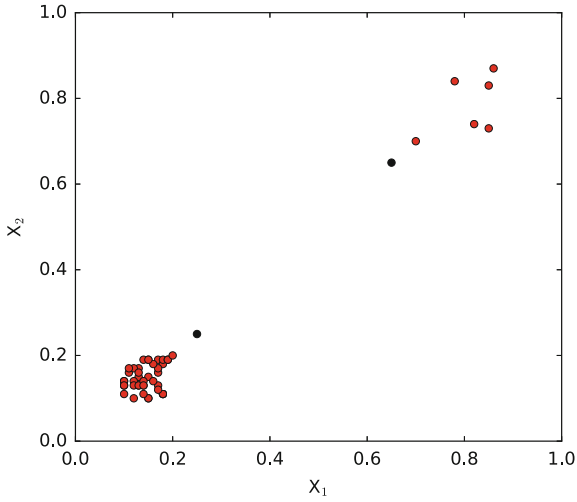


Fig. 3.3 Example outliers for local outlier factor

$$|\{x|x \in \{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N\} \wedge d(x, x_i) \leq k_{dist}(x_i)\}| \geq k \quad (3.12)$$

$$|\{x|x \in \{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N\} \wedge d(x, x_i) < k_{dist}(x_i)\}| \leq (k-1) \quad (3.13)$$

In other words, there should be $k-1$ points or less with a distance less than k_{dist} and at least one point with the same distance. The set of neighbors within this distance is called k_{dist_nh}:

$$k_{dist\_nh}(x_i) = \{x|x \in \{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N\} \wedge d(x, x_i) \leq k_{dist}(x_i)\} \quad (3.14)$$

We define the reachability distance of a point x_i to another point x as:

$$k_{reach\_dist}(x_i, x) = max(k_{dist}(x), d(x, x_i)) \quad (3.15)$$

This expresses that a reachability distance is the real distance if the point x_i is not among the k nearest points of x (in that case the value for $d(x, x_i)$ will be larger than $k_{dist}(x)$) and otherwise it is k_{dist} of that point, so we set the distance value of all points within $k_{dist}(x)$ equal to $k_{dist}(x)$. Next, consider the local reachability density around our point x_i:

$$k_{lrd}(x_i) = 1/\left(\frac{\sum_{x \in k_{dist\_nh}(x_i)} k_{reach\_dist}(x_i, x)}{|k_{dist\_nh}(x_i)|}\right) \quad (3.16)$$

In our definition, we look at the neighbors x of x_i within k_{dist}, their reachability distance to x_i, and divide this by the number of neighbors. Intuitively this says something on how close point x_i is to its neighbors. We divide 1 by this number, so the lower the average distance to ones neighbors, the higher the value. In order to see how much of an outlier the point is compared to its neighbors, we consider the local reachability density of those neighbors and compare them:

$$k_{lof}(x_i) = \frac{\sum_{x \in k_{dist\_nh}(x_i)} \frac{k_{lrd}(x)}{k_{lrd}(x_i)}}{|k_{dist\_nh}(x_i)|} \quad (3.17)$$

We compare the values for our point x_i and our neighboring points. Remember that a high value for k_{lrd} represents a closer proximity to neighbors. This formula expresses that the higher the scores for x_i on the k_{lrd} compared to its neighbors, the lower the local outlier factor will become, which makes perfect sense.

3.2 Imputation of Missing Values

Obviously, our dataset could contain a lot of missing values. This could be caused by a lot of outliers which were removed, or possibly by sensors not providing information at certain points in time. There are different ways to replace these missing values, we refer to this process as *imputation*.

The first approach we can take is to impute the *mean* value of an attribute calculated over the instances where the value is known. This is a common approach. The approach does have disadvantages when we have data with a lot of extreme values, as this severely impacts the value of the mean. The *median* is a robust alternative for these cases as it is less sensitive to extreme values. Note that all these approaches target numerical values. For categorical values, we can use the *mode*.

A more sophisticated approach is to predict the missing value for attribute j of instance i (x_i^j) using statistical models such as linear regression. In general, there are two ways this can be done:

1. we consider the values for the other attributes of the same instance and predict x_i^j using those values (i.e. $x_i^1, \ldots, x_i^{j-1}, x_i^{j+1}, \ldots, x_i^p \rightarrow x_i^j$);
2. we take the previous (and possibly next) values of the same attribute. For the latter we need a temporal dataset ($\mathbf{X}^{\mathcal{T}}$). Hence, we predict in the following way: $x_1^j, \ldots, x_{i-1}^j, x_{i+1}^j, \ldots, x_N^j \rightarrow x_i^j$.

We do not need to use the entire set of attributes (first case) or instances (second case) but can also consider a subset. A simple example of the second approach is to take the previous and next value of the specific attribute and average the values (again assuming numerical values), i.e.

$$x_i^j = \frac{x_{i-1}^j + x_{i+1}^j}{2} \tag{3.18}$$

Or in case we know that measurement x_{i-k}^j is the last available measurement while x_{i+l}^j is the first next measurement we can compute it in the following way (assuming a fixed sampling rate of our data):

$$x_i^j = x_{i-k}^j + k \cdot \frac{x_{i+l}^j - x_{i-k}^j}{(k+l)} \tag{3.19}$$

This is a simple form of linear interpolation and works under the assumption that x_i^j follows a linear trend. Of course, this does not work for the first and last time points. In these cases we can either use the next (or previous) couple of time point and extrapolate the trend from those points. The method described above belongs to the set of parametric imputation methods that rely on making assumptions on the distributions of the data as well as the underlying regression relationships. Instead, it is also possible to use non-parametric approaches to imputation. We can think of

these methods as fitting locally-weighted regressions to the data. That is why these approaches are also called local imputation schemes [3].

Some approaches allow for the usage of values of multiple attributes over time. An example of this is the Kalman filter which we will discuss in the next section.

3.3 A Combined Approach: The Kalman Filter

An approach that identifies outliers and also replaces these with new values is the Kalman filter (cf. [66]). The Kalman filter provides a model for the expected values based on historical data and estimates how noisy a new measurement is by comparing the observed values with the predicted values. Imagine Arnold running through his favorite park in Amsterdam, the Vondelpark. We continuously obtain GPS signals on his whereabouts. Suddenly we see a strange measurement: Arnold is supposed to have moved one kilometer in 10 seconds. While we should obviously never underestimate Arnold's physical shape we also know he is neither superman nor a Ferrari. The Kalman filter can find this strange anomaly and will insert a more reliable value instead. This can also work really well in a real time setting. Again, we need a dataset with temporal ordering for this to work.

Let us dive into the method in a bit more detail. In the Kalman filter, we distinguish between a latent state s_t and the measurements that can be performed based on the state, in our case x_t, or x_t^j if we want to consider single measurements. In our example case, the state of Arnold would be the actual presence at a certain location, his actual physical state, etcetera. Our measurement would be the GPS coordinate, activity level, heart rate and so on. We can express the next value of a state as:

$$s_t = F_t s_{t-1} + B_t u_t + w_t \tag{3.20}$$

Here, F_t and B_t are matrices while s_t is a vector that represents the previous value of the state, u_t is the control input to the state (we might want to adjust the state, e.g. by sending a message to Arnold, we will return to this in Chap. 9), and w_t represents process white noise. F_t expresses how the previous state influences the new state (by means of weights associated for each component of the state) while B_t represents how the control input influences the different components of the next state.

The value for the measurement associated with the state is:

$$x_t = H_t s_t + v_t \tag{3.21}$$

H_t is again a matrix while v_t is the measurement white noise. The noise values are assumed to be taken from a multivariate normal distribution with covariance matrix Q_t (representing the process noise covariance) and R_t (the measurement noise covariance) respectively:

$$w_t = \mathcal{N}(0, Q_t) \tag{3.22}$$

$$v_t = \mathcal{N}(0, R_t) \tag{3.23}$$

Given this model, we can start to make predictions of the next state. Let $\hat{s}_{t|t-1}$ represent the estimation of the state value s_t given observations up to $t-1$ (i.e. an a priori estimate) and $\hat{s}_{t|t}$ the a posteriori estimate given all observations including the observation at time t.

Remember that the noise plays an important role, since it determines the uncertainty we encounter when estimating the state. We assume that we have a matrix containing the level of noise or error we expect to see, denoted as P_t. The matrix contains the covariance and variances of the Gaussian probability density functions that characterize the error in our predictions. $P_{t|t-1}$ represents our a priori estimation of the error while $P_{t|t}$ is our a posteriori estimation. We start by making our predictions:

$$\hat{s}_{t|t-1} = F_t \hat{s}_{t-1|t-1} + B_t u_t \tag{3.24}$$

$$P_{t|t-1} = \mathbb{E}[(s_t - \hat{s}_{t|t-1})(s_t - \hat{s}_{t|t-1})^T] \tag{3.25}$$

Equation 3.24 is easy to understand given the previous Eq. 3.20 for the next value of the state. Equation 3.25 updates our estimation on the error based on our estimation of the state. Note that we cannot directly compute $P_{t|t-1}$ this way as we do not know s_t, but you can derive the value of $P_{t|t-1}$ in a different way. It is beyond the scope of this book to go into detail on this. The equation above does provide some intuition on the meaning of $P_{t|t-1}$.

The next step is to update our model based on our a priori predictions and the actual observations we perform. Here our matrix H_t plays an important role, recall that this is the mapping from a state to measurements associated with that state. First, we can determine the difference between a measurement x_t and our a priori estimate for the measurement, we will refer to this as e_t:

$$e_t = x_t - H^T \hat{s}_{t|t-1} \tag{3.26}$$

And we can create an estimation for the covariance for our measurements S_t given our matrix H_t:

$$S_t = H_t P_{t-1|t-1} H_t^T + R_t \tag{3.27}$$

In order to make the best estimate of the state given the error we observe between the actual and expected measurements we can define the *optimal Kalman gain*, defined as:

$$K_t = P_{t|t-1} H_t^T S_t^{-1} \tag{3.28}$$

And we use this to define the a posteriori estimation of the state based on our a priori estimate and the error between our model and the actual measurements:

$$\hat{s}_{t|t} = \hat{s}_{t|t-1} + K_t e_t \tag{3.29}$$

Finally, we update our covariance matrix:

$$P_{t|t} = (I - K_t H_t) P_{t|t-1} \tag{3.30}$$

The sequence of equations might be a bit overwhelming, but we hope that the intuition behind the approach has become clear: Basically, we distinguish between a state, that is not directly observed, and a measurement. Kalman filtering is then a set of transition equations that uses measurements to describe how the state is evolving. There are ample books and tutorials that discuss Kalman filtering in more detail [130].

We can use the Kalman filter by letting it run over our data, and then analyze when the deviation (i.e. e_t) is too big. In case it is (apparently it is noise), we can use our estimate and otherwise we use the original value. Of course, we keep on updating our model independent of whether we accept a measurement or not since this allows us to better estimate the noise levels.

Next to using the Kalman filter to detect and impute outliers, it can also be used for sensor fusion to extract information from a combination of sensors. Think again of our friend Arnold we could fuse data from various sensors (e.g. GPS location, accelerometer, step counter, speed) to determine his exact location, being our latent state. See [111] for an example of how to use the Kalman filter for sensor fusion.

3.4 Transformation

The next approaches for handling noisy data is to transform our data in a way that subtle noise (not the huge outliers we have seen before) is filtered and the parts of our data that explain most of the variance are identified. We will explain two approaches: the lowpass filter (which can be applied to individual attributes) and Principal Component Analysis, which works across the entire dataset.

3.4.1 Lowpass Filter

The lowpass filter can be applied to data that is of temporal nature (i.e. $\mathbf{X}^{\mathcal{T}}$), and assumes that there is a form of periodicity. Think about accelerometer data for example, if we are walking, we will see periodic measurements in our accelerometer data at a frequency around 1 Hz as walking is a repetitive pattern. When we process our data, we can filter out such periodic constituents based upon their frequency. We could say for instance that any measurement we perform that is at a higher frequency than our walking behavior is irrelevant and can be considered as noise (they might actually hamper our machine learning process). Hence, we want to remove the data originating from a form of periodicity above a certain frequency, and leave the data at lower frequencies untouched. How we move from measurement values over time to frequencies is explained in more detail in Chap. 4 when we discuss Fourier Transformations. An example for cutting out parts of the frequency spectrum and the corresponding periodic behavior is the Butterworth filter. Assuming f_c to be the cutoff frequency we represent the transfer function G (usually specified in dB due to the traditional application in the audio domain) of the signal with frequency f by the following (simplified) equation:

$$|G(f)|^2 = \frac{1}{1 + (f/f_c)^{2n}} \tag{3.31}$$

We see that the higher the frequency f, the lower the magnitude of the transfer function G, which is exactly what we want: we want to filter out the high frequency data and let the low frequency data pass. The parameter n is the order of the filter. The higher the order, the more steeply the magnitude of the frequencies above the cutoff frequency f_c drop. Let us consider an application of the filter to exemplify its working.

Figure 3.4 shows an example that combines two sinusoid functions with different frequencies, one with a frequency of 1 Hz and one with a much lower frequency, namely 0.1 Hz. We apply a Butterworth filter with a cutoff frequency of 0.5 Hz (i.e. everything with a higher frequency is filtered) to the combined data and see that we obtain the single low frequency sinusoid: we have filtered out the high frequency data. Lowpass filters have for instance been used in [7, 32, 36, 128] to clean up the data.

3.4.2 Principal Component Analysis

The Butterworth filter addresses individual attributes, transforms the signal into the frequency domain, and then picks a specific part of the frequency spectrum (e.g. low frequencies). We can also consider a set of attributes at the same time and extract features that explain most of the variation observed over all attributes. To

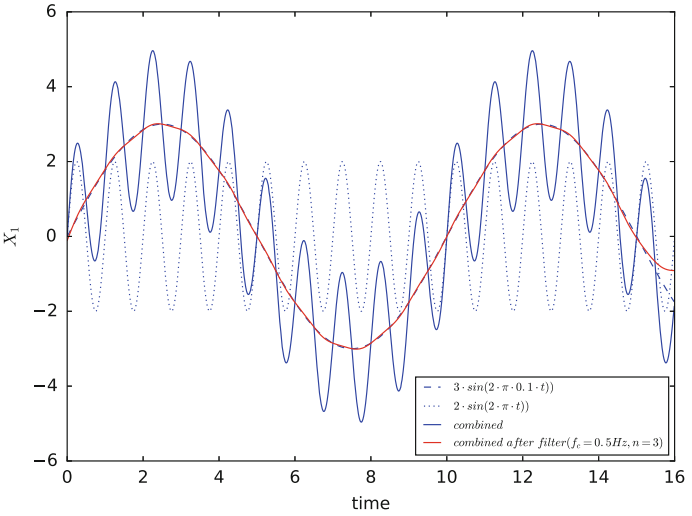


Fig. 3.4 Example application of Butterworth filter

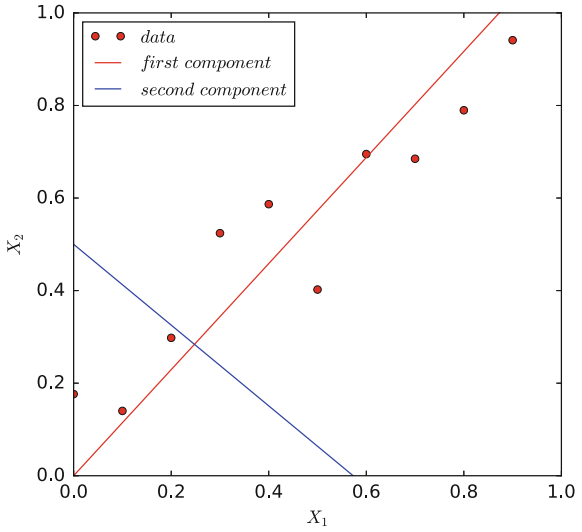


Fig. 3.5 Example dataset for principal component analysis

understand this, let us consider Fig. 3.5, in which we have two attributes X_1 and X_2. The measurements might represent the accelerometer data of Arnold and are expressed by the red points. We see that an increase in the measurement value for X_1 (acceleration on the x-axis) goes hand in hand with an increasing value for X_2 (acceleration on the y-axis).

How can we explain the variance in the data? We can actually see that the solid red line describes our data in a reasonably precise way. If we would know the equation

of this line, and we would project all points onto that line (so we ignore the distance to the line) we would be able to express our data points by a single value X_{new} on the line instead of value pairs. In fact, the distance from the line could be noise, which we could get rid of in one go as well. The other line (the blue line) is a line perpendicular (at an angle of 90^o) to our previously found line. It can be used as a secondary axis to express the distance of a point from our previous line. If we use both, we obviously do not lose any information, and hence, we might not get rid of the noise. Here, we only exemplify this procedure for two attributes, however, it is applicable to an arbitrary number of attributes.

The main goal of Principal Component Analysis (PCA) (see e.g. [64]) is to find vectors that represent these lines (or hyperplanes if we have more than two attributes) and order them in terms of how much variance in the data is explained. There are different ways to find these vectors. We will explain one. We start by defining the co-variance matrix of our data. The covariance between an attribute X_i and X_j is defined as follows:

$$cov(X_i, X_j) = \frac{\sum_{n=1}^{N}(x_n^i - \bar{X}_i)(x_n^j - \bar{X}_j)}{N} \tag{3.32}$$

where $\bar{X}_k$ is the mean value for attribute X_k:

$$\bar{X}_k = \frac{\sum_{n=1}^{N} x_n^k}{N} \tag{3.33}$$

The covariance matrix is then defined in the following way:

$$C = \begin{pmatrix} cov(X_1, X_1) & \cdots & cov(X_1, X_p) \\ \vdots & \ddots & \vdots \\ cov(X_p, X_1) & \cdots & cov(X_p, X_p) \end{pmatrix} \tag{3.34}$$

The covariance matrix expresses how the values of our attributes relate to each other, i.e. to what extent values are are correlated. The covariance of the same attribute (i.e. $cov(X_i, X_i)$) is the variance of the attribute. If we divide each element of the covariance matrix by the product of the per attribute standard deviations, i.e. $\sigma_i \sigma_j$, we obtain the matrix of all correlation coefficients. If an element of this matrix is 1, values of both attributes across our instances deviate from the mean in the same way (i.e. if we have a low value of one attribute compared to the mean, the same holds for the other attribute). In case of a correlation of -1 they behave in precisely the opposite way. Note that our matrix is symmetric ($cov(X_i, X_j) = cov(X_j, X_i)$).

How does this help us to find our line in Fig. 3.5 to describe the data? Well, once we have obtained this matrix we can find its *eigenvectors*. These are vectors that when

they are multiplied with our covariance matrix, we end up with (a multiplication of) our original vector, i.e.

$$\begin{pmatrix} cov(X_1, X_1) & \cdots & cov(X_1, X_p) \\ \vdots & \ddots & \vdots \\ cov(X_p, X_1) & \cdots & cov(X_p, X_p) \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ \vdots \\ v_p \end{pmatrix} = f \cdot \begin{pmatrix} v_1 \\ \vdots \\ v_p \end{pmatrix} \tag{3.35}$$

Here, the factor of the multiplication f is the *eigenvalue* of the *eigenvector*. To avoid counting multiplications of our eigenvectors twice we only consider the normalized eigenvectors:

$$\begin{pmatrix} sv_1 \\ \vdots \\ sv_p \end{pmatrix} = \begin{pmatrix} v_1 \\ \vdots \\ v_p \end{pmatrix} \div \sqrt{v_1^2 + \cdots + v_p^2} \tag{3.36}$$

Typically, each covariance matrix of dimensions $p \times p$ (remember that we have p attributes) has exactly p normalized eigenvectors. These eigenvectors are perpendicular. To find these eigenvectors ample approaches are available which we will not discuss here. These eigenvectors in fact are precisely the lines we have considered in our problem setting before. The normalized eigenvector with the highest eigenvalue is the line (or hyperplane for $p > 2$ attributes) that explains most variance in the data. This is called the principal component (hence the name of the approach).

Let $\{sv_1^i, \ldots, sv_p^i\}$ represent the values of the eigenvector with the ith highest eigenvalue f_i. We can project our data to these new axes (just as we have explained for the example). If we select $n \leq p$ eigenvectors we obtain a new (projected) dataset as follows:

$$\begin{pmatrix} x_1^1 & \cdots & x_1^p \\ \vdots & \ddots & \vdots \\ x_N^1 & \cdots & x_N^p \end{pmatrix} \cdot \begin{pmatrix} sv_1^1 & \cdots & sv_1^n \\ \vdots & \ddots & \vdots \\ sv_p^1 & \cdots & sv_p^n \end{pmatrix} = \begin{pmatrix} \mathbb{X}_1^1 & \cdots & \mathbb{X}_1^n \\ \vdots & \ddots & \vdots \\ \mathbb{X}_N^1 & \cdots & \mathbb{X}_N^n \end{pmatrix} \tag{3.37}$$

In the matrix on the right hand side of Eq. 3.37, each row still represents an instance of our original dataset. However, we have reduced it to n attributes instead of p.

If we take $n < p$, we do lose some information. Given our ordering however, the last eigenvectors are not likely to explain a lot of variance. Typically, the first number of components (i.e. those with the highest eigenvalues) explain most of it. In the case study we will see a typical example and illustrate that there is often a clear cut-off point.

Let us return to the previous example from Fig. 3.5. We have already visualized the principal component (solid red line) and the second component (dashed blue line). The principal component is specified as $< sv_1^1, sv_2^1 >=< 0.7044, 0.7089 >$ with $f_1 = 22.13$ and $< sv_1^2, sv_2^2 >=< -0.7089, 0.7044 >$ with $f_2 = 0.0392$. We can see that the eigenvalue of the first vector is much higher. This makes a

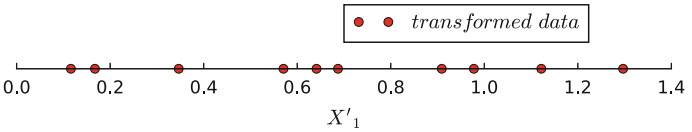


Fig. 3.6 Our example data after application of the first principal component

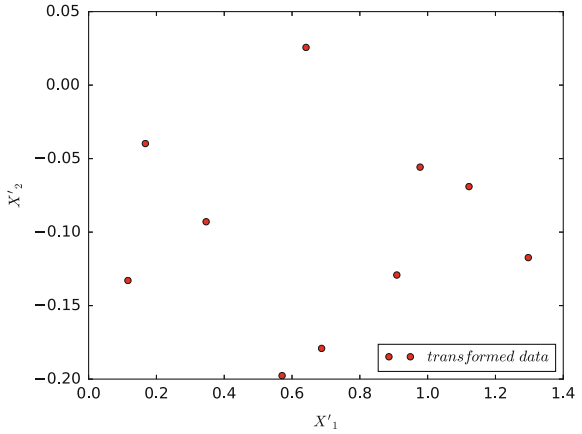


Fig. 3.7 Our example data after application of the first two (i.e. all) principal components

lot of sense, as it explains nearly all variance while the second one does not significantly contribute. If we apply only the first component and transform the data, we obtain the new dataset visualized in Fig. 3.6. Hence, we only have a single dimension representing the positioning of a data point on the line spanned by the first principal component. Figure 3.7 visualizes the new dataset when we use both components. Here, only the values have changed due to the two new axes. We see PCA being applied in various research papers related to the quantified self, including [16, 19].

Great, we are now able to extract useful features from our data using PCA. The big disadvantage is that we often lose the ability to interpret our models as we now have a new space with new values that are not immediately interpretable by a domain expert. In low dimensional cases it can be helpful to look for correlations between original attributes and the principal components to support interpretation.

3.5 Case Study

Let us consider the crowdsignals dataset again. We will now iteratively try the different approaches we have presented in this section and select the most appropriate ones to process our dataset. We will pass the various approaches in the same order they were explained before. First, we will show how to apply outlier detection in this dataset and filter extreme values. Then, we will impute missing values. Next, we will

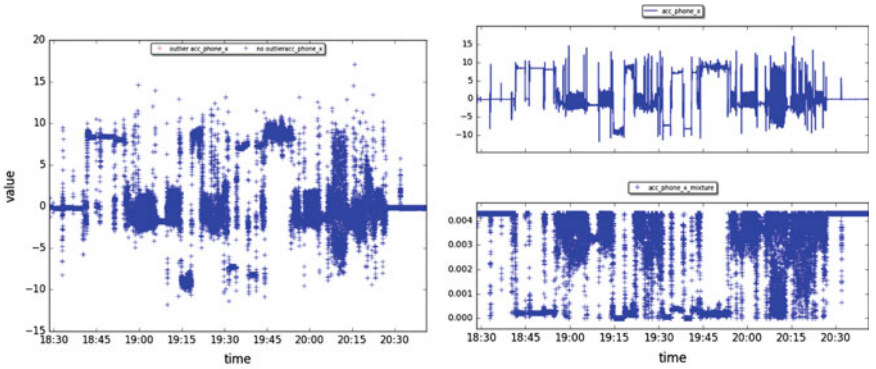
look at Kalman filtering as an integrated alternative approach to the previous steps. Finally, we end with the application of lowpass filtering and principal component analysis.

3.5.1 Outlier Detection

We have seen various outlier detection algorithms, but which one is best for our current dataset? To determine this, we will explore two representative types of measurements, namely the *acc_phone_x* which varies widely (similar to the other accelerometer measurements, magnetometers and gyroscope), and *light_phone_lux* which seems to be pretty stable except for a few extreme values (similar to heart rate and pressure). First, we try to explore the parameter setting of the different algorithms that result in reasonable detection of outliers. We do this by visual inspection and by seeing whether points we would visually consider to be outliers are indeed flagged without flagging points that seem normal. Figure 3.8 shows our four outlier approaches applied to *acc_phone_x* while Fig. 3.9 shows the same for *light_phone_lux*. These are the type of figures we use for our visual inspection. Note that we consider attributes in isolation here while our distance-based approaches would allow us to look at combinations of attributes. We have made this choice because we want to compare all approaches, and we still obtain good results in terms of finding outliers. To generate Figs. 3.8 and 3.9 we used the following parameter settings:

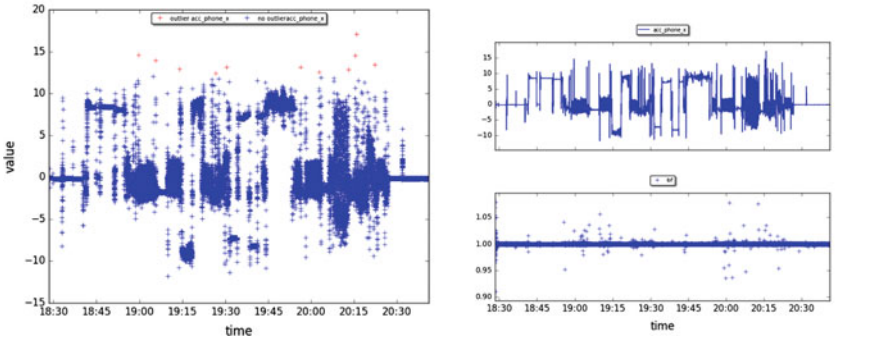
- Chauvenet's criterion: we set the value $c = 2$, according to the traditional Chauvenet criterion.
- Mixture models: we use 3 mixture components and a single iteration of the algorithm
- Simple distance-based approach: we set $d_{min} = 0.1$ and $f_{min} = 0.99$ and use Euclidean distance.
- Local outlier factor: we use 5 for the value of k and Euclidean distance.

In Figs. 3.8 and 3.9 we can see that Chauvenet's criterion does signal outliers for the *light_phone_lux* attribute: we find 33 outliers that seem to make sense. For the *acc_phone_x* we do not find any outliers, and visual inspection indeed shows that there are not very clear outliers. Note that we did not explicitly check whether the distribution of the values follow the normal distribution in this case, but the visual inspection does show that the outliers found are appropriate. You can explore this issue more in the exercises. The mixture models seems to work fine for *light_phone_lux* as well: extreme and rare values get a probability of observing of around 0. For *acc_phone_x* we again do not see very clear outliers; this is a sign that very obvious outliers are indeed missing. Our simple distance-based outlier detection finds outliers for the two examples: we see some outliers for both cases (27 for *light_phone_lux* and 11 for *acc_phone_x*). Finally, the local outlier factor does show changes in values



(a) Chauvenet's criterion

(b) Mixture model (upper part shows the observed values, bottom part shows the probability of observing the data)

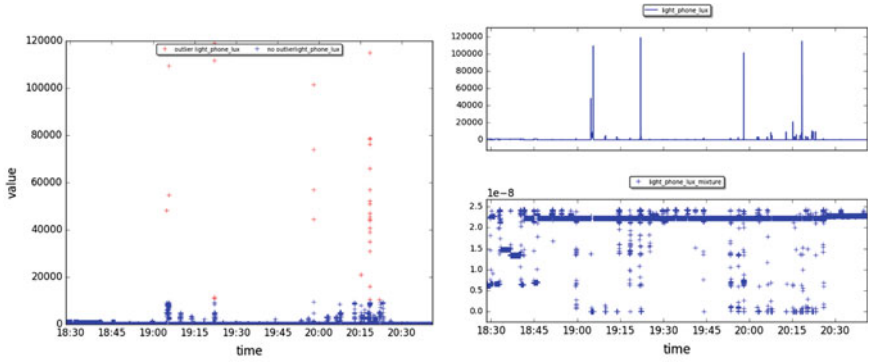


(c) Simple distance-based approach

(d) Local outlier factor (upper part shows the observed values, bottom part shows the k-lof score)

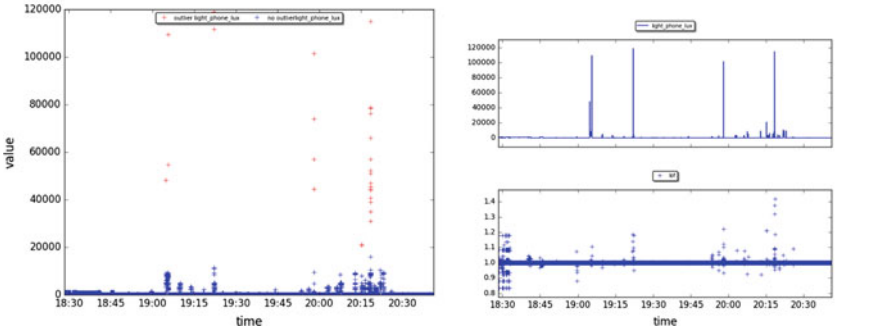
Fig. 3.8 Outlier for the attribute *acc_phone_x*

for outliers, but is in our opinion less clear compared to the simpler distance-based approach. In addition, it is computationally more demanding. Based on our observations, we have decided to apply a filtering of the outliers (replacing them with an unknown value) using the Chauvenet criterion: we want to be on the safe side and not throw away data points for which it is not so obvious that they are outliers. We apply this to all attributes except for the labels (that are just binary and do not contain outliers). Using a single parameter across all attributes has a severe risk which we are completely aware of, but visual inspection showed that the outliers that were removed seemed fairly reasonable.



(a) Chauvenet's criterion

(b) Mixture model (upper part shows the observed values, bottom part shows the probability of observing the data)



(c) Simple distance-based approach

(d) Local outlier factor (upper part shows the observed values, bottom part shows the k-lof score)

Fig. 3.9 Outlier for the attribute *light_phone_lux*

3.5.2 Missing Value Imputation

Now that we have removed the extreme values, we are left with a number of missing values. One option is removing the instances that contain missing values. However, this would result in a loss of valuable data. Therefore, we consider imputation of the missing values. We have seen that the heart rate attribute contains most missing values, so let us use the heart rate as an example. Figure 3.10 shows the result of imputation by using the mean and interpolation. Clearly, for this type of time series, interpolation techniques are preferred for imputation: it results in much more natural values. This holds for temporal sequences in general, but if we do not have this temporal ordering it is impossible. We apply this to all attributes with missing values (except the label attributes).

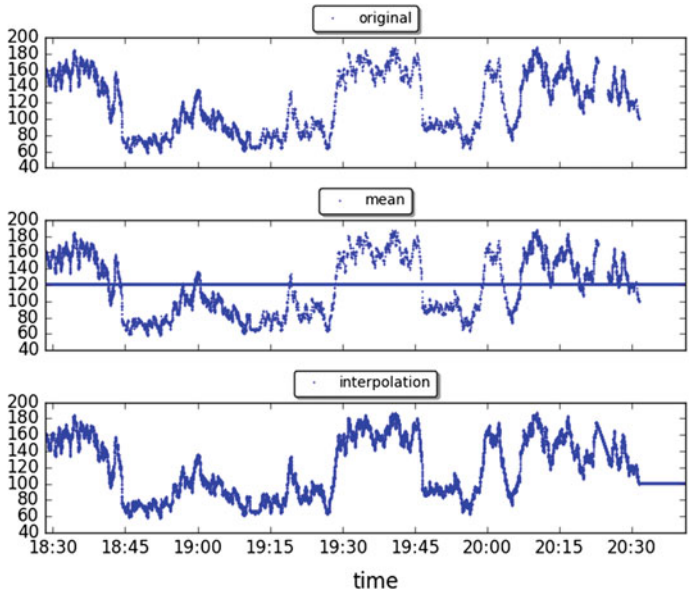


Fig. 3.10 Missing value imputation for *hr_watch_rate*. The *top panel* shows the original data followed by imputation using the mean, and interpolation respectively

3.5.3 Kalman Filter

As an alternative to our two step approach (identifying outliers and imputing missing values) we can also apply the Kalman filter to perform both tasks simultaneously. However, we do not have a known model that relates our observations to true values. Therefore, we are required to use a very simple model which directly links observations to real values per attribute. Other parameters of the Kalman filter (e.g. Q_t, R_t) can be automatically tuned towards the dataset. Despite its simplicity, this approach is able to capture whether a measured value deviates from the expectations and can replace it with the expected value based on the past values. In case of an unknown value it can impute its predicted value.

Figure 3.11 shows an example for the attribute *acc_phone_x*. We see that the application of the filter results in the values being dampened. While this can be useful in some cases, we will not further pursue this avenue for this dataset but stick to our outlier detection with the simple distance based approach and imputation by interpolation. In case a model would be known that relates (multiple) measurements to true values (or states) the filter could certainly be very useful. Also in an online setting (where data comes in on the fly) where we might not have access to a complete dataset, or do not have sufficient time to run our outlier detection algorithms, Kalman filtering could be useful.

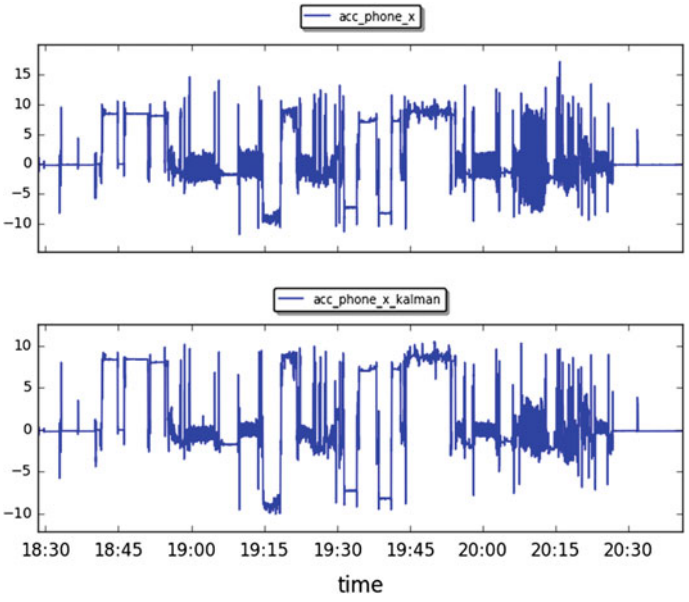


Fig. 3.11 Kalman filter applied to *acc_phone_x*, the *top graph* shows the original values, the *bottom* the values after applying the Kalman filter (note the differences in scales)

3.5.4 Data Transformation

The Butterworth lowpass filter allows us to remove some of the high frequency noise we potentially have in our dataset that might disturb our learning process. We would only expect this for the accelerometers, magnetometers, and the gyroscope. Let us focus on one accelerometer measurement, namely *acc_phone_x*. Given that we know that walking behavior has a frequency between 1 and 1.5 Hz, we look at the influence of filtering data with a frequency above 1.5 Hz. We use an order of 20 in order to guarantee that most of this noise is removed. Figure 3.12 shows the result. We indeed see that the data we obtain after filtering seems much cleaner (and easier to learn from) so we will use the filtered data in the remainder of the book.

Finally, we can try to find the principal components that explain most of the variance in our measurements (note that we consider all measurements/attributes here). If we include the value with respect to those components instead of the raw measurements, it might enable us to achieve better predictive capabilities. Of course, we should only use our predictors and not include our eventual targets. We apply principal component analysis to all our attributes except for the labels (target for classification) and heart rate (target for regression). You can see that we simplify things a bit as we could have also created separate sets for the regression and classification problems, but we do not feel that would add anything given that most of the variance is in the other attributes.

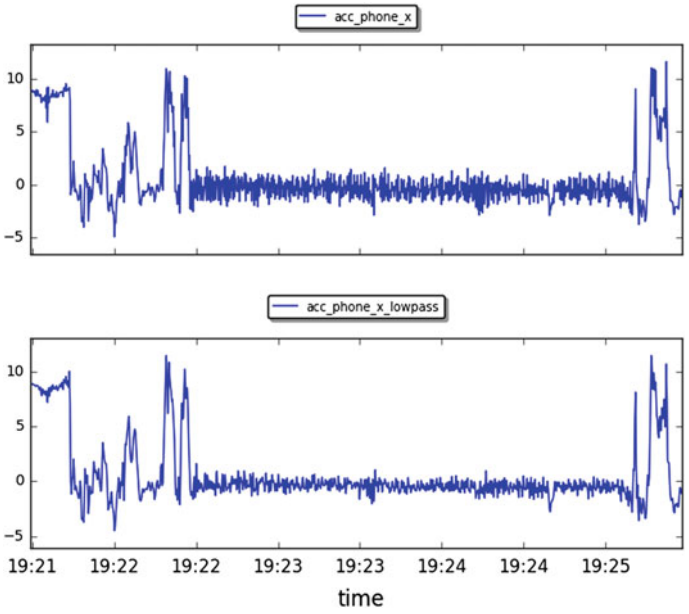


Fig. 3.12 Original data (*top*) and filtered data for *acc_phone_x* with frequencies above 1.5 Hz filtered. Note the time scale, we zoomed into part of the data

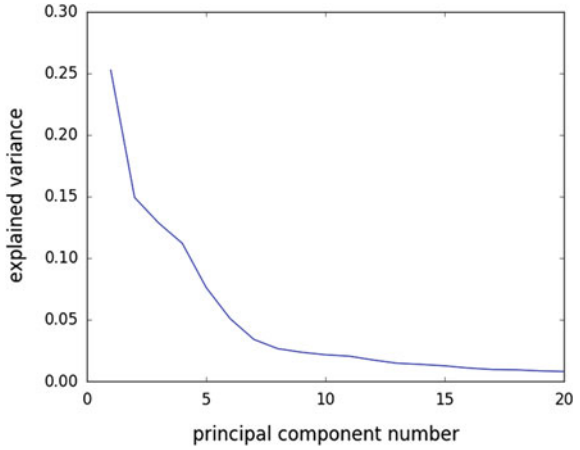


Fig. 3.13 Explained variance by principal components ranked on importance

Figure 3.13 shows the explained variance by the different principal components. We clearly observe that the explained variance declines after 7 components (this is sometimes referred to as the elbow). We therefore decide to select 7 components and include the value of each of the components (for each time point) into our dataset.

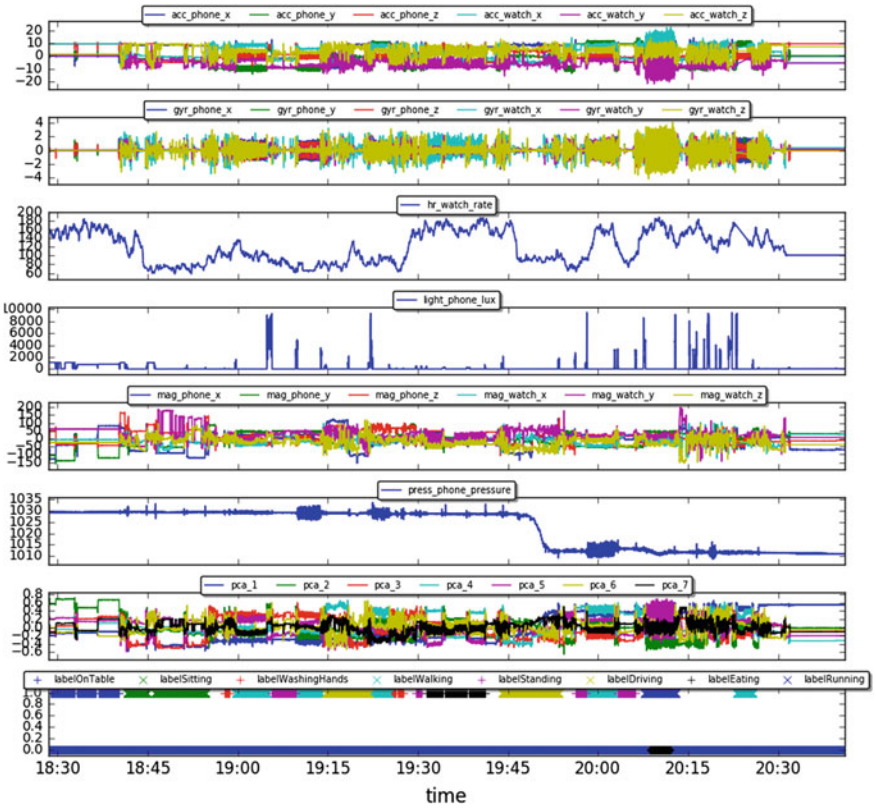


Fig. 3.14 Dataset after Chap. 3

Finally, the processed dataset after all steps we have just explained is shown in Fig. 3.14. Note the change in the scales of some axes due to the removal of the outliers.

3.6 Exercises

3.6.1 Pen and Paper

1. In our quantified self setting, we might face datasets of different users. In the description of the techniques in the chapter, we have focused on a single dataset only. When we apply these approaches, should we apply the techniques (and make choices) based on individual user datasets, or should we apply them on the

combination of all datasets? Provide at least two arguments in favor of each these two options.
2. We have seen two types of outlier detection algorithms: distance and distribution based. In what situations would it be better to apply a distance based outlier detection algorithm over a distribution-based approach?
3. In the simple distance-based approach we have seen two parameters, namely f_{min} and d_{min}. Explain the way in which you would find appropriate values for these parameters.
4. The local outlier factor algorithm is quite complex. Find out what the computational complexity of the algorithm is and discuss ways to improve the scalability of the approach.
5. We have seen that the Kalman filter assumes a model that relates observations to states. Imagine that we do not have such a model, but that we just map the observations and states one-by-one in a direct way (in fact we have done this for the crowdsignals case). Explain what the Kalman filter will entail when we take such an approach, so how does it update its model and what values would it predict?

3.6.2 Coding

1. One of the criteria to allow for applying Chauvenet's criterion is that the data follows a normal distribution. We did not actually verify this in our application of the filter. Study for at least two sequences of sensory values in the crowdsignals data (including the *acc_phone_x* we used in our case study) whether they are indeed normally distributed.
2. To generate Figs. 3.8 and 3.9 we have used the parameter settings described in Sect. 3.5.1. Vary the constant c (smaller and larger values) of the Chauvenet's criterion and study the dependency of the number of detected outliers on c. Repeat this for the other three methods presented for outlier detection. Use the source code from book's website, that generated the figures, as a starting point for the analysis.
3. Use a model-based approach to impute the heart rate
4. Similarly to what we have done for our crowdsignals dataset, apply the techniques that have been discussed in this chapter to the dataset you have collected yourself. Write down your observations and argue for certain choices you have made.
5. In line with the previous question, do the same for the case you found covering the data of multiple people. Think about the answer you gave to one of the *pen and paper* questions: how should you tackle the issue with multiple datasets?